

Intel® Stratix® 10 Avalon®-ST and Single Root I/O Virtualization (SR-IOV) Interface for PCI Express* Solutions User Guide

Updated for Intel® Quartus® Prime Design Suite: **18.0**



Subscribe

Send Feedback

UG-20032 | 2018.09.04

Latest document on the web: [PDF](#) | [HTML](#)



Contents

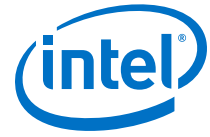
1. Introduction.....	6
1.1. Avalon-ST Interface with Optional SR-IOV for PCIe Introduction.....	6
1.2. Features.....	7
1.3. Release Information	8
1.4. Device Family Support	9
1.5. Recommended Speed Grades.....	10
1.6. Performance and Resource Utilization	11
1.7. Transceiver Tiles.....	11
1.8. PCI Express IP Core Package Layout.....	12
1.9. Channel Availability.....	16
2. Quick Start Guide.....	18
2.1. Design Components.....	18
2.2. Directory Structure.....	19
2.3. Generating the Design Example.....	20
2.4. Simulating the Design Example.....	22
2.5. Compiling the Design Example and Programming the Device.....	23
2.6. Installing the Linux Kernel Driver.....	23
2.7. Running the Design Example Application.....	24
3. Interface Overview.....	26
3.1. Avalon-ST RX Interface.....	29
3.2. Avalon-ST TX Interface.....	29
3.3. TX Credit Interface.....	29
3.4. TX and RX Serial Data.....	29
3.5. Clocks.....	29
3.6. Function-Level Reset (FLR) Interface.....	30
3.7. Control Shadow Interface for SR-IOV.....	30
3.8. Configuration Extension Bus Interface.....	30
3.9. Hard IP Reconfiguration Interface.....	30
3.10. Interrupt Interfaces.....	31
3.11. Power Management Interface.....	31
3.12. Reset.....	31
3.13. Transaction Layer Configuration Interface.....	31
3.14. PLL Reconfiguration Interface.....	31
3.15. PIPE Interface (Simulation Only).....	32
4. Parameters	33
4.1. Stratix 10 Avalon-ST Settings.....	34
4.2. Multifunction and SR-IOV System Settings.....	34
4.3. Base Address Registers.....	34
4.4. Device Identification Registers.....	35
4.5. TPH/ATS Capabilities.....	36
4.6. PCI Express and PCI Capabilities Parameters.....	37
4.6.1. Device Capabilities.....	37
4.6.2. Link Capabilities	38
4.6.3. MSI and MSI-X Capabilities	38
4.6.4. Slot Capabilities	39



4.6.5. Power Management	40
4.6.6. Vendor Specific Extended Capability (VSEC).....	40
4.7. Configuration, Debug and Extension Options.....	41
4.8. PHY Characteristics	41
4.9. Example Designs.....	42
5. Designing with the IP Core.....	43
5.1. Generation.....	43
5.2. Simulation.....	43
5.2.1. Selecting Serial or PIPE Simulation	44
5.3. IP Core Generation Output (Intel Quartus Prime Pro Edition).....	44
5.4. Integration and Implementation.....	47
5.4.1. Clock Requirements.....	47
5.4.2. Reset Requirements.....	47
5.5. Required Supporting IP Cores.....	48
5.5.1. Hard Reset Controller.....	48
5.5.2. TX PLL.....	48
5.6. Channel Layout and PLL Usage.....	49
6. Block Descriptions.....	55
6.1. Interfaces.....	56
6.1.1. TLP Header and Data Alignment for the Avalon-ST RX and TX Interfaces.....	58
6.1.2. Avalon-ST 256-Bit RX Interface.....	59
6.1.3. Avalon-ST 512-Bit RX Interface.....	64
6.1.4. Avalon-ST 256-Bit TX Interface.....	67
6.1.5. Avalon-ST 512-Bit TX Interface.....	72
6.1.6. TX Credit Interface.....	74
6.1.7. Interpreting the TX Credit Interface.....	75
6.1.8. Clocks.....	77
6.1.9. Update Flow Control Timer and Credit Release.....	77
6.1.10. Function-Level Reset (FLR) Interface	77
6.1.11. Resets.....	78
6.1.12. Interrupts.....	80
6.1.13. Control Shadow Interface for SR-IOV.....	80
6.1.14. Transaction Layer Configuration Space Interface.....	81
6.1.15. Configuration Extension Bus Interface.....	84
6.1.16. Hard IP Status Interface.....	85
6.1.17. Hard IP Reconfiguration.....	87
6.1.18. Power Management Interface.....	88
6.1.19. Serial Data Interface.....	89
6.1.20. PIPE Interface.....	89
6.1.21. Test Interface.....	92
6.1.22. PLL IP Reconfiguration.....	93
6.1.23. Message Handling.....	94
6.2. Errors reported by the Application Layer.....	94
6.2.1. Error Handling.....	95
6.3. Power Management.....	96
6.3.1. Endpoint D3 Entry	96
6.3.2. End Point D3 Exit.....	97
6.3.3. Exit from D3 hot	97
6.3.4. Exit from D3 cold.....	97



6.3.5. Active State Power Management.....	97
6.4. Transaction Ordering.....	97
6.4.1. TX TLP Ordering.....	97
6.4.2. RX TLP Ordering.....	98
6.5. RX Buffer.....	98
6.5.1. Retry Buffer.....	99
6.5.2. Configuration Retry Status.....	99
7. Interrupts.....	100
7.1. Interrupts for Endpoints.....	100
7.1.1. MSI and Legacy Interrupts	100
7.1.2. MSI-X	103
7.1.3. Implementing MSI-X Interrupts.....	103
7.1.4. Legacy Interrupts	105
8. Registers.....	106
8.1. Configuration Space Registers.....	106
8.1.1. Register Access Definitions.....	108
8.1.2. PCI Configuration Header Registers.....	109
8.1.3. PCI Express Capability Structures.....	110
8.1.4. Intel Defined VSEC Capability Header	113
8.1.5. General Purpose Control and Status Register.....	114
8.1.6. Uncorrectable Internal Error Status Register	115
8.1.7. Uncorrectable Internal Error Mask Register.....	115
8.1.8. Correctable Internal Error Status Register	116
8.1.9. Correctable Internal Error Mask Register	117
8.1.10. SR-IOV Virtualization Extended Capabilities Registers Address Map.....	118
9. Testbench and Design Example	127
9.1. Endpoint Testbench	128
9.1.1. Endpoint Testbench for SR-IOV.....	130
9.2. Test Driver Module	130
9.3. Root Port BFM Overview	130
9.3.1. BFM Memory Map	132
9.3.2. Configuration Space Bus and Device Numbering	132
9.3.3. Configuration of Root Port and Endpoint	132
9.3.4. Issuing Read and Write Transactions to the Application Layer	139
9.4. BFM Procedures and Functions	139
9.4.1. ebfm_barwr Procedure	139
9.4.2. ebfm_barwr_imm Procedure	140
9.4.3. ebfm_barrrd_wait Procedure	140
9.4.4. ebfm_barrrd_nowt Procedure	141
9.4.5. ebfm_cfgwr_imm_wait Procedure	141
9.4.6. ebfm_cfgwr_imm_nowt Procedure	142
9.4.7. ebfm_cfgrrd_wait Procedure	142
9.4.8. ebfm_cfgrrd_nowt Procedure	143
9.4.9. BFM Configuration Procedures.....	143
9.4.10. BFM Shared Memory Access Procedures	145
9.4.11. BFM Log and Message Procedures	147
9.4.12. Verilog HDL Formatting Functions	150



10. Troubleshooting and Observing the Link.....	154
10.1. Troubleshooting.....	154
10.1.1. Simulation Fails To Progress Beyond Polling.Active State.....	154
10.1.2. Hardware Bring-Up Issues	154
10.1.3. Link Training	154
10.1.4. Link Hangs in L0 State.....	155
10.1.5. Use Third-Party PCIe Analyzer	156
10.1.6. BIOS Enumeration Issues	156
10.2. PCIe Link Inspector Overview.....	157
10.2.1. PCIe Link Inspector Hardware	157
A. PCI Express Core Architecture.....	171
A.1. Transaction Layer	171
A.2. Data Link Layer	172
A.3. Physical Layer	174
B. TX Credit Adjustment Sample Code.....	177
C. Document Revision History.....	179
C.1. Document Revision History for the Intel Stratix 10 Avalon-ST and Single-Root I/O Virtualization (SR-IOV) Interfaces for PCI Express Solutions User Guide.....	179

1. Introduction

1.1. Avalon-ST Interface with Optional SR-IOV for PCIe Introduction

Intel® Stratix® 10 FPGAs include a configurable, hardened protocol stack for PCI Express* that is compliant with *PCI Express Base Specification 3.0*. The Avalon®-ST Intel Stratix 10 Hard IP for PCI Express IP Core supports Gen1, Gen2 and Gen3 data rates and x1, x2, x4, x8, or x16 configurations.

Figure 1. Intel Stratix 10 PCIe Variant with Avalon-ST Interface

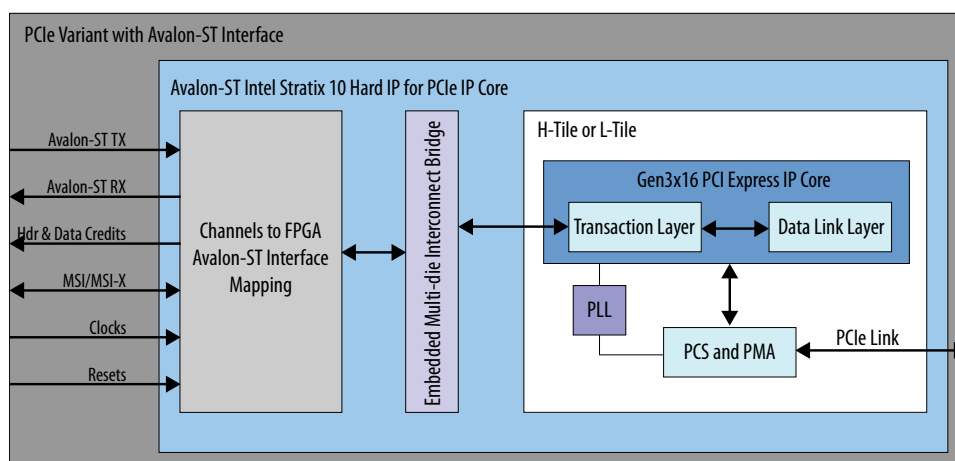
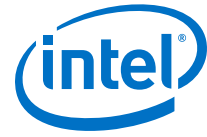


Table 1. PCI Express Data Throughput

The following table shows the theoretical link bandwidth of a PCI Express link for Gen1, Gen2, and Gen3 for 1, 2, 4, 8, and 16 lanes. This table provides bandwidths for a single transmit (TX) or receive (RX) channel. The numbers double for duplex operation. The protocol specifies 2.5 giga-transfers per second (GT/s) for Gen1, 5.0 GT/s for Gen2, and 8.0 GT/s for Gen3. Gen1 and Gen2 use 8B/10B encoding which introduces a 20% overhead. Gen3 uses 128b/130b encoding which introduces about 1.6% overhead.

	Link Width				
	x1	x2	x4	x8	x16
PCI Express Gen1 (2.5 Gbps)	2	4	8	16	32
PCI Express Gen2 (5.0 Gbps)	4	8	16	32	64
PCI Express Gen3 (8.0 Gbps)	7.87	15.75	31.5	63	126



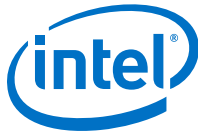
Related Information

- [Introduction to Intel FPGA IP Cores](#)
Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- [Generating a Combined Simulator Setup Script](#)
Create simulation scripts that do not require manual updates for software or IP version upgrades.
- [Project Management Best Practices](#)
Guidelines for efficient management and portability of your project and IP files.
- [Creating a System with the Platform Designer](#)
Provides information about the Platform Designer system integration tool that simplifies the tasks of defining and integrating customized IP cores.

1.2. Features

The Avalon-ST Intel Stratix 10 Hard IP for PCI Express IP Core supports the following features:

- Complete protocol stack including the Transaction, Data Link, and Physical Layers implemented as hard IP.
- x1, x2, x4, x8 and x16 configurations with Gen1, Gen2, or Gen3 lane rates for Native Endpoints.
- Avalon-ST 256-bit interface to the Application Layer except for Gen3 x16 variants.
- Avalon-ST 512-bit interface at 250 MHz to the Application Layer for Gen3 x16 variants.
- Instantiation as a stand-alone IP core from the Intel Quartus® Prime Pro Edition IP Catalog or as part of a system design in Platform Designer.
- Dynamic design example generation.
- Configuration via Protocol (CvP) providing separate images for configuration of the periphery and core logic.
- PHY interface for PCI Express (PIPE) or serial interface simulation using IEEE encrypted models.
- Testbench bus functional model (BFM) supporting x1, x2, x4, and x8 configurations. The x16 configuration downtrains to x8 for Intel (internally created) testbench.
- Support for a Gen3x16 simulation model that you can use in an Avery testbench. The Avery testbench is capable of simulating all 16 lanes. For more information, refer to [AN-811: Using the Avery BFM for PCI Express Gen3x16 Simulation on Intel Stratix 10 Devices](#).
- Altera debug master endpoint (ADME). For more information, refer to [Intel Stratix 10 L- and H-Tile Transceiver PHY User Guide](#).
- Dedicated 69.5 kilobyte (KB) receive buffer.
- End-to-end cyclic redundancy check (ECRC).
- Advanced Error Reporting (AER) for PFs.
Note: In Intel Stratix 10 devices, Advanced Error Reporting is always enabled in the PCIe Hard IP for both the L- and H-Tile transceivers.
- Base address register (BAR) checking logic.



New features in the Intel Quartus Prime Pro Edition 18.0 Software Release

- SR-IOV support for H-Tile devices.
- Separate Configuration Spaces for up to four PCIe Physical Functions (PFs) and a maximum of 2048 Virtual Functions (VFs) for the PFs in H-Tile devices.
- Address Translation Services (ATS) and TLP Processing Hints (TPH) capabilities.
- Control Shadow Interface to read the current settings for some of the VF Control Register fields in the PCI and PCI Express Configuration Spaces.
- Function Level Reset (FLR) for PFs and VFs.
- Message Signaled Interrupts (MSI) for PFs.
- MSI-X for PFs and VFs.
- A PCIe* Link Inspector including the following features:
 - Read and write access to the Configuration Space registers.
 - LTSSM monitoring.
 - Read and write access to PCS and PMA registers.
- Hardware support for dynamically-generated design examples.
- A Linux software driver to test the dynamically-generated design examples.

Note: The purpose of the *Intel Stratix 10 Avalon-ST and Single Root I/O Virtualization (SR-IOV) Interfaces for Solutions User Guide* is to explain how to use this IP. For a detailed understanding of the PCIe protocol, please refer to the *PCI Express Base Specification*.

Related Information

- [Transceiver Tiles](#) on page 11
- [AN-811: Using the Avery BFM for PCI Express Gen3x16 Simulation on Intel Stratix 10 Devices](#)
- [Configuration via Protocol \(CvP\) Implementation User Guide](#)
Provides separate images for periphery and core, reduces system costs, and improves security for the core bitstream.
- [PCI Express Base Specification 3.0](#)
- [Single Root I/O Virtualization and Sharing Specification 1.1](#)

1.3. Release Information

Table 2. Hard IP for PCI Express Release Information

Item	Description
Version	Intel Quartus Prime Pro Edition 18.0 Software
Release Date	May 2018
Ordering Codes	No ordering code is required



Intel verifies that the current version of the Intel Quartus Prime Pro Edition software compiles the previous version of each IP core, if this IP core was included in the previous release. Intel reports any exceptions to this verification in the *Intel IP Release Notes* or clarifies them in the Intel Quartus Prime Pro Edition IP Update tool. Intel does not verify compilation with IP core versions older than the previous release.

Related Information

Timing and Power Models

Reports the default device support levels in the current version of the Quartus Prime Pro Edition software.

1.4. Device Family Support

The following terms define device support levels for Intel FPGA IP cores:

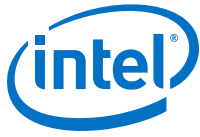
- **Advance support**—the IP core is available for simulation and compilation for this device family. Timing models include initial engineering estimates of delays based on early post-layout information. The timing models are subject to change as silicon testing improves the correlation between the actual silicon and the timing models. You can use this IP core for system architecture and resource utilization studies, simulation, pinout, system latency assessments, basic timing assessments (pipeline budgeting), and I/O transfer strategy (data-path width, burst depth, I/O standards tradeoffs).
- **Preliminary support**—the IP core is verified with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. It can be used in production designs with caution.
- **Final support**—the IP core is verified with final timing models for this device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.

Table 3. Device Family Support

Device Family	Support Level
Intel Stratix 10	Preliminary support.
Other device families	No support. Refer to the <i>Intel PCI Express Solutions</i> web page on the Intel website for support information on other device families.

Related Information

[PCI Express Solutions Web Page](#)



1.5. Recommended Speed Grades

Table 4. Intel Stratix 10 Recommended Speed Grades for All Avalon-ST Widths and Frequencies

The recommended speed grades are for production parts.

Lane Rate	Link Width	Interface Width	Application Clock Frequency (MHz)	Recommended Speed Grades
Gen1	x1, x2, x4, x8, x16	256 bits	125	-1, -2, -3
Gen2	x1, x2, x4, x8	256 bits	125	-1, -2, -3
Gen2	x16	256 bits	250	-1, -2
Gen3	x1, x2, x4	256 bits	125	-1, -2, -3
	x8	256 bits	250	-1, -2
	x16	512 bits	250	-1, -2

1.6. Performance and Resource Utilization

The SR-IOV Bridge and Gen3 x16 adapter are implemented in soft logic, requiring FPGA fabric resources. Resource utilization numbers are not available in the current release.

Related Information

Running the Fitter

For information on Fitter constraints.

1.7. Transceiver Tiles

Intel Stratix 10 introduces several transceiver tile variants to support a wide variety of protocols.

Figure 2. Intel Stratix 10 Transceiver Tile Block Diagram

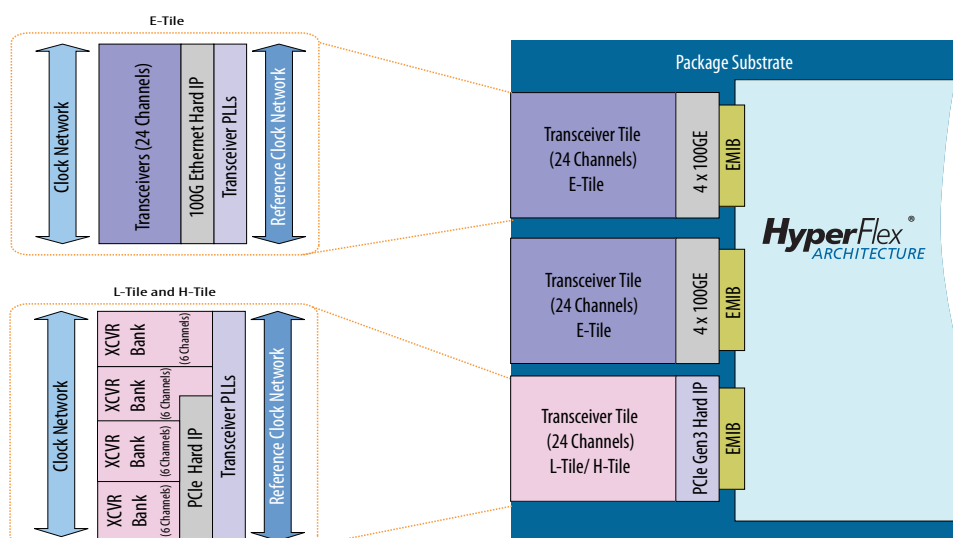


Table 5. Transceiver Tiles Channel Types

Tile	Device Type	Channel Capability		Channel Hard IP Access
		Chip-to-Chip	Backplane	
L-Tile	GX	26 Gbps (NRZ)	12.5 Gbps (NRZ)	PCIe Gen3x16
H-Tile	GX	28.3 Gbps (NRZ)	28.3 Gbps (NRZ)	PCIe Gen3x16
E-Tile	GXE	30 Gbps (NRZ), 56 Gbps (PAM-4)	30 Gbps (NRZ), 56 Gbps (PAM-4)	100G Ethernet

L-Tile and H-Tile

Both L and H transceiver tiles contain four transceiver banks-with a total of 24 duplex channels, eight ATX PLLs, eight fPLLs, eight CMU PLLs, a PCIe Hard IP block, and associated input reference and transmitter clock networks. L and H transceiver tiles also include 10GBASE-KR/40GBASE-KR4 FEC block in each channel.

L-Tiles have transceiver channels that support up to 26 Gbps chip-to-chip or 12.5 Gbps backplane applications. H-Tiles have transceiver channels to support 28 Gbps applications. H-Tile channels support fast lock-time for Gigabit-capable passive optical network (GPON).

Intel Stratix 10 GX/SX devices incorporate L-Tiles or H-Tiles. Package migration is available with Intel Stratix 10 GX/SX from L-Tile to H-Tile variants.

E-Tile

E-Tiles are designed to support 56 Gbps with PAM-4 signaling or up to 30 Gbps backplane with NRZ signaling. E-Tiles do not include any PCIe Hard IP blocks.

1.8. PCI Express IP Core Package Layout

Intel Stratix 10 devices have high-speed transceivers implemented on separate transceiver tiles. The transceiver tiles are on the left and right sides of the device.

Each 24-channel transceiver L- or H- tile includes one Gen3 x16 PCIe IP Core implemented in hardened logic. The following figures show the layout of PCIe IP cores in Intel Stratix 10 devices. Both L- and H-tiles are orange. E-tiles are green.

Figure 3. Intel Stratix 10 GX/SX Devices with 4 PCIe Hard IP Cores and 96 Transceiver Channels

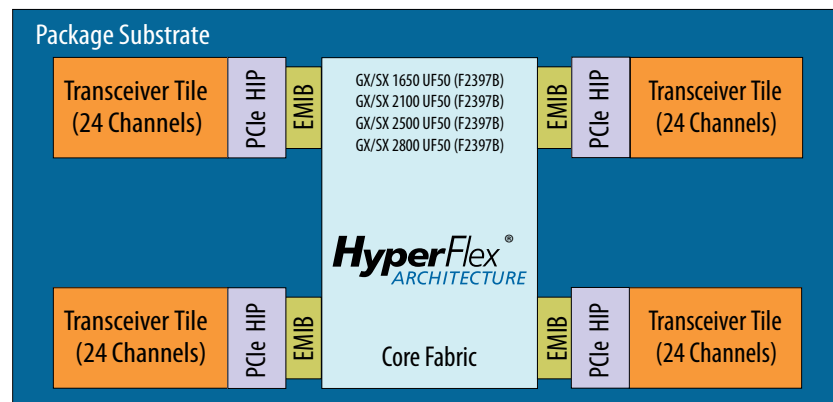


Figure 4. Intel Stratix 10 GX/SX Devices with 2 PCIe Hard IP Cores and 48 Transceiver Channels

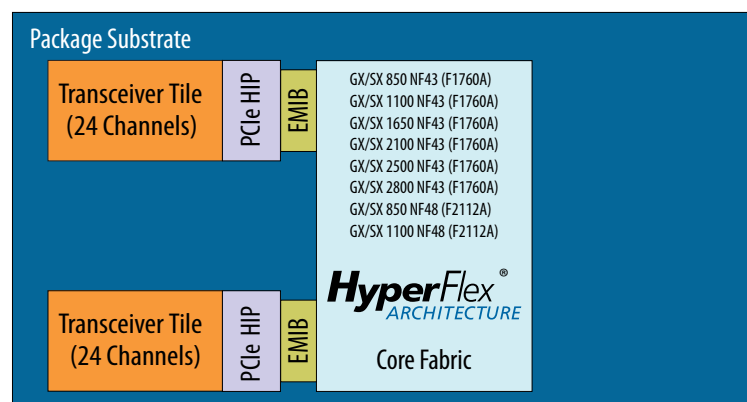


Figure 5. Intel Stratix 10 GX/SX Devices with 2 PCIe Hard IP Cores and 48 Transceiver Channels - Transceivers on Both Sides

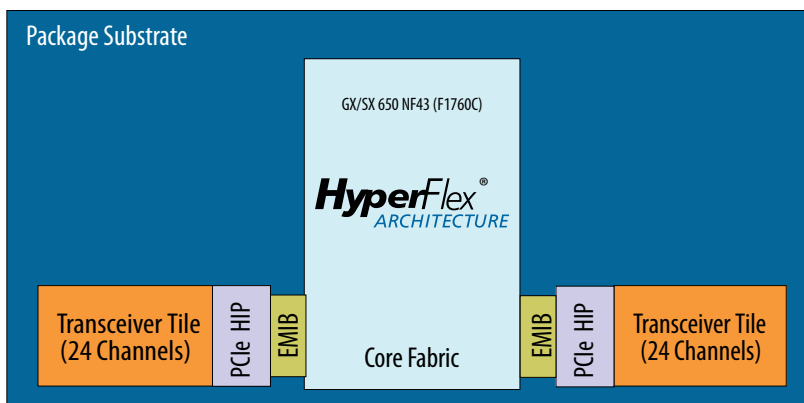
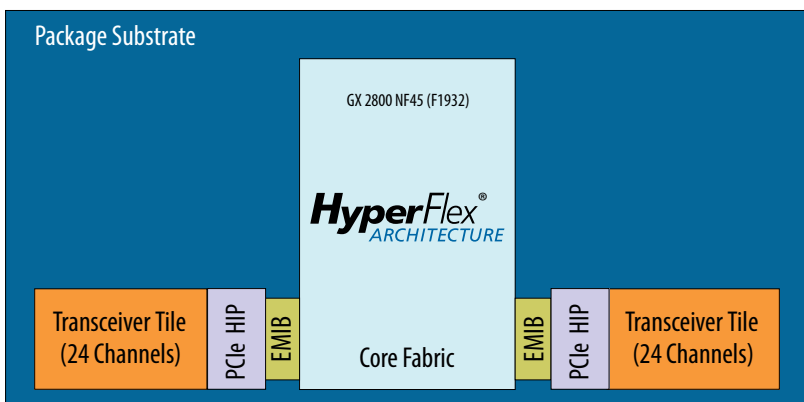


Figure 6. Intel Stratix 10 Migration Device with 2 Transceiver Tiles and 48 Transceiver Channels



Note:

1. Intel Stratix 10 migration device contains 2 L-Tiles which match Intel Arria[®] 10 migration device.

Figure 7. Intel Stratix 10 GX/SX Devices with 1 PCIe Hard IP Core and 24 Transceiver Channels

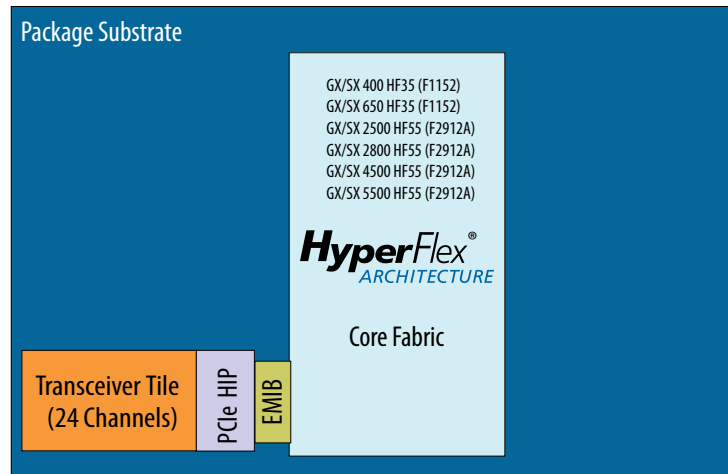
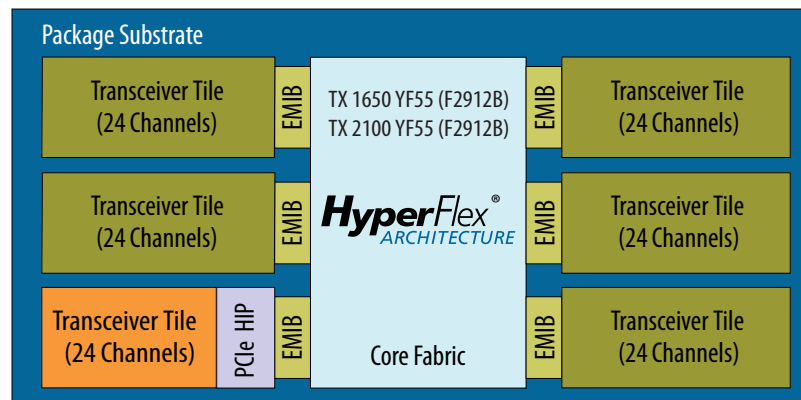


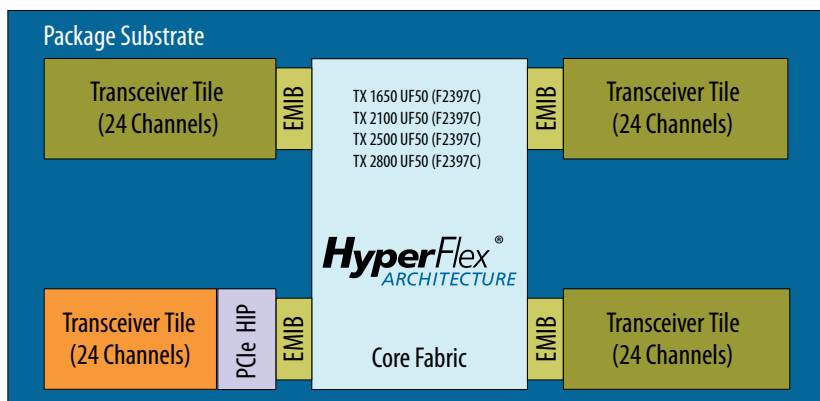
Figure 8. Intel Stratix 10 TX Devices with 1 PCIe Hard IP Core and 144 Transceiver Channels



Note:

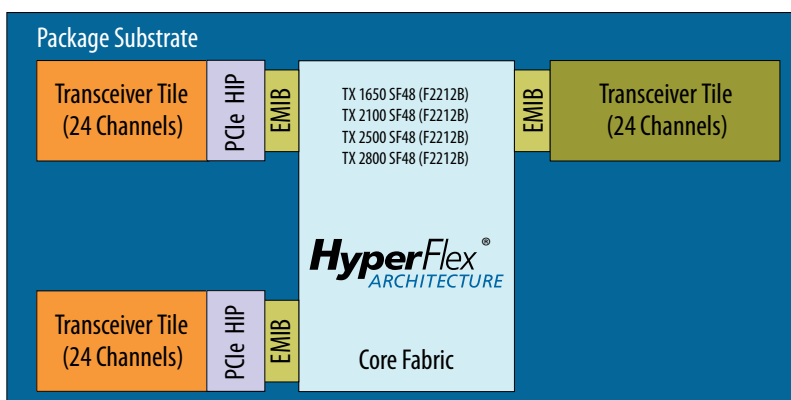
1. Intel Stratix 10 TX Devices use a combination of E-Tiles and H-Tiles.
2. Five E-Tiles support 56G PAM-4 and 30G NRZ backplanes.
3. One H-Tile supports up to 28.3G backplanes and PCIe up to Gen3 x16.

Figure 9. Intel Stratix 10 TX Devices with 1 PCIe Hard IP Core and 96 Transceiver Channels



- Note:**
1. Intel Stratix 10 TX Devices use a combination of E-Tiles and H-Tiles.
 2. Three E-Tiles support 56G PAM-4 and 30G NRZ backplanes.
 3. One H-Tile supports up to 28.3G backplanes PCIe up to Gen3 x16..

Figure 10. Intel Stratix 10 TX Devices with 2 PCIe Hard IP Cores and 72 Transceiver Channels



- Note:**
1. Intel Stratix 10 TX Devices use a combination of E-Tiles and H-Tiles.
 2. One E-Tile support 56G PAM-4 and 30G NRZ backplanes.
 3. Two H-Tiles supports up to 28.3G backplanes PCIe up to Gen3 x16..

Related Information

[Stratix 10 GX/SX Device Overview](#)

For more information about Stratix 10 devices.

1.9. Channel Availability

PCIe Hard IP Channel Restrictions

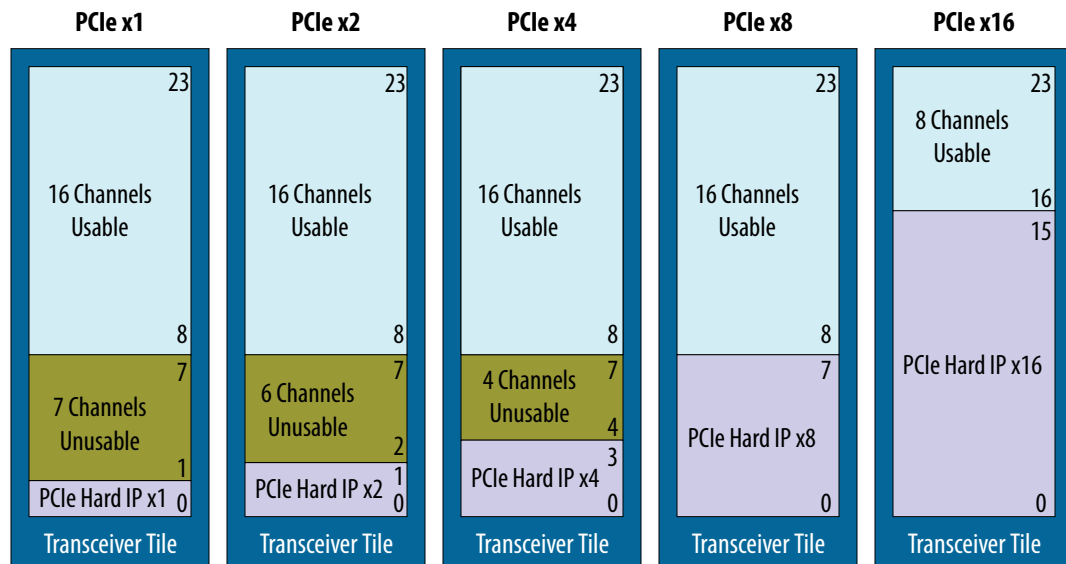
Each L- or H-Tile transceiver tile contains one PCIe Hard IP block. The following table and figure show the possible PCIe Hard IP channel configurations, the number of unusable channels, and the number of channels available for other protocols. For example, a PCIe x4 variant uses 4 channels and 4 additional channels are unusable.

Table 6. Unusable Channels

PCIe Hard IP Configuration	Number of Unusable Channels	Usable Channels
PCIe x1	7	16
PCIe x2	6	16
PCIe x4	4	16
PCIe x8	0	16
PCIe x16	0	8

Note: The PCIe Hard IP uses at least the bottom eight Embedded Multi-Die Interconnect Bridge (EMIB) channels, no matter how many PCIe lanes are enabled. Thus, these EMIB channels become unavailable for other protocols.

Figure 11. PCIe Hard IP Channel Configurations Per Transceiver Tile



The table below maps all transceiver channels to PCIe Hard IP channels in available tiles.

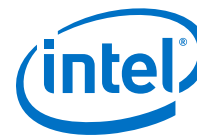


Table 7. PCIe Hard IP channel mapping across all tiles

Tile Channel Sequence	PCIe Hard IP Channel	Index within I/O Bank	Bottom Left Tile Bank Number	Top Left Tile Bank Number	Bottom Right Tile Bank Number	Top Right Tile Bank Number
23	N/A	5	1F	1N	4F	4N
22	N/A	4	1F	1N	4F	4N
21	N/A	3	1F	1N	4F	4N
20	N/A	2	1F	1N	4F	4N
19	N/A	1	1F	1N	4F	4N
18	N/A	0	1F	1N	4F	4N
17	N/A	5	1E	1M	4E	4M
16	N/A	4	1E	1M	4E	4M
15	15	3	1E	1M	4E	4M
14	14	2	1E	1M	4E	4M
13	13	1	1E	1M	4E	4M
12	12	0	1E	1M	4E	4M
11	11	5	1D	1L	4D	4L
10	10	4	1D	1L	4D	4L
9	9	3	1D	1L	4D	4L
8	8	2	1D	1L	4D	4L
7	7	1	1D	1L	4D	4L
6	6	0	1D	1L	4D	4L
5	5	5	1C	1K	4C	4K
4	4	4	1C	1K	4C	4K
3	3	3	1C	1K	4C	4K
2	2	2	1C	1K	4C	4K
1	1	1	1C	1K	4C	4K
0	0	0	1C	1K	4C	4K

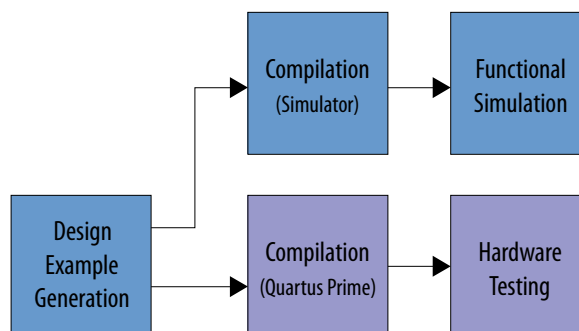
PCIe Soft IP Channel Usage

PCI Express soft IP PIPE-PHY cores available from third-party vendors are not subject to the channel usage restrictions described above. Refer to [Intel FPGA > Products > Intellectual Property](#) for more information about soft IP cores for PCI Express.

2. Quick Start Guide

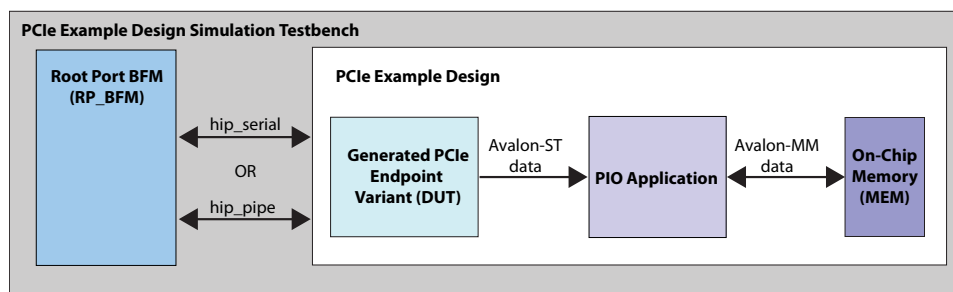
Using Intel Quartus Prime software, you can generate a programmed I/O (PIO) design example for the Avalon-ST Intel Stratix 10-GX Hard IP for PCI Express IP core. The generated design example reflects the parameters that you specify. The PIO example transfers data from a host processor to a target device. It is appropriate for low-bandwidth applications. This design example automatically creates the files necessary to simulate and compile in the Intel Quartus Prime software. You can download the compiled design to the Intel Stratix 10-GX FPGA Development Board. To download to custom hardware, update the Intel Quartus Prime Settings File (.qsf) with the correct pin assignments .

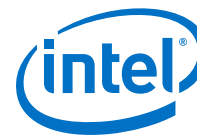
Figure 12. Development Steps for the Design Example



2.1. Design Components

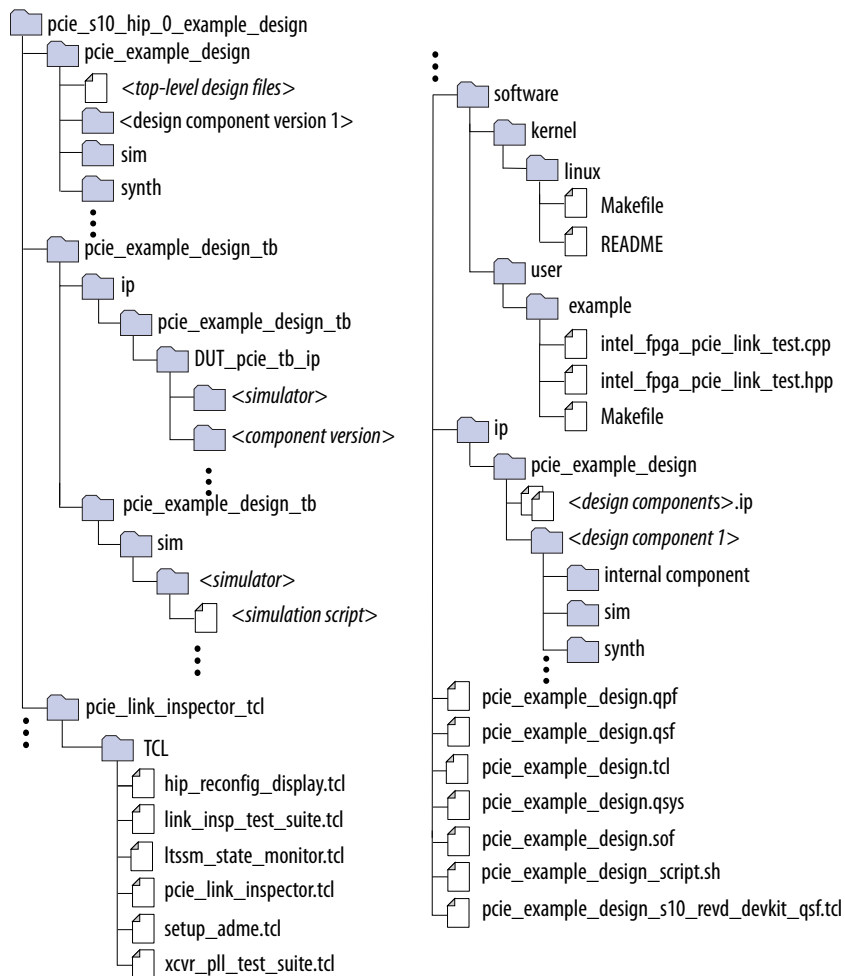
Figure 13. Block Diagram for the Platform Designer PIO Design Example Simulation Testbench





2.2. Directory Structure

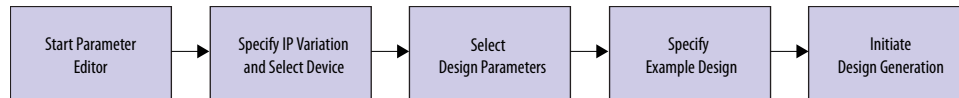
Figure 14. Directory Structure for the Generated Design Example



2.3. Generating the Design Example

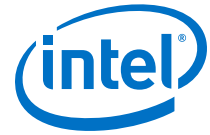
Follow these steps to generate your design:

Figure 15. Procedure



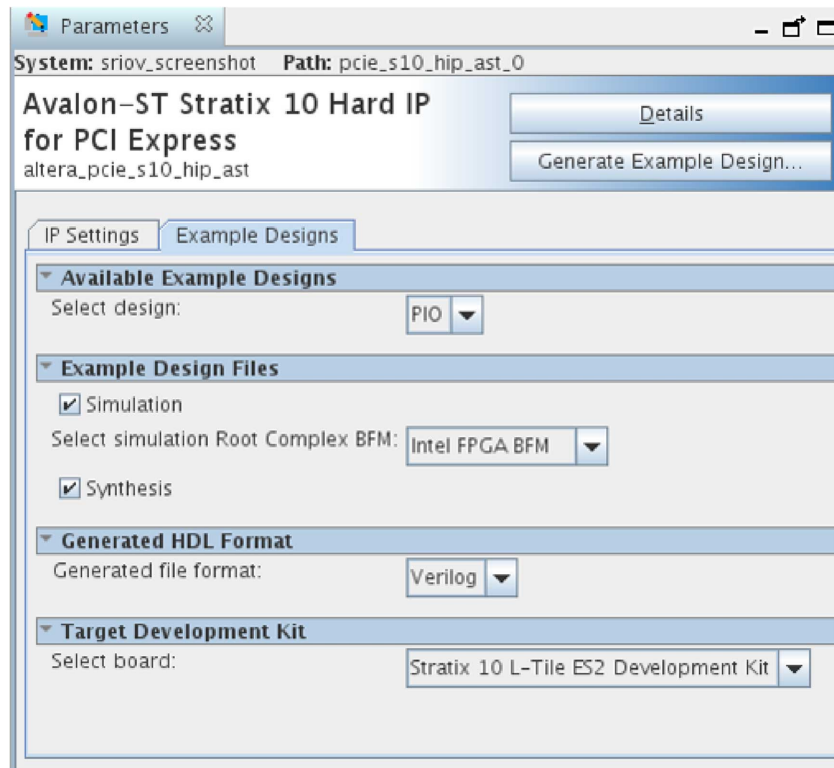
1. In the Intel Quartus Prime Pro Edition software, create a new project (**File ► New Project Wizard**).
2. Specify the **Directory**, **Name**, and **Top-Level Entity**.
3. For **Project Type**, accept the default value, **Empty project**. Click **Next**.
4. For **Add Files** click **Next**.
5. For **Family, Device & Board Settings** under **Family**, select **Intel Stratix 10 (GX/SX/MX/TX)** and the **Target Device** for your design.
6. Click **Finish**.
7. In the IP Catalog locate and add the **Avalon-ST Intel Stratix 10 Hard IP for PCI Express**.
8. In the **New IP Variant** dialog box, specify a name for your IP. Click **Create**.
9. On the **IP Settings** tabs, specify the parameters for your IP variation.
10. On the **Example Designs** tab, make the following selections:
 - a. For **Available Example Designs**, select **PIO**.
 - b. For **Example Design Files**, turn on the **Simulation** and **Synthesis** options. If you do not need these simulation or synthesis files, leaving the corresponding option(s) turned off significantly reduces the example design generation time.
 - c. If you have selected a x16 configuration, for **Select simulation Root Complex BFM**, choose the appropriate BFM:
 - **Intel FPGA BFM**: for all configurations up to Gen3 x8. This bus functional model (BFM) supports x16 configurations by downtraining to x8.
 - **Third-party BFM**: for x16 configurations if you want to simulate all 16 lanes using a third-party BFM. Refer to [AN-811: Using the Avery BFM for PCI Express Gen3x16 Simulation on Intel Stratix 10 Devices](#) for information about simulating with the Avery BFM.
 - d. For **Generated HDL Format**, only Verilog is available in the current release.
 - e. For **Target Development Kit**, select the appropriate option.

Note: If you select **None**, the generated design example targets the device you specified in Step 5 above. If you intend to test the design in hardware, make the appropriate pin assignments in the .qsf file. You can also use the pin planner tool to make pin assignments.
11. Select **Generate Example Design** to create a design example that you can simulate and download to hardware. If you select one of the Intel Stratix 10 development boards, the device on that board overwrites the device previously selected in the Intel Quartus Prime project if the devices are different. When the prompt asks you to specify the directory for your example design, you can accept



the default directory, `<example_design>/pcie_s10_hip_avmm_bridge_0_example_design`, or choose another directory.

Figure 16. Example Design Tab



When you generate an Intel Stratix 10 example design, a file called `recommended_pinassignments_s10.txt` is created in the directory `pcie_s10_hip_avmm_bridge_0_example_design`.⁽¹⁾

12. Click **Finish**. You may save your `.ip` file when prompted, but it is not required to be able to use the example design.
13. The prompt, **Recent changes have not been generated. Generate now?**, allows you to create files for simulation and synthesis of the IP core variation that you specified in Step 9 above. Click **No** if you only want to work with the design example you have generated.
14. Close the dummy project.

⁽¹⁾ This file contains the recommended pin assignments for all the pins in the example design. If you select a development kit option in the pull-down menu for **Target Development Kit**, the pin assignments in the `recommended_pinassignments_s10.txt` file match those that are in the `.qsf` file in the same directory. If you chose **NONE** in the pull-down menu, the `.qsf` file does not contain any pin assignment. In this case, you can copy the pin assignments in the `recommended_pinassignments_s10.txt` file to the `.qsf` file. You can always change any pin assignment in the `.qsf` file to satisfy your design or board requirements.

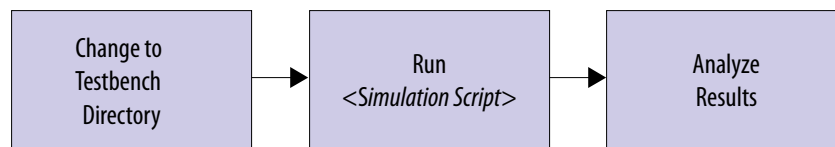
15. Open the example design project.
16. Compile the example design project to generate the .sof file for the complete example design. This file is what you download to a board to perform hardware verification.
17. Close your example design project.

Related Information

[AN-811: Using the Avery BFM for PCI Express Gen3x16 Simulation on Intel Stratix 10 Devices](#)

2.4. Simulating the Design Example

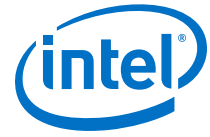
Figure 17. Procedure



1. Change to the testbench simulation directory, pcie_example_design_tb.
2. Run the simulation script for the simulator of your choice. Refer to the table below.
3. Analyze the results.

Table 8. Steps to Run Simulation

Simulator	Working Directory	Instructions
ModelSim*	<example_design>/ pcie_example_design_tb/ pcie_example_design_tb/sim/mentor/	1. Invoke vsim 2. do msim_setup.tcl 3. ld_debug 4. run -all 5. A successful simulation ends with the following message, "Simulation stopped due to successful completion!"
VCS*	<example_design>/ pcie_example_design_tb/ pcie_example_design_tb/sim/ synopsys/vcs	1. sh vcs_setup.sh USER_DEFINED_COMPILE_OPTIONS="" USER_DEFINED_ELAB_OPTIONS="-x1rm\ uniq_prior_final" 2. A successful simulation ends with the following message, "Simulation stopped due to successful completion!"
NCSim*	<example_design>/ pcie_example_design_tb/ pcie_example_design_tb/sim/cadence	1. sh ncsim_setup.sh USER_DEFINED_SIM_OPTIONS="" 2. A successful simulation ends with the following message, "Simulation stopped due to successful completion!"
Xcelium* Parallel Simulator	<example_design>/ pcie_example_design_tb/ pcie_example_design_tb/sim/xcelium	1. sh xcelium_setup.sh USER_DEFINED_SIM_OPTIONS="" USER_DEFINED_ELAB_OPTIONS = "-NOWARN\ CSINF1" 2. A successful simulation ends with the following message, "Simulation stopped due to successful completion!"



This testbench simulates up to x8 variants. It supports x16 variants by down-training to x8. To simulate all lanes of a x16 variant, you can create a simulation model using the Platform Designer to use in an Avery testbench. For more information refer to *AN-811: Using the Avery BFM for PCI Express Gen3x16 Simulation on Intel Stratix 10 Devices*.

The simulation reports, "Simulation stopped due to successful completion" if no errors occur.

Related Information

[AN-811: Using the Avery BFM for PCI Express Gen3x16 Simulation on Intel Stratix 10 Devices](#)

2.5. Compiling the Design Example and Programming the Device

1. Navigate to <project_dir>/pcie_s10_hip_avmm_bridge_0_example_design/ and open pcie_example_design.qpf.
2. On the Processing menu, select **Start Compilation**.
3. After successfully compiling your design, program the targeted device with the Programmer.

2.6. Installing the Linux Kernel Driver

Before you can test the design example in hardware, you must install the Linux kernel driver. You can use this driver to perform the following tests:

- A PCIe link test that performs 100 writes and reads
- Memory space DWORD⁽²⁾ reads and writes
- Configuration Space DWORD reads and writes

In addition, you can use the driver to change the value of the following parameters:

- The BAR being used
- The selects device by specifying the bus, device and function (BDF) numbers for the required device

The driver also allows you to enable SR-IOV for H-Tile devices.

Complete the following steps to install the kernel driver:

1. Navigate to ./software/kernel/linux under the example design generation directory.
2. Change the permissions on the install, load, and unload files:

```
$ chmod 777 install load unload
```
3. Install the driver:

⁽²⁾ Throughout this user guide, the terms word, DWORD and QWORD have the same meaning that they have in the PCI Express Base Specification. A word is 16 bits, a DWORD is 32 bits, and a QWORD is 64 bits.

- ```
$ sudo ./install
```
- Verify the driver installation:  

```
$ lsmod | grep intel_fpga_pcie_drv
```

Expected result:

```
intel_fpga_pcie_drv 17792 0
```
  - Verify that Linux recognizes the PCIe design example:  

```
$ lspci -d 1172:0000 -v | grep intel_fpga_pcie_drv
```

*Note:* If you have changed the Vendor ID, substitute the new Vendor ID for Intel's Vendor ID in this command.

Expected result:

```
Kernel driver in use: intel_fpga_pcie_drv
```

## 2.7. Running the Design Example Application

- Navigate to `./software/user/example` under the design example directory.
- Compile the design example application:  

```
$ make
```
- Run the test:  

```
$./intel_fpga_pcie_link_test
```

You can run the Intel FPGA IP PCIe link test in manual or automatic mode.

  - In automatic mode, the application automatically selects the device. The test selects the Intel Stratix 10 PCIe device with the lowest BDF by matching the Vendor ID. The test also selects the lowest available BAR.
  - In manual mode, the test queries you for the bus, device, and function number and BAR.

For the Intel Stratix 10 GX Development Kit, you can determine the BDF by typing the following command:

```
$ lspci -d 1172
```
- Here are sample transcripts for automatic and manual modes:

```
Intel FPGA PCIe Link Test - Automatic Mode
Version 2.0
0: Automatically select a device
1: Manually select a device

>0
Opened a handle to BAR 0 of a device with BDF 0x100

0: Link test - 100 writes and reads
1: Write memory space
2: Read memory space
3: Write configuration space
4: Read configuration space
5: Change BAR
6: Change device
7: Enable SR-IOV
8: Do a link test for every enabled virtual function
 belonging to the current device
9: Perform DMA
```



```
10: Quit program

> 0
Doing 100 writes and 100 reads . .
Number of write errors: 0
Number of read errors: 0
Number of DWORD mismatches: 0

Intel FPGA PCIe Link Test - Manual Mode
Version 1.0
0: Automatically select a device
1: Manually select a device

> 1
Enter bus number:
> 1
Enter device number:
> 0
Enter function number:
> 0
BDF is 0x100
Enter BAR number (-1 for none):
> 4
Opened a handle to BAR 4 of a device with BDF 0x100
```

## Related Information

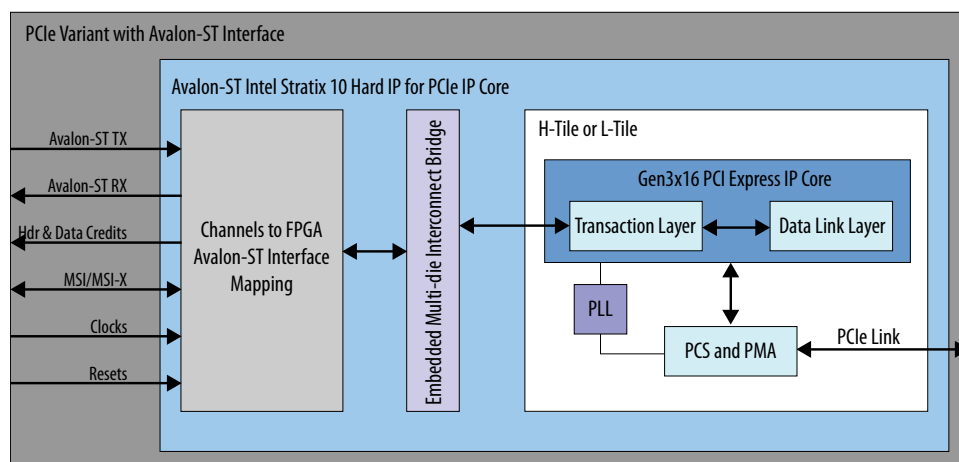
### [PCIe Link Inspector Overview](#)

Use the PCIe Link Inspector to monitor the link at the Physical, Data Link and Transaction Layers.

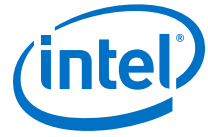
### 3. Interface Overview

The *PCI Express Base Specification 3.0* defines a packet interface for communication between a Root Port and an Endpoint. When you select the Avalon-ST interface, Transaction Layer Packets (TLP) transfer data between the Root Port and an Endpoint using the Avalon-ST TX and RX interfaces. The interfaces are named from the point-of-view of the user logic.

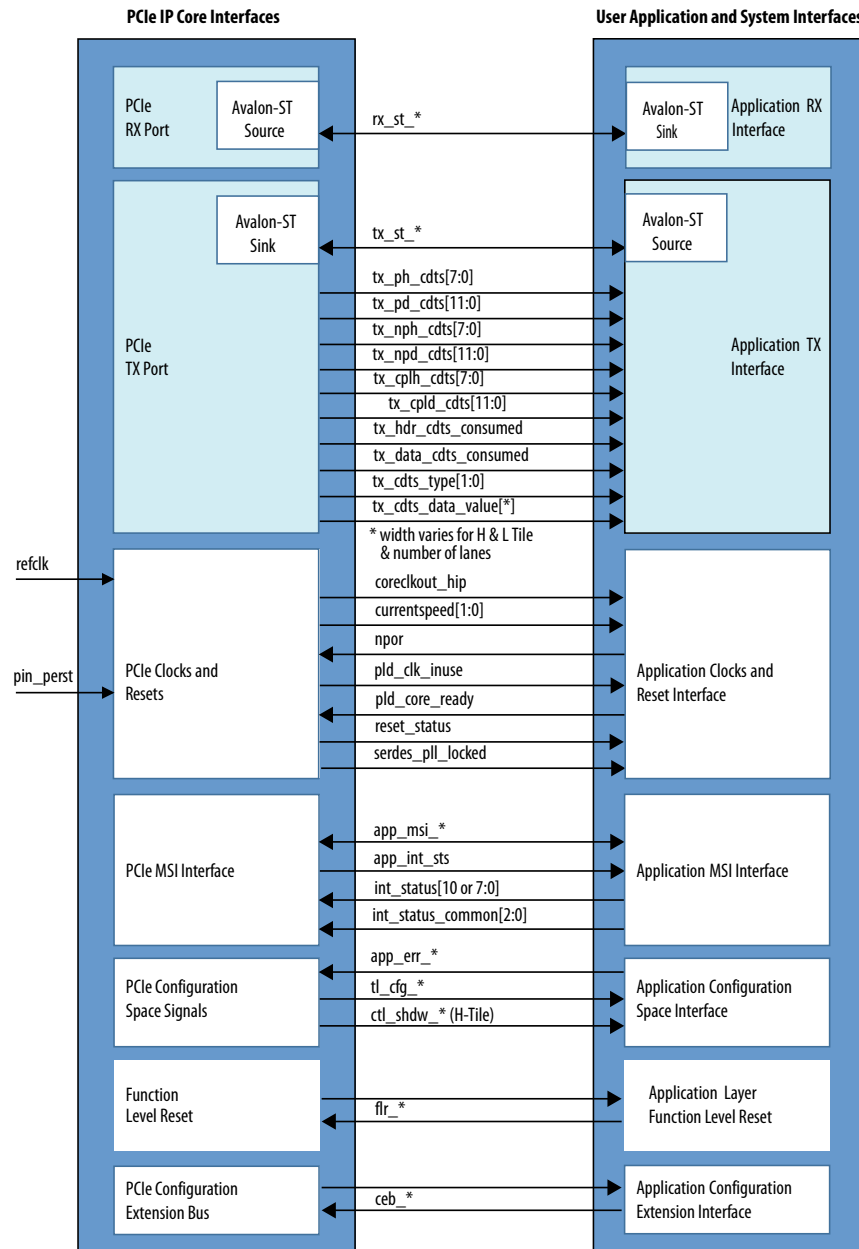
**Figure 18. Intel Stratix 10 Top-Level Interfaces**



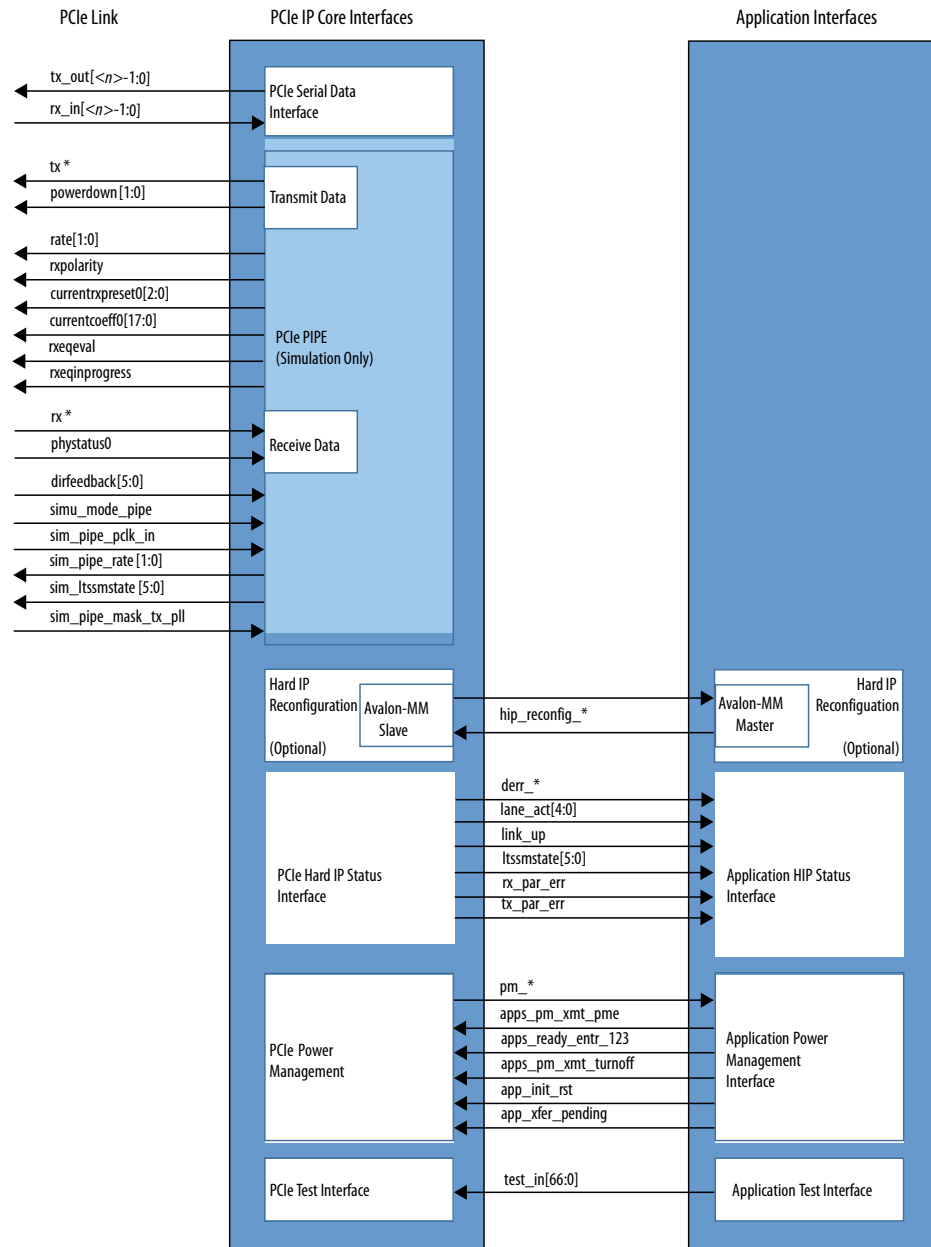
The following figures show the PCIe hard IP Core top-level interfaces and the connections to the Application Layer and system.



**Figure 19. Connections: User Application to Avalon-ST Intel Stratix 10 Hard IP for PCIe IP Core**



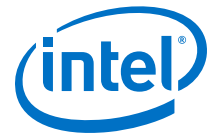
**Figure 20. Connections: PCIe Link and Reconfiguration, Power and Test Interfaces**



The following sections introduce these interfaces. Refer to the *Interfaces* section in the *Block Description* chapter for detailed descriptions and timing diagrams.

#### Related Information

- [Avalon-ST 256-Bit RX Interface](#) on page 59
- [Interfaces](#) on page 56
- [PCI Express Base Specification 3.0](#)



### 3.1. Avalon-ST RX Interface

The Transaction Layer transfers TLPs to the Application on this interface. The Application must assert `rx_st_ready` before transfers can begin.

This interface is not strictly Avalon-ST compliant, and does not have a well-defined `ready_latency`. For all variants other than the Gen3 x16 variant, the latency between the assertion or de-assertion of `rx_st_ready` and the corresponding de-assertion or assertion of `rx_st_valid` can be up to 17 cycles. Once `rx_st_ready` deasserts, `rx_st_valid` deasserts within 17 cycles. Once `rx_st_ready` reasserts, `rx_st_valid` resumes data transfer within 17 cycles. To achieve the best performance the Application must include a receive buffer large enough to avoid the deassertion of `rx_st_ready`. Refer to *Avalon-ST RX Interface* for more information.

For the Gen3 x16 variant, once `rx_st_ready` deasserts, `rx_st_valid` deasserts within 18 cycles. Once `rx_st_ready` reasserts, `rx_st_valid` resumes data transfer within 18 cycles.

**Note:** Throughout this user guide the terms Application and Application Layer refer to your logic that interfaces with the PCIe Transaction Layer.

### 3.2. Avalon-ST TX Interface

The Application transmits TLPs to the Transaction Layer of the IP core on this interface. The Transaction Layer must assert `tx_st_ready` before transmission begins. Transmission of a packet must be uninterrupted when `tx_st_ready` is asserted. The `readyLatency` of this interface is three `coreclkout_hip` cycles. For more detailed information about the Avalon-ST interface, refer to *Avalon-ST TX Interface*. The packet layout is shown in detail in the *Block Description* chapter.

### 3.3. TX Credit Interface

The Transaction Layer TX interface transmits TLPs in the same order as they were received from the Application. To optimize performance, the Application can perform credit-based checking before submitting requests for transmission, allowing the Application to reorder packets to improve throughput. Application reordering is optional. The Transaction Layer always performs a credit check before transmitting any TLP.

### 3.4. TX and RX Serial Data

This differential, serial interface is the physical link between a Root Port and an Endpoint. The PCIe IP Core supports 1, 2, 4, 8, or 16 lanes. Each lane includes a TX and RX differential pair. Data is striped across all available lanes.

### 3.5. Clocks

The *PCI Express Card Electromechanical Specification Revision 2.0* defines the input reference clock. It is a 100 MHz  $\pm 300$  ppm. The motherboard drives this clock to the PCIe card. The PCIe card is not required to use the reference clock received from the connector. However, the PCIe card must provide a 100 MHz  $\pm 300$  ppm to the `refclk` input pin.

**Table 9. Application Clock Frequency**

| Data Rate                    | Interface Width | coreclkout_hip Frequency |
|------------------------------|-----------------|--------------------------|
| Gen1 x1, x2, x4, x8, and x16 | 256 bits        | 125 MHz                  |
| Gen2 x1, x2, x4, and x8      | 256 bits        | 125 MHz                  |
| Gen2 x16                     | 256 bits        | 250 MHz                  |
| Gen3 x1, x2, and x4          | 256 bits        | 125 MHz                  |
| Gen3 x8                      | 256 bits        | 250 MHz                  |
| Gen3 x16                     | 512 bits        | 250 MHz                  |

### 3.6. Function-Level Reset (FLR) Interface

Use the function-level reset interface to reset individual SR-IOV Functions.

### 3.7. Control Shadow Interface for SR-IOV

The control shadow interface provides access to the current settings for some of the VF Control Register fields in the PCI and PCI Express Configuration Spaces located in the SR-IOV Bridge.

Use the interface for the following reasons:

- To monitor specific VF registers using the `ctl_shdw_update` output and the associated output signals.
- To monitor all VF registers using the `ctl_shdw_req_all` input to request a full scan of the register fields for all active VFs.

### 3.8. Configuration Extension Bus Interface

Use the Configuration Extension Bus to add capability structures to the IP core internal Configuration Spaces. Configuration TLPs with a destination register byte address of 0xC00 and higher are routed to the Configuration Extension Bus interface.

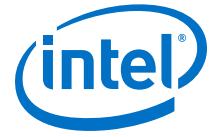
**Note:** The Configuration Extension Bus interface is not available if the parameter **Enable SR-IOV Support** under the **Multifunction and SR-IOV System Settings** tab is set to **On**.

### 3.9. Hard IP Reconfiguration Interface

Use this bus to dynamically modify the values of configuration registers that are read-only at run time.

The PCI Express link cannot be reset after changing the values of the read-only configuration registers of the Hard IP because the registers will be restored to their original values after reset.

The Hard IP reconfiguration interface is not accessible when the IP is in reset (i.e., when `reset_status = 1`).



## 3.10. Interrupt Interfaces

The PCIe IP core support Message Signaled Interrupts (MSI), MSI-X interrupts, and Legacy interrupts. MSI and legacy interrupts are *mutually exclusive*.

MSI uses the TLP single DWORD memory writes to implement interrupts. This interrupt mechanism conserves pins because it does not use separate wires for interrupts. In addition, the single DWORD provides flexibility for the data presented in the interrupt message. The MSI Capability structure is stored in the Configuration Space and is programmed using Configuration Space accesses.

The Application generates MSI-X messages which are single DWORD memory writes. The MSI-X Capability structure points to an MSI-X table structure and MSI-X PBA structure which are stored in memory. This scheme is different than the MSI capability structure, which contains all the control and status information for the interrupts.

Enable Legacy interrupts by programming the Interrupt Disable bit (bit[10]) of the Configuration Space Command to 1'b0. When legacy interrupts are enabled, the IP core emulates INTx interrupts using virtual wires. The `app_int_sts` port controls legacy interrupt generation.

## 3.11. Power Management Interface

Software uses the power management interface to control the power management state machine. The power management output signals indicate the current power state. The IP core supports the two mandatory power states: D0 full power and D3 preparation for loss of power. It does not support the optional D1 and D2 low-power states.

## 3.12. Reset

This interface indicates when the clocks are stable and FPGA configuration is complete. The PCIe IP core receives two reset inputs:

- `pin_perst` is the active low reset driven from the PCIe motherboard. Logic on the motherboard autonomously generates this fundamental reset signal.
- `npwr` is an active low reset signal. The Application drives this reset signal.

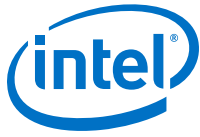
The PCIe IP core reset logic requires a free-running clock input. This free-running clock becomes stable after the secure device manager (SDM) block asserts `iocsrrdy_dly` indicating that the I/O Control and Status registers programming is complete.

## 3.13. Transaction Layer Configuration Interface

This interface provides time-domain multiplexed (TD) access to a subset of the values stored in the Configuration Space registers.

## 3.14. PLL Reconfiguration Interface

The PLL reconfiguration interface is an Avalon-MM slave interface. Use this bus to dynamically modify the value of PLL registers that are read-only at run time.



This interface is available when you turn on **Enable Transceiver dynamic reconfiguration** on the **Configuration, Debug and Extension Options** tab using the parameter editor.

To ensure proper system operation, reset or repeat device enumeration of the PCIe link after changing the value of read-only PLL registers.

### **3.15. PIPE Interface (Simulation Only)**

This is a 32-bit parallel interface between the PCIe IP Core and PHY. It carries the TLP data before it is serialized. It is available for simulation only and provides more visibility for debugging.

**Note:** You cannot change the width of the PIPE interface.

## 4. Parameters

This chapter provides a reference for all the parameters of the Intel Stratix 10 Hard IP for PCI Express IP core.

**Table 10. Design Environment Parameter**

Starting in Intel Quartus Prime 18.0, there is a new parameter **Design Environment** in the parameters editor window.

| Parameter                 | Value                    | Description                                                                                                                                                                                                                                                                                                                            |
|---------------------------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Design Environment</b> | <b>Standalone System</b> | <p>Identifies the environment that the IP is in.</p> <ul style="list-style-type: none"> <li>The <b>Standalone</b> environment refers to the IP being in a standalone state where all its interfaces are exported.</li> <li>The <b>System</b> environment refers to the IP being instantiated in a Platform Designer system.</li> </ul> |

**Table 11. System Settings**

| Parameter                         | Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Application Interface Type</b> | <b>Avalon-ST</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Selects the interface to the Application Layer.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Hard IP Mode</b>               | <b>Gen3x16, 512-bit interface, 250 MHz</b><br><b>Gen3x8, 256-bit interface, 250 MHz</b><br><b>Gen3x4, 256-bit interface, 125 MHz</b><br><b>Gen3x2, 256-bit interface, 125 MHz</b><br><b>Gen3x1, 256-bit interface, 125 MHz</b><br><b>Gen2x16, 256-bit interface, 250 MHz</b><br><b>Gen2x8, 256-bit interface, 125 MHz</b><br><b>Gen2x4, 256-bit interface, 125 MHz</b><br><b>Gen2x2, 256-bit interface, 125 MHz</b><br><b>Gen2x1, 256-bit interface, 125 MHz</b><br><b>Gen1x16, 256-bit interface, 125 MHz</b><br><b>Gen1x8, 256-bit interface, 125 MHz</b><br><b>Gen1x4, 256-bit interface, 125 MHz</b><br><b>Gen1x2, 256-bit interface, 125 MHz</b><br><b>Gen1x1, 256-bit interface, 125 MHz</b> | <p>Selects the following elements:</p> <ul style="list-style-type: none"> <li>The lane data rate. Gen1, Gen2, and Gen3 are supported</li> <li>The Application Layer interface frequency</li> </ul> <p>The width of the data interface between the hard IP Transaction Layer and the Application Layer implemented in the FPGA fabric.</p> <p><i>Note:</i> If the <b>Mode</b> selected is not available for the configuration chosen, an error message displays in the <b>Message</b> pane.</p> |
| <b>Port type</b>                  | <b>Native Endpoint Root Port</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | <p>Specifies the port type.</p> <p>The Endpoint stores parameters in the Type 0 Configuration Space. The Root Port stores parameters in the Type 1 Configuration Space.</p> <p><i>Note:</i> Although the <b>Root Port</b> option is available, it is not yet fully tested. Intel recommends that you not select it in the current release.</p>                                                                                                                                                 |

## 4.1. Stratix 10 Avalon-ST Settings

**Table 12. System Settings for PCI Express**

| Parameter                                              | Value         | Description                                                                                                                                                                                                                                                       |
|--------------------------------------------------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Enable Avalon-ST reset output port</b>              | <b>On/Off</b> | When <b>On</b> , the generated reset output port <code>clr_st</code> has the same functionality as the <code>hip_ready_n</code> port included in the Hard IP Reset interface. This option is available for backwards compatibility with Arria 10 devices.         |
| <b>Enable byte parity ports on Avalon-ST interface</b> | <b>On/Off</b> | When <b>On</b> , the RX and TX datapaths are parity protected. Parity is odd. The Application Layer must provide valid byte parity in the Avalon-ST TX direction.<br>This parameter is only available for the Avalon-ST Intel Stratix 10 Hard IP for PCI Express. |

## 4.2. Multifunction and SR-IOV System Settings

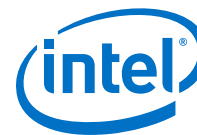
**Table 13. Multifunction and SR-IOV System Settings**

| Parameter                                                      | Value            | Description                                                                                                                                                           |
|----------------------------------------------------------------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Total Physical Functions (PFs) :</b>                        | 1 - 4            | Supports 1 - 4 PFs (in H-Tile devices).                                                                                                                               |
| <b>Enable SR-IOV Support</b>                                   | <b>On/Off</b>    | When <b>On</b> , the variant supports multiple VFs. When <b>Off</b> , supports PFs only. SR-IOV is only available in H-Tile devices.                                  |
| <b>Total Virtual Functions Assigned to Physical Functions:</b> | 1 - 2048         | Total number of VFs assigned to a PF. The sum of VFs assigned to PF0, PF1, PF2, and PF3 cannot exceed the 2048 VFs total.                                             |
| <b>System Supported Page Size:</b>                             | <b>4 KB-4 MB</b> | Specifies the page sizes supported. Sets the Supported Page Sizes register of the SR-IOV Capability structure.<br>Intel recommends that you accept the default value. |

**Note:** Throughout this document, the terminology "Physical Function" and "PF" refer to what is defined as "Physical Function" in the *PCI Express SR-IOV Specification* when SR-IOV is enabled, and to what is defined simply as "Function" in the *PCI Express Base Specification* when SR-IOV is not enabled.

**Note:** In a multifunction application, if two or more functions initiate Memory Read requests with identical tag values, the Intel Stratix 10 Avalon-ST Hard IP for PCIe IP Core only stores the latest Completion that it receives. Therefore, even though Completions for Memory Reads from different functions have different requester ID fields, if they share the same tag, only the last Completion with that tag value to arrive is preserved. To avoid lost Completion packets, the user application must not use the same tag for Memory Read transactions across multiple functions.

## 4.3. Base Address Registers



**Table 14. BAR Registers**

| Parameter     | Value                                                                    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type          | Disabled<br>64-bit prefetchable memory<br>32-bit non-prefetchable memory | <p>If you select 64-bit prefetchable memory, 2 contiguous BARs are combined to form a 64-bit prefetchable BAR; you must set the higher numbered BAR to <b>Disabled</b>. A non-prefetchable 64-bit BAR is not supported because in a typical system, the maximum non-prefetchable memory window is 32 bits.</p> <p>Defining memory as prefetchable allows contiguous data to be fetched ahead. Prefetching memory is advantageous when the requestor may require more data from the same region than was originally requested. If you specify that a memory is prefetchable, it must have the following 2 attributes:</p> <ul style="list-style-type: none"> <li>Reads do not have side effects such as changing the value of the data read</li> <li>Write merging is allowed</li> </ul> <p><i>Note:</i> BAR0 is not available if the internal descriptor controller is enabled.</p> |
| Size          | 16 Bytes–8 EBytes                                                        | Specifies the size of the address space accessible to the BAR.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Expansion ROM | Disabled<br>4 KBytes - 16 MBytes                                         | Specifies the size of the option ROM.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

## 4.4. Device Identification Registers

The following table lists the default values of the read-only registers in the PCI\* Configuration Header Space. You can use the parameter editor to set the values of these registers. At run time, you can change the values of these registers using the optional Hard IP Reconfiguration block signals.

You can specify Device ID registers for each Physical Function.

**Table 15. PCI Header Configuration Space Registers**

| Register Name       | Default Value | Description                                                                                                                                                                                                                                                                                         |
|---------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Vendor ID           | 0x00001172    | Sets the read-only value of the Vendor ID register. This parameter cannot be set to 0xFFFF per the <i>PCI Express Base Specification</i> . Address offset: 0x000.                                                                                                                                   |
| Device ID           | 0x00000000    | Sets the read-only value of the Device ID register. Address offset: 0x000.                                                                                                                                                                                                                          |
| VF 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. Address offset: 0x008.                                                                                                                                                                                                                        |
| Class code          | 0x00000000    | Sets the read-only value of the Class Code register. You must set this register to a non-zero value to ensure correct operation. Address offset: 0x008.                                                                                                                                             |
| Subsystem Vendor ID | 0x00000000    | Sets the read-only value of Subsystem Vendor ID register in the PCI Type 0 Configuration Space. This parameter cannot be set to 0xFFFF per the <i>PCI Express Base Specification</i> . This value is assigned by PCI-SIG to the device manufacturer. This value is only used in Root Port variants. |

*continued...*

| Register Name              | Default Value | Description                                                                                                                                                                 |
|----------------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                            |               | Address offset: 0x02C.                                                                                                                                                      |
| <b>Subsystem Device ID</b> | 0x00000000    | Sets the read-only value of the Subsystem Device ID register in the PCI Type 0 Configuration Space. This value is only used in Root Port variants.<br>Address offset: 0x02C |

## 4.5. TPH/ATS Capabilities

### TLP Processing Hints (TPH) Overview

TPH support PFs that target a TLP towards a specific processing resource such as a host processor or cache hierarchy. Steering Tags (ST) provide design-specific information about the host or cache structure.

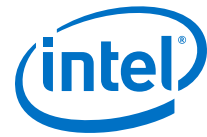
Software programs the Steering Tag values that are stored in an ST table. You can store the ST Table in the MSI-X Table or a custom location. For more information about Steering Tags, refer to *Section 6.17.2 Steering Tags* of the *PCI Express Base Specification, Rev. 3.0*. After analyzing the traffic of your system, you may be able to use TPH hints to improve latency or reduce traffic congestion.

### Address Translation Services (ATS) Overview

ATS extends the PCIe protocol to support an address translation agent (TA) that translates DMA addresses to cached addresses in the device. The translation agent can be located in or above the Root Port. Locating translated addresses in the device minimizes latency and provides a scalable, distributed caching system that improves I/O performance. The Address Translation Cache (ATC) located in the device reduces the processing load on the translation agent, enhancing system performance. For more information about ATS, refer to *Address Translation Services Revision 1.1*

**Table 16. PF0 - PF4 TPH/ATS**

| Parameter                                  | Value                                       | Description                                                                            |
|--------------------------------------------|---------------------------------------------|----------------------------------------------------------------------------------------|
| <b>Enable Address Translation Services</b> | <b>On/Off</b>                               | When <b>On</b> , the PF supports ATS.                                                  |
| <b>Enable TLP Processing Hints (TPH)</b>   | <b>On/Off</b>                               | When <b>On</b> , the PF supports TPH.                                                  |
| <b>Interrupt Mode</b>                      | <b>On/Off</b>                               | When <b>On</b> , an MSI-X interrupt vector number selects the steering tag.            |
| <b>Device Specific Mode</b>                | <b>On/Off</b>                               | When <b>On</b> , the TPH Requestor Capability structure stores the steering tag table. |
| <b>Steering Tag Table Location</b>         | <b>ST table not present<br/>MSI-X table</b> | When <b>On</b> , the MSI-X table stores the steering tag table .                       |
| <b>Steering Tag Table size</b>             | <b>0-2047</b>                               | Specifies the number of 2-byte steering table entries.                                 |

**Table 17. PF0 - PF4 VF TPH/ATS**

All VFs assigned to a PF must have the same settings.

| Parameter                           | Value                               | Description                                                                                  |
|-------------------------------------|-------------------------------------|----------------------------------------------------------------------------------------------|
| Enable Address Translation Services | On/Off                              | When <b>On</b> , the PF supports ATS.                                                        |
| Enable TLP Processing Hints (TPH)   | On/Off                              | When <b>On</b> , the PF supports TPH.                                                        |
| Interrupt Mode                      | On/Off                              | When <b>On</b> , an MSI-X interrupt vector number selects the steering tag.                  |
| Device Specific Mode                | On/Off                              | When <b>On</b> , the TPH Requestor Capability structure stores the steering tag table.       |
| Steering Tag Table Location         | ST table not present<br>MSI-X table | Selects the location of the steering tag table in <b>Device Specific Mode</b> is <b>On</b> . |
| Steering Tag Table size             | 0-2047                              | Specifies the number of 2-byte steering table entries.                                       |

#### Related Information

- [PCI Express Base Specification Revision 3.0](#)
- [Address Translation Services Revision 1.1](#)

## 4.6. PCI Express and PCI Capabilities Parameters

This group of parameters defines various capability properties of the IP core. Some of these parameters are stored in the PCI Configuration Space - PCI Compatible Configuration Space. The byte offset indicates the parameter address.

### 4.6.1. Device Capabilities

**Table 18. Device Registers**

| Parameter                       | Possible Values                                   | Default Value | Address | Description                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------|---------------------------------------------------|---------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Maximum payload sizes supported | 128 bytes<br>256 bytes<br>512 bytes<br>1024 bytes | 512 bytes     | 0x074   | Specifies the maximum payload size supported. This parameter sets the read-only value of the max payload size supported field of the Device Capabilities register.                                                                                                                                                                                                            |
| PF0 Support extended tag field  | On<br>Off                                         | Off           |         | When you turn this option <b>On</b> , the core supports 256 tags, improving the performance of high latency systems. Turning this option on turns on the Extended Tag bit in the Configuration Space Device Capabilities register.<br><br>The IP core tracks tags for Non-Posted Requests. The tracking clears when the IP core receives the last Completion TLP for a MemRd. |

## 4.6.2. Link Capabilities

Table 19. Link Capabilities

| Parameter                         | Value  | Description                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Link port number (Root Port only) | 0x01   | Sets the read-only value of the port number field in the Link Capabilities register. This parameter is for Root Ports only. It should not be changed.                                                                                                                                                                                                                                |
| Slot clock configuration          | On/Off | When you turn this option <b>On</b> , indicates that the Endpoint uses the same physical reference clock that the system provides on the connector. When <b>Off</b> , the IP core uses an independent clock regardless of the presence of a reference clock on the connector. This parameter sets the Slot Clock Configuration bit (bit 12) in the PCI Express Link Status register. |

## 4.6.3. MSI and MSI-X Capabilities

Table 20. MSI and MSI-X Capabilities

| Parameter                 | Value              | Address      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------|--------------------|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MSI messages requested    | 1, 2, 4, 8, 16, 32 | 0x050[31:16] | Specifies the number of messages the Application Layer can request. Sets the value of the Multiple Message Capable field of the Message Control register. Only PFs support MSI. When you enabled SR-IOV, PFs must use MSI-X.                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>MSI-X Capabilities</b> |                    |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Implement MSI-X           | On/Off             |              | When <b>On</b> , adds the MSI-X capability structure, with the parameters shown below. When you enable SR-IOV, you must enable MSI-X.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|                           | <b>Bit Range</b>   |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Table size                | [10:0]             | 0x068[26:16] | System software reads this field to determine the MSI-X Table size $\langle n \rangle$ , which is encoded as $\langle n-1 \rangle$ . For example, a returned value of 2047 indicates a table size of 2048. This field is read-only in the MSI-X Capability Structure. Legal range is 0–2047 ( $2^{11}$ ). VF's share a common Table Size. VF Table BIR/Offset, and PBA BIR/Offset are fixed at compile time. BAR4 accesses these tables. The table Offset field = 0x600. The PBA Offset field = 0x400 for SRIOV. You must implement an MSI-X table. If you do not intend to use MSI-X, you may program the table size to 1. |
| Table offset              | [31:0]             |              | Points to the base of the MSI-X Table. The lower 3 bits of the table BAR indicator (BIR) are set to zero by software to form a 64-bit qword-aligned offset. This field is read-only.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <i>continued...</i>       |                    |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |



| Parameter                             | Value  | Address | Description                                                                                                                                                                                                                                                                                                      |
|---------------------------------------|--------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Table BAR indicator</b>            | [2:0]  |         | Specifies which one of a function's BARs, located beginning at 0x10 in Configuration Space, is used to map the MSI-X table into memory space. This field is read-only. Legal range is 0–5.                                                                                                                       |
| <b>Pending bit array (PBA) offset</b> | [31:0] |         | Used as an offset from the address contained in one of the function's Base Address registers to point to the base of the MSI-X PBA. The lower 3 bits of the PBA BIR are set to zero by software to form a 32-bit qword-aligned offset. This field is read-only in the MSI-X Capability Structure. <sup>(3)</sup> |
| <b>Pending BAR indicator</b>          | [2:0]  |         | Specifies the function Base Address registers, located beginning at 0x10 in Configuration Space, that maps the MSI-X PBA into memory space. This field is read-only in the MSI-X Capability Structure. Legal range is 0–5.                                                                                       |

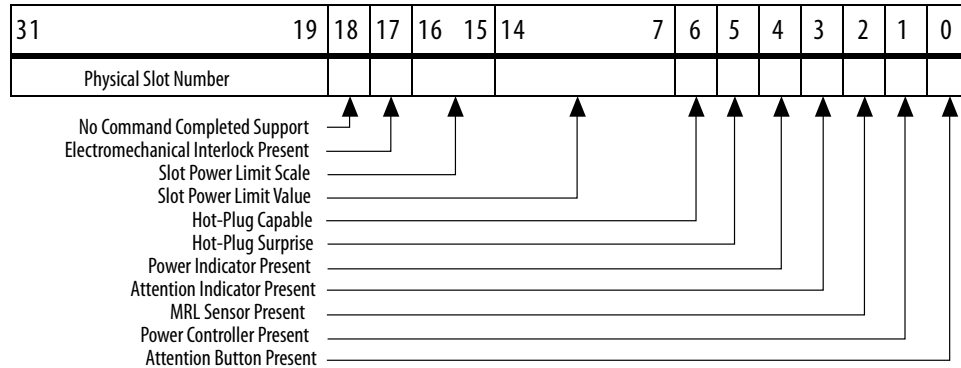
#### 4.6.4. Slot Capabilities

**Table 21. Slot Capabilities**

| Parameter                | Value         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Use Slot register</b> | <b>On/Off</b> | This parameter is only supported in Root Port mode. The slot capability is required for Root Ports if a slot is implemented on the port. Slot status is recorded in the PCI Express Capabilities register.<br>Defines the characteristics of the slot. You turn on this option by selecting <b>Enable slot capability</b> . Refer to the figure below for bit definitions.                                                                                                                |
| <b>Slot power scale</b>  | 0–3           | Specifies the scale used for the <b>Slot power limit</b> . The following coefficients are defined: <ul style="list-style-type: none"> <li>0 = 1.0x</li> <li>1 = 0.1x</li> <li>2 = 0.01x</li> <li>3 = 0.001x</li> </ul> The default value prior to hardware and firmware initialization is b'00. Writes to this register also cause the port to send the Set_Slot_Power_Limit Message.<br>Refer to Section 6.9 of the <i>PCI Express Base Specification Revision</i> for more information. |
| <b>Slot power limit</b>  | 0–255         | In combination with the <b>Slot power scale value</b> , specifies the upper limit in watts on power supplied by the slot. Refer to Section 7.8.9 of the <i>PCI Express Base Specification</i> for more information.                                                                                                                                                                                                                                                                       |
| <b>Slot number</b>       | 0–8191        | Specifies the slot number.                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

<sup>(3)</sup> Throughout this user guide, the terms word, DWORD and qword have the same meaning that they have in the *PCI Express Base Specification*. A word is 16 bits, a DWORD is 32 bits, and a qword is 64 bits.

Figure 21. Slot Capability



## 4.6.5. Power Management

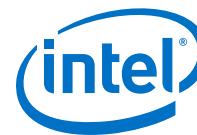
Table 22. Power Management Parameters

| Parameter                       | Value                                                                                                                                                                                                        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Endpoint L0s acceptable latency | <b>Maximum of 64 ns</b><br><b>Maximum of 128 ns</b><br><b>Maximum of 256 ns</b><br><b>Maximum of 512 ns</b><br><b>Maximum of 1 us</b><br><b>Maximum of 2 us</b><br><b>Maximum of 4 us</b><br><b>No limit</b> | <p>This design parameter specifies the maximum acceptable latency that the device can tolerate to exit the L0s state for any links between the device and the root complex. It sets the read-only value of the Endpoint L0s acceptable latency field of the Device Capabilities Register (0x084).</p> <p>This Endpoint does not support the L0s or L1 states. However, in a switched system there may be links connected to switches that have L0s and L1 enabled. This parameter is set to allow system configuration software to read the acceptable latencies for all devices in the system and the exit latencies for each link to determine which links can enable Active State Power Management (ASPM). This setting is disabled for Root Ports.</p> <p>The default value of this parameter is 64 ns. This is a safe setting for most designs.</p>               |
| Endpoint L1 acceptable latency  | <b>Maximum of 1 us</b><br><b>Maximum of 2 us</b><br><b>Maximum of 4 us</b><br><b>Maximum of 8 us</b><br><b>Maximum of 16 us</b><br><b>Maximum of 32 us</b><br><b>Maximum of 64 ns</b><br><b>No limit</b>     | <p>This value indicates 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's internal buffering. It sets the read-only value of the Endpoint L1 acceptable latency field of the Device Capabilities Register.</p> <p>This Endpoint does not support the L0s or L1 states. However, a switched system may include links connected to switches that have L0s and L1 enabled. This parameter is set to allow system configuration software to read the acceptable latencies for all devices in the system and the exit latencies for each link to determine which links can enable Active State Power Management (ASPM). This setting is disabled for Root Ports.</p> <p>The default value of this parameter is 1 <math>\mu</math>s. This is a safe setting for most designs.</p> |

## 4.6.6. Vendor Specific Extended Capability (VSEC)

Table 23. VSEC

| Parameter                                                     | Value        | Description                                                                                                                                       |
|---------------------------------------------------------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| User ID register from the Vendor Specific Extended Capability | Custom value | Sets the read-only value of the 16-bit User ID register from the Vendor Specific Extended Capability. This parameter is only valid for Endpoints. |



## 4.7. Configuration, Debug and Extension Options

**Table 24. Configuration, Debug and Extension Options**

| Parameter                                                                 | Value         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------------------------------------------------------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Enable hard IP dynamic reconfiguration of PCIe read-only registers</b> | <b>On/Off</b> | <p>When <b>On</b>, you can use the Hard IP reconfiguration bus to dynamically reconfigure Hard IP read-only registers. For more information refer to <i>Hard IP Reconfiguration Interface</i>.</p> <p>With this parameter set to <b>On</b>, the hip_reconfig_clk port is visible on the block symbol of the Avalon-MM Hard IP component. In the System Contents window, connect a clock source to this hip_reconfig_clk port. For example, you can export hip_reconfig_clk and drive it with a free-running clock on the board whose frequency is in the range of 100 to 125 MHz. Alternatively, if your design includes a clock bridge driven by such a free-running clock, the out_clk of the clock bridge can be used to drive hip_reconfig_clk.</p> |
| <b>Enable transceiver dynamic reconfiguration</b>                         | <b>On/Off</b> | <p>When <b>On</b>, provides an Avalon-MM interface that software can drive to change the values of transceiver registers.</p> <p>With this parameter set to <b>On</b>, the xcvr_reconfig_clk, reconfig_pll0_clk, and reconfig_pll1_clk ports are visible on the block symbol of the Avalon-MM Hard IP component. In the System Contents window, connect a clock source to these ports. For example, you can export these ports and drive them with a free-running clock on the board whose frequency is in the range of 100 to 125 MHz. Alternatively, if your design includes a clock bridge driven by such a free-running clock, the out_clk of the clock bridge can be used to drive these ports.</p>                                                |
| <b>Enable Native PHY, ATX PLL, and fPLL ADME for Toolkit</b>              | <b>On/Off</b> | <p>When <b>On</b>, the generated IP includes an embedded Altera Debug Master Endpoint (ADME) that connects internally to an Avalon-MM slave interface for dynamic reconfiguration. The ADME can access the transceiver reconfiguration space. It can perform certain test and debug functions via JTAG using the System Console.</p>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Enable PCIe Link Inspector</b>                                         | <b>On/Off</b> | <p>When <b>On</b>, the PCIe Link Inspector is enabled. Use this interface to monitor the PCIe link at the Physical, Data Link and Transaction layers. You can also use the Link Inspector to reconfigure some transceiver registers. You must turn on <b>Enable transceiver dynamic reconfiguration</b>, <b>Enable dynamic reconfiguration of PCIe read-only registers</b> and <b>Enable Native PHY, ATX PLL, and fPLL ADME for</b> to use this feature.</p> <p>For more information about using the PCIe Link Inspector refer to <i>Link Inspector Hardware</i> in the <i>Troubleshooting and Observing Link Status</i> appendix.</p>                                                                                                                  |

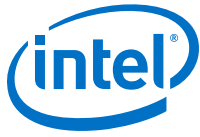
### Related Information

- [Hard IP Reconfiguration](#) on page 87
- [PCIe Link Inspector Hardware](#) on page 157

## 4.8. PHY Characteristics

**Table 25. PHY Characteristics**

| Parameter                                           | Value                | Description                                                                                                                                                                                                 |
|-----------------------------------------------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Gen2 TX de-emphasis</b>                          | <b>3.5dB<br/>6dB</b> | <p>Specifies the transmit de-emphasis for Gen2. Intel recommends the following settings:</p> <ul style="list-style-type: none"> <li>• 3.5dB: Short PCB traces</li> <li>• 6.0dB: Long PCB traces.</li> </ul> |
| <b>VCCR/VCCT supply voltage for the transceiver</b> | <b>1_1V<br/>1_0V</b> | <p>Allows you to report the voltage supplied by the board for the transceivers.</p>                                                                                                                         |



## 4.9. Example Designs

Table 26. Example Designs

| Parameter                 | Value                                                                                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------|----------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Available Example Designs | PIO                                                                                                | The <b>DMA</b> example design uses the Write Data Mover, Read Data Mover, and a custom Descriptor Controller. The <b>Simple DMA</b> example design does not use the Write Data Mover and Read Data Mover, and uses a standard Intel Descriptor Controller.<br>When you select the PIO option, the generated design includes a target application including only downstream transactions.<br>The <b>PIO</b> design example is the only option for the Avalon-ST interface. |
| Simulation                | On/Off                                                                                             | When <b>On</b> , the generated output includes a simulation model.                                                                                                                                                                                                                                                                                                                                                                                                        |
| Synthesis                 | On/Off                                                                                             | When <b>On</b> , the generated output includes a synthesis model.                                                                                                                                                                                                                                                                                                                                                                                                         |
| Generated HDL format      | Verilog/VHDL                                                                                       | Only Verilog HDL is available in the current release.                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Target Development Kit    | None<br>Intel Stratix 10 H-Tile ES1 Development Kit<br>Intel Stratix 10 L-Tile ES2 Development Kit | Select the appropriate development board.<br>If you select one of the development boards, system generation overwrites the device you selected with the device on that development board.<br><i>Note:</i> If you select <b>None</b> , system generation does not make any pin assignments. You must make the assignments in the .qsf file.                                                                                                                                |

## 5. Designing with the IP Core

### 5.1. Generation

You can use the the Intel Quartus Prime Pro Edition IP Catalog or the Platform Designer to define and generate an Avalon-ST Intel Stratix 10 Hard IP for PCI Express IP Core custom component.

For information about generating your custom IP refer to the topics listed below.

#### Related Information

- [Parameters](#) on page 33
- [Creating a System with the Platform Designer](#)
- [Generating the Design Example](#) on page 20
- [Stratix 10 Product Table](#)

### 5.2. Simulation

The Intel Quartus Prime Pro Edition software optionally generates a functional simulation model, a testbench or design example, and vendor-specific simulator setup scripts when you generate your parameterized PCI Express IP core. For Endpoints, the generation creates a Root Port BFM.

**Note:** Root Port example design generation is not supported in this release of Intel Quartus Prime Pro Edition.

The Intel Quartus Prime Pro Edition supports the following simulators.

**Table 27. Supported Simulators**

| Vendor          | Simulator                      | Version      | Platform       |
|-----------------|--------------------------------|--------------|----------------|
| Aldec           | Active-HDL *                   | 10.3         | Windows        |
| Aldec           | Riviera-PRO *                  | 2016.10      | Windows, Linux |
| Cadence         | Incisive Enterprise * (NCSim*) | 15.20        | Linux          |
| Cadence         | Xcelium* Parallel Simulator    | 17.04.014    | Linux          |
| Mentor Graphics | ModelSim PE*                   | 10.5c        | Windows        |
| Mentor Graphics | ModelSim SE*                   | 10.5c        | Windows, Linux |
| Mentor Graphics | QuestaSim*                     | 10.5c        | Windows, Linux |
| Synopsys        | VCS/VCS MX*                    | 2016,06-SP-1 | Linux          |

**Note:** The Intel testbench and Root Port BFM provide a simple method to do basic testing of the Application Layer logic that interfaces to the PCIe IP variation. This BFM allows you to create and run simple task stimuli with configurable parameters to exercise basic functionality of the example design. The testbench and Root Port BFM are not intended to be a substitute for a full verification environment. Corner cases and certain traffic profile stimuli are not covered. To ensure the best verification coverage possible, Intel suggests strongly that you obtain commercially available PCI Express verification IP and tools, or do your own extensive hardware testing or both.

Refer to the *Example Design for Avalon-ST Stratix 10 Hard IP for PCI Express* chapter to create a simple custom example design using the parameters that you specify.

#### Related Information

- [Generating the Design Example](#) on page 20
- [Introduction to Intel FPGA IP Cores, Simulating Intel FPGA IP Cores](#)
- [Simulation Quick-Start](#)

### 5.2.1. Selecting Serial or PIPE Simulation

The parameter `serial_sim_hwycl` in `<testbench_dir>/pcie_<dev>_hip_avst_0_example_design/pcie_example_design_tb/ip/pcie_example_design_tb/DUT_pcie_tb_ip/altera_pcie_<dev>_tbed_<ver>/sim/altpcie_s10_tbed_hwycl.v` determines the simulation mode. When 1'b1, the simulation is serial. When 1'b0, the simulation runs in the 32-bit parallel PIPE mode.

### 5.3. IP Core Generation Output (Intel Quartus Prime Pro Edition)

The Intel Quartus Prime software generates the following output file structure for individual IP cores that are not part of a Platform Designer system.

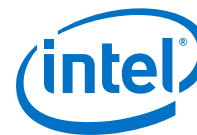


Figure 22. Individual IP Core Generation Output (Intel Quartus Prime Pro Edition)

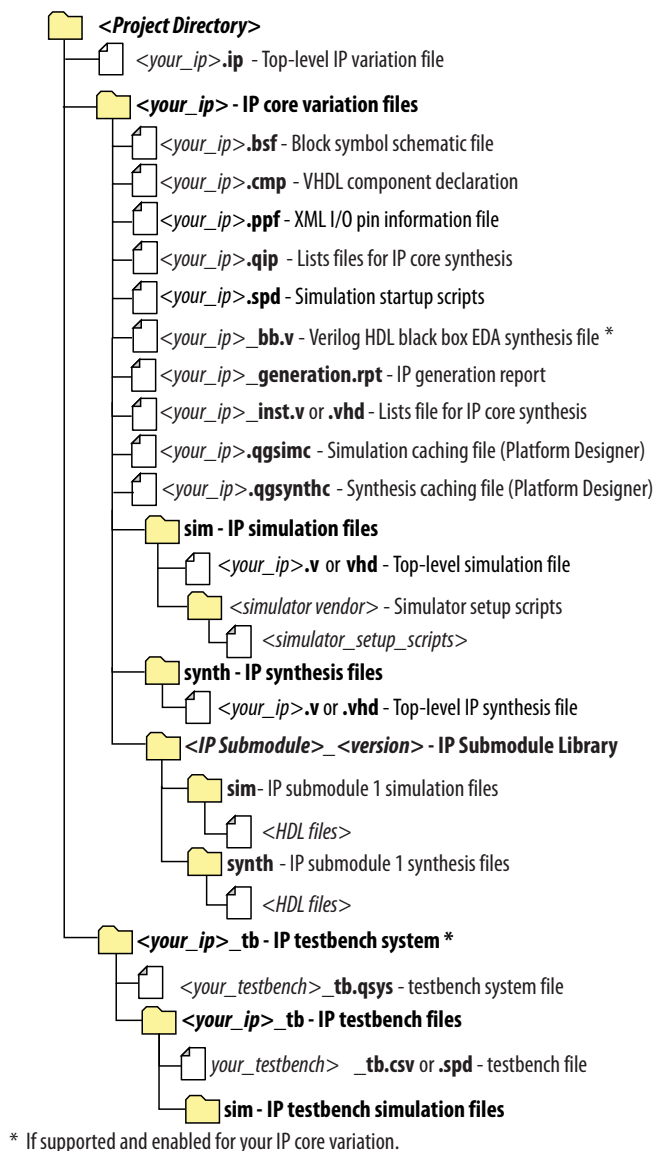
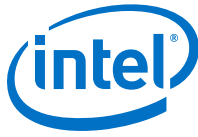
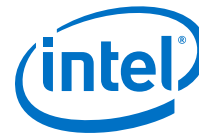


Table 28. Output Files of Intel FPGA IP Generation

| File Name                | Description                                                                                                                                                                                                |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <your_ip>.ip             | Top-level IP variation file that contains the parameterization of an IP core in your project. If the IP variation is part of a Platform Designer system, the parameter editor also generates a .qsys file. |
| <your_ip>.cmp            | The VHDL Component Declaration (.cmp) file is a text file that contains local generic and port definitions that you use in VHDL design files.                                                              |
| <your_ip>_generation.rpt | IP or Platform Designer generation log file. Displays a summary of the messages during IP generation.                                                                                                      |
| continued...             |                                                                                                                                                                                                            |



| File Name                                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <your_ip>.qgsimc (Platform Designer systems only)  | Simulation caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip regeneration of the HDL.                                                                                                                                                                                                                                                   |
| <your_ip>.qgsynth (Platform Designer systems only) | Synthesis caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip regeneration of the HDL.                                                                                                                                                                                                                                                    |
| <your_ip>.qip                                      | Contains all information to integrate and compile the IP component.                                                                                                                                                                                                                                                                                                                                                                                                              |
| <your_ip>.csv                                      | Contains information about the upgrade status of the IP component.                                                                                                                                                                                                                                                                                                                                                                                                               |
| <your_ip>.bsf                                      | A symbol representation of the IP variation for use in Block Diagram Files (.bdf).                                                                                                                                                                                                                                                                                                                                                                                               |
| <your_ip>.spd                                      | Input file that ip-make-simscript requires to generate simulation scripts. The .spd file contains a list of files you generate for simulation, along with information about memories that you initialize.                                                                                                                                                                                                                                                                        |
| <your_ip>.ppf                                      | The Pin Planner File (.ppf) stores the port and node assignments for IP components you create for use with the Pin Planner.                                                                                                                                                                                                                                                                                                                                                      |
| <your_ip>_bb.v                                     | Use the Verilog blackbox (_bb.v) file as an empty module declaration for use as a blackbox.                                                                                                                                                                                                                                                                                                                                                                                      |
| <your_ip>_inst.v or _inst.vhd                      | HDL example instantiation template. Copy and paste the contents of this file into your HDL file to instantiate the IP variation.                                                                                                                                                                                                                                                                                                                                                 |
| <your_ip>.regmap                                   | If the IP contains register information, the Intel Quartus Prime software generates the .regmap file. The .regmap file describes the register map information of master and slave interfaces. This file complements the .sopcinfo file by providing more detailed register information about the system. This file enables register display views and user customizable statistics in System Console.                                                                            |
| <your_ip>.svd                                      | Allows HPS System Debug tools to view the register maps of peripherals that connect to HPS within a Platform Designer system.<br>During synthesis, the Intel Quartus Prime software stores the .svd files for slave interface visible to the System Console masters in the .sof file in the debug session. System Console reads this section, which Platform Designer queries for register map information. For system slaves, Platform Designer accesses the registers by name. |
| <your_ip>.v <your_ip>.vhd                          | HDL files that instantiate each submodule or child IP core for synthesis or simulation.                                                                                                                                                                                                                                                                                                                                                                                          |
| mentor/                                            | Contains a msim_setup.tcl script to set up and run a ModelSim simulation.                                                                                                                                                                                                                                                                                                                                                                                                        |
| aldec/                                             | Contains a Riviera-PRO* script rivierapro_setup.tcl to setup and run a simulation.                                                                                                                                                                                                                                                                                                                                                                                               |
| /synopsys/vcs<br>/synopsys/vcsmx                   | Contains a shell script vcs_setup.sh to set up and run a VCS simulation.<br>Contains a shell script vcsmx_setup.sh and synopsys_sim.setup file to set up and run a VCS MX simulation.                                                                                                                                                                                                                                                                                            |
| /cadence                                           | Contains a shell script ncsim_setup.sh and other setup files to set up and run an NCSim simulation.                                                                                                                                                                                                                                                                                                                                                                              |
| /xcelium                                           | Contains an Xcelium* Parallel simulator shell script xcelium_setup.sh and other setup files to set up and run a simulation.                                                                                                                                                                                                                                                                                                                                                      |
| /submodules                                        | Contains HDL files for the IP core submodule.                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <IP submodule>/                                    | Platform Designer generates /synth and /sim sub-directories for each IP submodule directory that Platform Designer generates.                                                                                                                                                                                                                                                                                                                                                    |



## 5.4. Integration and Implementation

### 5.4.1. Clock Requirements

The Avalon-ST Stratix 10 Hard IP for PCI Express IP Core has a single 100 MHz input clock and a single output clock. An additional clock is available for PIPE simulations only.

#### refclk

Each instance of the PCIe IP core has a dedicated refclk input signal. This input reference clock can be sourced from any reference clock in the transceiver tile. Refer to the *Stratix 10 GX, MX, TX and SX Device Family Pin Connection Guidelines* for additional information on termination and valid locations.

#### coreclkout\_hip

This output clock is a fixed frequency clock that drives the Application Layer. The output clock frequency is derived from the maximum link width and maximum link rate of the PCIe IP core.

**Table 29. Application Layer Clock Frequency for All Combinations of Link Width, Data Rate and Application Layer Interface Widths**

| Maximum Link Rate | Maximum Link Width  | Avalon-ST Interface Width | coreclkout_hip Frequency |
|-------------------|---------------------|---------------------------|--------------------------|
| Gen1              | x1, x2, x4, x8, x16 | 256                       | 125 MHz                  |
| Gen2              | x1, x2, x4, x8      | 256                       | 125 MHz                  |
| Gen2              | x16                 | 256                       | 250 MHz                  |
| Gen3              | x1, x2, x4          | 256                       | 125 MHz                  |
| Gen3              | x8                  | 256                       | 250 MHz                  |
| Gen3              | x16                 | 512                       | 250 MHz                  |

#### sim\_pipe\_pclk\_in

This input clock is for PIPE simulation only. Derived from the refclk input, sim\_pipe\_pclk\_in is the PIPE interface clock for PIPE mode simulation.

### 5.4.2. Reset Requirements

The Avalon-ST Intel Stratix 10 Hard IP for PCI Express IP Core has two, asynchronous, active low reset inputs, npor and pin\_perst. Both reset the Transaction, Data Link and Physical Layers.

#### npor

The Application Layer drives the npor reset input to the PCIe IP core. If you choose to design an Application Layer does not drive npor, you must tie this output to 1'b1. The npor signal resets all registers and state machines to their initial values.

### pin\_perst

This is the PCI Express Fundamental Reset signal. Asserting this signal returns all registers and state machines to their initialization values. Each instance of the PCIe IP core has a dedicated pin\_perst pin. You must connect the pin\_perst of each hard IP instance to the corresponding nPERST pin of the device. These pins have the following location.

- NPERSTL0 : Bottom Left PCIe IP core and Configuration via Protocol (CvP)
- NPERSTL1: Middle Left PCIe IP core (When available)
- NPERSTL2: Top Left PCIe IP core (When available)
- NPERSTR0: Bottom Right PCIe IP core (When available)
- NPERSTR1: Middle Right PCIe IP core (When available)
- NPERSTR2: Top Right PCIe IP core (When available)

For maximum compatibility, always use the bottom PCIe IP core on the left side of the device first. This is the only location that supports CvP using the PCIe link.

**Note:** CvP is not available in the Quartus Prime Pro – Stratix 10 Edition 17.1 Interim Release

### reset\_status

When asserted, this signal indicates that the PCIe IP core is in reset. The reset\_status signal is synchronous to coreclkout\_hip. It is active high.

### clr\_st

This signal has the same functionality as reset\_status. It is provided for backwards compatibility with Arria 10 devices. It is active high.

## 5.5. Required Supporting IP Cores

Stratix 10 Hard IP for PCI Express IP core designs always include the Hard Reset Controller and TX PLL IP cores. System generation automatically adds these components to the generated design.

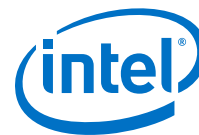
### 5.5.1. Hard Reset Controller

The Hard Reset Controller generates the reset for the PCIe IP core logic, transceivers, and Application Layer. To meet 100 ms PCIe configuration time, the Hard Reset Controller interfaces with the SDM. This allows the PCIe Hard IP to be configured first so that PCIe link training occurs when the FPGA fabric is still being configured.

### 5.5.2. TX PLL

For Gen1 or Gen2, the PCIe IP core uses the fractional PLL ( fPLL) for the TX PLL. For Gen3, the PCIe Hard IP core uses the fPLL when operating at the Gen1 or Gen2 data rate. It uses the advanced transmit (ATX) PLL when operating at the Gen3 data rate.

**Note:** The ATX PLL is sometimes called the LC PLL, where LC refers to inductor/capacitor.



## 5.6. Channel Layout and PLL Usage

The following figures show the channel layout and PLL usage for Gen1, Gen2 and Gen3, x1, x2, x4, x8 and x16 variants of the Intel Stratix 10 IP core. The channel layout is the same for the Avalon-ST and Avalon-MM interfaces to the Application Layer.

**Note:** All of the PCIe hard IP instances in Stratix 10 devices are Gen3 x16. Channels 8-15 are available for other protocols when fewer than 16 channels are used. Refer to *Channel Availability* for more information.

**Figure 23. Gen1 and Gen2 x1**

|         |               |               |       |              |
|---------|---------------|---------------|-------|--------------|
| fPLL1   | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 |       |              |
|         | PMA Channel 3 | PCS Channel 3 |       |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 |       |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 |       |              |
|         | PMA Channel 0 | PCS Channel 0 |       |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 15 | PCIe Hard IP |
|         | PMA Channel 3 | PCS Channel 3 | Ch 14 |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 13 |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 12 |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 11 |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 | Ch 10 |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 9  |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 8  |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 7  |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 6  |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 5  |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 | Ch 4  |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 3  |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 2  |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 1  |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 0  |              |
|         | PMA Channel 0 | PCS Channel 0 |       |              |

HRC  
connects to  
fPLL0

Figure 24. Gen1 and Gen2 x2

|         |               |               |       |              |
|---------|---------------|---------------|-------|--------------|
| fPLL1   | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 |       |              |
|         | PMA Channel 3 | PCS Channel 3 |       |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 |       |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 |       |              |
|         | PMA Channel 0 | PCS Channel 0 |       |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 15 | PCIe Hard IP |
|         | PMA Channel 3 | PCS Channel 3 | Ch 14 |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 13 |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 12 |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 11 |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 | Ch 10 |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 9  |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 8  |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 7  |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 6  |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 5  |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 | Ch 4  |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 3  |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 2  |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 1  |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 0  |              |
|         | PMA Channel 0 | PCS Channel 0 |       |              |

HRC connects to fPLL0

Figure 25. Gen1 and Gen2 x4

|         |               |               |       |              |
|---------|---------------|---------------|-------|--------------|
| fPLL1   | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 |       |              |
|         | PMA Channel 3 | PCS Channel 3 |       |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 |       |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 |       |              |
|         | PMA Channel 0 | PCS Channel 0 |       |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 15 | PCIe Hard IP |
|         | PMA Channel 3 | PCS Channel 3 | Ch 14 |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 13 |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 12 |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 11 |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 | Ch 10 |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 9  |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 8  |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 7  |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 6  |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 5  |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 | Ch 4  |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 3  |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 2  |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 1  |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 0  |              |
|         | PMA Channel 0 | PCS Channel 0 |       |              |

HRC connects to fPLL0

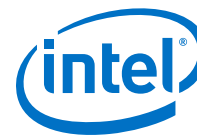


Figure 26. Gen1 and Gen2 x8

|         |               |               |       |              |
|---------|---------------|---------------|-------|--------------|
| fPLL1   | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 |       |              |
|         | PMA Channel 3 | PCS Channel 3 |       |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 |       |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 |       |              |
|         | PMA Channel 0 | PCS Channel 0 |       |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 |       |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 15 | PCIe Hard IP |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 14 |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 13 |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 12 |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 | Ch 11 |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 10 |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 9  |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 8  |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 7  |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 6  |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 | Ch 5  |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 4  |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 3  |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 2  |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 1  |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 0  |              |

Figure 27. Gen1 and Gen2 x16

|         |               |               |       |              |
|---------|---------------|---------------|-------|--------------|
| fPLL1   | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 |       |              |
|         | PMA Channel 3 | PCS Channel 3 |       |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 |       |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 |       |              |
|         | PMA Channel 0 | PCS Channel 0 |       |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 |       |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 15 | PCIe Hard IP |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 14 |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 13 |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 12 |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 | Ch 11 |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 10 |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 9  |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 8  |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 7  |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 6  |              |
| fPLL1   | PMA Channel 5 | PCS Channel 5 | Ch 5  |              |
| ATXPLL1 | PMA Channel 4 | PCS Channel 4 | Ch 4  |              |
|         | PMA Channel 3 | PCS Channel 3 | Ch 3  |              |
| fPLL0   | PMA Channel 2 | PCS Channel 2 | Ch 2  |              |
| ATXPLL0 | PMA Channel 1 | PCS Channel 1 | Ch 1  |              |
|         | PMA Channel 0 | PCS Channel 0 | Ch 0  |              |

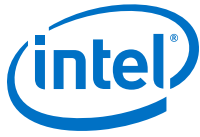


Figure 28. Gen3 x1

|                |               |               |       |              |
|----------------|---------------|---------------|-------|--------------|
| fPLL1          | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 |       |              |
|                | PMA Channel 3 | PCS Channel 3 |       |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 |       |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 |       |              |
|                | PMA Channel 0 | PCS Channel 0 |       |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 15 | PCIe Hard IP |
|                | PMA Channel 3 | PCS Channel 3 | Ch 14 |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 13 |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 | Ch 12 |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 11 |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 | Ch 10 |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 9  |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 8  |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 7  |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 | Ch 6  |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 5  |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 | Ch 4  |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 3  |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 2  |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 1  |              |
| ATXPLL0 (Gen3) | PMA Channel 1 | PCS Channel 1 | Ch 0  |              |
|                | PMA Channel 0 | PCS Channel 0 |       |              |

HRC connects to fPLL0 & ATXPLL0

Figure 29. Gen3 x2

|                |               |               |       |              |
|----------------|---------------|---------------|-------|--------------|
| fPLL1          | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 |       |              |
|                | PMA Channel 3 | PCS Channel 3 |       |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 |       |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 |       |              |
|                | PMA Channel 0 | PCS Channel 0 |       |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 15 | PCIe Hard IP |
|                | PMA Channel 3 | PCS Channel 3 | Ch 14 |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 13 |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 | Ch 12 |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 11 |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 | Ch 10 |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 9  |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 8  |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 7  |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 | Ch 6  |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 5  |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 | Ch 4  |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 3  |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 2  |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 1  |              |
| ATXPLL0 (Gen3) | PMA Channel 1 | PCS Channel 1 | Ch 0  |              |
|                | PMA Channel 0 | PCS Channel 0 |       |              |

HRC connects to fPLL0 & ATXPLL0

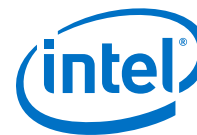


Figure 30. Gen3 x4

|                |               |               |       |              |
|----------------|---------------|---------------|-------|--------------|
| fPLL1          | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 |       |              |
|                | PMA Channel 3 | PCS Channel 3 |       |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 |       |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 |       |              |
|                | PMA Channel 0 | PCS Channel 0 |       |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 15 | PCIe Hard IP |
|                | PMA Channel 3 | PCS Channel 3 | Ch 14 |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 13 |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 | Ch 12 |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 11 |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 | Ch 10 |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 9  |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 8  |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 7  |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 | Ch 6  |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 5  |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 | Ch 4  |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 3  |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 2  |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 1  |              |
| ATXPLL0 (Gen3) | PMA Channel 1 | PCS Channel 1 | Ch 0  |              |
|                | PMA Channel 0 | PCS Channel 0 |       |              |

Figure 31. Gen3 x8

|                |               |               |       |              |
|----------------|---------------|---------------|-------|--------------|
| fPLL1          | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 |       |              |
|                | PMA Channel 3 | PCS Channel 3 |       |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 |       |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 |       |              |
|                | PMA Channel 0 | PCS Channel 0 |       |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 15 | PCIe Hard IP |
|                | PMA Channel 3 | PCS Channel 3 | Ch 14 |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 13 |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 | Ch 12 |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 11 |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 | Ch 10 |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 9  |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 8  |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 7  |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 | Ch 6  |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 5  |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 | Ch 4  |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 3  |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 2  |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 1  |              |
| ATXPLL0 (Gen3) | PMA Channel 1 | PCS Channel 1 | Ch 0  |              |
|                | PMA Channel 0 | PCS Channel 0 |       |              |

**Figure 32. Gen3 x16**

|                |               |               |       |              |
|----------------|---------------|---------------|-------|--------------|
| fPLL1          | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 |       |              |
|                | PMA Channel 3 | PCS Channel 3 |       |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 |       |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 |       |              |
|                | PMA Channel 0 | PCS Channel 0 |       |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 |       |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 |       |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 15 | PCIe Hard IP |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 14 |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 | Ch 13 |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 12 |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 | Ch 11 |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 10 |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 9  |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 8  |              |
| ATXPLL0 (Gen3) | PMA Channel 1 | PCS Channel 1 | Ch 7  |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 6  |              |
| fPLL1          | PMA Channel 5 | PCS Channel 5 | Ch 5  |              |
| ATXPLL1        | PMA Channel 4 | PCS Channel 4 | Ch 4  |              |
|                | PMA Channel 3 | PCS Channel 3 | Ch 3  |              |
| fPLL0          | PMA Channel 2 | PCS Channel 2 | Ch 2  |              |
| ATXPLL0        | PMA Channel 1 | PCS Channel 1 | Ch 1  |              |
|                | PMA Channel 0 | PCS Channel 0 | Ch 0  |              |

HRC connects to fPLL0 & ATXPLL1 middle XCVR bank

### Related Information

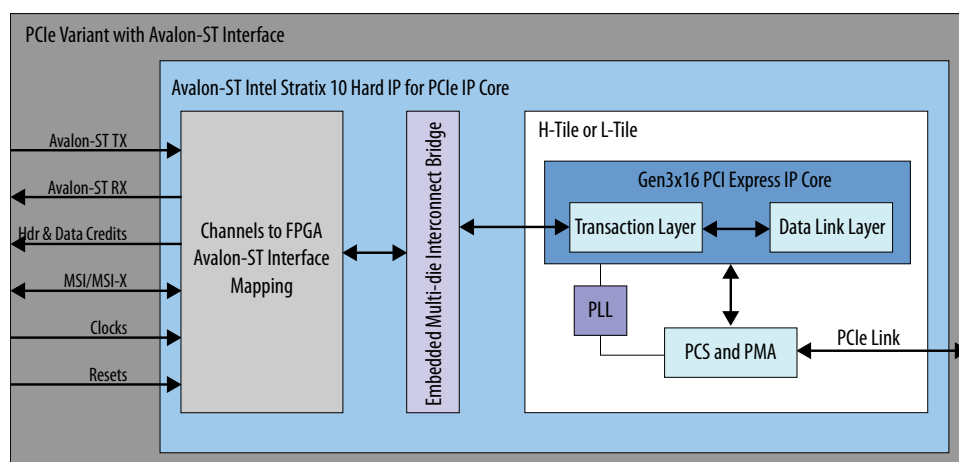
[Channel Availability](#) on page 16

## 6. Block Descriptions

The Intel Stratix 10 Hard IP for PCI Express implements the complete PCI Express protocol stack as defined in the *PCI Express Base Specification*. The protocol stack includes the following layers:

- **Transaction Layer**—The Transaction Layer contains the Configuration Space, which manages communication with the Application Layer, the RX and TX channels, the RX buffer, and flow control credits.
- **Data Link Layer**—The Data Link Layer, located between the Physical Layer and the Transaction Layer, manages packet transmission and maintains data integrity at the link level. Specifically, the Data Link Layer performs the following tasks:
  - Manages transmission and reception of Data Link Layer Packets (DLLPs)
  - Generates all transmission link cyclical redundancy code (LCRC) values and checks all LCRCs during reception
  - Manages the retry buffer and retry mechanism according to received ACK/NAK Data Link Layer packets
  - Initializes the flow control mechanism for DLLPs and routes flow control credits to and from the Transaction Layer
- **Physical Layer**—The Physical Layer initializes the speed, lane numbering, and lane width of the PCI Express link according to packets received from the link and directives received from higher layers. The following figure provides a high-level block diagram.

**Figure 33. Intel Stratix 10 Hard IP for PCI Express Using the Avalon-ST Interface**



Each channel of the Physical Layer is paired with an Embedded Multi-die Interconnect Bridge (EMIB) module. The FPGA fabric interfaces to the PCI Express IP core through the EMIB.

## 6.1. Interfaces

This section describes the top-level interfaces in the Avalon-ST Stratix 10 Hard IP for PCIe IP core.

The colors and variables in the figure below specify the following information:

- Orange text: signal is only available for L-Tile devices
- Deep red/brown text: signal is only available for H-Tile devices
- Blue text: PIPE interface signals, only available for simulation
- $\langle w \rangle$ : the width of the Avalon-ST data interface
- $\langle n \rangle$ : 2 for the 512-bit interface and 1 for the 256-bit interface
- $\langle r \rangle$ : 6 for the 512-bit interface and 3 for the 256-bit

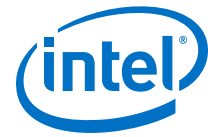
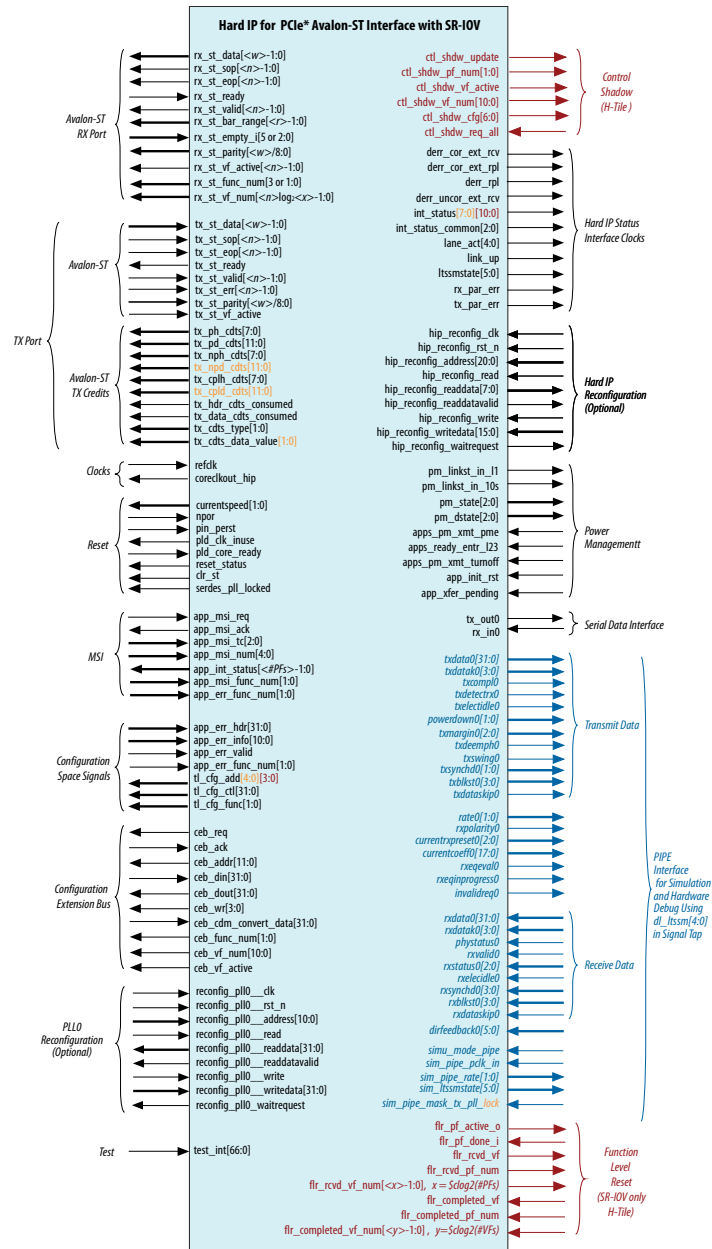


Figure 34. Avalon-ST Intel Stratix 10 Hard IP for PCI Express Top-Level Signals



### 6.1.1. TLP Header and Data Alignment for the Avalon-ST RX and TX Interfaces

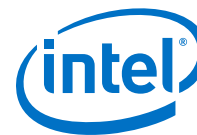
The TLP header and data is packed on the TX and RX interfaces.

The ordering of bytes in the header and data portions of packets is different. The first byte of the header dword is located in the most significant byte of the dword. The first byte of the data dword is located in the least significant byte of the dword on the data bus.

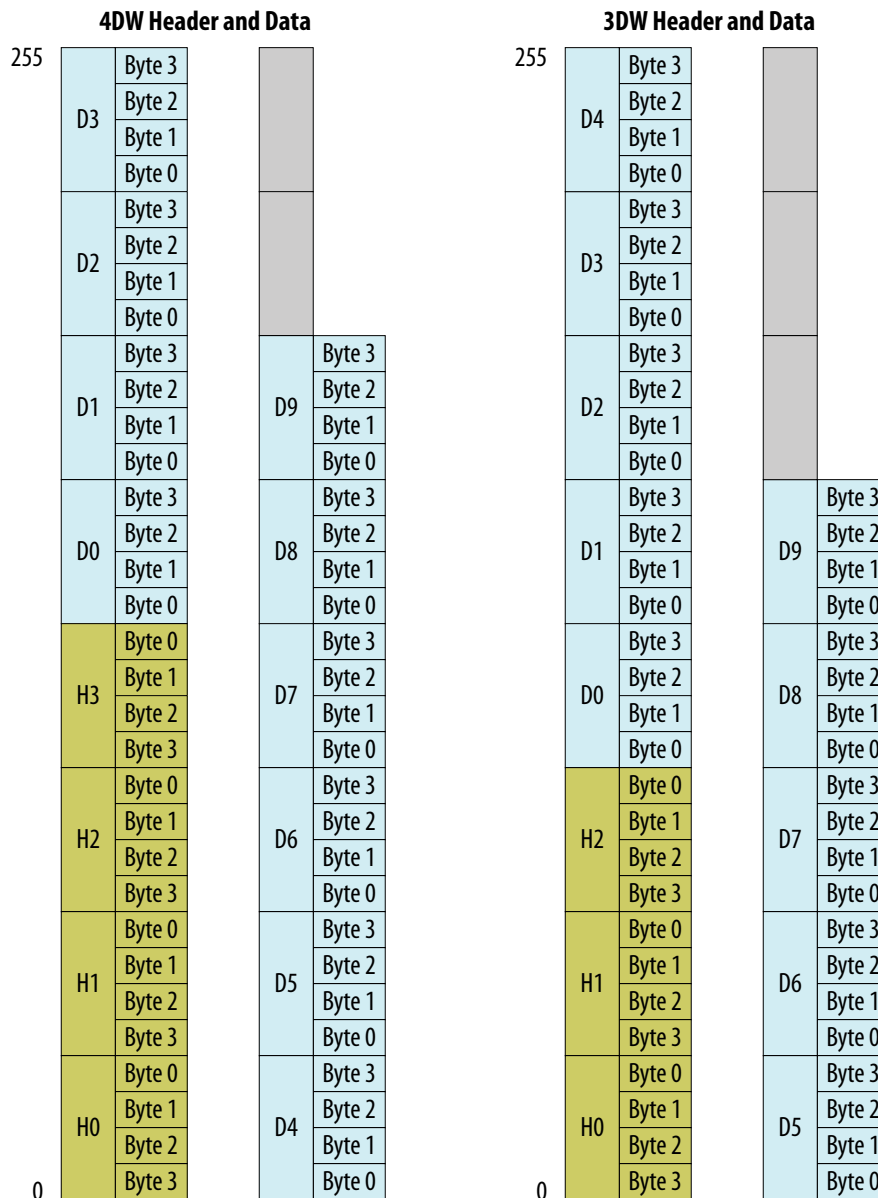
**Table 30. Mapping Avalon-ST Packets to PCI Express TLPs**

| Packet  | TLP                                                                                   |
|---------|---------------------------------------------------------------------------------------|
| Header0 | pcie_hdr_byte0, pcie_hdr_byte1, pcie_hdr_byte2, pcie_hdr_byte3                        |
| Header1 | pcie_hdr_byte4, pcie_hdr_byte5, pcie_hdr_byte6, pcie_hdr_byte7                        |
| Header2 | pcie_hdr_byte8, pcie_hdr_byte9, pcie_hdr_byte10, pcie_hdr_byte11                      |
| Header3 | pcie_hdr_byte12, pcie_hdr_byte13, pcie_hdr_byte14, pcie_hdr_byte15                    |
| Data0   | pcie_data_byte3, pcie_data_byte2, pcie_data_byte1, pcie_data_byte0                    |
| Data1   | pcie_data_byte7, pcie_data_byte6, pcie_data_byte5, pcie_data_byte4                    |
| Data2   | pcie_data_byte11, pcie_data_byte10, pcie_data_byte9, pcie_data_byte8                  |
| Data<n> | pcie_data_byte<4n+3>, pcie_data_byte<4n+2>, pcie_data_byte<4n+1>, pcie_data_byte0<4n> |

The following figure illustrates the mapping of Avalon-ST packets to PCI Express TLPs for a three-dword header and a four-dword header. In the figure, H0 to H3 are header dwords, and D0 to D9 are data dwords.



**Figure 35. Three and Four DWord Header and Data on the TX and RX Interfaces**



**Note:** Unlike in previous devices, in Intel Stratix 10 data is not qword aligned. If you are porting your application from an earlier implementation, you must update your application to compensate for this change.

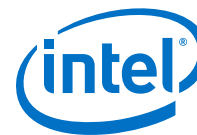
### 6.1.2. Avalon-ST 256-Bit RX Interface

The Application Layer receives data from the Transaction Layer of the PCIe IP core over the Avalon-ST RX interface. This is a 256-bit interface.

**Table 31. 256-Bit Avalon-ST RX Datapath**

| Signal                                        | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rx_st_data[255:0]                             | Output    | Receive data bus. The Application Layer receives data from the Transaction Layer on this bus. The data on this bus is valid when rx_st_valid is asserted.<br><i>Refer to the TLP Header and Data Alignment for the Avalon-ST TX and Avalon-ST RX Interfaces for the layout of TLP headers and data.</i>                                                                                                                                                                                                                                               |
| rx_st_sop                                     | Output    | Marks the first cycle of the TLP when both rx_st_sop and rx_st_valid are asserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| rx_st_eop                                     | Output    | Marks the last cycle of the TLP when both rx_st_eop and rx_st_valid are asserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| rx_st_ready                                   | Input     | Indicates that the Application Layer is ready to accept data. The Application Layer deasserts this signal to throttle the data stream.                                                                                                                                                                                                                                                                                                                                                                                                                |
| rx_st_valid                                   | Output    | Qualifies rx_st_data into the Application Layer.<br>The rx_st_ready to rx_st_valid latency for Stratix 10 devices is 17 cycles. When rx_st_ready deasserts, rx_st_valid will deassert within 17 cycles. When rx_st_ready reasserts, rx_st_valid will reassert within 17 cycles if there is more data to send. To achieve the best throughput, Intel recommends that you size the RX buffer to avoid the deassertion of rx_st_ready. Refer to <i>Avalon-ST RX Interface rx_st_valid Deasserts</i> for a timing diagram that illustrates this behavior. |
| rx_st_bar_range[2:0]                          | Output    | Specifies the bar for the TLP being output. The following encodings are defined: <ul style="list-style-type: none"> <li>• 000: Memory Bar 0</li> <li>• 001: Memory Bar 1</li> <li>• 010: Memory Bar 2</li> <li>• 011: Memory Bar 3</li> <li>• 100: Memory Bar 4</li> <li>• 101: Memory Bar 5</li> <li>• 110: I/O BAR</li> <li>• 111: Expansion ROM BAR</li> </ul> The data on this bus is valid when rx_st_sop and rx_st_valid are both asserted.                                                                                                     |
| rx_st_vf_active H-Tile                        | Output    | When asserted, the received TLP targets a VF bar. Valid if rx_st_sop and rx_st_valid are asserted. When deasserted, the TLP targets a PF and the rx_st_func_num port drives the function number.<br>Valid when multiple virtual functions are enabled.                                                                                                                                                                                                                                                                                                |
| rx_st_func_num[1:0] H-Tile                    | Output    | Specifies the target physical function number of the received TLP. The application uses this information to route packets for both request and completion TLPs. For completion TLPs, specifies the PF number of the requestor for this completion TLP. If the TLP targets a VF[<m>, <n>], this bus carries the PF<m> information.<br>Valid when multiple physical functions are enabled.<br>These outputs are qualified by rx_st_sop and rx_st_valid.                                                                                                 |
| rx_st_vf_num[log <sub>2</sub> <x>-1:0] H-Tile | Output    | Specifies the target VF number rx_st_data[255:0] of the received TLP. The application uses this information for both request and completion TLPs. For completion TLPs, specifies the VF number of the requestor for this completion TLP. <x> is the number of VFs.<br>Valid when rx_st_vf_active is asserted. If the TLP targeting at VF[<m>, <n>] this bus carries the VF<n> information.<br>Valid when multiple virtual functions are enabled.                                                                                                      |

*continued...*



| Signal           | Direction | Description                                                                                                                                             |
|------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  |           | These outputs are qualified by rx_st_sop and rx_st_valid.                                                                                               |
| rx_st_empty[2:0] |           | Specifies the number of dwords that are empty, valid during cycles when the rx_st_eop signal is asserted. Not interpreted when rx_st_eop is deasserted. |

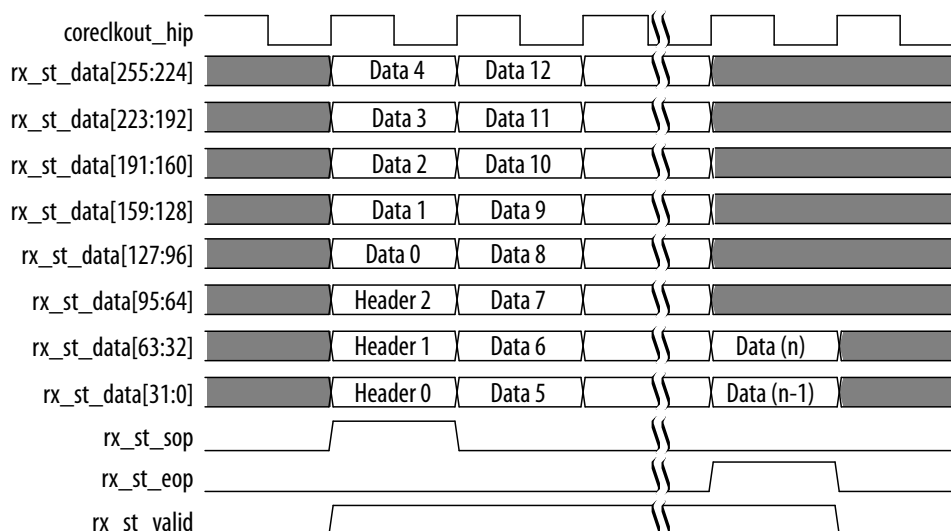
#### Related Information

- [Avalon-ST RX Interface rx\\_st\\_valid Deasserts](#) on page 63
- [Configuration Status Interface](#)  
For multifunction variants, the Configuration Status Interface specifies the settings of the various configuration register fields of the functions.

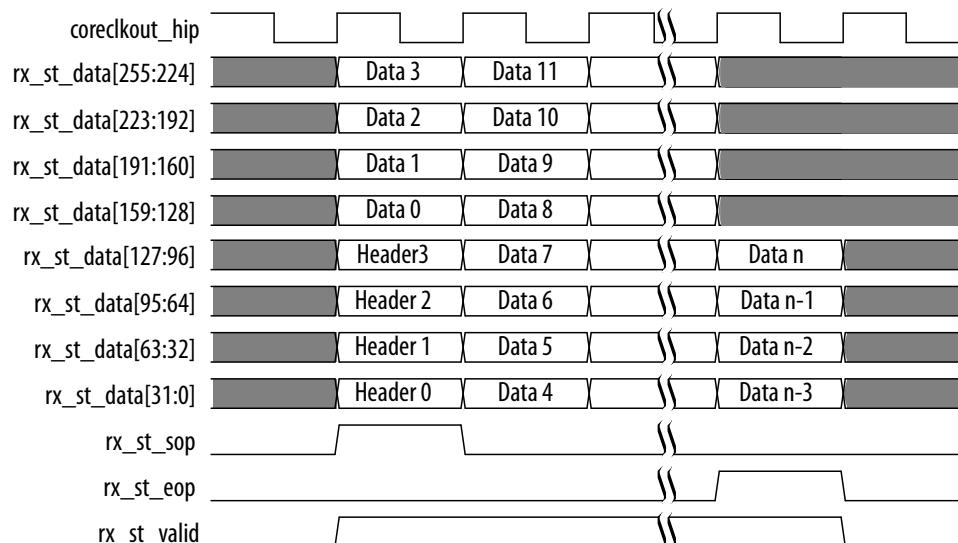
#### 6.1.2.1. Avalon-ST RX Interface Three- and Four-Dword TLPs

These timing diagrams illustrate the layout of headers and data for the Avalon-ST RX interface.

**Figure 36. Avalon-ST RX Interface Cycle Definition for Three-Dword Header TLPs**



**Figure 37. Avalon-ST RX Interface Cycle Definition for Four-Dword Header TLPs**

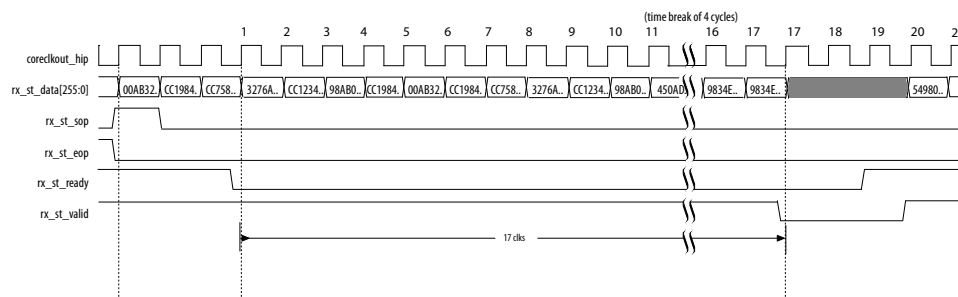


#### 6.1.2.2. Avalon-ST RX Interface rx\_st\_ready Deasserts for the 256-Bit Interface

This timing diagram illustrates the timing of the RX interface when the application throttles the Stratix 10 Hard IP for PCI Express by deasserting rx\_st\_ready. The rx\_st\_valid signal deasserts within 17 cycles of the rx\_st\_ready deassertion for variants other than the Gen3 x16 variant, and within 18 cycles for the Gen3 x16 variant. The rx\_st\_valid signal reasserts within 17 cycles after rx\_st\_ready reasserts if there is more data to send for variants other than the Gen3 x16 variant, and within 18 cycles for the Gen3 x16 variant. rx\_st\_data is held until the application is able to accept it.

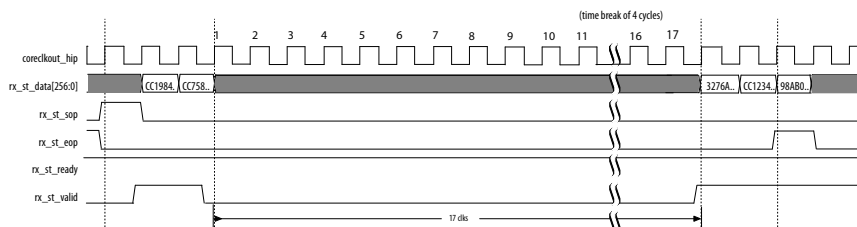
To achieve the best throughput, Intel recommends that you size the RX buffer in your application logic to avoid the deassertion of rx\_st\_ready.

**Figure 38. Avalon-ST RX Interface rx\_st\_valid Deasserts for the 256-Bit Interface**



## Avalon-ST RX Interface rx\_st\_valid Reasserts for the 256-Bit Interface

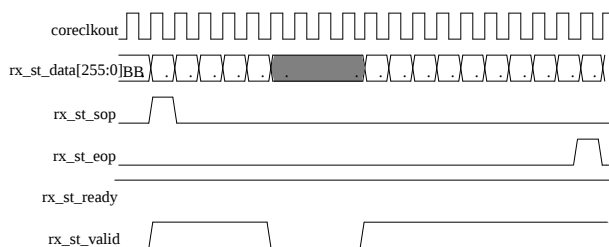
Figure 39.



### 6.1.2.3. Avalon-ST RX Interface rx\_st\_valid Deasserts

This timing diagram illustrates the deassertion of the rx\_st\_valid signal, even when rx\_st\_ready remains asserted.

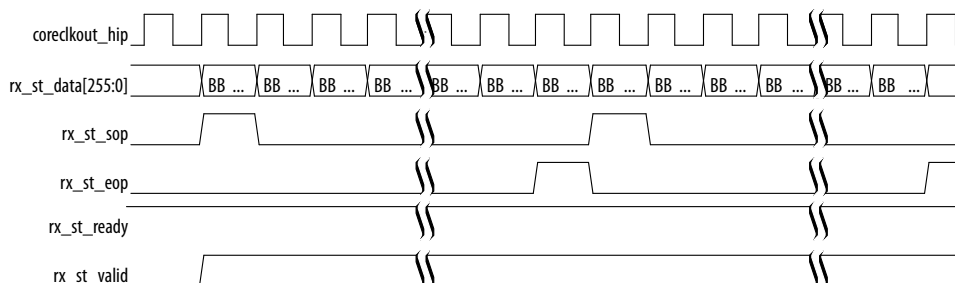
Figure 40. Avalon-ST RX Interface rx\_st\_valid Deasserts



### 6.1.2.4. Avalon-ST RX Back-to-Back Transmission

This timing diagram illustrates back-to-back transmission on the Avalon-ST RX interface with no idle cycles between the assertion of rx\_st\_eop and rx\_st\_sop.

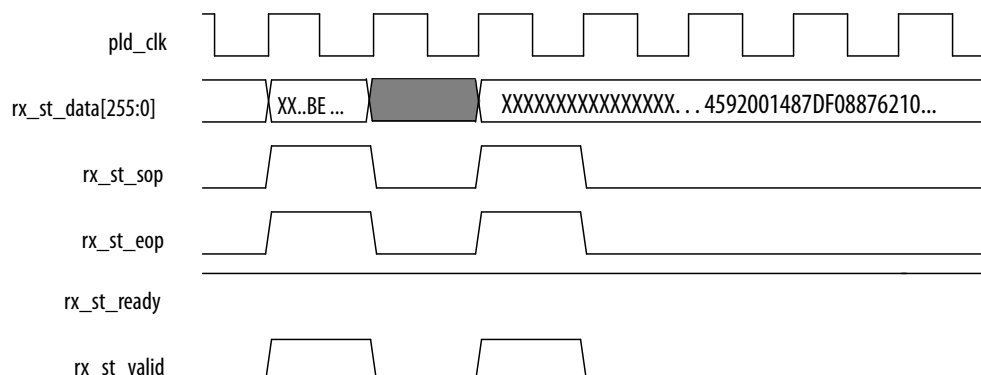
Figure 41. Avalon-ST RX Back-to-Back Transmission



### 6.1.2.5. Avalon-ST RX Interface Single-Cycle TLPs

This timing diagram illustrates two single-cycle TLPs.

**Figure 42. Avalon-ST RX Single-Cycle TLPs**



### 6.1.3. Avalon-ST 512-Bit RX Interface

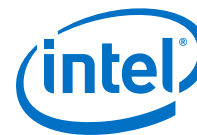
The Application Layer receives data from the Transaction Layer of the PCI Express IP core over the Avalon-ST RX interface. The 512-bit interface supports Gen3 x16 variants and uses a 250 MHz clock, simplifying timing closure.

The 512-bit interface supports two locations for the beginning of a TLP, bit[0] and bit[256]. The interface supports two TLPs per cycle only when an end-of-packet cycle occurs in the lower 256 bits. In other words, a TLP can start on bit [256] only if a rx\_st\_eop pulse occurs in the lower 256 bits.

**Note:** This interface is not strictly Avalon-compliant because it supports two rx\_st\_sop signals and two rx\_st\_eop signals per cycle.

**Table 32. 512-Bit Avalon-ST RX Datapath**

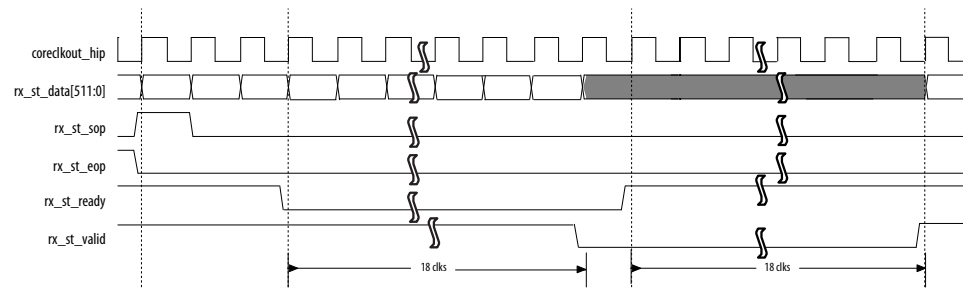
| Signal              | Direction | Description                                                                                                                                                                                                                                                                                                                                                              |
|---------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rx_st_data_o[511:0] | Output    | Receive data bus. The Application Layer receives data from the Transaction Layer on this bus. For large TLPs, the Application Layer drives 512-bit data on rx_st_data[511:0] until the end-of-packet cycle.<br>For TLPs with an end-of-packet cycle in the lower 256 bits, the 512-bit interface supports a start-of-packet cycle in the upper 256 bits.                 |
| rx_st_sop[1:0]      | Output    | Signals the first cycle of the TLP when asserted in conjunction the corresponding bit of rx_st_valid. The following encodings are defined: <ul style="list-style-type: none"> <li>rx_st_sop[1]: When asserted, indicates the start of a TLP on rx_st_data[511:256].</li> <li>rx_st_sop[0]: When asserted, indicates the start of a TLP on rx_st_data[255:0].</li> </ul>  |
| rx_st_eop[1:0]      | Output    | Signals the last cycle of the TLP when asserted in conjunction with the corresponding bit of rx_st_valid[1:0]. The following encodings are defined: <ul style="list-style-type: none"> <li>rx_st_eop[1]: When asserted, signals the end of a TLP on rx_st_data[511:256].</li> <li>rx_st_sop[0]: When asserted, signals the end of a TLP on rx_st_data[255:0].</li> </ul> |
| rx_st_ready_i       | Input     | Indicates that the Application Layer is ready to accept data. The Application Layer deasserts this signal to apply backpressure to the data stream.                                                                                                                                                                                                                      |
| continued...        |           |                                                                                                                                                                                                                                                                                                                                                                          |



| Signal                 | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rx_st_valid_o[1:0]     | Output    | <p>Qualifies rx_st_data_o into the Application Layer.</p> <p>The rx_st_ready_i to rx_st_valid_o[1:0] latency for Stratix 10 devices is 18 cycles. When rx_st_ready_i deasserts, rx_st_valid_o[1:0] will deassert within 18 cycles. When rx_st_ready_i reasserts, rx_st_valid_o will reassert within 18 cycles if there is more data to send. To achieve the best throughput, Intel recommends that you size the RX buffer in your application logic to avoid the deassertion of rx_st_ready_o.</p> <p>The <i>Relationship Between rx_st_ready and rx_st_valid for the 512-bit Avalon-ST Interface</i> timing diagram below illustrates the relationship between rx_st_ready_i and rx_st_valid_o.</p> |
| rx_st_bar_range_o[5:0] | Output    | <p>Specifies the BAR for the TLP being output. The following encodings are defined:</p> <ul style="list-style-type: none"> <li>rx_st_bar_range_o[5:3]: Specifies BAR for rx_st_data[511:256].</li> <li>rx_st_bar_range_o[2:0]: Specifies BAR for rx_st_data[255:0].</li> </ul> <p>For each BAR range, the following encodings are defined:</p> <ul style="list-style-type: none"> <li>000: Memory BAR 0</li> <li>001: Memory BAR 1</li> <li>010: Memory BAR 2</li> <li>011: Memory BAR 3</li> <li>100: Memory BAR 4</li> <li>101: Memory BAR 5</li> <li>110: I/O BAR</li> <li>111: Expansion ROM BAR</li> </ul> <p>These outputs are valid when both rx_st_sop and rx_st_valid are asserted.</p>     |
| rx_st_empty_o[5:0]     | Output    | <p>Specifies the number of dwords that are empty during cycles when the rx_st_eop_o[1:0] signal is asserted. Not valid when rx_st_eop[1:0] is deasserted. The following encodings are defined:</p> <ul style="list-style-type: none"> <li>rx_st_empty_o[5:3]: Specifies empty dwords for the high-order packet.</li> <li>rx_st_empty_o[2:0]: Specifies empty dwords for the low-order packet.</li> </ul>                                                                                                                                                                                                                                                                                             |
| rx_st_parity_o[63:0]   | Output    | <p>Byte parity for rx_st_data_o. Bit 0 corresponds to rx_st_data_o[7:0], bit 1 corresponds to rx_st_data_o[15:8] and so on.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| continued...           |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

| Signal                                         | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------------------------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rx_st_vf_active[1:0] H-Tile                    | Output    | When asserted, the received TLP targets a VF bar. Valid when rx_st_sop is asserted. When deasserted, the TLP targets a PF and the rx_st_func_num port drives the physical function number.<br>For the 512-bit interface, Bit [0] corresponds to the rx_st_data[255:0] and bit 1 corresponds to rx_st_data[511:256]<br>Valid when multiple physical functions are enabled.                                                                                                                                                                                                                                                                                              |
| rx_st_func_num[3:0] H-Tile                     | Output    | <ul style="list-style-type: none"> <li>rx_st_func_num[3:2]: Specifies the physical function number for rx_st_data[511:256].</li> <li>rx_st_func_num[1:0]: Specifies the physical function number for rx_st_data[255:0]</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| rx_st_vf_num[log <sub>2</sub> <x>)-1:0] H-Tile | Output    | Specifies the target VF number for the received TLP. The application uses this information for both request and completion TLPs. For completion TLPs, specifies the VF number of the requester for this completion TLP. <x> is the number of VFs.<br>The following encodings are defined: <ul style="list-style-type: none"> <li>rx_st_vf_num[21:11]: Specifies the VF number for rx_st_data[511:256]</li> <li>rx_st_vf_num[10:0]: Specifies the VF number for rx_st_data[255:0]</li> </ul> Valid when rx_st_vf_active is asserted. If the TLP targeting at VF[<m>, <n>] this bus carries the VF<n> information.<br>Valid when multiple virtual functions are enabled. |

**Figure 43. Relationship Between rx\_st\_ready and rx\_st\_valid for the 512-bit Avalon-ST Interface**

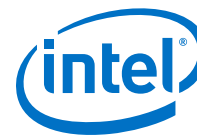


#### Related Information

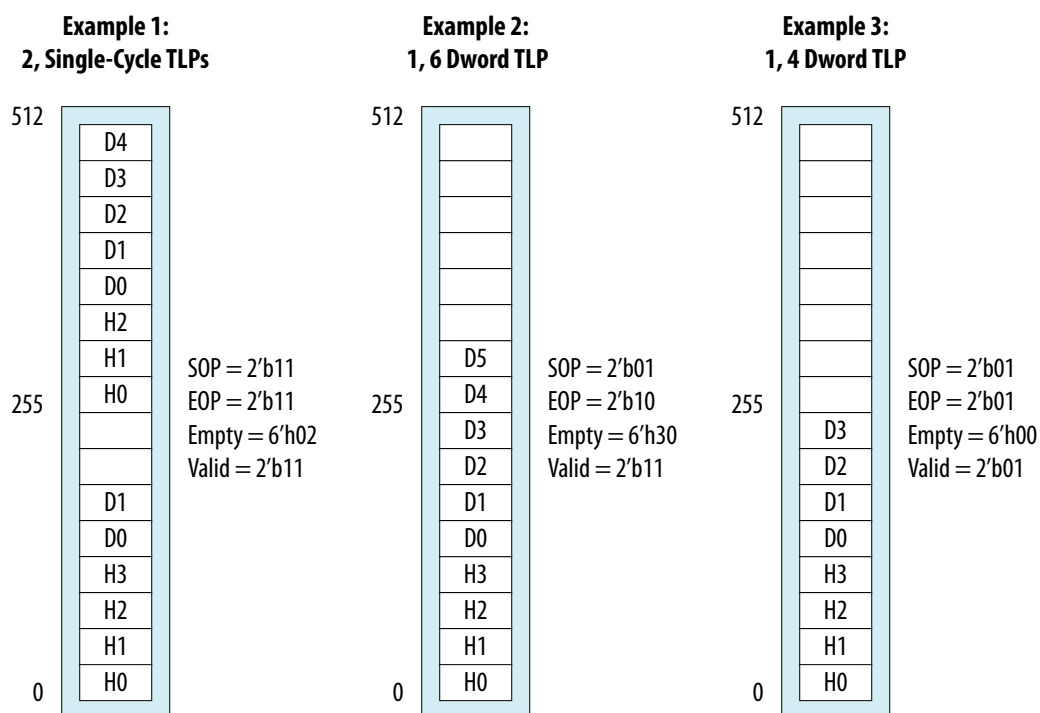
- [Avalon-ST RX Interface rx\\_st\\_valid Deasserts](#) on page 63
- [Configuration Status Interface](#)  
For multifunction variants, the Configuration Status Interface specifies the settings of the various configuration register fields of the functions.

#### 6.1.3.1. PCIe TLP Layout Using the 512-Bit Interface

The following figure illustrates TLP layout and provides three examples of TLP transmission with the correct encodings for the rx\_st\_sop[1:0], rx\_st\_eop[1:0], rx\_st\_valid\_o[1:0] and rx\_st\_empty\_o[5:0] signals.



**Figure 44. Receiving PCIe TLPs Using 512-bit Avalon-ST Interface**



#### Example 1: Two, Small TLPs

Example 1 illustrates the transmission of two, single-cycle TLPs. The TLP transmitted in the low-order bits has two empty dwords. The TLP transmitted in the high-order bits has no empty dwords.

#### Example 2: One, 6-DWord TLP

Example 2 shows the transmission of one, 6-dword PCI Express. The low-order bits provide the header and the first four dwords of data. The high-order bits have the final two dwords of data. The `rx_st_empty_o[5:3]` vector indicates six empty dwords in the high-order bits. The `rx_st_eop[1]` bit indicates the end of the TLP in the high-order bits.

#### Example 3: One, 4-DWord TLP

Example 3 shows a single cycle packet in the low-order bits and no transmission in the high-order bits. The `rx_st_valid_o[1]` bit indicates that the high-order bits do not drive valid data.

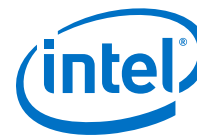
### 6.1.4. Avalon-ST 256-Bit TX Interface

The Application Layer transfers data to the Transaction Layer of the PCIe IP core over the Avalon-ST TX interface. This is a 256-bit interface.

**Table 33. 256-Bit Avalon-ST TX Datapath**

| Signal            | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tx_st_data[255:0] | Input     | <p>Data for transmission. The Application Layer must provide a properly formatted TLP on the TX interface. Valid when tx_st_valid is asserted (subject to the ready latency, see below). The mapping of message TLPs is the same as the mapping of Transaction Layer TLPs with 4 dword headers. The number of data cycles must be correct for the length and address fields in the header. Issuing a packet with an incorrect number of data cycles results in the TX interface hanging and becoming unable to accept further requests. In addition, the TLP must be no larger than the negotiated Max Payload size.</p> <p>For the TLP requester ID field, bits[31:16] in dword1 specify the following information:</p> <ul style="list-style-type: none"> <li>Bits[19:16]: Specify the PF number.</li> <li>Bits[31:20]: Specify the VF number. The VF number is valid when tx_st_vf_active is asserted. If the request is from VF[&lt;m&gt;,&lt;n&gt;], then bits[19:16] drive the PF number, &lt;m&gt;, and bits [31:20] drive the VF number, &lt;n&gt;. The tx_st_vf_active signal asserts in the same cycle as tx_st_sop and tx_st_valid.</li> </ul> <p>Refer to the <i>TLP Header and Data Alignment for the Avalon-ST TX and Avalon-ST RX Interfaces</i> for the layout of TLP headers and data.</p> |
| tx_st_sop         | Input     | Indicates first cycle of a TLP when asserted together with tx_st_valid.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| tx_st_eop         | Input     | Indicates last cycle of a TLP when asserted together with tx_st_valid.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| tx_st_ready       | Output    | <p>Indicates that the Transaction Layer is ready to accept data for transmission. The core deasserts this signal to throttle the data stream. tx_st_ready may be asserted during reset. The Application Layer should wait at least 2 clock cycles after the reset is released before issuing packets on the Avalon-ST TX interface. The reset_status signal can also be used to monitor when the IP core has come out of reset.</p> <p>If tx_st_ready is asserted by the Transaction Layer on cycle &lt;n&gt;, then &lt;n&gt; + readyLatency is a ready cycle, during which the Application Layer may assert tx_st_valid and transfer data.</p> <p>If tx_st_ready is deasserted by the Transaction Layer on cycle &lt;n&gt;, then the Application Layer must deassert tx_st_valid within the readyLatency number of cycles after cycle &lt;n&gt;.</p> <p>The readyLatency is 3 coreclkout_hip cycles.</p> <p>This interface is not strictly Avalon-ST compliant.</p>                                                                                                                                                                                                                                                                                                                                        |
| tx_st_valid       | Input     | <p>Clocks tx_st_data to the core when tx_st_ready is also asserted. Between tx_st_sop and the corresponding tx_st_eop, tx_st_valid must not be deasserted, except in response to tx_st_ready deassertion. When tx_st_ready deasserts in the middle of a packet, this signal must deassert exactly 3 coreclkout_hip cycles later. When tx_st_ready reasserts, and tx_st_data is in mid-TLP, this signal must reassert within 3 cycles. The figure entitled <i>Avalon-ST TX Interface tx_st_ready Deasserts</i> illustrates the timing of this signal.</p> <p>To facilitate timing closure, Intel recommends that you register both the tx_st_ready and tx_st_valid signals.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| tx_st_err         | Input     | <p>Forces an error on transmitted TLP. This signal is used to nullify a packet. To nullify a packet, assert this signal for 1 cycle with tx_st_eop. When a packet is nullified, the following packet should not be transmitted until the next clock cycle.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

*continued...*



| Signal                 | Direction | Description                                                                                                                                                                                                           |
|------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                        |           | <i>Note:</i> You cannot nullify a packet with 8 DW or less of data.                                                                                                                                                   |
| tx_st_parity[31:0]     | Input     | The IP core supports byte parity. Each bit represents odd parity of the associated byte of the tx_st_data bus. For example, bit[0] corresponds to tx_st_data[7:0], bit[1] corresponds to tx_st_data[15:8], and so on. |
| tx_st_vf_active H-Tile | Input     | When asserted, the transmitting TLP is for a VF. When deasserted, the transmitting TLP is for a PF. Valid when tx_st_sop is asserted.<br>Valid when multiple virtual functions are enabled.                           |

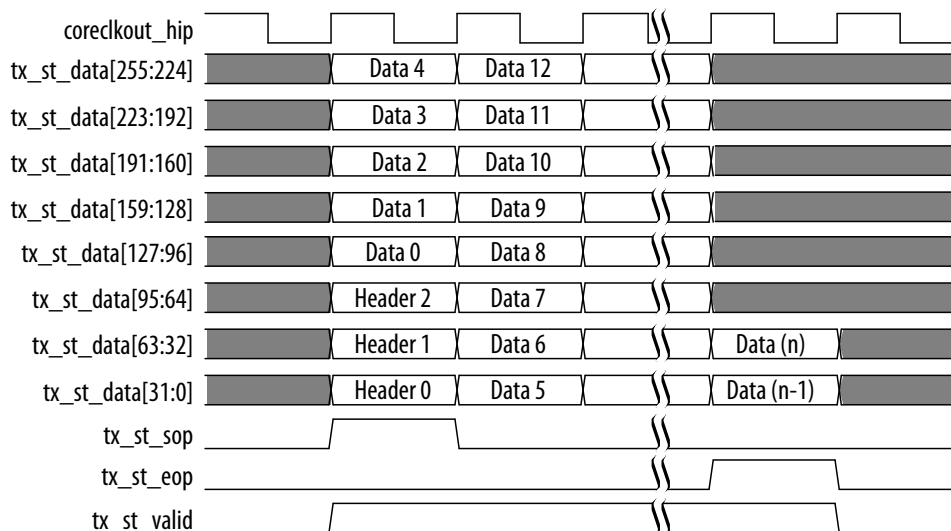
#### Related Information

- [Avalon-ST TX Interface tx\\_st\\_ready Deassertion](#) on page 70
- [Configuration Status Interface](#)  
For multifunction variants, the Configuration Status Interface specifies the settings of the various configuration register fields of the functions.
- [TLP Header and Data Alignment for the Avalon-ST RX and TX Interfaces](#) on page 58

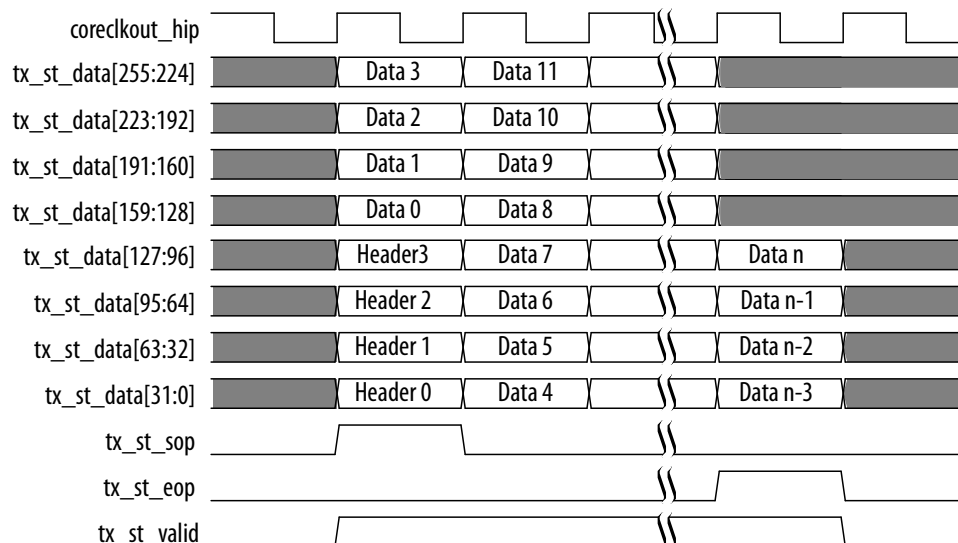
#### 6.1.4.1. Avalon-ST TX Three- and Four-Dword TLPs

These timing diagrams illustrate the layout of headers and data for the Avalon-ST TX interface.

**Figure 45. Avalon-ST TX Interface Cycle Definition for Three-Dword Header TLPs**



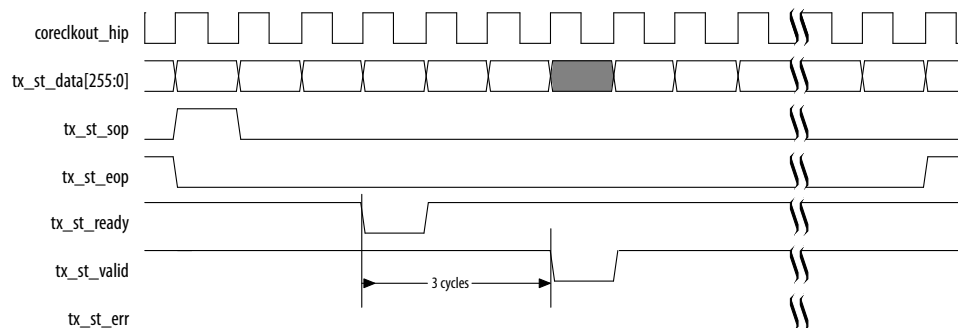
**Figure 46. Avalon-ST TX Interface Cycle Definition for Four-Dword Header TLPs**



#### 6.1.4.2. Avalon-ST TX Interface tx\_st\_ready Deassertion

This timing diagram illustrates the timing of the TX interface when the Stratix 10 Hard IP for PCI Express pauses the application layer by deasserting tx\_st\_ready. The timing diagram shows a readyLatency of 3 cycles. The application deasserts tx\_st\_valid after three cycles.

**Figure 47. Avalon-ST TX Interface tx\_st\_ready Deassertion**

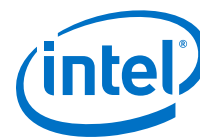


#### 6.1.4.3. Assertion and Deassertion of Avalon-ST TX Interface tx\_st\_valid

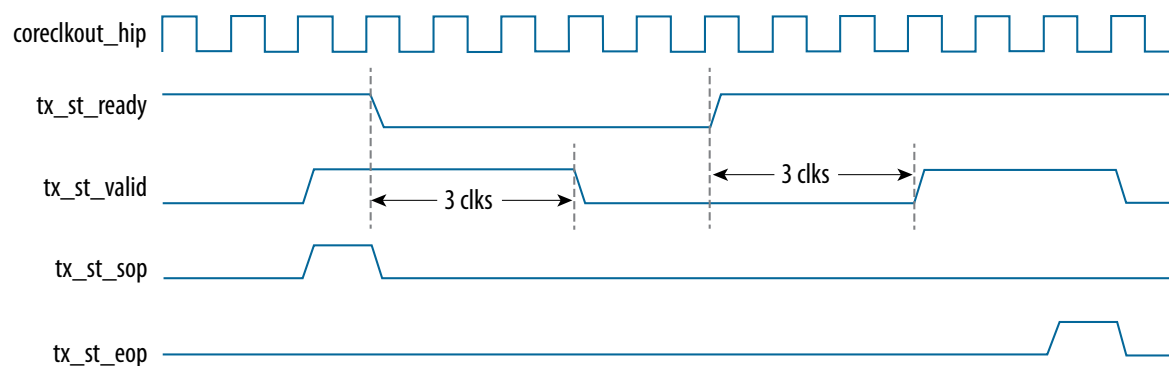
You must not deassert tx\_st\_valid between the tx\_st\_sop and tx\_st\_eop on a ready cycle. For the definition of a ready cycle, refer to *Avalon Interface Specifications*.

**Note:** This is an additional requirement for the PCIe Hard IP that is not compliant to the Avalon-ST standard.

The following timing diagram shows an example where tx\_st\_ready deasserts in the middle of a packet and then reasserts, causing tx\_st\_valid to also deassert and then reassert.



**Figure 48. Proper Assertion and Deassertion of tx\_st\_valid**



### 6.1.5. Avalon-ST 512-Bit TX Interface

The Application Layer transfers data to the Transaction Layer of the PCI Express IP core over the Avalon-ST TX interface. The 512-bit interface supports Gen3 x16 variants and uses a 250 MHz clock, simplifying timing closure.

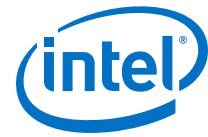
The 512-bit interface supports two locations for the beginning of a TLP, bit[0] and bit[256]. The interface supports multiple TLPs per cycle only when an end-of-packet cycle occurs in the lower 256 bits.

**Note:** This interface is not strictly Avalon-compliant because it supports two tx\_st\_sop signals and two tx\_st\_eop signals per cycle, and because there is a requirement of no ready and invalid cycle in the middle of a packet.

**Table 34. 512-Bit Avalon-ST TX Datapath**

| Signal              | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tx_st_data_i[511:0] | Input     | <p>Application Layer data for transmission. The Application Layer must provide a properly formatted TLP on the TX interface. Valid when the tx_st_valid_o signal is asserted. The mapping of message TLPs is the same as the mapping of Transaction Layer TLPs with 4 dword headers. The number of data cycles must be correct for the length and address fields in the header. Issuing a packet with an incorrect number of data cycles results in the TX interface hanging and becoming unable to accept further requests.</p> <p>For the TLP requester ID field, bits[31:16] in dword1 specify the following information:</p> <ul style="list-style-type: none"> <li>Bits[19:16]: Specify the PF number.</li> <li>Bits[31:20]: Specify the VF number. The VF number is valid when tx_st_vf_active is asserted. If the request is from VF[&lt;m&gt;, &lt;n&gt;], then bits[19:16] drive the PF number, &lt;m&gt;, and bits [31:20] drive the VF number, &lt;n&gt;. The tx_st_vf_active signal asserts in the same cycle as tx_st_sop and tx_st_valid.</li> </ul> <p>For TLPs with an end-of-packet cycle in the lower 256 bits, the 512-bit interface supports a start-of-packet cycle in the upper 256 bits. If a TLP starts on bit 256, bits [319:304] specify the completer ID for Completion packets.</p> |
| tx_st_sop_i[1:0]    | Input     | <p>Indicates the first cycle of a TLP when asserted in conjunction with the corresponding bit of tx_st_valid_o. The following encodings are defined:</p> <ul style="list-style-type: none"> <li>tx_st_sop_i[1]: When asserted indicates the start of a TLP in tx_st_data[511:256].</li> <li>tx_st_sop_i[0]: When asserted indicates the start of a TLP in tx_st_data[255:0].</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| tx_st_eop_i[1:0]    | Input     | <p>Indicates the end of a TLP when asserted in conjunction with the corresponding bit of tx_st_valid_o[1:0]. The following encodings are defined:</p> <ul style="list-style-type: none"> <li>tx_st_eop_i[1]: When asserted indicates the end of a TLP in tx_st_data[511:256].</li> <li>tx_st_eop_i[0]: When asserted indicates the end of a TLP in tx_st_data_i[255:0].</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| tx_st_ready_o       | Output    | <p>Indicates that the Transaction Layer is ready to accept data for transmission. The core deasserts this signal to apply backpressure to the data stream. The Application Layer should wait at least 2 clock cycles after the reset is</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

*continued...*



| Signal                      | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             |           | <p>released before issuing packets on the Avalon-ST TX interface. The Application Layer can monitor the reset_status signal to determine when the IP core has come out of reset.</p> <p>If tx_st_ready_o is asserted by the Transaction Layer on cycle &lt;n&gt;, then &lt;n&gt; + readyLatency is a ready cycle, during which the Application Layer may assert tx_st_valid_i and transfer data.</p> <p>If the Transaction Layer deasserts tx_st_ready_o on cycle &lt;n&gt;, then the Application Layer must deassert tx_st_valid_i within a readyLatency number of cycles after cycle &lt;n&gt;.</p> <p>The readyLatency is 3 coreclkout_hip cycles.</p>                                                                                                                                                                                  |
| tx_st_valid_i[1:0]          | Input     | <p>Clocks tx_st_data_i into the core on ready cycles. Between tx_st_sop_i and tx_st_eop_i, the tx_st_valid_i signal must not be deasserted in the middle of a TLP except in response to tx_st_ready deassertion. When tx_st_ready_o deasserts in the middle of a packet, this signal must deassert exactly 3 coreclkout_hip cycles later because the readyLatency is 3 cycles for this interface. When tx_st_ready_o reasserts, and tx_st_data is in mid-TLP, this signal must reassert on the next ready cycle. The figure entitled <i>Avalon-ST TX Interface tx_st_ready Deasserts</i> illustrates the timing of this signal. The behavior of this signal is the same for the 256- and 512-bit interfaces.</p> <p>To facilitate timing closure, Intel recommends that you register both the tx_st_ready_o and tx_st_valid_i signals.</p> |
| tx_st_err_i[1:0]            | Input     | <p>When asserted, indicates an error on transmitted TLP. This signal is asserted with tx_st_eop_i and nullifies a packet. The following encodings are defined:</p> <ul style="list-style-type: none"> <li>tx_st_err_i[1]: Specifies an error in tx_st_data[511:256].</li> <li>tx_st_err_i[0]: Specifies an error in tx_st_data[255:0].</li> </ul> <p><i>Note:</i> You cannot nullify a packet with 8 DW or less of data.</p>                                                                                                                                                                                                                                                                                                                                                                                                               |
| tx_st_parity_i[63:0]        | Input     | <p>Byte parity for tx_st_data_i. Bit 0 corresponds to tx_st_data_i[7:0], bit 1 corresponds to tx_st_data_i[15:8], and so on.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| tx_st_vf_active[1:0] H-Tile | Input     | <p>When asserted, the transmitting TLP is for a VF. When deasserted, the transmitting TLP is for a PF. Valid when tx_st_sop is asserted.</p> <p>Valid when multiple functions are enabled.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

#### Related Information

- [Avalon-ST TX Interface tx\\_st\\_ready Deassertion](#) on page 70
- [Configuration Status Interface](#)  
For multifunction variants, the Configuration Status Interface specifies the settings of the various configuration register fields of the functions.

### 6.1.6. TX Credit Interface

Before a TLP can be transmitted, flow control logic verifies that the link partner's RX port has sufficient buffer space to accept it. The TX Credit interface reports the link partner's available RX buffer space to the Application. It reports the space available in units called Flow Control credits.

Flow Control credits are defined for the following TLP categories:

- Posted transactions - TLPs that do not require a response
- Non-Posted transactions - TLPs that require a completion
- Completions - TLPs that respond to non-posted transactions

**Table 35. Categorization of Transactions Types**

| TLP Type                                                                               | Category    |
|----------------------------------------------------------------------------------------|-------------|
| Memory Write                                                                           | Posted      |
| Memory Read<br>Memory Read Lock                                                        | Non-posted  |
| I/O Read<br>I/O Write                                                                  | Non-posted  |
| Configuration Read<br>Configuration Write                                              | Non-posted  |
| Message                                                                                | Posted      |
| Completions<br>Completions with Data<br>Completion Locked<br>Completion Lock with Data | Completions |
| Fetch and Add AtomicOp                                                                 | Non-posted  |

**Table 36. Flow Control Credits**

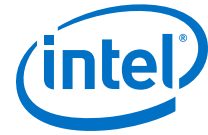
The following table translates flow control credits to dwords. The *PCI Express Base Specification* defines a dword as four bytes.

| Credit Type                 | Number of Dwords |
|-----------------------------|------------------|
| Header credit - completions | 4 dwords         |
| Header credit - requests    | 5 dwords         |
| Data credits                | 4 dwords         |

**Table 37. TX Flow Control Credit Interface**

The IP core provides TX credit information to the Application Layer. To optimize performance, the Application can perform credit-based checking before submitting requests for transmission, and reorder packets to improve throughput. The IP core always checks for sufficient TX credits before transmitting any TLP.

| Signal                   | Direction | Description                                                        |
|--------------------------|-----------|--------------------------------------------------------------------|
| tx_ph_cdts[7:0]          | Output    | Header credit net value for the flow control (FC) posted requests. |
| tx_pd_cdts[11:0]         | Output    | Data credit net value for the FC posted requests.                  |
| tx_nph_cdts[7:0]         | Output    | Header credit net value for the FC non-posted requests.            |
| tx_npd_cdts[11:0] L-Tile | Output    | Data credit net value for the FC non-posted requests.              |
| continued...             |           |                                                                    |



| Signal                                                                                                                                                                                  | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                                         |           | The tx_npd_cdts[11:0] is not available for H-Tile devices. You can monitor the non-posted header credits to determine if there is sufficient space to transmit the next non-posted data TLP.                                                                                                                                                                                                                                                                                                                      |
| tx_cplh_cdts[7:0]                                                                                                                                                                       | Output    | Header credit net value for the FC Completion. A value of 0 indicates infinite Completion header credits.                                                                                                                                                                                                                                                                                                                                                                                                         |
| tx_cp1d_cdts[11:0] L-Tile                                                                                                                                                               | Output    | Data credit net value for the FC Completion. A value of 0 indicates infinite Completion data credit.<br>The tx_cp1d_cdts[11:0] is not available for H-Tile devices. You can monitor the completion header credits to determine if there is sufficient space to transmit the next non-posted data TLP. The completion TLP size is either the request data size or 64 bytes when the completion is split.<br>For Root Ports or non peer-to-peer Endpoints as the link partner, assume that this credit is infinite. |
| tx_hdr_cdts_consumed[1:0]                                                                                                                                                               | Output    | Asserted for 1 coreclkout_hip cycle, for each header credit consumed by the application layer traffic. Note that credits the Hard IP consumes for internally generated Completions or Messages are not tracked in this signal.<br>For the Gen3 x16 512-bit interface, tx_hdr_cdts_consumed[1] is for the higher bus and tx_hdr_cdts_consumed[0] is for the lower bus.<br>There is only one tx_hdr_cdts_consumed signal for the 256-bit interface.                                                                 |
| tx_data_cdts_consumed[1:0]                                                                                                                                                              | Output    | Asserted for 1 coreclkout_hip cycle, for each data credit consumed. Note that credits the Hard IP consumes for internally generated Completions or Messages are not tracked in this signal.<br>For the Gen3 x16 512-bit interface, tx_data_cdts_consumed[1] is for the higher bus and tx_data_cdts_consumed[0] is for the lower bus.<br>There is only one tx_data_cdts_consumed signal for the 256-bit interface.                                                                                                 |
| tx_cdts_type[<n>-1:0]                                                                                                                                                                   | Output    | Specifies the credit type shown on the tx_cdts_data_value[1:0] bus. The following encodings are defined: <ul style="list-style-type: none"> <li>2'b00: Posted credits</li> <li>2'b01: Non-posted credits<sup>(4)</sup></li> <li>2'b10: Completion credits</li> <li>2'b11: Reserved</li> </ul> For the Gen3 x16 512-bit interface, tx_cdts_type[3:2] is for the higher bus and tx_cdts_type[1:0] is for the lower bus.                                                                                             |
| tx_cdts_data_value[3:0] L-Tile, 16 lanes<br>tx_cdts_data_value[1:0] L-Tile, 8 lanes or fewer<br>tx_cdts_data_value[1:0] H-Tile, 16 lanes<br>tx_cdts_data_value H-Tile, 8 lanes or fewer | Output    | For H-Tile: 1 = 2 data credits consumed; 0 = 1 data credit consumed.<br>For L-Tile: The value of tx_cdts_data_value+1 specifies the data credit consumed.<br>For both H- and L-Tiles: Only valid when tx_data_cdts_consumed asserts.<br>For the Gen3 x16 512-bit interface, tx_cdts_data_value[1] is for the higher bus and tx_cdts_data_value[0] is for the lower bus.                                                                                                                                           |

### 6.1.7. Interpreting the TX Credit Interface

The Intel Stratix 10 PCIe TX credit interface reports the number of Flow Control credits available.

<sup>(4)</sup> The PCIe IP core does not consume non-posted credits.

The following equation defines the available buffer space of the link partner:

$$RX\_buffer\_space = (credit\_limit - credits\_consumed) + released\_credits$$

where:

$$credits\_consumed = credits\_consumed_{application} + credits\_consumed_{PCIe\_IP\_core}$$

The hard IP core consumes a small number of posted credits for the following reasons:

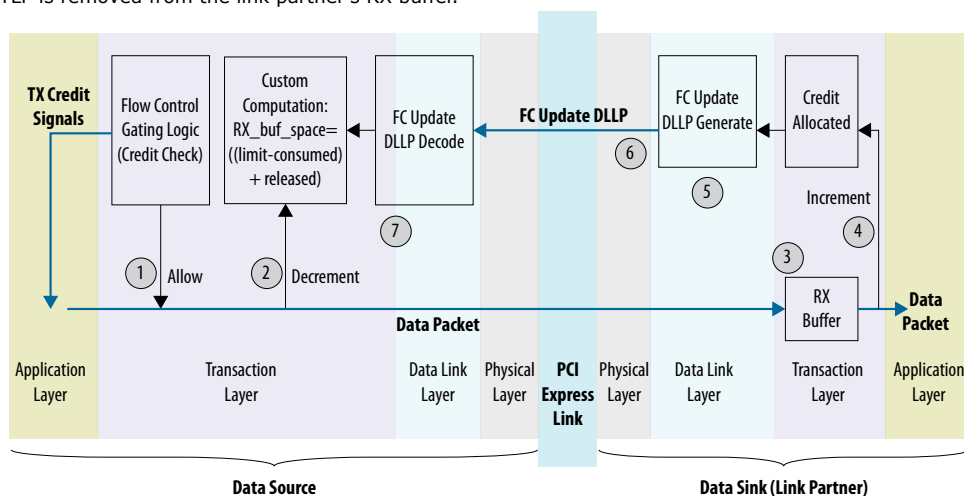
- To send completion TLPs for Configuration Requests targeting internal registers
- To transmit posted TLPs for error and interrupt Messages

The hard IP core does not report the internally consumed credits to the Application. Without knowing the hard IP core's internal usage, the Application cannot maintain a completely accurate count of Posted or Completion credits.

The hard IP core does not consume non-posted credits. Consequently, it is possible to maintain an accurate count of the available non-posted credits. Refer to the *TX Credit Adjustment Sample Code* for code that calculates the number of non-posted credits available to the Application. This RTL recovers the updated Flow Control credits from the remote link. It drives the value of the link partner's `RX_buffer_space` for non-posted header and data credits on `tx_nph_cdt`s and `tx_npd_cdt`s, respectively.

**Figure 49. Flow Control Credit Update Loop**

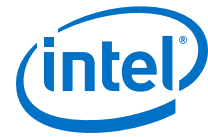
The following figure provides an overview of the steps to determine credits available and release credits after the TLP is removed from the link partner's RX buffer.



**Note:** To avoid a potential deadlock or performance degradation, the Application should check available credits before sending a TLP to the Stratix 10 Hard IP for PCI Express IP Core. If the Application cannot send Non-Posted packets, it must be allowed to send other types of packets.

### Related Information

[TX Credit Adjustment Sample Code](#) on page 177



## 6.1.8. Clocks

**Table 38. Clocks**

| Signal         | Direction | Description                                                                                                                                                                                                                                                      |                          |
|----------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|
| refclk         | Input     | This is the input reference clock for the IP core as defined by the <i>PCI Express Card Electromechanical Specification Revision 2.0</i> . The frequency is 100 MHz ±300 ppm. To meet the PCIe 100 ms wake-up time requirement, this clock must be free-running. |                          |
| coreclkout_hip | Output    | This clock drives the Data Link, Transaction, and Application Layers. For the Application Layer, the frequency depends on the data rate and the number of lanes as specified in the table                                                                        |                          |
|                |           | Data Rate                                                                                                                                                                                                                                                        | coreclkout_hip Frequency |
|                |           | Gen1 x1, x2, x4, x8, and x16                                                                                                                                                                                                                                     | 125 MHz                  |
|                |           | Gen2 x1, x2, x4, and x8,                                                                                                                                                                                                                                         | 125 MHz                  |
|                |           | Gen2 x16                                                                                                                                                                                                                                                         | 250 MHz                  |
|                |           | Gen3 x1, x2, and x4                                                                                                                                                                                                                                              | 125 MHz                  |
|                |           | Gen3 x8                                                                                                                                                                                                                                                          | 250 MHz                  |
|                |           |                                                                                                                                                                                                                                                                  |                          |

## 6.1.9. Update Flow Control Timer and Credit Release

The IP core releases credits on a per-clock-cycle basis as it removes TLPs from the local RX Buffer.

## 6.1.10. Function-Level Reset (FLR) Interface

The function-level reset (FLR) interface can reset the individual SR-IOV functions.

**Table 39. Function-Level Reset (FLR) Interface**

| Signal                         | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| f1r_pf_active_o[<n>-1:0]H-Tile | Output    | The SR-IOV Bridge asserts f1r_pf_active_o when bit 15 of the PCIe Device Control Register is set. Bit 15 is the FLR field. Once asserted, the f1r_pf_active_o signal remains high until the Application Layer sets f1r_pf_done_i high for the associated function.<br>The Application Layer must perform actions necessary to clear any pending transactions associated with the function being reset. The Application Layer must assert f1r_pf_done_i to indicate it has completed the FLR actions and is ready to re-enable the PF. |
| f1r_pf_done_i[<n>-1:0]H-Tile   | Input     | <n> is the number of PFs.<br>When asserted for one or more cycles, indicates that the Application Layer has completed resetting all the logic associated with the PF. Bit 0 is for PF0. Bit 1 is for PF1, and so on. Users decode the FLR write to PF register through the cfg write broadcast bus to know the PF should be FLR-ed. When f1r_pf_active_o is asserted, the Application Layer must assert f1r_completed within 100 microseconds to re-enable the function.                                                              |
| f1r_rcvd_vf H-Tile             | Output    | The SR-IOV Bridge asserts this output port for one cycle when a 1 is being written into the PCIe Device Control Register FLR field, bit[15], of the of a VF. f1r_rcvd_pf_num and f1r_rcvd_vf_num drive PF number and the VF offset associated with the Function being reset.                                                                                                                                                                                                                                                          |

*continued...*

| Signal                                                                 | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------------------------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                        |           | The Application Layer responds to a pulse on this output by clearing any pending transactions associated with the VF being reset. It then asserts <code>f1r_completed_vf</code> to indicate that it has completed the FLR actions and is ready to re-enable the VF.                                                                                                                             |
| <code>f1r_rcvd_pf_num[&lt;n&gt;-1:0]</code> H-Tile                     | Output    | When <code>f1r_rcvd_vf</code> is asserted, this output specifies the PF number associated with the VF being reset.<br><n> is the PF number.                                                                                                                                                                                                                                                     |
| <code>f1r_rcvd_vf_num[log<sub>2</sub>&lt;n&gt;-1:0]</code> H-Tile      | Output    | When <code>f1r_rcvd_vf</code> is asserted, this output specifies the VF number offset associated with the VF being reset.<br><n> is the number of VFs.                                                                                                                                                                                                                                          |
| <code>f1r_completed_vf</code> H-Tile                                   | Input     | When asserted, indicates that the Application Layer has completed resetting all the logic associated with <code>f1r_completed_vf_num[&lt;n&gt;-1:0]</code> .<br>When <code>f1r_active_vf[&lt;n&gt;]</code> asserts, the Application Layer it must assert the corresponding bit of <code>f1r_completed_vf</code> within 100 microseconds to re-enable the VF.<br><n> is the total number of VFs. |
| <code>f1r_completed_pf_num[&lt;n&gt;-1:0]</code> H-Tile                | Input     | When <code>f1r_completed_vf</code> is asserted, this input specifies the PF number associated with the VF that has completed .<br><n> is the number of PFs.                                                                                                                                                                                                                                     |
| <code>f1r_completed_vf_num[log<sub>2</sub>&lt;n&gt;-1:0]</code> H-Tile | Input     | When <code>f1r_completed_vf</code> is asserted, this input specifies the VF number associated with the VF that has completed its FLR.<br><n> is the total number of VFs.                                                                                                                                                                                                                        |

Figure 50. FLR Interface Timing Diagram for Physical Functions

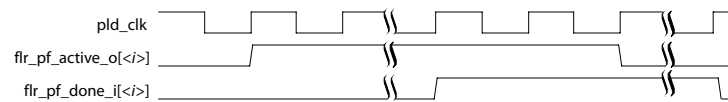
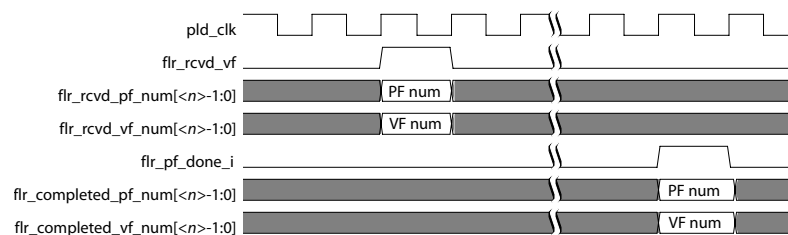


Figure 51. FLR Interface Timing Diagram for Virtual Functions



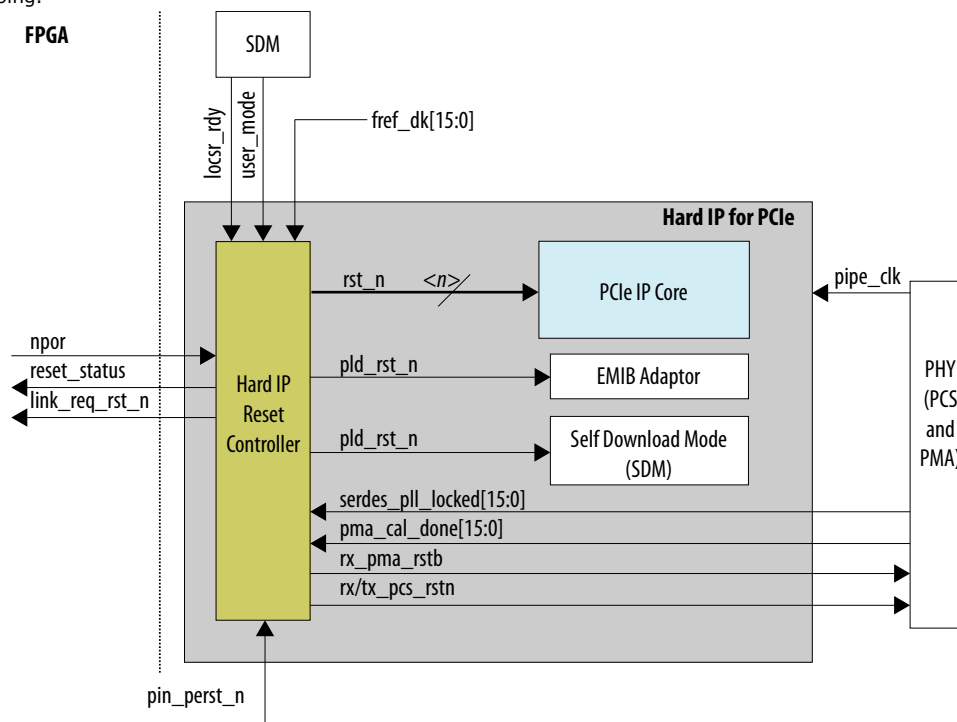
### 6.1.11. Resets

An embedded hard reset controller (HRC) generates the reset for PCIe core logic, PMA, PCS, and Application. To support the 100 ms PCIe wake-up time, the embedded reset controller connects to the SDM. This connection allows the FPGA periphery to configure and begin operation before the FPGA fabric is programmed.

The reset logic requires a free running clock that is stable and available to the IP core at configuration time.

**Figure 52. Intel Stratix 10 Hard IP for PCI Express Reset Controller**

The signals shown in the Hard IP for PCIe Reset Controller are implemented in hard logic and not available for probing.



**Table 40. Resets**

| Signal            | Direction | Description                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| currentspeed[1:0] | Output    | Indicates the current speed of the PCIe module. The following encodings are defined: <ul style="list-style-type: none"> <li>2'b00: Undefined</li> <li>2'b01: Gen1</li> <li>2'b10: Gen2</li> <li>2'b11: Gen3</li> </ul>                                                                                                                                                         |
| npor              | Input     | The Application Layer drives this active low reset signal. npor resets the entire IP core, PCS, PMA, and PLLs. npor should be held for a minimum of 20 ns. Gen3 x16 variants, should hold npor for at least 10 cycles. This signal is edge, not level sensitive; consequently, a low value on this signal does not hold custom logic in reset. This signal cannot be disabled. |
| pin_perst         | Input     | Active low reset from the PCIe reset pin of the device. Resets the datapath and control registers.                                                                                                                                                                                                                                                                             |
| pld_clk_inuse     | Output    | This reset signal has the same effect as reset_status. This signal is provided for backwards compatibility with Arria 10 devices.                                                                                                                                                                                                                                              |
| pld_core_ready    | Input     | When asserted, indicates that the Application Layer is ready. The IP core can releases reset after this signal is asserted.                                                                                                                                                                                                                                                    |

*continued...*

| Signal            | Direction | Description                                                                                                                                                                                                                                                                 |
|-------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| reset_status      | Output    | Active high reset status. When high, indicates that the IP core is not ready for usermode. reset_status is deasserted only when npor is deasserted and the IP core is not in reset. Use reset_status to drive the reset of your application. Synchronous to coreclkout_hip. |
| clr_st            | Output    | clr_st has the same functionality as reset_status. It is provided for backwards compatibility with previous device families.                                                                                                                                                |
| serdes_pll_locked | Output    | When asserted, indicates that the PLL that generates coreclkout_hip is locked. In pipe simulation mode this signal is always asserted.                                                                                                                                      |

## 6.1.12. Interrupts

### 6.1.12.1. MSI and Legacy Interrupts

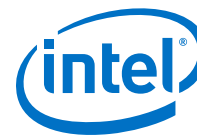
Message Signaled Interrupts (MSI) interrupts are signaled on the PCI Express link using a single dword Memory Write TLP.

**Table 41. MSI and Legacy Interrupts**

| Signal                       | Direction | Description                                                                                                                                                                                                                                                                                                         |
|------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| app_msi_req                  | Input     | Application Layer MSI request. Assertion causes an MSI posted write TLP to be generated based on the MSI configuration register values and the app_msi_tc and app_msi_num input ports. Deasserts when app_msi_ack acknowledges the request.                                                                         |
| app_msi_ack                  | Output    | The IP core acknowledges the app_msi_req request. Asserts for 1 cycle to acknowledge the Application Layer's request for an MSI interrupt. The app_msi_req request deasserts as soon as it receives this signal.                                                                                                    |
| app_msi_tc[2:0]              | Input     | Application Layer MSI traffic class. This signal indicates the traffic class used to send the MSI (unlike INTX interrupts, any traffic class can be used to send MSIs).                                                                                                                                             |
| app_msi_num[4:0]             | Input     | MSI number of the Application Layer. The application uses the app_msi_num bus to indicate the offset between the base message data and the MSI to send. When multiple message mode is enabled, it sets the lower five bits of the MSI Data register. Only bits that the MSI Message Control register enables apply. |
| app_int_sts[3:0] H-Tile      | Input     | Controls legacy interrupts. Assertion of app_int_sts causes an Assert_INTx message TLP to be generated and sent upstream. Deassertion of app_int_sts causes a Deassert_INTx message TLP to be generated and sent upstream.<br>When you enable multiple PFs, bit 0 is for PF0, bit 1 is for PF1, and so on.          |
| app_msi_func_num[1:0] H-Tile | Input     | Specifies the function number requesting an MSI transmission.                                                                                                                                                                                                                                                       |
| app_err_func_num[1:0] H-Tile | Input     | Specifies the function number that is asserting the app_err_valid signal.                                                                                                                                                                                                                                           |

### 6.1.13. Control Shadow Interface for SR-IOV

The control shadow interface provides access to the current settings for some of the VF Control Register fields in the PCI and PCI Express Configuration Spaces located in the SR-IOV Bridge. This interface is only available for H-Tile devices.



Use this interface for the following purposes:

- To monitor specific VF registers using the `ctl_shdw_update` output and the associated output signals defined below.
- To monitor all VF registers using the `ctl_shdw_req_all` input to request a full scan of the register fields for all active VFs.

| Signal                                      | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ctl_shdw_update</code>                | Output    | The SR-IOV Bridge asserts this output for 1 clock cycle when one or more of the register fields being monitored is updated. The <code>ctl_shdw_cfg</code> outputs drive the new values. <code>ctl_shdw_pf_num</code> , <code>ctl_shdw_vf_num</code> , and <code>ctl_shdw_vf_active</code> identify the VF and its PF.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>ctl_shdw_pf_num[&lt;n&gt;-1:0]</code> | Output    | Identifies the PF whose register settings are on the <code>ctl_shdw_cfg</code> outputs. When the function is a VF, this input specifies the PF number to which the VF is attached.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>ctl_shdw_vf_active</code>             | Output    | When asserted, indicates that the function whose register settings are on the <code>ctl_shdw_cfg</code> outputs is a VF. <code>ctl_shdw_vf_num</code> drives the VF number offset.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>ctl_shdw_vf_num[10:0]</code>          | Output    | Identifies the VF number offset of the VF whose register settings are on <code>ctl_shdw_cfg</code> outputs when <code>ctl_shdw_vf_active</code> is asserted. Its value ranges from 0-(<n>-1), where <n> is the number of VFs attached to the associated PF.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>ctl_shdw_cfg[6:0]</code>              | Output    | When <code>ctl_shdw_update</code> is asserted, this output provides the current settings of the register fields of the associated function. The bits specify the following register fields: <ul style="list-style-type: none"> <li>• [0]: Bus Master Enable, bit[2] of the PCI Command Register.</li> <li>• [1]: MSI-X function mask field, bit[14] of the MSI-X Message Control register</li> <li>• [2]: MSI-X enable field, bit[15] of the MSI-X Message Control register</li> <li>• [4:3]: TPH Steering Tag (ST) Mode Select field, bits[1:0] of the TPH Requester Control register</li> <li>• [5]: TPH Requester Enable field, bit[8] of the TPH Requester Control register</li> <li>• [6]: Enable field, bit[15] of the ATS Control register</li> </ul> |
| <code>ctl_shdw_req_all</code>               | Input     | When asserted, requests a complete scan of the register fields being monitored for all active Functions. When the <code>ctl_shdw_req_all</code> input is asserted, the SR-IOV bridge cycles through each VF. It provides the current values of all register fields. If a Configuration Write occurs during a scan, the SR-IOV Bridge interrupts the scan to output the new setting. It then resumes the scan, continuing sequentially from the updated VF setting.<br><br>The SR-IOV Bridge checks the state of <code>ctl_shdw_req_all</code> at the end of each scan cycle. It starts a new scan cycle if this input is asserted. Connect this input to logic 1 to scan the functions continuously.                                                         |

#### 6.1.14. Transaction Layer Configuration Space Interface

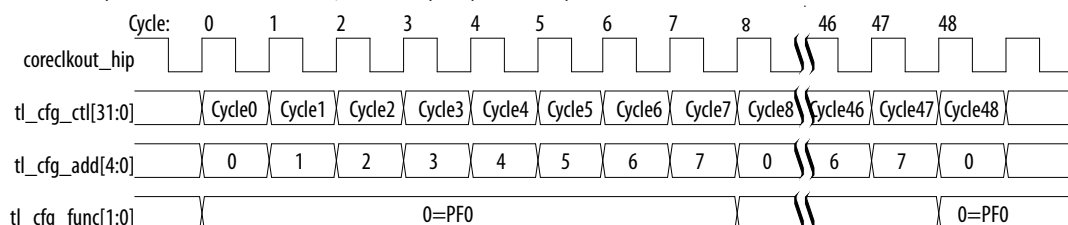
The Transaction Layer (TL) bus provides a subset of the information stored in the Configuration Space. Use this information in conjunction with the `app_err*` signals to understand TLP transmission problems.

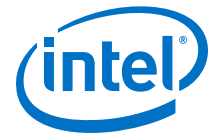
**Table 42. Configuration Space Signals**

| Signal                                           | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tl_cfg_add[3:0] H-Tile<br>tl_cfg_add[4:0] L-Tile | Output    | Address of the TLP register. This signal is an index indicating which Configuration Space register information is being driven onto tl_cfg_ctl. Refer to <i>H-Tile Multiplexed Configuration Register Information Available on tl_cfg_ctl</i> or <i>E-Tile Multiplexed Configuration Register Information Available on tl_cfg_ctl</i> for the available information as appropriate.<br><br>Address of the TLP register. This signal is an index indicating which Configuration Space register information is being driven onto tl_cfg_ctl. Refer to <i>L-Tile Multiplexed Configuration Register Information Available on tl_cfg_ctl</i> for the available information. |
| tl_cfg_ctl[31:0]                                 | Output    | The tl_cfg_ctl signal is multiplexed and contains a subset of contents of the Configuration Space registers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| tl_cfg_func[1:0]                                 | Output    | Specifies the function whose Configuration Space register values are being driven tl_cfg_ctl[31:0]. The following encodings are defined: <ul style="list-style-type: none"> <li>2'b00: Physical Function (PF0)</li> <li>2'b01: PF1 for H-Tile, reserved for L-Tile</li> <li>2'b10: PF2 for H-Tile, reserved for L-Tile</li> <li>2'b11: PF3 for H-Tile, reserved for L-Tile</li> </ul>                                                                                                                                                                                                                                                                                   |
| app_err_hdr[31:0]                                | Input     | Header information for the error TLP. Four, 4-byte transfers send this information to the IP core.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| app_err_info[10:0]                               | Input     | The Application can optionally provide the following information: <ul style="list-style-type: none"> <li>app_err_info[0]: Malformed TLP</li> <li>app_err_info[1]: Receiver Overflow</li> <li>app_err_info[2]: Unexpected Completion</li> <li>app_err_info[3]: Completer Abort</li> <li>app_err_info[4]: Completer Timeout</li> <li>app_err_info[5]: Unsupported Request</li> <li>app_err_info[6]: Poisoned TLP Received</li> <li>app_err_info[7]: AtomicOp Egress Blocked</li> <li>app_err_info[8]: Uncorrectable Internal Error</li> <li>app_err_info[9]: Correctable Internal Error</li> <li>app_err_info[10]: Advisory Non-Fatal Error</li> </ul>                    |
| app_err_valid                                    | Input     | When asserted, indicates that the data on app_err_info[10:0] is valid. For multi-function variants, the app_err_func_num specifies the function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

**Figure 53. Configuration Space Register Access Timing**

Information on the Transaction Layer (TL) bus is time-division multiplexed (TDM). When tl\_cfg\_func[1:0] = 2'b00, tl\_cfg\_ctl[31:0] drives PF0 Configuration Space register values for eight consecutive cycles. The next 40 cycles are reserved. Then, the 48-cycle pattern repeats.





**Table 43. L-Tile Multiplexed Configuration Register Information Available on tl\_cfg\_ctl**

| TD<br>M | 31                                                                                                      | 24 | 23                                                                                 | 16 | 15                                                                                              | 8 | 7                                                                                                                                                                 | 0 |
|---------|---------------------------------------------------------------------------------------------------------|----|------------------------------------------------------------------------------------|----|-------------------------------------------------------------------------------------------------|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|
| 0       | [28:24]: Device Number<br>[29]: Relax order enable<br>[30]: No snoop enable<br>[31]: IDO request enable |    | Bus Number                                                                         |    | [13:8]: Auto negotiation link width<br>[14]: IDO completion enable<br>[15]: Memory space enable |   | Device Control<br>[2:0]: Max payload size<br>[5:3]: Max rd req size<br>[6]: Extended tag enable<br>[7]: Bus master enable                                         |   |
| 1       | [28:24]AER IRQ Msg num<br>[29]: cfg_send_corr_err<br>[30]: cfg_send_nf_err<br>[31]: cfg_send_f_rr       |    | [16]: RCB cntl<br>[17]: cfg_pm_no_soft_rst<br>[23:18]: auto negotiation link width |    | [12:8]: PCIe cap interrupt msg num<br>[13]: interrupt disable<br>[15:14]: Reserved.             |   | [1:0]: Sys power ind. cntl<br>[3:2]: Sys atten ind cntl<br>[4]: Sys power cntl<br>[7:5]: Reserved                                                                 |   |
| 2       | Index of start VF[6:0]                                                                                  |    | Num VFs                                                                            |    | [4:1]: STU<br>[11:8]: ATS<br>[15:12]: auto negotiation link speed                               |   | [0]: VF enable<br>[1]: TPH enable<br>[3:2]: TPH ST mode[1:0]<br>[4]: Atomic request enable<br>[5]: ARI forward enable<br>[6]: ATS cache enable<br>[7]: ATS STU[0] |   |
| 3       | MSI Address Lower                                                                                       |    |                                                                                    |    |                                                                                                 |   |                                                                                                                                                                   |   |
| 4       | MSI Address Upper                                                                                       |    |                                                                                    |    |                                                                                                 |   |                                                                                                                                                                   |   |
| 5       | MSI Mask                                                                                                |    |                                                                                    |    |                                                                                                 |   |                                                                                                                                                                   |   |
| 6       | MSI Data                                                                                                |    |                                                                                    |    | Reserved                                                                                        |   | [0]: MSI enable<br>[1]: 64-bit MSI<br>[4:2]: Multiple MSI enable<br>[5]: MSI-X enable<br>[6]: MSI-X func mask                                                     |   |
| 7       | Reserved                                                                                                |    |                                                                                    |    | [5:0]: Auto negotiation link width<br>[9:6]: Auto negotiation link speed                        |   |                                                                                                                                                                   |   |

**Table 44. H-Tile Multiplexed Configuration Register Information Available on tl\_cfg\_ctl**

Information on the Transaction Layer (TL) bus is time-division multiplexed (TDM). The TL bus displays the information for each of the 4 PFs and their associated VFs in 10 consecutive cycles. Then, the 40-cycle pattern repeats.

| TD<br>M      | 31                                                                                             | 24 | 23         | 16 | 15                                                                                                                                                                                                  | 8                                                                                                                 | 7                                                                                                      | 0 |
|--------------|------------------------------------------------------------------------------------------------|----|------------|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|---|
| 0            | [28:24]: Device number<br>[29]: Relaxed Ordering en<br>[30]: No Snoop en<br>[31]: (IDO) req en |    | Bus Number |    | [ 8]: unsupported_req_rpt_en<br>[ 9]: corr_err_rpt_en<br>[10]: nonfatal_err_rpt_en<br>[11]: fatal_err_rpt_en<br>[12]: serr_err<br>[13]: perr_en<br>[14]: IDO completion en<br>[15]: Memory space en | Device Control<br>[2:0]: Max payload size<br>[5:3]: Max rd req size<br>[6]: Extended tag en<br>[7]: Bus master en |                                                                                                        |   |
| 1            | Number of VFs[15:0]                                                                            |    |            |    | [12:8]: PCIe Capability<br>IRQ Msg Num<br>[13]: IRQ disable<br>[14]: Rd Cmpl Boundary (RCB) cntl                                                                                                    |                                                                                                                   | [1:0]: System ind power cntl<br>[3:2]: Sys atten ind cntl<br>[4]: System power cntl<br>[7:5]: Reserved |   |
| continued... |                                                                                                |    |            |    |                                                                                                                                                                                                     |                                                                                                                   |                                                                                                        |   |

| TD<br>M | 31                                                                                         | 24 | 23 | 16 | 15                                                                                                 | 8 | 7                                                                                                                  | 0 |
|---------|--------------------------------------------------------------------------------------------|----|----|----|----------------------------------------------------------------------------------------------------|---|--------------------------------------------------------------------------------------------------------------------|---|
|         |                                                                                            |    |    |    | [15]: pm_no_soft_rst                                                                               |   |                                                                                                                    |   |
| 2       | [16]: Reserved<br>[27:17]: Index of Start VF[10:0]<br>[31:28]: Auto negotiation link speed |    |    |    | [8]: ATS cache en<br>[13:9]: ATS STU[4:0]<br>[15:14]: Reserved                                     |   | [0]: VF en<br>[2:1]: TPH en<br>[5:3]: TPH ST mode<br>[6:] Atomic req en<br>[7]: ARI forward enable                 |   |
| 3       | MSI Address Lower                                                                          |    |    |    |                                                                                                    |   |                                                                                                                    |   |
| 4       | MSI Address Upper                                                                          |    |    |    |                                                                                                    |   |                                                                                                                    |   |
| 5       | MSI Mask                                                                                   |    |    |    |                                                                                                    |   |                                                                                                                    |   |
| 6       | MSI Data                                                                                   |    |    |    | [12:8]: AER IRQ Msg Num<br>[13]: cfg_send_cor_err<br>[14]: cfg_send_nf_err<br>[15]: cfg_send_f_err |   | [0]: MSI en<br>[1]: 64-bit MSI<br>[4:2]: Multiple MSI en<br>[5]: MSI-X en<br>[6]: MSI-X func mask<br>[7]: Reserved |   |
| 7       | AER Uncorrectable Error Mask                                                               |    |    |    |                                                                                                    |   |                                                                                                                    |   |
| 8       | AER Correctable Error Mask                                                                 |    |    |    |                                                                                                    |   |                                                                                                                    |   |
| 9       | AER Uncorrectable Error Severity                                                           |    |    |    |                                                                                                    |   |                                                                                                                    |   |

### 6.1.15. Configuration Extension Bus Interface

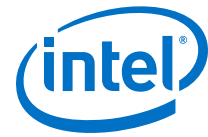
Use the Configuration Extension Bus to add capability structures to the IP core's internal Configuration Spaces. Configuration TLPs with a destination register byte address of 0xC00 and higher route to the Configuration Extension Bus interface. Report the Completion Status Successful Completion (SC) on the Configuration Extension Bus. The IP core then generates a Completion to transmit on the link.

Use the app\_err\_info[8] signal included in the *Transaction Layer Configuration Space Interface* to report uncorrectable internal errors.

**Note:** The Configuration Extension Bus interface is not available if the parameter **Enable SR-IOV Support** under the **Multifunction and SR-IOV System Settings** tab is set to **On**.

**Note:** The IP core does not apply ID-based Ordering (IDO) bits to the internally generated Completions.

| Signal         | Direction | Description                                                                                                                       |
|----------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------|
| ceb_req        | Output    | When asserted, indicates a valid Configuration Extension Bus access cycle. Deasserted when ceb_ack is asserted.                   |
| ceb_ack        | Input     | Asserted to acknowledge ceb_req. The Application must implement this logic.                                                       |
| ceb_addr[11:0] | Output    | Address bus to the external register block. The width of the address bus is the value you select for the CX_LBC_EXT_AW parameter. |
| ceb_din[31:0]  | Input     | Read data.                                                                                                                        |
| continued...   |           |                                                                                                                                   |



| Signal                     | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ceb_cdm_convert_data[31:0] | Input     | Acts as a mask. If the value of a bit is 1, overwrite the value of the VF register with the value of the corresponding PF register at that bit position. If the value is 0, do not overwrite the bit.<br>This signal is available for H-Tile only.                                                                                                                                                                                                           |
| ceb_dout[31:0]             | Output    | Data to be written.                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| ceb_wr[3:0]                | Output    | Indicates the configuration register access type, read or write. For writes, CEB_wr also indicates the byte enables: The following encodings are defined: <ul style="list-style-type: none"> <li>4'b000: Read</li> <li>4'b0001: Write byte 0</li> <li>4'b0010: Write byte 1</li> <li>4'b0100: Write byte 2</li> <li>4'b1000: Write byte 3</li> <li>4'b1111: Write all bytes.</li> </ul> Combinations of byte enables, for example, 4'b 0101b are also valid. |
| ceb_vf_num[10:0]           | Output    | The VF of the current CEB access.<br>This signal is available for H-Tile only.                                                                                                                                                                                                                                                                                                                                                                               |
| ceb_vf_active              | Output    | When asserted, indicates a VF is active.<br>This signal is available for H-Tile only.                                                                                                                                                                                                                                                                                                                                                                        |
| ceb_func_num[1:0]          | Output    | The PF number of the current CEB access.<br>This signal is available for H-Tile only.                                                                                                                                                                                                                                                                                                                                                                        |

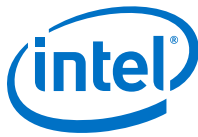
### 6.1.16. Hard IP Status Interface

Hard IP Status: This optional interface includes the following signals that are useful for debugging, including: link status signals, interrupt status signals, TX and RX parity error signals, correctable and uncorrectable error signals.

**Table 45. Hard IP Status Interface**

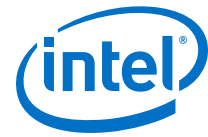
| Signal                                                                               | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------------------------------------------------------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| derr_cor_ext_rcv                                                                     | Output    | When asserted, indicates that the RX buffer detected a 1-bit (correctable) ECC error. This is a pulse stretched output.                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| derr_cor_ext_rpl                                                                     | Output    | When asserted, indicates that the retry buffer detected a 1-bit (correctable) ECC error. This is a pulse stretched output.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| derr_rpl                                                                             | Output    | When asserted, indicates that the retry buffer detected a 2-bit (uncorrectable) ECC error. This is a pulse stretched output.                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| derr_uncor_ext_rcv                                                                   | Output    | When asserted, indicates that the RX buffer detected a 2-bit (uncorrectable) ECC error. This is a pulse stretched output.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| int_status[10:0](H-Tile)<br>int_status[7:0] (L-Tile)<br>int_status_pf1[7:0] (L-Tile) | Output    | The int_status[3:0] signals drive legacy interrupts to the application (for H-Tile).<br>The int_status[10:4] signals provide status for other interrupts (for H-Tile).<br>The int_status[3:0] signals drive legacy interrupts to the application for PF0 (for L-Tile).<br>The int_status[7:4] signals provide status for other interrupts for PF0 (for L-Tile).<br>The int_status_pf1[3:0] signals drive legacy interrupts to the application for PF1 (for L-Tile).<br>The int_status_pf1[7:4] signals provide status for other interrupts for PF1 (for L-Tile). |

*continued...*



| Signal                              | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                     |           | <p>The following signals are defined:</p> <ul style="list-style-type: none"> <li>• <code>int_status[0]</code>: Interrupt signal A</li> <li>• <code>int_status[1]</code>: Interrupt signal B</li> <li>• <code>int_status[2]</code>: Interrupt signal C</li> <li>• <code>int_status[3]</code>: Interrupt signal D</li> <li>• <code>int_status[4]</code>: Specifies a Root Port AER error interrupt. This bit is set when the <code>cfg_aer_rc_err_msi</code> or <code>cfg_aer_rc_err_int</code> signal asserts. This bit is cleared when software writes 1 to the register bit or when <code>cfg_aer_rc_err_int</code> is deasserts.</li> <li>• <code>int_status[5]</code>: Specifies the Root Port PME interrupt status. It is set when <code>cfg_pme_msi</code> or <code>cfg_pme_int</code> asserts. It is cleared when software writes a 1 to clear or when <code>cfg_pme_int</code> deasserts.</li> <li>• <code>int_status[6]</code>: Asserted when a hot plug event occurs and Power Management Events (PME) are enabled. (PMEs are typically used to revive the system or a function from a low power state.)</li> <li>• <code>int_status[7]</code>: Specifies the hot plug event interrupt status.</li> <li>• <code>int_status[8]</code>: Specifies the interrupt status for the Link Autonomous Bandwidth Status register. H-Tile only.</li> <li>• <code>int_status[9]</code>: Specifies the interrupt status for the Link Bandwidth Management Status register. H-Tile only.</li> <li>• <code>int_status[10]</code>: Specifies the interrupt status for the Link Equalization Request bit in the Link Status register. H-Tile only.</li> </ul> |
| <code>int_status_common[2:0]</code> | Output    | <p>Specifies the interrupt status for the following registers. When asserted, indicates that an interrupt is pending:</p> <ul style="list-style-type: none"> <li>• <code>int_status_common[0]</code>: Autonomous bandwidth status register.</li> <li>• <code>int_status_common[1]</code>: Bandwidth management status register.</li> <li>• <code>int_status_common[2]</code>: Link equalization request bit in the link status register.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>lane_act[4:0]</code>          | Output    | <p>Lane Active Mode: This signal indicates the number of lanes that configured during link training. The following encodings are defined:</p> <ul style="list-style-type: none"> <li>• 5'b0 0001: 1 lane</li> <li>• 5'b0 0010: 2 lanes</li> <li>• 5'b0 0100: 4 lanes</li> <li>• 5'b0 1000: 8 lanes</li> <li>• 5'b1 0000: 16 lanes</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>link_up</code>                | Output    | When asserted, the link is up.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>ltssmstate[5:0]</code>        | Output    | <p>Link Training and Status State Machine (LTSSM) state: The LTSSM state machine encoding defines the following states:</p> <ul style="list-style-type: none"> <li>• 6'h00 - Detect.Quiet</li> <li>• 6'h01 - Detect.Active</li> <li>• 6'h02 - Polling.Active</li> <li>• 6'h03 - Polling.Compliance</li> <li>• 6'h04 - Polling.Configuration</li> <li>• 6'h05 - PreDetect.Quiet</li> <li>• 6'h06 - Detect.Wait</li> <li>• 6'h07 - Configuration.Linkwidth.Start</li> <li>• 6'h08 - Configuration.Linkwidth.Accept</li> <li>• 6'h09 - Configuration.Lanenum.Wait</li> <li>• 6'h0A - Configuration.Lanenum.Accept</li> <li>• 6'h0B - Configuration.Complete</li> <li>• 6'h0C - Configuration.Idle</li> <li>• 6'h0D - Recovery.RcvrLock</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

*continued...*



| Signal     | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |           | <ul style="list-style-type: none"> <li>6'h0E - Recovery.Speed</li> <li>6'h0F - Recovery.RcvrCfg</li> <li>6'h10 - Recovery.Idle</li> <li>6'h20 - Recovery.Equalization Phase 0</li> <li>6'h21 - Recovery.Equalization Phase 1</li> <li>6'h22 - Recovery.Equalization Phase 2</li> <li>6'h23 - Recovery.Equalization Phase 3</li> <li>6'h11 - L0</li> <li>6'h12 - L0s</li> <li>6'h13 - L123.SendIdle</li> <li>6'h14 - L1.Idle</li> <li>6'h15 - L2.Idle</li> <li>6'h16 - L2.TransmitWake</li> <li>6'h17 - Disabled.Entry</li> <li>6'h18 - Disabled.Idle</li> <li>6'h19 - Disabled</li> <li>6'h1A - Loopback.Entry</li> <li>6'h1B - Loopback.Active</li> <li>6'h1C - Loopback.Exit</li> <li>6'h1D - Loopback.Exit.Timeout</li> <li>6'h1E - HotReset.Entry</li> <li>6'h1F - Hot.Reset</li> </ul> |
| rx_par_err | Output    | Asserted for a single cycle to indicate that a parity error was detected in a TLP at the input of the RX buffer. This error is logged as an uncorrectable internal error in the VSEC registers. For more information, refer to <i>Uncorrectable Internal Error Status Register</i> . If this error occurs, you must reset the Hard IP because parity errors can leave the Hard IP in an unknown state.                                                                                                                                                                                                                                                                                                                                                                                      |
| tx_par_err | Output    | Asserted for a single cycle to indicate a parity error during TX TLP transmission. The IP core transmits TX TLP packets even when a parity error is detected.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

#### Related Information

[Uncorrectable Internal Error Status Register](#) on page 115

### 6.1.17. Hard IP Reconfiguration

The Hard IP reconfiguration interface is an Avalon-MM slave interface with a 21-bit address and an 8-bit data bus. You can use this bus to dynamically modify the value of configuration registers that are read-only at run time. To ensure proper system operation, reset or repeat device enumeration of the PCI Express link after changing the value of read-only configuration registers of the Hard IP.

**Table 46. Hard IP Reconfiguration Signals**

| Signal                     | Direction | Description                                                                                                                                    |
|----------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------|
| hip_reconfig_clk           | Input     | Reconfiguration clock. The frequency range for this clock is 100–125 MHz.                                                                      |
| hip_reconfig_rst_n         | Input     | Active-low Avalon-MM reset for this interface.                                                                                                 |
| hip_reconfig_address[20:0] | Input     | The 21-bit reconfiguration address. When the Hard IP reconfiguration feature is enabled, the hip_reconfig_address[20:0] bits are programmable. |

*continued...*

| Signal                      | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             |           | <p>Some bits have the same functions in both H-Tile and L-Tile:</p> <ul style="list-style-type: none"> <li>hip_reconfig_address[11:0]: Provide full byte access to the 4 Kbytes PCIe configuration space.</li> </ul> <p><i>Note:</i> For the address map of the PCIe configuration space, refer to the <i>Configuration Space Registers</i> section in the <i>Registers</i> chapter.</p> <ul style="list-style-type: none"> <li>hip_reconfig_address[20]: Should be set to 1'b1 to indicate PCIe space access.</li> </ul> <p>Some bits have different functions in H-Tile versus L-Tile:</p> <p>For H-Tile:</p> <ul style="list-style-type: none"> <li>hip_reconfig_address[13:12]: Provide the PF number. Since the H-Tile can support up to four PFs, two bits are needed to encode the PF number.</li> <li>hip_reconfig_address[19:14]: Reserved. They must be driven to 0.</li> </ul> <p>For L-Tile:</p> <ul style="list-style-type: none"> <li>hip_reconfig_address[12]: Provides the PF number. Since the L-Tile only supports up to two PFs, one bit is sufficient to encode the PF number.</li> <li>hip_reconfig_address[19:13]: Reserved. They must be driven to 0.</li> </ul> |
| hip_reconfig_read           | Input     | Read signal. This interface is not pipelined. You must wait for the return of the hip_reconfig_readdata[7:0] from the current read before starting another read operation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| hip_reconfig_readdata[7:0]  | Output    | 8-bit read data. hip_reconfig_readdata[7:0] is valid on the third cycle after the assertion of hip_reconfig_read.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| hip_reconfig_readdatavalid  | Output    | When asserted, the data on hip_reconfig_readdata[7:0] is valid.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| hip_reconfig_write          | Input     | Write signal.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| hip_reconfig_writedata[7:0] | Input     | 8-bit write model.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| hip_reconfig_waitrequest    | Output    | When asserted, indicates that the IP core is not ready to respond to a request.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

### Related Information

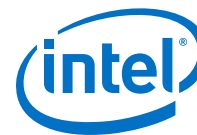
Configuration Space Registers on page 106

## 6.1.18. Power Management Interface

Software programs the Device into a D-state by writing to the Power Management Control and Status register in the PCI Power Management Capability Structure. The IP core supports the required PCI D0 and D3 Power Management states. D1 and D2 states are not supported.

**Table 47. Power Management Interface Signals**

| Signal           | Direction | Description                                                 |
|------------------|-----------|-------------------------------------------------------------|
| pm_linkst_in_l1  | Output    | When asserted, indicates that the link is in the L1 state.  |
| pm_linkst_in_l0s | Output    | When asserted, indicates that the link is in the L0s state. |
| pm_state[2:0]    | Output    | Specifies the current power state.                          |
| pm_dstate[2:0]   | Output    | Specifies the power management D-state for PF0.             |
| continued...     |           |                                                             |



| Signal              | Direction | Description                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| apps_pm_xmt_pme     | Input     | Wake Up. The Application Layer asserts this signal for 1 cycle to wake up the Power Management Capability (P MC) state machine from a D1, D2 or D3 power state. Upon wake-up, the core sends a PM_PME Message. This port is functionally identical to outband_pwrup_cmd. You can use this signal or outband_pwrup_cmd to request a return from a low-power state to D0. |
| apps_ready_entr_l23 | Input     | When asserted, the data on hip_reconfig_readdata[7:0] is valid.                                                                                                                                                                                                                                                                                                         |
| apps_pm_xmt_turnoff | Input     | Application Layer request to generate a PM_Turn_Off message. The Application Layer must assert this signal for one clock cycle. The IP core does not return an acknowledgment or grant signal. The Application Layer must not pulse this signal again until the previous message has been transmitted.                                                                  |
| app_init_rst        | Input     | Application Layer request for a hot reset to downstream devices.                                                                                                                                                                                                                                                                                                        |
| app_xfer_pending    | Input     | When asserted, prevents the IP core from entering L1 state or initiates exit from the L1 state.                                                                                                                                                                                                                                                                         |

### 6.1.19. Serial Data Interface

The IP core supports 1, 2, 4, 8, or 16 lanes.

**Table 48. Serial Data Interface**

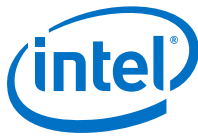
| Signal          | Direction | Description                  |
|-----------------|-----------|------------------------------|
| tx_out[<n-1>:0] | Output    | Transmit serial data output. |
| rx_in[<n-1>:0]  | Input     | Receive serial data input.   |

### 6.1.20. PIPE Interface

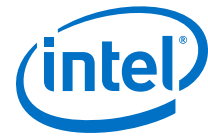
The Stratix 10 PIPE interface compiles with the *PHY Interface for the PCI Express Architecture PCI Express 3.0* specification.

**Table 49. PIPE Interface**

| Signal              | Direction | Description                                                                                                                                                 |
|---------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| txdata[31:0]        | Output    | Transmit data.                                                                                                                                              |
| txdatak[3:0]        | Output    | Transmit data control character indication.                                                                                                                 |
| txcompl             | Output    | Transmit compliance. This signal drives the TX compliance pattern. It forces the running disparity to negative in Compliance Mode (negative COM character). |
| txelecidle          | Output    | Transmit electrical idle. This signal forces the tx_out<n> outputs to electrical idle.                                                                      |
| txdetectrx          | Output    | Transmit detect receive. This signal tells the PHY layer to start a receive detection operation or to begin loopback.                                       |
| powerdown[1:0]      | Output    | Power down. This signal requests the PHY to change the power state to the specified state (P0, P0s, P1, or P2).                                             |
| txmargin[2:0]       | Output    | Transmit V <sub>OD</sub> margin selection. The value for this signal is based on the value from the Link Control 2 Register.                                |
| <i>continued...</i> |           |                                                                                                                                                             |



| Signal               | Direction | Description                                                                                                                                                                                                                                                                                        |
|----------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| txdeemp              | Output    | Transmit de-emphasis selection. The Intel Stratix 10 Hard IP for PCI Express sets the value for this signal based on the indication received from the other end of the link during the Training Sequences (TS). You do not need to change this value.                                              |
| txswing              | Output    | When asserted, indicates full swing for the transmitter voltage. When deasserted indicates half swing.                                                                                                                                                                                             |
| txsynchd[1:0]        | Output    | For Gen3 operation, specifies the receive block type. The following encodings are defined: <ul style="list-style-type: none"><li>2'b01: Ordered Set Block</li><li>2'b10: Data Block</li></ul> Designs that do not support Gen3 can ground this signal.                                             |
| txblkst[3:0]         | Output    | For Gen3 operation, indicates the start of a block in the transmit direction. pipe spec                                                                                                                                                                                                            |
| txdataskip           | Output    | For Gen3 operation. Allows the MAC to instruct the TX interface to ignore the TX data interface for one clock cycle. The following encodings are defined: <ul style="list-style-type: none"><li>1'b0: TX data is invalid</li><li>1'b1: TX data is valid</li></ul>                                  |
| rate[1:0]            | Output    | The 2-bit encodings have the following meanings: <ul style="list-style-type: none"><li>2'b00: Gen1 rate (2.5 Gbps)</li><li>2'b01: Gen2 rate (5.0 Gbps)</li><li>2'b1X: Gen3 rate (8.0 Gbps)</li></ul>                                                                                               |
| rxpolarity           | Output    | Receive polarity. This signal instructs the PHY layer to invert the polarity of the 8B/10B receiver decoding block.                                                                                                                                                                                |
| currentrxpreset[2:0] | Output    | For Gen3 designs, specifies the current preset.                                                                                                                                                                                                                                                    |
| currentcoeff[17:0]   | Output    | For Gen3, specifies the coefficients to be used by the transmitter. The 18 bits specify the following coefficients: <ul style="list-style-type: none"><li>[5:0]: <math>C_{-1}</math></li><li>[11:6]: <math>C_0</math></li><li>[17:12]: <math>C_{+1}</math></li></ul>                               |
| rxeqeval             | Output    | For Gen3, the PHY asserts this signal when it begins evaluation of the transmitter equalization settings. The PHY asserts Phystatus when it completes the evaluation. The PHY deasserts rxeqeval to abort evaluation.<br>Refer to the figure below for a timing diagram illustrating this process. |
| rxeqinprogress       | Output    | For Gen3, the PHY asserts this signal when it begins link training. The PHY latches the initial coefficients from the link partner.<br>Refer to the figure below for a timing diagram illustrating this process.                                                                                   |
| invalidreq           | Output    | For Gen3, indicates that the Link Evaluation feedback requested a TX equalization setting that is out-of-range. The PHY asserts this signal continually until the next time it asserts rxeqeval.                                                                                                   |
| rxdata[31:0]         | Input     | Receive data control. Bit 0 corresponds to the lowest-order byte of rxdata, and so on. A value of 0 indicates a data byte. A value of 1 indicates a control byte. For Gen1 and Gen2 only.                                                                                                          |
| rxdatak[3:0]         | Input     | Receive data control. This bus receives data on lane. Bit 0 corresponds to the lowest-order byte of rxdata, and so on. A value of 0 indicates a data byte. A value of 1 indicates a control byte. For Gen1 and Gen2 only.                                                                          |
| phystatus            | Input     | PHY status. This signal communicates completion of several PHY requests. pipe spec                                                                                                                                                                                                                 |
| continued...         |           |                                                                                                                                                                                                                                                                                                    |



| Signal              | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rxvalid             | Input     | Receive valid. This signal indicates symbol lock and valid data on rxdata and rxdatak.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| rxstatus[2:0]       | Input     | Receive status. This signal encodes receive status, including error codes for the receive data stream and receiver detection.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| rxelecidle          | Input     | Receive electrical idle. When asserted, indicates detection of an electrical idle. pipe spec                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| rxsynchd[3:0]       | Input     | For Gen3 operation, specifies the receive block type. The following encodings are defined: <ul style="list-style-type: none"> <li>2'b01: Ordered Set Block</li> <li>2'b10: Data Block</li> </ul> Designs that do not support Gen3 can ground this signal.                                                                                                                                                                                                                                                                                                                                                                                                    |
| rxblkst[3:0]        | Input     | For Gen3 operation, indicates the start of a block in the receive direction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| rxdataskip          | Input     | For Gen3 operation. Allows the PCS to instruct the RX interface to ignore the RX data interface for one clock cycle. The following encodings are defined: <ul style="list-style-type: none"> <li>1'b0: RX data is invalid</li> <li>1'b1: RX data is valid</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                         |
| dirfeedback[5:0]    | Input     | For Gen3, provides a Figure of Merit for link evaluation for H tile transceivers. The feedback applies to the following coefficients: <ul style="list-style-type: none"> <li>dirfeedback[5:4]: Feedback applies to <math>C_{+1}</math></li> <li>dirfeedback[3:2]: Feedback applies to <math>C_0</math></li> <li>dirfeedback[1:0]: Feedback applies to <math>C_{-1}</math></li> </ul> The following feedback encodings are defined: <ul style="list-style-type: none"> <li>2'b00: No change</li> <li>2'b01: Increment</li> <li>2'b10: Decrement</li> <li>2'b11: Reserved</li> </ul> Refer to the figure below for a timing diagram illustrating this process. |
| simu_mode_pipe      | Input     | When set to 1, the PIPE interface is in simulation mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| sim_pipe_pclk_in    | Input     | This clock is used for PIPE simulation only, and is derived from the refclk. It is the PIPE interface clock used for PIPE mode simulation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| sim_pipe_rate[1:0]  | Output    | The 2-bit encodings have the following meanings: <ul style="list-style-type: none"> <li>2'b00: Gen1 rate (2.5 Gbps)</li> <li>2'b01: Gen2 rate (5.0 Gbps)</li> <li>2'b10: Gen3 rate (8.0 Gbps)</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| sim_ltssmstate[5:0] | Output    | LTSSM state: The following encodings are defined: <ul style="list-style-type: none"> <li>6'h00 - Detect.Quiet</li> <li>6'h01 - Detect.Active</li> <li>6'h02 - Polling.Active</li> <li>6'h03 - Polling.Compliance</li> <li>6'h04 - Polling.Configuration</li> <li>6'h05 - PreDetect.Quiet</li> <li>6'h06 - Detect.Wait</li> <li>6'h07 - Configuration.Linkwidth.Start</li> <li>6'h08 - Configuration.Linkwidth.Accept</li> <li>6'h09 - Configuration.Lanenum.Wait</li> <li>6'h0A - Configuration.Lanenum.Accept</li> <li>6'h0B - Configuration.Complete</li> <li>6'h0C - Configuration.Idle</li> <li>6'h0D - Recovery.RcvrLock</li> </ul>                     |

*continued...*

| Signal                    | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           |           | <ul style="list-style-type: none"> <li>6'h0E - Recovery.Speed</li> <li>6'h0F - Recovery.RcvrCfg</li> <li>6'h10 - Recovery.Idle</li> <li>6'h20 - Recovery.Equalization Phase 0</li> <li>6'h21 - Recovery.Equalization Phase 1</li> <li>6'h22 - Recovery.Equalization Phase 2</li> <li>6'h23 - Recovery.Equalization Phase 3</li> <li>6'h11 - L0</li> <li>6'h12 - L0s</li> <li>6'h13 - L123.SendEIdle</li> <li>6'h14 - L1.Idle</li> <li>6'h15 - L2.Idle</li> <li>6'h16 - L2.TransmitWake</li> <li>6'h17 - Disabled.Entry</li> <li>6'h18 - Disabled.Idle</li> <li>6'h19 - Disabled</li> <li>6'h1A - Loopback.Entry</li> <li>6'h1B - Loopback.Active</li> <li>6'h1C - Loopback.Exit</li> <li>6'h1D - Loopback.Exit.Timeout</li> <li>6'h1E - HotReset.Entry</li> <li>6'h1F - Hot.Reset</li> </ul> |
| sim_pipe_mask_tx_pll_lock | Input     | <p>Should be active during rate change. This signal is used to mask the PLL lock signals. This interface is used only for PIPE simulations.</p> <p>In serial simulations, The Endpoint PHY drives this signal. For PIPE simulations, in the Intel testbench, The PIPE BFM drives this signal.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

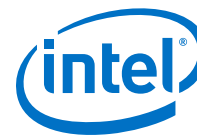
### Related Information

PHY Interface for the PCI Express Architecture PCI Express 3.0

## 6.1.21. Test Interface

The 256-bit test output interface is available only for x16 simulations. For x1, x2, x4, and x8 variants a 7-bit auxiliary test bus is available.

| Signal        | Direction | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| test_in[66:0] | Input     | <p>This is a multiplexer to select the test_out[255:0] and aux_test_out[6:0] buses. Driven from channels 8-15.</p> <p>The following encodings are defined:</p> <ul style="list-style-type: none"> <li>[66]: Reserved</li> <li>[65:58]: The multiplexer selects the EMIB adaptor</li> <li>[57:50]: The multiplexer selects configuration block</li> <li>[49:48]: The multiplexer selects clocks</li> <li>[47:46]: The multiplexer selects equalization</li> <li>[45:44]: The multiplexer selects miscellaneous functionality</li> <li>[43:42]: The multiplexer selects the PIPE adaptor</li> <li>[41:40]: The multiplexer selects for CvP.</li> <li>[39:31]: The multiplexer selects channels 7-1, aux_test_out[6:0]</li> <li>[30:3]: The multiplexer selects channels 15-8, test_out[255:0]</li> </ul> |
| continued...  |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |



| Signal                       | Direction | Description                                                                                                                                                                                                                              |
|------------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                              |           | <ul style="list-style-type: none"> <li>[2]: Results in the inversion of LCRC bit 0.</li> <li>[1]: Results in the inversion of ECRC bit 0</li> <li>[0]: Turns on <code>diag_fast_link_mode</code> to speed up simulation.</li> </ul>      |
| <code>test_out[255:0]</code> | Output    | <code>test_out[255:0]</code> routes to channels 8-15. Includes diagnostic signals from core, adaptor, clock, configuration block, equalization control, miscellaneous, reset, and pipe_adaptor modules. Available only for x16 variants. |

### 6.1.22. PLL IP Reconfiguration

The PLL reconfiguration interface is an Avalon-MM slave interface with an 11-bit address and a 32-bit data bus. Use this bus to dynamically modify the value of PLL registers that are read-only at run time.

To ensure proper system operation, reset or repeat device enumeration of the PCIe link after changing the value of read-only PLL registers.

These signals are present when you turn on **Enable Transceiver dynamic reconfiguration** on the **Configuration, Debug and Extension Options** tab using the parameter editor.

**Table 50. Hard IP Reconfiguration Signals**

The same set of signals are available for PLL1.

| Signal                                     | Direction | Description                                                                                                                                                                               |
|--------------------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>xcvr_reconfig_clk</code>             | Input     | Reconfiguration clock. The frequency range for this clock is 100–125 MHz.                                                                                                                 |
| <code>xcvr_reconfig_rst_n</code>           | Input     | Active-low Avalon-MM reset for this interface.                                                                                                                                            |
| <code>xcvr_reconfig_address[14:0]</code>   | Input     | The 11-bit reconfiguration address.                                                                                                                                                       |
| <code>xcvr_reconfig_read</code>            | Input     | Read signal. This interface is not pipelined. You must wait for the return of the <code>xcvr_reconfig_readdata[31:0]</code> from the current read before starting another read operation. |
| <code>xcvr_reconfig_readdata[31:0]</code>  | Output    | 32-bit read data. <code>xcvr_reconfig_readdata[31:0]</code> is valid on the third cycle after the assertion of <code>xcvr_reconfig_read</code> .                                          |
| <code>xcvr_reconfig_readdatavalid</code>   | Output    | When asserted, the data on <code>xcvr_reconfig_readdata[31:0]</code> is valid.                                                                                                            |
| <code>xcvr_reconfig_write</code>           | Input     | Write signal.                                                                                                                                                                             |
| <code>xcvr_reconfig_writedata[31:0]</code> | Input     | 32-bit write data.                                                                                                                                                                        |
| <code>xcvr_reconfig_waitrequest</code>     | Output    | When asserted, indicates that the IP core is not ready to respond to a request.                                                                                                           |

## 6.1.23. Message Handling

### 6.1.23.1. Endpoint Received Messages

| Message Type                        | Message              | Message Processing                                                                                                                                                                                                                                                   |
|-------------------------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Power Management                    | PME_Turn_Off         | Forwarded to the Application Layer on Avalon-ST RX interface. Also processed by the IP core.                                                                                                                                                                         |
| Slot Power Limit                    | Set_Slot_Power_Limit | Forwarded to the Application Layer on Avalon-ST RX interface.                                                                                                                                                                                                        |
| Vendor Defined with or without Data | Vendor_Type0         | Forwarded to the Application Layer on Avalon-ST RX interface. Not processed by core.<br>You can program the IP core to drop these Messages using the <code>virtual_drop_vendor0_msg</code> . When dropped, Vendor0 Messages are logged as Unsupported Requests (UR). |
| ATS                                 | ATS_Invalidate       | Forwarded to the Application Layer on Avalon-ST RX interface. Not processed by the IP core.                                                                                                                                                                          |
| Locked Transaction                  | Unlock Message       | Forwarded to the Application Layer on Avalon-ST RX interface. Not processed by the IP core.                                                                                                                                                                          |
| All Others                          | ---                  | Internally dropped by Endpoint and handled as an Unsupported Request.                                                                                                                                                                                                |

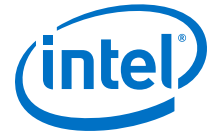
### 6.1.23.2. Endpoint Transmitted Messages

The Endpoint transmits Messages that it receives from the Application Layer on the Avalon-ST TX interface and internally generated messages. The Endpoint may generate Messages autonomously or in response to a request received on the : `app_pm_xmt_pme`, `app_int`, or `app_err_*` input ports.

| Message Type                        | Message                                | Message Processing                                                                                                                                                                                                                                                                                                             |
|-------------------------------------|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Power Management                    | PM_PME, PME_TO_Ack                     | The Application Layer transmits the PM_PME request via the <code>app_pm_xmt_pme</code> input.<br>The Application Layer must generate the PME_TO_Ack and transmit it on the Avalon-ST TX interface.<br>The Application Layer transmits the <code>app_ready_entr_l23</code> to indicate that it is ready to enter the L23 state. |
| Vendor Defined with or without Data | Vendor_Type0                           | The Application Layer must generate and transmit this on the Avalon-ST TX interface.                                                                                                                                                                                                                                           |
| ATS                                 | ATS_Request, ATS_Invalidate Completion | The Application Layer must generate and transmit these Messages on the Avalon-ST TX interface.                                                                                                                                                                                                                                 |
| INT                                 | INTx_Assert, INTx_Deassert             | The Application Layer transmits the INTx_Assert and INTx_Deassert Messages using the <code>app_int</code> interface.                                                                                                                                                                                                           |
| ERR                                 | ERR_COR, ERR_NONFATAL, ERR_FATAL       | The IP core transmits these error Messages autonomously when it detects internal errors. It also receives and forwards these errors when received from the Application Layer via the <code>app_err_*</code> interface.                                                                                                         |

## 6.2. Errors reported by the Application Layer

The Application Layer uses the `app_err_*` interface to report errors to the IP core.



The Application Layer reports the following types of errors to the IP core:

- Unexpected Completion
- Completer Abort
- CPL Timeout
- Unsupported Request
- Poisoned TLP received
- Uncorrected Internal Error, including ECC and parity errors flagged by the core
- Corrected Internal Error, including Corrected ECC errors flagged by the core
- Advisory NonFatal Error

For Advanced Error Reporting (AER), the Application Layer provide the information to log the TLP header and the error log request via the `app_err_*` interface.

The Application Layer completes the following steps to report an error to the IP core:

- Sets the corresponding status bits in the PCI Status register, and the PCIe Device Status register
- Sets the appropriate status bits and header log in the AER registers if AER is enabled
- Indicates the Error event to the upstream component:
  - Endpoints transmit an Message upstream
  - Root Ports assert `app_serr_out` to the Application Layer if an error is detected or if an error Message is received from a downstream component. The Root Port also forwards the error Message from the downstream component on the Avalon-ST RX interface. The Application Layer may choose to ignore this information. (Root Ports are not supported in the Quartus Prime Pro – Stratix 10 Edition 17.1 Interim Release.)

### 6.2.1. Error Handling

When the IP core detects an error in a received TLP, it generates a Completion. It sets the Completion status set to Completer Abort (CA) or Unsupported Request (UR).

The IP Core completes the following actions when it detects an error in a received TLP:

- Discards the TLP.
- Generates a Completion (for non-posted requests) with the Completion status set to CA or UR.
- Sets the corresponding status bits in the PCI Status register and the PCIe Device Status register.
- Sets the corresponding status bits and header log in the AER registers if AER is enabled.
- Indicates the Error event to the upstream component.
  - For Endpoints, the IP core sends an error Message upstream.
  - For Root Ports, the IP core asserts `app_serr_out` asserts to the Application Layer) when it detects an error or receives an error Message from a downstream component.

*Note:* The error Message from the downstream component is also forwarded on the Avalon-ST RX interface. The Application Layer may choose to ignore this information.

## 6.3. Power Management

The Stratix 10 Hard IP for PCI Express IP core supports the required PCI D0 and D3 Power Management states. It does not support the optional D1 and D2 Power Management states.

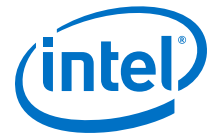
Software programs the Device into a D-state by writing to the Power Management Control and Status register in the PCI Power Management Capability Structure. The `pm_*` interface transmits the D-state to the Application Layer.

### 6.3.1. Endpoint D3 Entry

This topic outlines the D3 power-down procedure.

All transmission on the Avalon-ST TX and RX interfaces must have completed before IP core can begin the L1 request (Enter\_L1 DLLP). In addition, the RX Buffer must be empty and the Application Layer `app_xfer_pending` output must be deasserted.

1. Software writes the Power Management Control register to put the IP core to the D3<sub>hot</sub> state.
2. The Endpoint stops transmitting requests when it has been taken out of D0.
3. The link transitions to L1.
4. Software sends the PME\_Turn\_Off Message to the Endpoint to initiate power down. The Root Port transitions the link back to L0, and Endpoint receives the Message on the Avalon-ST RX interface.
5. The End Point transmits a PME\_TO\_Ack Message to acknowledge the Turn Off request.
6. When ready for power removal, (D3<sub>cold</sub>), the End Point asserts `app_ready_entr_l23`. The core sends the PM\_Enter\_L23 DLLP and initiates the Link transition to L3.



### 6.3.2. End Point D3 Exit

An End Point can exit the D3 state if the following two conditions are true: First, the PME\_Support setting in the Power Management Capabilities (PMC) register must enable PME notification. Second, software must set the PME\_en bit in the Power Management Control and Status (PMCSR) register.

### 6.3.3. Exit from D3 hot

The Power Management Capability register must enable D3<sub>Hot</sub> PME\_Support. In addition, software must set the PME\_en bit in the Power Management Control and Status register.

### 6.3.4. Exit from D3 cold

1. To issue a PM\_EVENT Message from the D3 cold state, the device must first issue a wakeup event (WAKE#) to request reapplication of power and the clock. The wakeup event triggers a fundamental reset which reinitializes the link to L0.
2. The Application Layer requests a wake-up event by asserting apps\_pm\_xmt\_pme. Asserting apps\_pm\_xmt\_pme causes the IP core to transmit a PM\_EVENT Message. In addition, the IP core sets the PME\_status bit in the Power Management Control and Status register to notify software that it has requested the wakeup.

The PCIe Link states are indicated on the pm\_\* interface. The LTSSM state is indicated on the ltssm\_state output.

### 6.3.5. Active State Power Management

Active State Power Management (ASPM) is not supported as indicated by the ASPM Support bits in the Link Capabilities register.

## 6.4. Transaction Ordering

### 6.4.1. TX TLP Ordering

TLPs on the Avalon-ST TX interface are transmitted in the order in which the Application Layer presents them. The IP core provides TX credit information to the Application Layer so that the Application Layer can perform credits-based reordering before submitting TLPs for transmission.

This reordering is optional. The IP core always checks for sufficient TX credits before transmitting any TLP. Ordering is not guaranteed between the following TLP transmission interfaces:

- Avalon
- MSI and MSI-X interrupt
- Internal Configuration Space TLPs

## 6.4.2. RX TLP Ordering

TLPs ordering on the Avalon-ST RX interface is determined by the available credits for each TLP type and the PCIe ordering rules. Refer to *Table 2-34 Ordering Rules Summary* in the *PCI Express Base Specification Revision 3.0* for a summary of the ordering rules.

The IP core implements relaxed ordering as described in the *PCI Express Base Specification Revision 3.0*. It does not perform ID-Based Ordering (IDO). The Application Layer can implement IDO reordering. It is possible for two different TLP types pending in the RX buffer to have equal priority. When this situation occurs, the IP core uses a fairness-based arbitration scheme to determine which TLP to forward to the Application Layer.

### Related Information

[PCI Express Base Specification 3.0](#)

## 6.5. RX Buffer

The Receive Buffer stores TLPs received from the PCI Express link. The RX Buffer stores the entire TLP before it forwards it to the Application Layer. Storing the entire TLP allows the IP to accomplish two things:

- The IP core can rate match the PCIe link to the Application Layer.
- The IP core can store TLPs until error checking is complete.

The 64 KB RX buffer has separate buffer space for Posted, Non-Posted and Completion TLPs. Headers and data also have separate allocations. The RX Buffer enables full bandwidth RX traffic for all three types of TLPs simultaneously.

**Table 51. Flow Control Credit Allocation**

The buffer allocation is fixed.

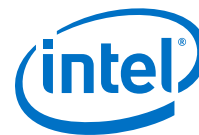
| RX Buffer Segment | Number of Credits                                                               | Buffer Size |
|-------------------|---------------------------------------------------------------------------------|-------------|
| Posted            | Posted headers: 127 credits<br>Posted data: 750 credits                         | ~14 KB      |
| Non-posted        | Non-posted headers credits: 115 credits<br>Non-posted data credits: 230 credits | ~5.5 KB     |
| Completions       | Completion headers: 770 credits<br>Completion data: 2500 credits                | ~50 KB      |

The RX buffer operates only in the Store and Forward Queue Mode. Bypass and Cut-through modes are not supported.

Flow control credit checking for the posted and non-posted buffer segments prevents RX buffer overflow. The *PCI Express Base Specification Revision 3.0* requires the IP core to advertise infinite Completion credits. The Application Layer must manage the Read Requests so as not to overflow the Completion buffer segment.

### Related Information

[PCI Express Base Specification 3.0](#)



### 6.5.1. Retry Buffer

The retry buffer stores a copy of a transmitted TLP until the remote device acknowledges the TLP using the ACK mechanism. In the case of an LCRC error on the receiving side, the remote device sends NAK DLLP to the transmitting side. The transmitting side retrieves the TLP from the retry buffer and resends it.

Retry buffer resources are only freed upon reception of an ACK DLLP.

### 6.5.2. Configuration Retry Status

The Stratix 10 Hard IP for PCI Express is part of the periphery image of the device. After power-up, the periphery image is loaded first. The End Point can then respond to Configuration Requests with Config Retry Status (CRS) to delay the enumeration process until after the FPGA fabric is configured.

## 7. Interrupts

---

### 7.1. Interrupts for Endpoints

The Intel Stratix 10 Hard IP for PCI Express provides support for PCI Express MSI, MSI-X, and legacy interrupts when configured in Endpoint mode. The MSI and legacy interrupts are *mutually exclusive*. After power up, the Hard IP block starts in legacy interrupt mode. Then, software decides whether to switch to MSI or MSI-X mode. To switch to MSI mode, software programs the `msi_enable` bit of the MSI Message Control Register to 1, (bit[16] of 0x050). You enable MSI-X mode, by turning on **Implement MSI-X** under the **PCI Express/PCI Capabilities** tab using the parameter editor. If you turn on the **Implement MSI-X** option, you should implement the MSI-X table structures at the memory space pointed to by the BARs.

*Note:*

Refer to section 6.1 of *PCI Express Base Specification* for a general description of PCI Express interrupt support for Endpoints.

#### Related Information

[PCI Express Base Specification 3.0](#)

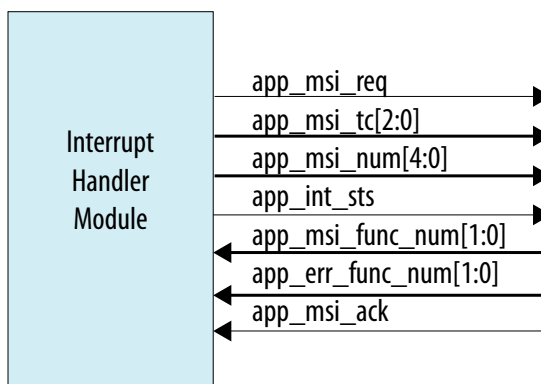
#### 7.1.1. MSI and Legacy Interrupts

The IP core generates single dword Memory Write TLPs to signal MSI interrupts on the PCI Express link. The Application Layer Interrupt Handler Module `app_msi_req` output port controls MSI interrupt generation. When asserted, it causes an MSI posted Memory Write TLP to be generated. The IP core constructs the TLP using information from the following sources:

- The MSI Capability registers
- The traffic class (`app_msi_tc`)
- The message data specified by `app_msi_num`

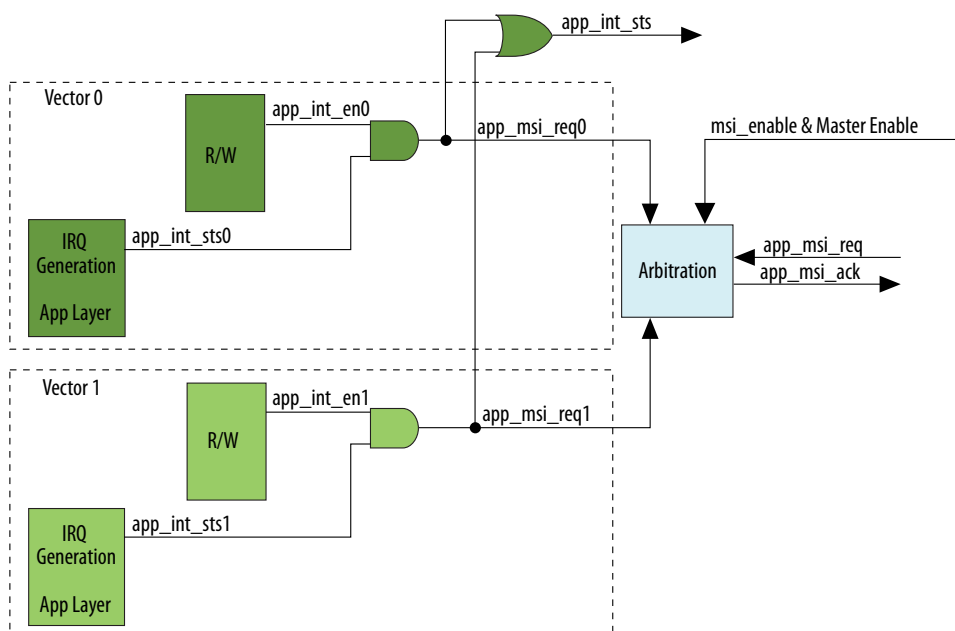
To enable MSI interrupts, the Application Layer must first set the `MSI enable` bit. Then, it must disable legacy interrupts by setting the `Interrupt Disable`, bit 10 of the Command register.

**Figure 54. Interrupt Handler Module in the Application Layer**



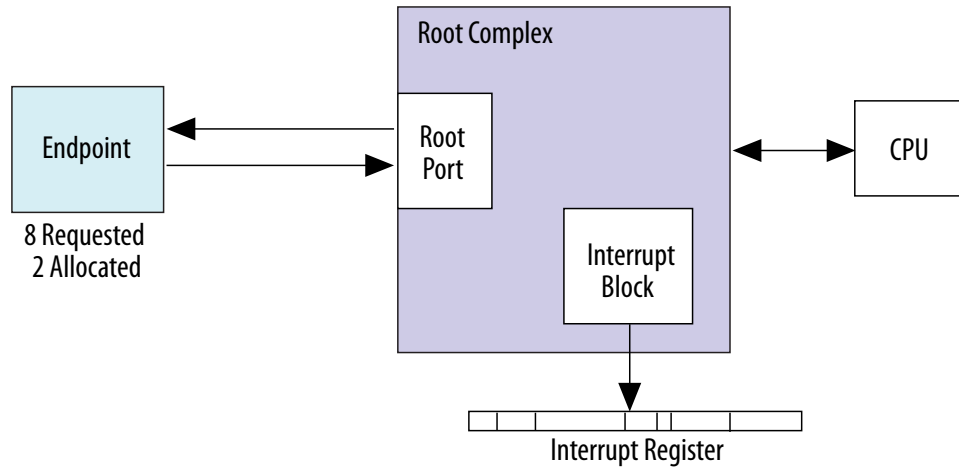
The following figure illustrates a possible implementation of the Interrupt Handler Module with a per vector enable bit. Alternatively, the Application Layer could implement a global interrupt enable instead of this per vector MSI.

**Figure 55. Example Implementation of the Interrupt Handler Block**



There are 32 possible MSI messages. The number of messages requested by a particular component does not necessarily correspond to the number of messages allocated. For example, in the following figure, the Endpoint requests eight MSIs but is only allocated two. In this case, you must design the Application Layer to use only two allocated messages.

**Figure 56. MSI Request Example**



The following table describes three example implementations. The first example allocates all 32 MSI messages. The second and third examples only allocate 4 interrupts.

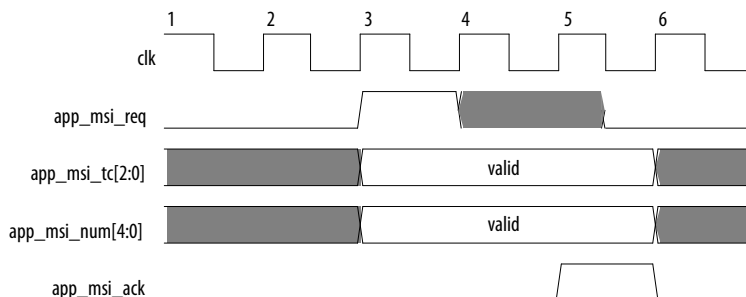
**Table 52. MSI Messages Requested, Allocated, and Mapped**

| MSI                                 | Allocated |     |     |
|-------------------------------------|-----------|-----|-----|
|                                     | 32        | 4   | 4   |
| System Error                        | 31        | 3   | 3   |
| Hot Plug and Power Management Event | 30        | 2   | 3   |
| Application Layer                   | 29:0      | 1:0 | 2:0 |

MSI interrupts generated for Hot Plug, Power Management Events, and System Errors always use Traffic Class 0. MSI interrupts generated by the Application Layer can use any Traffic Class. For example, a DMA that generates an MSI at the end of a transmission can use the same traffic control as was used to transfer data.

The following figure illustrates the interactions among MSI interrupt signals for the Root Port. The minimum latency possible between `app_msi_req` and `app_msi_ack` is one clock cycle. In this timing diagram `app_msi_req` can extend beyond `app_msi_ack` before deasserting. However, `app_msi_req` must be deasserted before or within the same clock as `app_msi_ack` is deasserted to avoid inferring a new interrupt.

**Figure 57. MSI Interrupt Signals Timing**



### 7.1.2. MSI-X

You can enable MSI-X interrupts by turning on **Implement MSI-X** under the **PCI Express/PCI Capabilities** heading using the parameter editor. If you turn on the **Implement MSI-X** option, you should implement the MSI-X table structures at the memory space pointed to by the BARs as part of your Application Layer.

The Application Layer transmits MSI-X interrupts on the Avalon-ST TX interface. MSI-X interrupts are single dword Memory Write TLPs. Consequently, the Last DW Byte Enable in the TLP header must be set to 4b'0000. MSI-X TLPs should be sent only when enabled by the MSI-X enable and the function mask bits in the Message Control for the MSI-X Configuration register. These bits are available on the `tl_cfg_ctl` output bus.

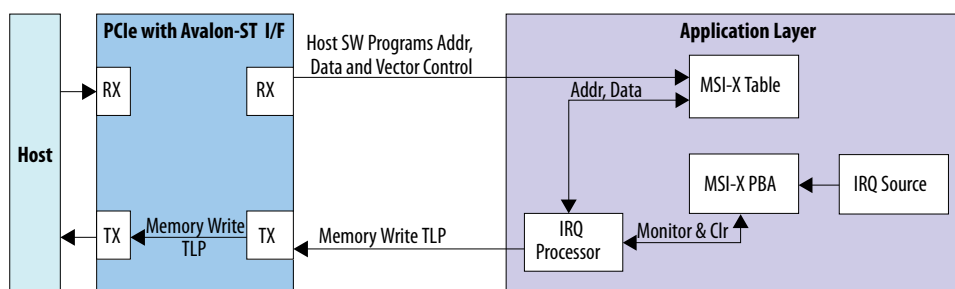
#### Related Information

- [PCI Local Bus Specification](#)
- [PCI Express Base Specification 3.0](#)

### 7.1.3. Implementing MSI-X Interrupts

Section 6.8.2 of the *PCI Local Bus Specification* describes the MSI-X capability and table structures. The MSI-X capability structure points to the MSI-X Table structure and MSI-X Pending Bit Array (PBA) registers. The BIOS sets up the starting address offsets and BAR associated with the pointer to the starting address of the MSI-X Table and PBA registers.

**Figure 58. MSI-X Interrupt Components**



1. Host software sets up the MSI-X interrupts in the Application Layer by completing the following steps:

- a. Host software reads the Message Control register at 0x050 register to determine the MSI-X Table size. The number of table entries is the  $\langle \text{value read} \rangle + 1$ .

The maximum table size is 2048 entries. Each 16-byte entry is divided in 4 fields as shown in the figure below. For multi-function variants, BAR4 accesses the MSI-X table. For all other variants, any BAR can access the MSI-X table. The base address of the MSI-X table must be aligned to a 4 KB boundary.

- b. The host sets up the MSI-X table. It programs MSI-X address, data, and masks bits for each entry as shown in the figure below.

**Figure 59. Format of MSI-X Table**

| DWORD 3        | DWORD 2      | DWORD 1               | DWORD 0         |               | Host Byte Addresses |
|----------------|--------------|-----------------------|-----------------|---------------|---------------------|
| Vector Control | Message Data | Message Upper Address | Message Address | Entry 0       | Base                |
| Vector Control | Message Data | Message Upper Address | Message Address | Entry 1       | Base + 1 × 16       |
| Vector Control | Message Data | Message Upper Address | Message Address | Entry 2       | Base + 2 × 16       |
|                |              |                       |                 | ⋮             | ⋮                   |
| Vector Control | Message Data | Message Upper Address | Message Address | Entry (N - 1) | Base + (N - 1) × 16 |

- c. The host calculates the address of the  $\langle n^{\text{th}} \rangle$  entry using the following formula:

$$\text{nth\_address} = \text{base address}[BAR] + 16\langle n \rangle$$

2. When Application Layer has an interrupt, it drives an interrupt request to the IRQ Source module.
3. The IRQ Source sets appropriate bit in the MSI-X PBA table.

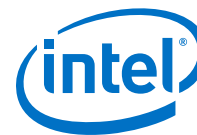
The PBA can use qword or dword accesses. For qword accesses, the IRQ Source calculates the address of the  $\langle m^{\text{th}} \rangle$  bit using the following formulas:

$$\begin{aligned} \text{qword address} &= \langle \text{PBA base addr} \rangle + 8(\text{floor}(\langle m \rangle / 64)) \\ \text{qword bit} &= \langle m \rangle \bmod 64 \end{aligned}$$

**Figure 60. MSI-X PBA Table**

| Pending Bit Array (PBA)                                            |                                   | Address                                     |
|--------------------------------------------------------------------|-----------------------------------|---------------------------------------------|
| Pending Bits 0 through 63                                          | QWORD 0                           | Base                                        |
| Pending Bits 64 through 127                                        | QWORD 1                           | Base + 1 × 8                                |
|                                                                    |                                   | ⋮                                           |
| Pending Bits $((N - 1) \text{ div } 64) \times 64$ through $N - 1$ | QWORD $((N - 1) \text{ div } 64)$ | Base + $((N - 1) \text{ div } 64) \times 8$ |

4. The IRQ Processor reads the entry in the MSI-X table.
  - a. If the interrupt is masked by the Vector\_Control field of the MSI-X table, the interrupt remains in the pending state.
  - b. If the interrupt is not masked, IRQ Processor sends Memory Write Request to the TX slave interface. It uses the address and data from the MSI-X table. If **Message Upper Address** = 0, the IRQ Processor creates a three-dword header. If the **Message Upper Address** > 0, it creates a 4-dword header.
5. The host interrupt service routine detects the TLP as an interrupt and services it.



#### Related Information

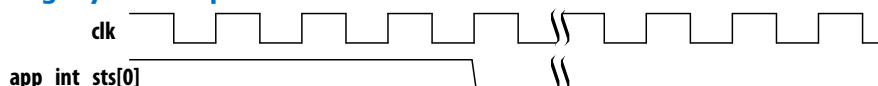
- Floor and ceiling functions
- PCI Local Bus Specification, Rev. 3.0

### 7.1.4. Legacy Interrupts

Legacy interrupts mimic the original PCI level-sensitive interrupts using *virtual wire* messages. The Intel Stratix 10 signals legacy interrupts on the PCIe link using Message TLPs. The term, INTx, refers collectively to the four legacy interrupts, INTA#, INTB#, INTC# and INTD#. The Intel Stratix 10 asserts `app_int_sts` to cause an Assert\_INTx Message TLP to be generated and sent upstream. Deassertion of `app_int_sts` causes a Deassert\_INTx Message TLP to be generated and sent upstream. To use legacy interrupts, you must clear the Interrupt Disable bit, which is bit 10 of the Command register. Then, turn off the MSI Enable bit.

The following figures illustrates interrupt timing for the legacy interface. The legacy interrupt handler asserts `app_int_sts` to instruct the Hard IP for PCI Express to send a Assert\_INTx message TLP.

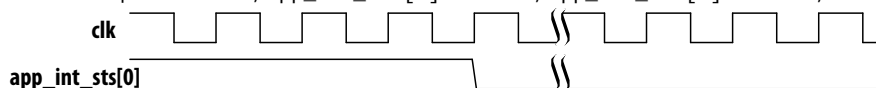
**Figure 61. Legacy Interrupt Assertion**



The following figure illustrates the timing for deassertion of legacy interrupts. The legacy interrupt handler asserts `app_int_sts` causing the Hard IP for PCI Express to send a Deassert\_INTx message.

**Figure 62. Legacy Interrupt Deassertion**

For multi-function implementations, `app_int_sts[0]` is for PF0, `app_int_sts[1]` is for PF1, and so on.

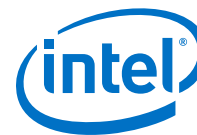


## 8. Registers

### 8.1. Configuration Space Registers

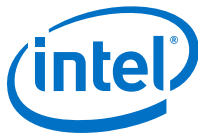
**Table 53. Correspondence between Configuration Space Capability Structures and the PCIe Base Specification Description**

| Byte Address | Configuration Space Register                                      | Corresponding Section in PCIe Specification                                                                                         |
|--------------|-------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| 0x000-0x03C  | PCI Header Type 0 Configuration Registers                         | Type 0 Configuration Space Header                                                                                                   |
| 0x040-0x04C  | Power Management                                                  | PCI Power Management Capability Structure                                                                                           |
| 0x050-0x05C  | MSI Capability Structure                                          | MSI Capability Structure, see also and <i>PCI Local Bus Specification</i>                                                           |
| 0x060-0x06C  | Reserved                                                          | N/A                                                                                                                                 |
| 0x070-0x0A8  | PCI Express Capability Structure                                  | PCI Express Capability Structure                                                                                                    |
| 0x0B0-0x0B8  | MSI-X Capability Structure                                        | MSI-X Capability Structure, see also and <i>PCI Local Bus Specification</i>                                                         |
| 0x0BC-0x0FC  | Reserved                                                          | N/A                                                                                                                                 |
| 0x100-0x134  | Advanced Error Reporting (AER) (for PFs only)                     | Advanced Error Reporting Capability                                                                                                 |
| 0x138-0x174  | Virtual Channel Capability Structure (Reserved)                   | Virtual Channel Capability                                                                                                          |
| 0x178-0x17C  | Alternative Routing-ID Implementation (ARI). Always on for SR-IOV | ARI Capability                                                                                                                      |
| 0x188-0x1B0  | Secondary PCI Express Extended Capability Header                  | PCI Express Extended Capability                                                                                                     |
| 0x1B4        | Reserved                                                          | N/A                                                                                                                                 |
| 0x1B8-0x1F4  | SR-IOV Capability Structure                                       | SR-IOV Extended Capability Header in <i>Single Root I/O Virtualization and Sharing Specification, Rev. 1.1</i>                      |
| 0x1F8-0x1D0  | Transaction Processing Hints (TPH) Requester Capability           | TLP Processing Hints (TPH)                                                                                                          |
| 0x1D4-0x280  | Reserved                                                          | N/A                                                                                                                                 |
| 0x284-0x288  | Address Translation Services (ATS) Capability Structure           | Address Translation Services Extended Capability (ATS) in <i>Single Root I/O Virtualization and Sharing Specification, Rev. 1.1</i> |
| 0xB80-0xBFC  | Intel-Specific                                                    | Vendor-Specific Header (Header only)                                                                                                |
| 0xC00        | Optional Custom Extensions                                        | N/A                                                                                                                                 |
| 0xC00        | Optional Custom Extensions                                        | N/A                                                                                                                                 |



**Table 54. Summary of Configuration Space Register Fields**

| Byte Address        | Hard IP Configuration Space Register             | Corresponding Section in PCIe Specification         |
|---------------------|--------------------------------------------------|-----------------------------------------------------|
| 0x000               | Device ID, Vendor ID                             | Type 0 Configuration Space Header                   |
| 0x004               | Status, Command                                  | Type 0 Configuration Space Header                   |
| 0x008               | Class Code, Revision ID                          | Type 0 Configuration Space Header                   |
| 0x00C               | Header Type, Cache Line Size                     | Type 0 Configuration Space Header                   |
| 0x010               | Base Address 0                                   | Base Address Registers                              |
| 0x014               | Base Address 1                                   | Base Address Registers                              |
| 0x018               | Base Address 2                                   | Base Address Registers                              |
| 0x01C               | Base Address 3                                   | Base Address Registers                              |
| 0x020               | Base Address 4                                   | Base Address Registers                              |
| 0x024               | Base Address 5                                   | Base Address Registers                              |
| 0x028               | Reserved                                         | N/A                                                 |
| 0x02C               | Subsystem ID, Subsystem Vendor ID                | Type 0 Configuration Space Header                   |
| 0x030               | Reserved                                         | N/A                                                 |
| 0x034               | Capabilities Pointer                             | Type 0 Configuration Space Header                   |
| 0x038               | Reserved                                         | N/A                                                 |
| 0x03C               | Interrupt Pin, Interrupt Line                    | Type 0 Configuration Space Header                   |
| 0x040               | PME_Support, D1, D2, etc.                        | PCI Power Management Capability Structure           |
| 0x044               | PME_en, PME_Status, etc.                         | Power Management Status and Control Register        |
| 0x050               | MSI-Message Control, Next Cap Ptr, Capability ID | MSI and MSI-X Capability Structures                 |
| 0x054               | Message Address                                  | MSI and MSI-X Capability Structures                 |
| 0x058               | Message Upper Address                            | MSI and MSI-X Capability Structures                 |
| 0x05C               | Reserved Message Data                            | MSI and MSI-X Capability Structures                 |
| 0x0B0               | MSI-X Message Control Next Cap Ptr Capability ID | MSI and MSI-X Capability Structures                 |
| 0x0B4               | MSI-X Table Offset BIR                           | MSI and MSI-X Capability Structures                 |
| 0x0B8               | Pending Bit Array (PBA) Offset BIR               | MSI and MSI-X Capability Structures                 |
| 0x100               | PCI Express Enhanced Capability Header           | Advanced Error Reporting Enhanced Capability Header |
| 0x104               | Uncorrectable Error Status Register              | Uncorrectable Error Status Register                 |
| 0x108               | Uncorrectable Error Mask Register                | Uncorrectable Error Mask Register                   |
| 0x10C               | Uncorrectable Error Mask Register                | Uncorrectable Error Severity Register               |
| 0x110               | Correctable Error Status Register                | Correctable Error Status Register                   |
| 0x114               | Correctable Error Mask Register                  | Correctable Error Mask Register                     |
| 0x118               | Advanced Error Capabilities and Control Register | Advanced Error Capabilities and Control Register    |
| 0x11C               | Header Log Register                              | Header Log Register                                 |
| 0x12C               | Root Error Command                               | Root Error Command Register                         |
| <i>continued...</i> |                                                  |                                                     |



| Byte Address | Hard IP Configuration Space Register                                      | Corresponding Section in PCIe Specification |
|--------------|---------------------------------------------------------------------------|---------------------------------------------|
| 0x130        | Root Error Status                                                         | Root Error Status Register                  |
| 0x134        | Error Source Identification Register Correctable Error Source ID Register | Error Source Identification Register        |
| 0x188        | Next Capability Offset, PCI Express Extended Capability ID                | Secondary PCI Express Extended Capability   |
| 0x18C        | Enable SKP OS, Link Equalization Req, Perform Equalization                | Link Control 3 Register                     |
| 0x190        | Lane Error Status Register                                                | Lane Error Status Register                  |
| 0x194:0x1B0  | Lane Equalization Control Register                                        | Lane Equalization Control Register          |
| 0xB80        | VSEC Capability Header                                                    | Vendor-Specific Extended Capability Header  |
| 0xB84        | VSEC Length, Revision, ID                                                 | Vendor-Specific Header                      |
| 0xB88        | Intel Marker                                                              | Intel-Specific Registers                    |
| 0xB8C        | JTAG Silicon ID DW0                                                       |                                             |
| 0xB90        | JTAG Silicon ID DW1                                                       |                                             |
| 0xB94        | JTAG Silicon ID DW2                                                       |                                             |
| 0xB98        | JTAG Silicon ID DW3                                                       |                                             |
| 0xB9C        | User Device and Board Type ID                                             |                                             |
| 0xBA0:0xBAC  | Reserved                                                                  |                                             |
| 0xBB0        | General Purpose Control and Status Register                               |                                             |
| 0xBB4        | Uncorrectable Internal Error Status Register                              |                                             |
| 0xBB8        | Uncorrectable Internal Error Mask Register                                |                                             |
| 0xBBC        | Correctable Error Status Register                                         |                                             |
| 0xBC0        | Correctable Error Mask Register                                           |                                             |
| 0xBC4:BD8    | Reserved                                                                  | N/A                                         |
| 0xC00        | Optional Custom Extensions                                                | N/A                                         |

#### Related Information

- [PCI Express Base Specification 3.0](#)
- [PCI Local Bus Specification](#)

### 8.1.1. Register Access Definitions

This document uses the following abbreviations when describing register access.

**Table 55. Register Access Abbreviations**

Sticky bits are not initialized or modified by hot reset or function-level reset.

| Abbreviation | Meaning               |
|--------------|-----------------------|
| RW           | Read and write access |
| RO           | Read only             |
| WO           | Write only            |
| continued... |                       |



| Abbreviation | Meaning                      |
|--------------|------------------------------|
| RW1C         | Read write 1 to clear        |
| RW1CS        | Read write 1 to clear sticky |
| RWS          | Read write sticky            |

### 8.1.2. PCI Configuration Header Registers

The *Correspondence between Configuration Space Registers and the PCIe Specification* lists the appropriate section of the *PCI Express Base Specification* that describes these registers.

**Figure 63. Configuration Space Registers Address Map**

| End Point Capability Structure | Required/Optional | Starting Byte Offset |
|--------------------------------|-------------------|----------------------|
| PCI Header Type 0              | Required          | 0x00                 |
| PCI Power Management           | Required          | 0x40                 |
| MSI                            | Optional          | 0x50                 |
| PCI Express                    | Required          | 0x70                 |
| MSI-X                          | Optional          | 0xB0                 |
| AER                            | Required          | 0x100                |
| Secondary PCIe                 | Required          | 0x188                |
| VSEC                           | Required          | 0xB80                |
| Custom Extensions              | Optional          | 0xC00                |

**Figure 64. PCI Configuration Space Registers - Byte Address Offsets and Layout**

|       | 31                  | 24          | 23 | 16 | 15            | 8                   | 7                    | 0 |  |
|-------|---------------------|-------------|----|----|---------------|---------------------|----------------------|---|--|
| 0x000 | Device ID           |             |    |    |               | Vendor ID           |                      |   |  |
| 0x004 | Status              |             |    |    |               | Command             |                      |   |  |
| 0x008 | Class Code          |             |    |    |               |                     | Revision ID          |   |  |
| 0x00C | 0x00                | Header Type |    |    | 0x00          |                     | Cache Line Size      |   |  |
| 0x010 | BAR Registers       |             |    |    |               |                     |                      |   |  |
| 0x014 | BAR Registers       |             |    |    |               |                     |                      |   |  |
| 0x018 | BAR Registers       |             |    |    |               |                     |                      |   |  |
| 0x01C | BAR Registers       |             |    |    |               |                     |                      |   |  |
| 0x020 | BAR Registers       |             |    |    |               |                     |                      |   |  |
| 0x024 | BAR Registers       |             |    |    |               |                     |                      |   |  |
| 0x028 | Reserved            |             |    |    |               |                     |                      |   |  |
| 0x02C | Subsystem Device ID |             |    |    |               | Subsystem Vendor ID |                      |   |  |
| 0x030 | Reserved            |             |    |    |               |                     |                      |   |  |
| 0x034 | Reserved            |             |    |    |               |                     | Capabilities Pointer |   |  |
| 0x038 | Reserved            |             |    |    |               |                     |                      |   |  |
| 0x03C | 0x00                |             |    |    | Interrupt Pin |                     | Interrupt Line       |   |  |

## Related Information

PCI Express Base Specification 3.0

### 8.1.3. PCI Express Capability Structures

The layout of the most basic Capability Structures are provided below. Refer to the *PCI Express Base Specification* for more information about these registers.

**Figure 65. Power Management Capability Structure - Byte Address Offsets and Layout**

|       |                       |    |                                        |    |                                     |   |               |   |
|-------|-----------------------|----|----------------------------------------|----|-------------------------------------|---|---------------|---|
|       | 31                    | 24 | 23                                     | 16 | 15                                  | 8 | 7             | 0 |
| 0x040 | Capabilities Register |    |                                        |    | Next Cap Ptr                        |   | Capability ID |   |
| 0x04C | Data                  |    | PM Control/Status<br>Bridge Extensions |    | Power Management Status and Control |   |               |   |

**Figure 66. MSI Capability Structure**

|       |                                                                                    |    |    |    |              |   |               |   |
|-------|------------------------------------------------------------------------------------|----|----|----|--------------|---|---------------|---|
|       | 31                                                                                 | 24 | 23 | 16 | 15           | 8 | 7             | 0 |
| 0x050 | Message Control<br>Configuration MSI Control Status<br>Register Field Descriptions |    |    |    | Next Cap Ptr |   | Capability ID |   |
| 0x054 | Message Address                                                                    |    |    |    |              |   |               |   |
| 0x058 | Message Upper Address                                                              |    |    |    |              |   |               |   |
| 0x05C | Reserved                                                                           |    |    |    | Message Data |   |               |   |

**Figure 67. PCI Express Capability Structure - Byte Address Offsets and Layout**

In the following table showing the PCI Express Capability Structure, registers that are not applicable to a device are reserved.

|       | 31                                | 24 23 | 16 15            | 8 7              | 0                           |
|-------|-----------------------------------|-------|------------------|------------------|-----------------------------|
| 0x070 | PCI Express Capabilities Register |       |                  | Next Cap Pointer | PCI Express Capabilities ID |
| 0x074 | Device Capabilities               |       |                  |                  |                             |
| 0x078 | Device Status                     |       | Device Control   |                  |                             |
| 0x07C | Link Capabilities                 |       |                  |                  |                             |
| 0x080 | Link Status                       |       | Link Control     |                  |                             |
| 0x084 | Slot Capabilities                 |       |                  |                  |                             |
| 0x088 | Slot Status                       |       | Slot Control     |                  |                             |
| 0x08C | Root Capabilities                 |       | Root Control     |                  |                             |
| 0x090 | Root Status                       |       |                  |                  |                             |
| 0x094 | Device Compatibilities 2          |       |                  |                  |                             |
| 0x098 | Device Status 2                   |       | Device Control 2 |                  |                             |
| 0x09C | Link Capabilities 2               |       |                  |                  |                             |
| 0x0A0 | Link Status 2                     |       | Link Control 2   |                  |                             |
| 0x0A4 | Slot Capabilities 2               |       |                  |                  |                             |
| 0x0A8 | Slot Status 2                     |       | Slot Control 2   |                  |                             |

**Figure 68. MSI-X Capability Structure**

|       | 31                                   | 24 23 | 16 15        | 8 7           | 3 2 | 0                                       |
|-------|--------------------------------------|-------|--------------|---------------|-----|-----------------------------------------|
| 0x0B0 | Message Control                      |       | Next Cap Ptr | Capability ID |     |                                         |
| 0x0B4 | MSI-X Table Offset                   |       |              |               |     | MSI-X Table BAR Indicator               |
| 0x0B8 | MSI-X Pending Bit Array (PBA) Offset |       |              |               |     | MSI-X Pending Bit Array - BAR Indicator |

**Figure 69. PCI Express AER Extended Capability Structure**

|       |                                                  |    |                                                  |   |
|-------|--------------------------------------------------|----|--------------------------------------------------|---|
|       | 31                                               | 16 | 15                                               | 0 |
| 0x100 | PCI Express Enhanced Capability Register         |    |                                                  |   |
| 0x104 | Uncorrectable Error Status Register              |    |                                                  |   |
| 0x108 | Uncorrectable Error Mask Register                |    |                                                  |   |
| 0x10C | Uncorrectable Error Severity Register            |    |                                                  |   |
| 0x110 | Correctable Error Status Register                |    |                                                  |   |
| 0x114 | Correctable Error Mask Register                  |    |                                                  |   |
| 0x118 | Advanced Error Capabilities and Control Register |    |                                                  |   |
| 0x11C | Header Log Register                              |    |                                                  |   |
| 0x12C | Root Error Command Register                      |    |                                                  |   |
| 0x130 | Root Error Status Register                       |    |                                                  |   |
| 0x134 | Error Source Identification Register             |    | Correctable Error Source Identification Register |   |

**Note:** Refer to the *Advanced Error Reporting Capability* section for more details about the PCI Express AER Extended Capability Structure.

#### Related Information

- [PCI Express Base Specification 3.0](#)
- [PCI Local Bus Specification](#)



### 8.1.4. Intel Defined VSEC Capability Header

The figure below shows the address map and layout of the Intel defined VSEC Capability.

**Figure 70. Vendor-Specific Extended Capability Address Map and Register Layout**

|     |                                              |       |               |                                    |   |
|-----|----------------------------------------------|-------|---------------|------------------------------------|---|
|     | 31                                           | 24 23 | 16 15         | 8 7                                | 0 |
| 00h | Next Cap Offset                              |       | Version       | PCI Express Extended Capability ID |   |
| 04h | VSEC Length                                  |       | VSEC Revision | VSEC ID                            |   |
| 08h | Intel Marker                                 |       |               |                                    |   |
| 0Ch | JTAG Silicon ID DW0                          |       |               |                                    |   |
| 10h | JTAG Silicon ID DW1                          |       |               |                                    |   |
| 14h | JTAG Silicon ID DW2                          |       |               |                                    |   |
| 18h | JTAG Silicon ID DW3                          |       |               |                                    |   |
| 1Ch | Reserved                                     |       |               | User Configurable Device/Board ID  |   |
| 20h | Reserved                                     |       |               |                                    |   |
| 24h | Reserved                                     |       |               |                                    |   |
| 28h | Reserved                                     |       |               |                                    |   |
| 2Ch | Reserved                                     |       |               |                                    |   |
| 30h | General-Purpose Control and Status           |       |               |                                    |   |
| 34h | Uncorrectable Internal Error Status Register |       |               |                                    |   |
| 38h | Uncorrectable Internal Error Mask Register   |       |               |                                    |   |
| 3Ch | Correctable Internal Error Status Register   |       |               |                                    |   |
| 40h | Correctable Internal Error Mask Register     |       |               |                                    |   |
| 44h | Reserved                                     |       |               |                                    |   |
| 48h | Reserved                                     |       |               |                                    |   |
| 4Ch | Reserved                                     |       |               |                                    |   |
| 50h | Reserved                                     |       |               |                                    |   |
| 54h | Reserved                                     |       |               |                                    |   |
| 58h | Reserved                                     |       |               |                                    |   |

**Table 56. Intel-Defined VSEC Capability Header - 0xB80**

| Bits    | Register Description                                                                                                  | Default Value | Access |
|---------|-----------------------------------------------------------------------------------------------------------------------|---------------|--------|
| [31:20] | Next Capability Pointer: Value is the starting address of the next Capability Structure implemented. Otherwise, NULL. | Variable      | RO     |
| [19:16] | Version. PCIe specification defined value for VSEC version.                                                           | 1             | RO     |
| [15:0]  | PCI Express Extended Capability ID. PCIe specification defined value for VSEC Capability ID.                          | 0x000B        | RO     |

#### 8.1.4.1. Intel Defined Vendor Specific Header

**Table 57. Intel defined Vendor Specific Header - 0xB84**

| Bits    | Register Description                                                             | Default Value | Access |
|---------|----------------------------------------------------------------------------------|---------------|--------|
| [31:20] | VSEC Length. Total length of this structure in bytes.                            | 0x5C          | RO     |
| [19:16] | VSEC. User configurable VSEC revision.                                           | Not available | RO     |
| [15:0]  | VSEC ID. User configurable VSEC ID. You should change this ID to your Vendor ID. | 0x1172        | RO     |

#### 8.1.4.2. Intel Marker

**Table 58. Intel Marker - 0xB88**

| Bits   | Register Description                                                                                                               | Default Value | Access |
|--------|------------------------------------------------------------------------------------------------------------------------------------|---------------|--------|
| [31:0] | Intel Marker - An additional marker for standard Intel programming software to be able to verify that this is the right structure. | 0x41721172    | RO     |

#### 8.1.4.3. JTAG Silicon ID

This read only register returns the JTAG Silicon ID. The Intel Programming software uses this JTAG ID to make ensure that it is programming the SRAM Object File (\*.sof) .

**Table 59. JTAG Silicon ID - 0xB8C-0xB98**

| Bits   | Register Description | Default Value<br>(5) | Access |
|--------|----------------------|----------------------|--------|
| [31:0] | JTAG Silicon ID DW3  | Unique ID            | RO     |
| [31:0] | JTAG Silicon ID DW2  | Unique ID            | RO     |
| [31:0] | JTAG Silicon ID DW1  | Unique ID            | RO     |
| [31:0] | JTAG Silicon ID DW0  | Unique ID            | RO     |

#### 8.1.4.4. User Configurable Device and Board ID

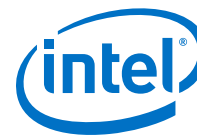
**Table 60. User Configurable Device and Board ID - 0xB9C**

| Bits   | Register Description                                    | Default Value           | Access |
|--------|---------------------------------------------------------|-------------------------|--------|
| [15:0] | Allows you to specify ID of the .sof file to be loaded. | From configuration bits | RO     |

#### 8.1.5. General Purpose Control and Status Register

This register provides up to eight I/O pins for Application Layer control and status requirements. This feature supports Partial Reconfiguration of the FPGA fabric. Partial Reconfiguration only requires one input and one output pin. The other seven I/Os make this interface extensible.

(5) Because the Silicon ID is a unique value, it does not have a global default value.



**Table 61. General Purpose Control and Status Register - 0xBB0**

| Bits    | Register Description                                                   | Default Value | Access |
|---------|------------------------------------------------------------------------|---------------|--------|
| [31:16] | Reserved.                                                              | N/A           | RO     |
| [15:8]  | General Purpose Status. The Application Layer can read status bits.    | 0             | RO     |
| [7:0]   | General Purpose Control. The Application Layer can write control bits. | 0             | RW     |

### 8.1.6. Uncorrectable Internal Error Status Register

This register reports the status of the internally checked errors that are uncorrectable. When these specific errors are enabled by the Uncorrectable Internal Error Mask register, they are forwarded as Uncorrectable Internal Errors. This register is for debug only. Only use this register to observe behavior, not to drive logic custom logic.

**Table 62. Uncorrectable Internal Error Status Register - 0xBB4**

This register is for debug only. It should only be used to observe behavior, not to drive custom logic.

| Bits    | Register Description                                           | Reset Value | Access |
|---------|----------------------------------------------------------------|-------------|--------|
| [31:13] | Reserved.                                                      | 0           | RO     |
| [12]    | Debug bus interface (DBI) access error status.                 | 0           | RW1CS  |
| [11]    | ECC error from Config RAM block.                               | 0           | RW1CS  |
| [10]    | Uncorrectable ECC error status for Retry Buffer.               | 0           | RO     |
| [9]     | Uncorrectable ECC error status for Retry Start of the TLP RAM. | 0           | RW1CS  |
| [8]     | RX Transaction Layer parity error reported by the IP core.     | 0           | RW1CS  |
| [7]     | TX Transaction Layer parity error reported by the IP core.     | 0           | RW1CS  |
| [6]     | Internal error reported by the FPGA.                           | 0           | RW1CS  |
| [5:4]   | Reserved.                                                      | 0           | RW1CS  |
| [3]     | Uncorrectable ECC error status for RX Buffer Header #2 RAM.    | 0           | RW1CS  |
| [2]     | Uncorrectable ECC error status for RX Buffer Header #1 RAM.    | 0           | RW1CS  |
| [1]     | Uncorrectable ECC error status for RX Buffer Data RAM #2.      | 0           | RW1CS  |
| [0]     | Uncorrectable ECC error status for RX Buffer Data RAM #1.      | 0           | RW1CS  |

### 8.1.7. Uncorrectable Internal Error Mask Register

The Uncorrectable Internal Error Mask register controls which errors are forwarded as internal uncorrectable errors.

**Table 63. Uncorrectable Internal Error Mask Register - 0xBB8**

The access code RWS stands for Read Write Sticky meaning the value is retained after a soft reset of the IP core.

| Bits    | Register Description          | Reset Value | Access |
|---------|-------------------------------|-------------|--------|
| [31:13] | Reserved.                     | 1b'0        | RO     |
| [12]    | Mask for Debug Bus Interface. | 1b'1        | RO     |

*continued...*

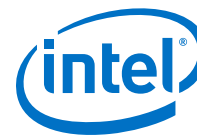
| Bits | Register Description                                                 | Reset Value | Access |
|------|----------------------------------------------------------------------|-------------|--------|
| [11] | Mask for ECC error from Config RAM block.                            | 1b'1        | RWS    |
| [10] | Mask for Uncorrectable ECC error status for Retry Buffer.            | 1b'1        | RO     |
| [9]  | Mask for Uncorrectable ECC error status for Retry Start of TLP RAM.  | 1b'1        | RWS    |
| [8]  | Mask for RX Transaction Layer parity error reported by IP core.      | 1b'1        | RWS    |
| [7]  | Mask for TX Transaction Layer parity error reported by IP core.      | 1b'1        | RWS    |
| [6]  | Mask for Uncorrectable Internal error reported by the FPGA.          | 1b'1        | RO     |
| [5]  | Reserved.                                                            | 1b'0        | RWS    |
| [4]  | Reserved.                                                            | 1b'1        | RWS    |
| [3]  | Mask for Uncorrectable ECC error status for RX Buffer Header #2 RAM. | 1b'1        | RWS    |
| [2]  | Mask for Uncorrectable ECC error status for RX Buffer Header #1 RAM. | 1b'1        | RWS    |
| [1]  | Mask for Uncorrectable ECC error status for RX Buffer Data RAM #2.   | 1b'1        | RWS    |
| [0]  | Mask for Uncorrectable ECC error status for RX Buffer Data RAM #1.   | 1b'1        | RWS    |

### 8.1.8. Correctable Internal Error Status Register

**Table 64. Correctable Internal Error Status Register - 0xBBBC**

The Correctable Internal Error Status register reports the status of the internally checked errors that are correctable. When these specific errors are enabled by the Correctable Internal Error Mask register, they are forwarded as Correctable Internal Errors. This register is for debug only. Only use this register to observe behavior, not to drive logic custom logic.

| Bits    | Register Description                                                                                                                                                | Reset Value | Access |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|--------|
| [31:12] | Reserved.                                                                                                                                                           | 0           | RO     |
| [11]    | Correctable ECC error status for Config RAM.                                                                                                                        | 0           | RW1CS  |
| [10]    | Correctable ECC error status for Retry Buffer.                                                                                                                      | 0           | RW1CS  |
| [9]     | Correctable ECC error status for Retry Start of TLP RAM.                                                                                                            | 0           | RW1CS  |
| [8]     | Reserved.                                                                                                                                                           | 0           | RO     |
| [7]     | Reserved.                                                                                                                                                           | 0           | RO     |
| [6]     | Internal Error reported by FPGA.                                                                                                                                    | 0           | RW1CS  |
| [5]     | Reserved                                                                                                                                                            | 0           | RO     |
| [4]     | PHY Gen3 SKP Error occurred. Gen3 data pattern contains SKP pattern (8'b10101010) is misinterpreted as a SKP OS and causing erroneous block realignment in the PHY. | 0           | RW1CS  |
| [3]     | Correctable ECC error status for RX Buffer Header RAM #2.                                                                                                           | 0           | RW1CS  |
| [2]     | Correctable ECC error status for RX Buffer Header RAM #1.                                                                                                           | 0           | RW1CS  |
| [1]     | Correctable ECC error status for RX Buffer Data RAM #2.                                                                                                             | 0           | RW1CS  |
| [0]     | Correctable ECC error status for RX Buffer Data RAM #1.                                                                                                             | 0           | RW1CS  |



## 8.1.9. Correctable Internal Error Mask Register

**Table 65. Correctable Internal Error Status Register - 0xBBBC**

The Correctable Internal Error Status register controls which errors are forwarded as Internal Correctable Errors.

| Bits    | Register Description                                               | Reset Value | Access |
|---------|--------------------------------------------------------------------|-------------|--------|
| [31:12] | Reserved.                                                          | 0           | RO     |
| [11]    | Mask for correctable ECC error status for Config RAM.              | 0           | RWS    |
| [10]    | Mask for correctable ECC error status for Retry Buffer.            | 1           | RWS    |
| [9]     | Mask for correctable ECC error status for Retry Start of TLP RAM.  | 1           | RWS    |
| [8]     | Reserved.                                                          | 0           | RO     |
| [7]     | Reserved.                                                          | 0           | RO     |
| [6]     | Mask for internal Error reported by FPGA.                          | 0           | RWS    |
| [5]     | Reserved                                                           | 0           | RO     |
| [4]     | Mask for PHY Gen3 SKP Error.                                       | 1           | RWS    |
| [3]     | Mask for correctable ECC error status for RX Buffer Header RAM #2. | 1           | RWS    |
| [2]     | Mask for correctable ECC error status for RX Buffer Header RAM #1. | 1           | RWS    |
| [1]     | Mask for correctable ECC error status for RX Buffer Data RAM #.    | 1           | RWS    |
| [0]     | Mask for correctable ECC error status for RX Buffer Data RAM #1.   | 1           | RWS    |

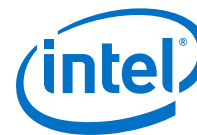
## 8.1.10. SR-IOV Virtualization Extended Capabilities Registers Address Map

Figure 71. SR-IOV Virtualization Extended Capabilities Registers

|       |                                            |                                     |       |                      |   |
|-------|--------------------------------------------|-------------------------------------|-------|----------------------|---|
|       | 31                                         | 24 23                               | 20 19 | 16 15                | 0 |
| 0x1B8 | SR-IOV Extended Capability Header Register |                                     |       |                      |   |
| 0x1BC | SR-IOV Capabilities                        |                                     |       |                      |   |
| 0x1C0 | SR-IOV Status                              |                                     |       | SR-IOV Control       |   |
| 0x1C4 | TotalVFs (RO)                              |                                     |       | InitialVFs (RO)      |   |
| 0x1C8 | RsvdP                                      | Function<br>Dependency<br>Link (RO) |       | NumVFs (RW)          |   |
| 0x1CC | VF Stride (RO)                             |                                     |       | First VF Offset (RO) |   |
| 0x1D0 | VF Device ID (RO)                          |                                     |       | RsvdP                |   |
| 0x1D4 | Supported Pages Sizes (RO)                 |                                     |       |                      |   |
| 0x1D8 | System Page Size (RW)                      |                                     |       |                      |   |
| 0x1DC | VF BAR0 (RW)                               |                                     |       |                      |   |
| 0x1E0 | VF BAR1 (RW)                               |                                     |       |                      |   |
| 0x1E4 | VF BAR2 (RW)                               |                                     |       |                      |   |
| 0x1E8 | VF BAR3 (RW)                               |                                     |       |                      |   |
| 0x1EC | VF BAR4 (RW)                               |                                     |       |                      |   |
| 0x1F0 | VF BAR5 (RW)                               |                                     |       |                      |   |

Table 66. SR-IOV Virtualization Extended Capabilities Registers

| Byte Address Offset                                                 | Name                                          | Description                                                                                                                                            |
|---------------------------------------------------------------------|-----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Alternative RID (ARI) Capability Structure</b>                   |                                               |                                                                                                                                                        |
| 0x178                                                               | ARI Enhanced Capability Header                | PCI Express Extended Capability ID for ARI and next capability pointer.                                                                                |
| 0x017C                                                              | ARI Capability Register, ARI Control Register | The lower 16 bits implement the ARI Capability Register and the upper 16 bits implement the ARI Control Register.                                      |
| <b>Single-Root I/O Virtualization (SR-IOV) Capability Structure</b> |                                               |                                                                                                                                                        |
| 0x1B8                                                               | SR-IOV Extended Capability Header             | PCI Express Extended Capability ID for SR-IOV and next capability pointer.                                                                             |
| 0x1BC                                                               | SR-IOV Capabilities Register                  | Lists supported capabilities of the SR-IOV implementation.                                                                                             |
| 0x1C0                                                               | SR-IOV Control and Status Registers           | The lower 16 bits implement the SR-IOV Control Register. The upper 16 bits implement the SR-IOV Status Register.                                       |
| 0x1C4                                                               | InitialVFs/TotalVFs                           | The lower 16 bits specify the initial number of VFs attached to PF0. The upper 16 bits specify the total number of PFs available for attaching to PF0. |
| <i>continued...</i>                                                 |                                               |                                                                                                                                                        |



| Byte Address Offset                                                          | Name                                             | Description                                                                                                                                                                      |
|------------------------------------------------------------------------------|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x1C8                                                                        | Function Dependency Link, NumVFs                 | The Function Dependency field describes dependencies between Physical Functions. The NumVFs field contains the number of VFs currently configured for use.                       |
| 0x1CC                                                                        | VF Offset/Stride                                 | Specifies the offset and stride values used to assign routing IDs to the VFs.                                                                                                    |
| 0x1D0                                                                        | VF Device ID                                     | Specifies VF Device ID assigned to the device.                                                                                                                                   |
| 0x1D4                                                                        | Supported Page Sizes                             | Specifies all page sizes supported by the device.                                                                                                                                |
| 0x1D8                                                                        | System Page Size                                 | Stores the page size currently selected.                                                                                                                                         |
| 0x1DC                                                                        | VF BAR 0                                         | VF Base Address Register 0. Can be used independently as a 32-bit BAR, or combined with VF BAR 1 to form a 64-bit BAR.                                                           |
| 0x1E0                                                                        | VF BAR 1                                         | VF Base Address Register 1. Can be used independently as a 32-bit BAR, or combined with VF BAR 0 to form a 64-bit BAR.                                                           |
| 0x1E4                                                                        | VF BAR 2                                         | VF Base Address Register 2. Can be used independently as a 32-bit BAR, or combined with VF BAR 3 to form a 64-bit BAR.                                                           |
| 0x1E8                                                                        | VF BAR 3                                         | VF Base Address Register 3. Can be used independently as a 32-bit BAR, or combined with VF BAR 2 to form a 64-bit BAR.                                                           |
| 0x1EC                                                                        | VF BAR 4                                         | VF Base Address Register 4. Can be used independently as a 32-bit BAR, or combined with VF BAR 5 to form a 64-bit BAR.                                                           |
| 0x1F0                                                                        | VF BAR 5                                         | VF Base Address Register 5. Can be used independently as a 32-bit BAR, or combined with VF BAR 4 to form a 64-bit BAR.                                                           |
| 0x1F4                                                                        | VF Migration State Array Offset                  | Not implemented.                                                                                                                                                                 |
| <b>Secondary PCI Express Extended Capability Structure (Gen3, PF 0 only)</b> |                                                  |                                                                                                                                                                                  |
| 0x280                                                                        | Secondary PCI Express Extended Capability Header | PCI Express Extended Capability ID for Secondary PCI Express Capability, and next capability pointer.                                                                            |
| 0x284                                                                        | Link Control 3 Register                          | Not implemented.                                                                                                                                                                 |
| 0x288                                                                        | Lane Error Status Register                       | Per-lane error status bits.                                                                                                                                                      |
| 0x28C                                                                        | Lane Equalization Control Register 0             | Transmitter Preset and Receiver Preset Hint values for Lanes 0 and 1 of remote device. These values are captured during Link Equalization.                                       |
| 0x290                                                                        | Lane Equalization Control Register 1             | Transmitter Preset and Receiver Preset Hint values for Lanes 2 and 3 of remote device. These values are captured during Link Equalization.                                       |
| 0x294                                                                        | Lane Equalization Control Register 2             | Transmitter Preset and Receiver Preset Hint values for Lanes 4 and 5 of remote device. These values are captured during Link Equalization.                                       |
| 0x298                                                                        | Lane Equalization Control Register 3             | Transmitter Preset and Receiver Preset Hint values for Lanes 6 and 7 of remote device. These values are captured during Link Equalization.                                       |
| <b>Transaction Processing Hints (TPH) Requester Capability Structure</b>     |                                                  |                                                                                                                                                                                  |
| 0x1F8                                                                        | TPH Requester Extended Capability Header         | PCI Express Extended Capability ID for TPH Requester Capability, and next capability pointer.                                                                                    |
| 0x1FC                                                                        | TPH Requester Capability Register                | PCI Express Extended Capability ID for TPH Requester Capability, and next capability pointer. This register contains the advertised parameters for the TPH Requester Capability. |
| 0x1D0                                                                        | TPH Requester Control Register                   | This register contains enable and mode select bits for the TPH Requester Capability.                                                                                             |
| <b>continued...</b>                                                          |                                                  |                                                                                                                                                                                  |

| Byte Address Offset                                            | Name                                             | Description                                                                                    |
|----------------------------------------------------------------|--------------------------------------------------|------------------------------------------------------------------------------------------------|
| <b>Address Translation Services (ATS) Capability Structure</b> |                                                  |                                                                                                |
| 0x284                                                          | ATS Extended Capability Header                   | PCI Express Extended Capability ID for ATS Capability, and next capability pointer.            |
| 0x288                                                          | ATS Capability Register and ATS Control Register | This location contains the 16-bit ATS Capability Register and the 16-bit ATS Control Register. |

### 8.1.10.1. ARI Enhanced Capability Header

**Table 67. ARI Enhanced Capability ID - 0x178**

| Bits    | Register Description<br>Default Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Default Value   | Access |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------|
| [15:0]  | PCI Express Extended Capability ID for ARI.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 0x000E          | RO     |
| [19:16] | Capability Version.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 0x1             | RO     |
| [31:20] | <ul style="list-style-type: none"> <li>If the number of VFs attached to this PFs is non-zero, this pointer points to the SR-IOV Capability, 0x200). Otherwise, its value is configured as follows: <ul style="list-style-type: none"> <li>PF0 with maximum link speed of 8.0 GT/s: Next Capability = Secondary PCIe, 0x280.</li> <li>PF0 with maximum link speed of 2.5 or 5.0 GT/s and TPH Requester Capability: Next Capability = TPH Requester, 0x300.</li> <li>PF0 with maximum link speed of 2.5 or 5.0 GT/s, with ATS Capability and no TPH Requester Capability : Next Capability = ATS, 0x3C0.</li> <li>PFs 1–3 with TPH Requester Capability: Next Capability = TPH Requester, 0x300.</li> <li>PFs 1–3 with ATS Capability: Next Capability = ATS, 0x3C0.</li> <li>All other cases: Next Capability = 0.</li> </ul> </li> </ul> | See description | RO     |

**Table 68. ARI Enhanced Capability Header and Control Register -0x17C**

| Bits    | Register Description                                                            | Default Value | Access |
|---------|---------------------------------------------------------------------------------|---------------|--------|
| [0]     | Specifies support for arbitration at the Function group level. Not implemented. | 0             | RO     |
| [7:1]   | Reserved.                                                                       | 0             | RO     |
| [15:8]  | ARI Next Function Pointer. Pointer to the next PF.                              | 1             | RO     |
| [31:16] | Reserved.                                                                       | 0             | RO     |



### 8.1.10.2. SR-IOV Enhanced Capability Registers

**Table 69. SR-IOV Extended Capability Header Register - 0x1B8**

| Bits    | Register Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Default Value            | Access |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|--------|
| [15:0]  | PCI Express Extended Capability ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 0x0010                   | RO     |
| [19:16] | Capability Version                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 1                        | RO     |
| [31:16] | <p>Next Capability Pointer: The value depends on data rate. If the number of VFs attached to this PFs is non-zero, this pointer points to the SR-IOV Extended Capability, 0x200. Otherwise, its value is configured as follows:</p> <ul style="list-style-type: none"> <li>PF0 has a maximum link speed of 8.0 GT/s: Next Capability = Secondary PCIe, 0x280.</li> <li>PF0 has a maximum link speed of 2.5 or 5.0 GT/s and TPH Requester Capability: Next Capability = TPH Requester, 0x1FC.</li> <li>PF0 has a maximum link speed of 2.5 or 5.0 GT/s, with ATS Capability and no TPH Requester Capability : Next Capability = ATS, 0x284.</li> <li>PFs 1–3 with TPH Requester Capability: Next Capability = TPH Requester, 0x1FC.</li> <li>PFs 1–3 with ATS Capability: Next Capability = ATS, 0x284.</li> <li>All other cases: Next Capability = 0.</li> </ul> | Set in Platform Designer | RO     |

**Table 70. SR-IOV Capabilities Register - 0x1BC**

| Bits   | Register Description            | Default Value                                                          | Access |
|--------|---------------------------------|------------------------------------------------------------------------|--------|
| [0]    | VF Migration Capable            | 0                                                                      | RO     |
| [1]    | ARI Capable Hierarchy Preserved | 1, for the lowest-numbered PF with SR-IOV Capability; 0 for other PFs. | RO     |
| [31:2] | Reserved                        | 0<br>Default Value                                                     | RO     |

**Table 71. SR-IOV Control and Status Registers - 0x1C0**

| Bits    | Register Description                            | Default Value | Access                                                                  |
|---------|-------------------------------------------------|---------------|-------------------------------------------------------------------------|
| [0]     | VF Enable                                       | 0             | RW                                                                      |
| [1]     | VF Migration Enable. Not implemented.           | 0             | RO                                                                      |
| [2]     | VF Migration Interrupt Enable. Not implemented. | 0             | RO                                                                      |
| [3]     | VF Memory Space Enable                          | 0             | RW                                                                      |
| [4]     | ARI Capable Hierarchy                           | 0             | RW, for the lowest-numbered PF with SR-IOV Capability; RO for other PFs |
| [15:5]  | Reserved                                        | 0             | RO                                                                      |
| [31:16] | SR-IOV Status Register. Not implemented         | 0             | RO                                                                      |

### 8.1.10.3. Initial VFs and Total VFs Registers

**Table 72. Initial VFs and Total VFs Registers - 0x1C4**

| Bits    | Description                                                              | Default Value            | Access |
|---------|--------------------------------------------------------------------------|--------------------------|--------|
| [15:0]  | Initial VFs. Specifies the initial number of VFs configured for this PF. | Same value as TotalVFs   | RO     |
| [31:16] | Total VFs. Specifies the total number of VFs attached to this PF.        | Set in Platform Designer | RO     |

**Table 73. Function Dependency Link and NumVFs Registers -0x1C8**

| Bit Location | Description                                                                                                                        | Default Value | Access |
|--------------|------------------------------------------------------------------------------------------------------------------------------------|---------------|--------|
| [15:0]       | NumVFs. Specifies the number of VFs enabled for this PF. Writable only when the VF Enable bit in the SR-IOV Control Register is 0. | 0             | RW     |
| [31:16]      | Function Dependency Link                                                                                                           | 0             | RO     |

**Table 74. VF Offset and Stride Registers -0x1CC**

| Bits    | Register Description | Default Value | Access |
|---------|----------------------|---------------|--------|
| [31:16] | VF Stride            | 1             | RO     |

### 8.1.10.4. VF Device ID Register

**Table 75. VF Device ID Register - 0x1D0**

| Bits    | Register Description | Default Value            | Access |
|---------|----------------------|--------------------------|--------|
| [15:0]  | Reserved             | 0                        | RO     |
| [31:16] | VF Device ID         | Set in Platform Designer | RO     |

### 8.1.10.5. Page Size Registers

**Table 76. Supported Page Size Register -0x1D4**

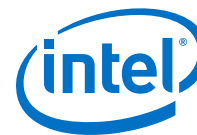
| Bits   | Register Description                                                   | Default Value            | Access |
|--------|------------------------------------------------------------------------|--------------------------|--------|
| [31:0] | Supported Page Sizes. Specifies the page sizes supported by the device | Set in Platform Designer | RO     |

**Table 77. System Page Size Register - 0x1D8**

| Bits   | Register Description                                            | Default Value            | Access |
|--------|-----------------------------------------------------------------|--------------------------|--------|
| [31:0] | Supported Page Sizes. Specifies the page size currently in use. | Set in Platform Designer | RO     |

### 8.1.10.6. VF Base Address Registers (BARs) 0-5

Each PF implements six BARs. You can specify BAR settings in Platform Designer. You can configure VF BARs as 32-bit memories. Or you can combine VF BAR0 and BAR1 to form a 64-bit memory BAR. VF BAR 0 may also be designated as prefetchable or non-prefetchable in Platform Designer. Finally, the address range of VF BAR 0 can be configured as any power of 2 between 128 bytes and 2 GB.



The contents of VF BAR 0 are described below:

**Table 78. VF BARs 0-5 - 0x1DC-0x1F0**

| Bits   | Register Description                                                                                                                                                                                                            | Default Value                          | Access          |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|-----------------|
| [0]    | Memory Space Indicator: Hardwired to 0 to indicate the BAR defines a memory address range.                                                                                                                                      | 0                                      | RO              |
| [1]    | Reserved. Hardwired to 0.                                                                                                                                                                                                       | 0                                      |                 |
| [2]    | Specifies the BAR size.: The following encodings are defined: <ul style="list-style-type: none"> <li>1'b0: 32-bit BAR</li> <li>1'b1: 64-bit BAR created by pairing BAR0 with BAR1, BAR2 with BAR3, or BAR4 with BAR5</li> </ul> | 0                                      | RO              |
| [3]    | When 1, indicates that the data within the address range refined by this BAR is prefetchable. When 1, indicates that the data is not prefetchable. Data is prefetchable if reading is guaranteed not to have side-effects .     | Prefetchable: 0<br>Non-Prefetchable: 1 | RO              |
| [7:4]  | Reserved. Hardwired to 0.                                                                                                                                                                                                       | 0                                      | RO              |
| [31:8] | Base address of the BAR. The number of writable bits is based on the BAR access size. For example, if bits [15:8] are hardwired to 0, if the BAR access size is 64 KB. Bits [31:16] can be read and written.                    | 0                                      | See description |

### 8.1.10.7. Secondary PCI Express Extended Capability Header

**Table 79. Secondary PCI Express Extended Capability Header - 0x188**

| Bits    | Register Description                                                                                                                                                                                                                                                                                                                                                            | Default Value      | Access |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|--------|
| [15:0]  | PCI Express Extended Capability ID.                                                                                                                                                                                                                                                                                                                                             | 0x0019             | RO     |
| [19:16] | Capability Version.                                                                                                                                                                                                                                                                                                                                                             | 0x1                | RO     |
| [31:20] | Next Capability Pointer. The following values are possible: <ul style="list-style-type: none"> <li>If TPH Requester Capability is supported by the Function, its Next Capability = TPH Requester, 0x1FC.</li> <li>Otherwise, if the ATS Capability is supported by the Function, its Next Capability = ATS, 0x284.</li> <li>In all other cases: Next Capability = 0.</li> </ul> | 0x1FC, 0x284, or 0 | RO     |

### 8.1.10.8. Lane Status Registers

**Table 80. Lane Error Status Register - 0x18C**

| Bits   | Register Description                                                                                                                                                                                                                                                                                          | Default Value | Access |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------|
| [7:0]  | Lane Error Status: Each 1 indicates an error was detected in the corresponding lane. Only Bit 0 is implemented when the link width is 1. Bits [1:0] are implemented when the link width is 2, and so on. The other bits read as 0. This register is present only in PF0 when the maximum data rate is 8 Gbps. | 0             | RW1CS  |
| [31:8] | Reserved                                                                                                                                                                                                                                                                                                      | 0             | RO     |

**Table 81. Lane Equalization Control Registers 0–3 - 0x190-0x19C**

This register contains the Transmitter Preset and the Receiver Preset Hint values. The Training Sequences capture these values during Link Equalization. This register is present only in PF0 when the maximum data rate is 8 Gbps. Lane Equalization Control Registers 0 at address 0x20C records values for lanes 0 and 1. Lane Equalization Control Registers 0 at address 0x20C records values for lanes 2 and 3, and so on.

| Bits    | Register Description                      | Default Value                                    | Access |
|---------|-------------------------------------------|--------------------------------------------------|--------|
| [6:0]   | Reserved                                  | 0x7F                                             | RO     |
| [7]     | Reserved                                  | 0                                                | RO     |
| [11:8]  | Upstream Port Lane 0 Transmitter Preset   | 0xF                                              | RO     |
| [14:12] | Upstream Port Lane 0 Receiver Preset Hint | 0x7                                              | RO     |
| [15]    | Reserved                                  | 0                                                | RO     |
| [22:16] | Reserved                                  | 0x7F                                             | RO     |
| [23]    | Reserved                                  | 0                                                | RO     |
| [27:24] | Upstream Port Lane 1 Transmitter Preset   | 0xF when link width > 1<br>0 when link width = 1 | RO     |
| [30:28] | Upstream Port Lane 1 Receiver Preset Hint | 0x7 when link width > 1<br>0 when link width = 1 | RO     |
| [31]    | Reserved                                  | 0                                                | RO     |

#### 8.1.10.9. Transaction Processing Hints (TPH) Requester Enhanced Capability Header

**Table 82. Transaction Processing Hints (TPH) Requester Enhanced Capability Header Register - 0x1F80x1F8**

| Bits    | Register Description                                                           | Default Value | Access |
|---------|--------------------------------------------------------------------------------|---------------|--------|
| [31:20] | Next Capability Pointer: Points to ATS Capability when preset, NULL otherwise. | 0x0017        | RO     |
| [19:16] | Capability Version.                                                            | 1             | RO     |
| [15:0]  | PCI Express Extended Capability ID.                                            |               | RO     |

#### 8.1.10.10. TPH Requester Capability Register

**Table 83. TPH Requester Capability Register - 0x1FC**

| Bits    | Register Description                                                                                                                                                                                                                                 | Default Value            | Access |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|--------|
| [31:27] | Reserved.                                                                                                                                                                                                                                            | 0                        | RO     |
| [26:16] | ST Table Size: Specifies the number of entries in the Steering Tag Table. When set to 0, the table has 1 entry. When set to 1, the table has 2 entries. The maximum table size is 2048 entries when located in the MSI-X table Each entry is 8 bits. | Set in Platform Designer | RO     |

*continued...*



| Bits    | Register Description                                                                                                                                                                                                                                                                                                                                                                                                             | Default Value            | Access |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|--------|
| [15:11] | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                         | 0                        | RO     |
| [10:9]  | ST Table Location: Setting this field indicates if a Steering Tag Table is implemented for this Function. The following encodings are defined: <ul style="list-style-type: none"> <li>2'b'00: ST Table not present</li> <li>2'b'01: ST Table stored in the TPH Requestor Capability Structure</li> <li>2'b'10: ST values stored in the MSI-X Table in client RAM</li> <li>2'b'11: Reserved</li> </ul> Valid settings are 0 or 2. | Set in Platform Designer | RO     |
| [8]     | Extended TPH Requester Supported: When set to 1, indicates that the function is capable of generating requests with 16-bit Steering Tags, using TLP Prefix. This bit is permanently set to 0.                                                                                                                                                                                                                                    | 0                        | RO     |
| [7:3]   | Reserved.                                                                                                                                                                                                                                                                                                                                                                                                                        | 0                        | RO     |
| [2]     | Device-Specific Mode Supported: A setting of 1 indicates that the function supports the Device-Specific Mode for TPH Steering Tag generation. The client typically chooses the Steering Tag values from the ST Table, but is not required to do so.                                                                                                                                                                              | Set in Platform Designer | RO     |
| [1]     | Interrupt Vector Mode Supported: A setting of 1 indicates that the function supports the Interrupt Vector Mode for TPH Steering Tag generation. In the Interrupt Vector Mode, Steering Tags are attached to MSI/MSI-X interrupt requests. The MSI/MSI-X interrupt vector number selects the Steering Tag for each interrupt.                                                                                                     | Set in Platform Designer | RO     |
| [0]     | No ST Mode Supported: When set to 1, indicates that the function supports the <i>No ST Mode</i> for the generation of TPH Steering Tags. In the No ST Mode, the device must use a Steering Tag value of 0 for all requests.<br><br>This bit is hardwired to 1, because all TPH Requesters are required to support the No ST Mode of operation.                                                                                   | 1                        | RO     |

### 8.1.10.11. TPH Requester Control Register

**Table 84. TPH Requester Control Register - 0x1D0**

| Bits   | Register Description                                                                                                                                                                                                                              | Default Value | Access |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------|
| [31:9] | Reserved.                                                                                                                                                                                                                                         | 0             | RO     |
| [8]    | TPH Requester Enable: When set to 1, the Function can generate requests with Transaction Processing Hints.                                                                                                                                        | 0             | RW     |
| [7:3]  | Reserved.                                                                                                                                                                                                                                         | 0             | RO     |
| [2:0]  | ST Mode. The following encodings are defined: <ul style="list-style-type: none"> <li>3'b'000: No Steering Tag Mode</li> <li>3'b'001: Interrupt Vector Mode</li> <li>3'b'010: Device-specific mode</li> <li>3'b'011 - 3'b'111: Reserved</li> </ul> | 0             | RW     |

### 8.1.10.12. Address Translation Services ATS Enhanced Capability Header

**Table 85. Address Translation Services ATS Enhanced Capability Header Register - 0x284**

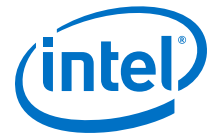
| Bits    | Register Description                     | Default Value | Access |
|---------|------------------------------------------|---------------|--------|
| [31:20] | Next Capability Pointer: Points to NULL. | 0             | RO     |
| [19:16] | Capability Version.                      | 1             | RO     |
| [15:0]  | PCI Express Extended Capability ID       | 0x003C        | RO     |

### 8.1.10.13. ATS Capability Register and ATS Control Register

**Table 86. ATS Capability Register and ATS Control Register -0x288**

The lower 16 bits are the ATS Capability Register and the upper 16 bits are the ATS Control Register

| Bits   | Register Description                                                                                                                                                                                                                                                                                                                            | Default Value            | Access |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|--------|
| [15]   | Enable bit. When set, the Function can cache translations.                                                                                                                                                                                                                                                                                      | 0                        | RW     |
| [14:5] | Reserved.                                                                                                                                                                                                                                                                                                                                       | 0                        | RO     |
| [4:0]  | Smallest Translation Unit (STU): This value specifies the minimum number of 4096-byte blocks specified in a Translation Completion or Invalidate Request. This is a power of 2 multiplier. The number of blocks is $2^{\text{STU}}$ . A value of 0 indicates one block and a value of 0x1F indicates $2^{31}$ blocks, or 8 terabyte (TB) total. | 0                        | RW     |
| [15:6] | Reserved.                                                                                                                                                                                                                                                                                                                                       | 0                        | RO     |
| [5]    | Page Aligned Request: If set, indicates the untranslated address is always aligned to a 4096-byte boundary. This bit is hardwired to 1.                                                                                                                                                                                                         | 1                        | RO     |
| [4:0]  | Invalidate Queue Depth: The number of Invalidate Requests that the Function can accept before throttling the upstream connection. If 0, the Function can accept 32 Invalidate Requests.                                                                                                                                                         | Set in Platform Designer | RO     |



## 9. Testbench and Design Example

---

This chapter introduces the Endpoint design example including a testbench, BFM, and a test driver module. You can create this design example using design flows described in *Quick Start Guide*.

This testbench simulates up to x8 variants. It supports x16 variants by down-training to x8. To simulate all lanes of a x16 variant, you can create a simulation model in Platform Designer to use in an Avery testbench. For more information refer to *AN-811: Using the Avery BFM for PCI Express x16 Simulation on Intel Stratix 10 Devices*.

This testbench simulates up to x8 variants. It supports x16 variants by down-training to x8. To simulate all lanes of a x16 variant, you can create a simulation model in Platform Designer to use in an Avery testbench. For more information refer to *AN-811: Using the Avery BFM for PCI Express x16 Simulation on Intel Stratix 10 Devices*.

When configured as an Endpoint variation, the testbench instantiates a design example and a Root Port BFM which provides the following functions:

- A configuration routine that sets up all the basic configuration registers in the Endpoint. This configuration allows the Endpoint application to be the target and initiator of PCI Express transactions.
- A Verilog HDL procedure interface to initiate PCI Express transactions to the Endpoint.

This testbench simulates a single Endpoint DUT.

The testbench uses a test driver module, `altpciethb_bfm_rp_<gen>_x8.sv`, to exercise the target memory. At startup, the test driver module displays information from the Root Port Configuration Space registers, so that you can correlate to the parameters you specified using the parameter editor.

**Note:**

The Intel testbench and Root Port BFM provide a simple method to do basic testing of the Application Layer logic that interfaces to the variation. This BFM allows you to create and run simple task stimuli with configurable parameters to exercise basic functionality of the Intel example design. The testbench and Root Port BFM are not intended to be a substitute for a full verification environment. Corner cases and certain traffic profile stimuli are not covered. Refer to the items listed below for further details. To ensure the best verification coverage possible, Intel suggests strongly that you obtain commercially available PCI Express verification IP and tools, or do your own extensive hardware testing or both.

Your Application Layer design may need to handle at least the following scenarios that are not possible to create with the Intel testbench and the Root Port BFM:

- It is unable to generate or receive Vendor Defined Messages. Some systems generate Vendor Defined Messages. Consequently, you must design the Application Layer to process them. The Hard IP block passes these messages on to the Application Layer which, in most cases should ignore them.
- It can only handle received read requests that are less than or equal to the currently set **Maximum payload size** option specified under **PCI Express/PCI Capabilities** heading under the **Device** tab using the parameter editor. Many systems are capable of handling larger read requests that are then returned in multiple completions.
- It always returns a single completion for every read request. Some systems split completions on every 64-byte address boundary.
- It always returns completions in the same order the read requests were issued. Some systems generate the completions out-of-order.
- It is unable to generate zero-length read requests that some systems generate as flush requests following some write transactions. The Application Layer must be capable of generating the completions to the zero length read requests.
- It uses fixed credit allocation.
- It does not support parity.
- It does not support multi-function designs.

#### Related Information

- [Quick Start Guide](#) on page 18
- [AN-811: Using the Avery BFM for PCI Express x16 Simulation on Intel Stratix 10 Devices](#)

## 9.1. Endpoint Testbench

You can create an Endpoint design for inclusion in the testbench using design flows described in the *Quick Start Guide*. This testbench uses the parameters that you specify in the *Quick Start Guide*.

This testbench simulates up to an  $\times 8$  PCI Express link using either the PIPE interface of the Endpoint or the serial PCI Express interface. The testbench design does not allow more than one PCI Express link to be simulated at a time. The following figure presents a high level view of the design example.

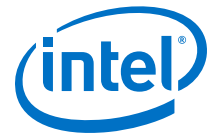
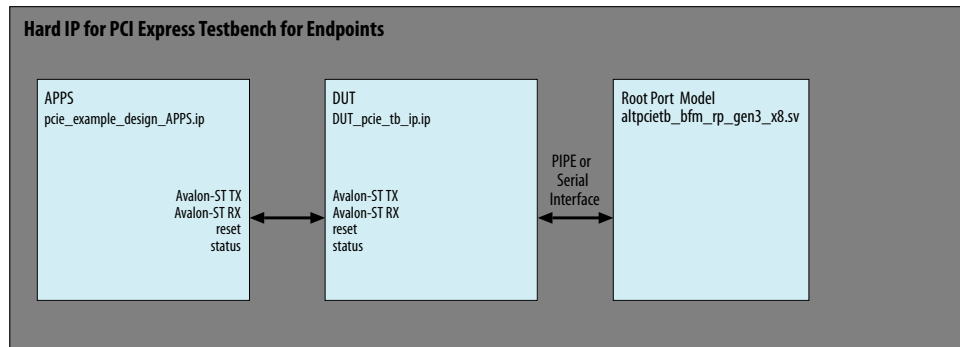


Figure 72. Design Example for Endpoint Designs



The top-level of the testbench instantiates the following main modules:

- `altpcieth_bfm_rp_<gen>_x8.sv` —This is the Root Port PCIe BFM. This is the module that you modify to vary the transactions sent to the example Endpoint design or your own design.

```
//Directory path
<project_dir>/pcie_<dev>_hip_ast_0_example_design/
pcie_example_design_tb/ip/pcie_example_design_tb/DUT_pcie_tb_ip/
altera_pcie_<dev>_tbed_<ver>/sim
```

**Note:** If you modify the RP BFM, you must also make the appropriate corresponding changes the APPs module.

- `pcie_example_design_DUT.ip`: This is the Endpoint design with the parameters that you specify.

```
//Directory path
<project_dir>/pcie_<dev>_hip_ast_0_example_design/ip/
pcie_example_design
```

- `pcie_example_design_APPS.ip`: This module is a target and initiator of transactions.

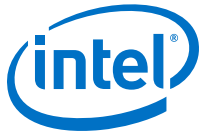
```
//Directory path
<project_dir>/pcie_<dev>_hip_ast_0_example_design/ip/
pcie_example_design/
```

- `altpcieth_bfm_cfpb.v`: This module supports Configuration Space Bypass mode. It drives TLPs to the custom Configuration Space.

```
//Directory path
<project_dir>/pcie_<dev>_hip_ast_0_example_design/
pcie_example_design_tb/ip/pcie_example_design_tb/DUT_pcie_tb_ip/
altera_pcie_<dev>_tbed_<ver>/sim
```

In addition, the testbench has routines that perform the following tasks:

- Generates the reference clock for the Endpoint at the required frequency.
- Provides a PCI Express reset at start up.



**Note:** Before running the testbench, you should set the `serial_sim_hwctl` parameter in `<testbench_dir>/pcie_<dev>_hip_avst_0_example_design/pcie_example_design_tb/ip/pcie_example_design_tb/DUT_pcie_tb_ip/altera_pcie_<dev>_tbed_<ver>/sim/altpcie__tbed_hwctl.v`. Set to 1 for serial simulation and 0 for PIPE simulation.

### 9.1.1. Endpoint Testbench for SR-IOV

The Endpoint testbench for SR-IOV supports up to two PFs and 32 VFs per PF.

First, the testbench configures the link and accesses then Configuration Space. Then, the testbench performs the following tests:

1. A single memory write followed by a single memory read to each PF and VF.
2. For PF0 only, the testbench drives memory writes to each VF and followed by memory reads of all VFs.

## 9.2. Test Driver Module

The test driver module, `altpcie_<dev>_tbed_hwctl.v`, instantiates the top-level BFM, `altpcieth_bfm_top_rp.v`.

The top-level BFM completes the following tasks:

1. Instantiates the driver and monitor.
2. Instantiates the Root Port BFM.
3. Instantiates either the PIPE or serial interfaces.

The configuration module, `altpcieth_bfm_configure.v` performs the following tasks:

1. Configures assigns the BARs.
2. Configures the Root Port and Endpoint.
3. Displays comprehensive Configuration Space, BAR, MSI and MSI-X, AER, settings..

### Related Information

[BFM Procedures and Functions](#) on page 139

## 9.3. Root Port BFM Overview

The basic Root Port BFM provides Verilog HDL task-based interface to request transactions to issue on the PCI Express link. The Root Port BFM also handles requests received from the PCI Express link. The following figure shows the most important modules in the Root Port BFM.

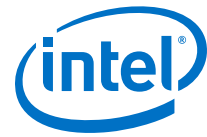
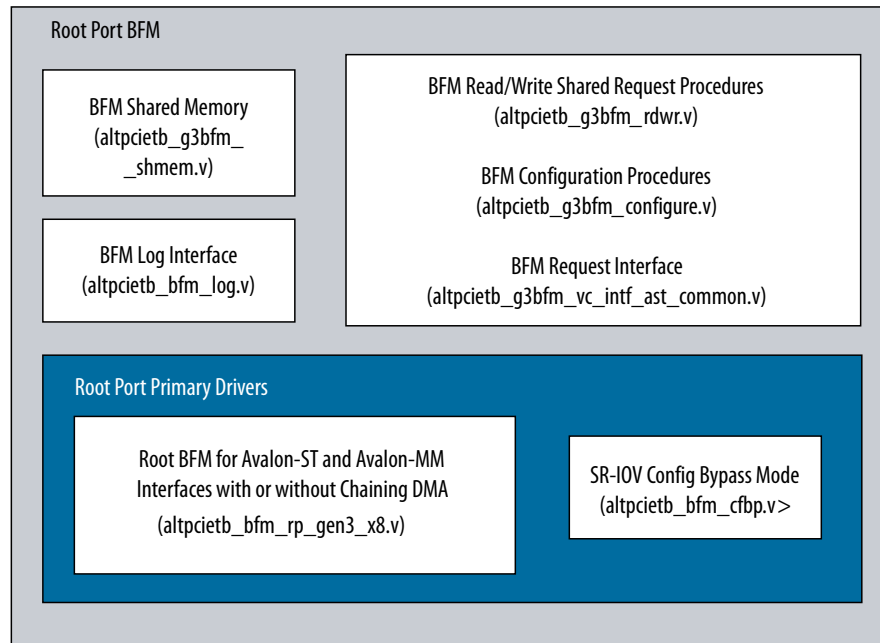


Figure 73. Root Port BFM



These modules implement the following functionality:

- BFM Log Interface, `altpcieth_bfm_log.v` and `altlpcieth_bfm_rp_<gen>_x8.v`: The BFM log functions provides routine for writing commonly formatted messages to the simulator standard output and optionally to a log file. It also provides controls that stop simulation on errors. For details on these procedures, refer to *BFM Log and Message Procedures*.
- BFM Read/Write Request Functions, `altpcieth_bfm_rp_<gen>_x8.sv`: These functions provide the basic BFM calls for PCI Express read and write requests. For details on these procedures, refer to *BFM Read and Write Procedures*.
- BFM Log Interface, `altpcieth_bfm_log.v` and `altlpcieth_bfm_rp_<gen>_x8.v`: The BFM log functions provides routine for writing commonly formatted messages to the simulator standard output and optionally to a log file. It also provides controls that stop simulation on errors. For details on these procedures, refer to *BFM Log and Message Procedures*.
- BFM Configuration Functions, `altpcieth_g3bfm_configure.v`: These functions provide the BFM calls to request configuration of the PCI Express link and the Endpoint Configuration Space registers. For details on these procedures and functions, refer to *BFM Configuration Procedures*.

- BFM shared memory, `altpcieth_g3bfm_shmem_common.v`: This module provides the Root Port BFM shared memory. It implements the following functionality:
  - Provides data for TX write operations
  - Provides data for RX read operations
  - Receives data for RX write operations
  - Receives data for received completions

Refer to *BFM Shared Memory Access Procedures* to learn more about the procedures to read, write, fill, and check the shared memory from the BFM driver.

- BFM Request Interface, `altpcieth_g3bfm_req_intf.v`: This interface provides the low-level interface between the `altpcieth_g3bfm_rdw` and `altpcieth_bfm_configure` procedures or functions and the Root Port RTL Model. This interface stores a write-protected data structure containing the sizes and the values programmed in the BAR registers of the Endpoint. It also stores other critical data used for internal BFM management. You do not need to access these files directly to adapt the testbench to test your Endpoint application.
- Avalon-ST Interfaces, `altpcieth_g3bfm_vc_intf_ast_common.v`: These interface modules handle the Root Port interface model. They take requests from the BFM request interface and generate the required PCI Express transactions. They handle completions received from the PCI Express link and notify the BFM request interface when requests are complete. Additionally, they handle any requests received from the PCI Express link, and store or fetch data from the shared memory before generating the required completions.

#### Related Information

- [BFM Shared Memory Access Procedures](#) on page 145
- [BFM Configuration Procedures](#) on page 143
- [BFM Log and Message Procedures](#) on page 147

### 9.3.1. BFM Memory Map

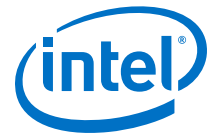
The BFM shared memory is 2 MBs. The BFM shared memory maps to the first 2 MBs of I/O space and also the first 2 MBs of memory space. When the Endpoint application generates an I/O or memory transaction in this range, the BFM reads or writes the shared memory.

### 9.3.2. Configuration Space Bus and Device Numbering

Enumeration assigns the Root Port interface device number 0 on internal bus number 0. Use the `ebfm_cfg_rp_ep` to assign the Endpoint to any device number on any bus number (greater than 0). The specified bus number is the secondary bus in the Root Port Configuration Space.

### 9.3.3. Configuration of Root Port and Endpoint

Before you issue transactions to the Endpoint, you must configure the Root Port and Endpoint Configuration Space registers.



The `ebfm_cfg_rp_ep` procedure executes the following steps to initialize the Configuration Space:

1. Sets the Root Port Configuration Space to enable the Root Port to send transactions on the PCI Express link.
2. Sets the Root Port and Endpoint PCI Express Capability Device Control registers as follows:
  - a. Disables Error Reporting in both the Root Port and Endpoint. The BFM does not have error handling capability.
  - b. Enables Relaxed Ordering in both Root Port and Endpoint.
  - c. Enables Extended Tags for the Endpoint if the Endpoint has that capability.
  - d. Disables Phantom Functions, Aux Power PM, and No Snoop in both the Root Port and Endpoint.
  - e. Sets the Max Payload Size to the value that the Endpoint supports because the Root Port supports the maximum payload size.
  - f. Sets the Root Port Max Read Request Size to 4 KB because the example Endpoint design supports breaking the read into as many completions as necessary.
  - g. Sets the Endpoint Max Read Request Size equal to the Max Payload Size because the Root Port does not support breaking the read request into multiple completions.
3. Assigns values to all the Endpoint BAR registers. The BAR addresses are assigned by the algorithm outlined below.
  - a. I/O BARs are assigned smallest to largest starting just above the ending address of BFM shared memory in I/O space and continuing as needed throughout a full 32-bit I/O space.
  - b. The 32-bit non-prefetchable memory BARs are assigned smallest to largest, starting just above the ending address of BFM shared memory in memory space and continuing as needed throughout a full 32-bit memory space.
  - c. The value of the `addr_map_4GB_limit` input to the `ebfm_cfg_rp_ep` procedure controls the assignment of the 32-bit prefetchable and 64-bit prefetchable memory BARs. The default value of the `addr_map_4GB_limit` is 0.

If the `addr_map_4GB_limit` input to the `ebfm_cfg_rp_ep` procedure is set to 0, then the `ebfm_cfg_rp_ep` procedure assigns the 32-bit prefetchable memory BARs largest to smallest, starting at the top of 32-bit memory space and continuing as needed down to the ending address of the last 32-bit non-prefetchable BAR.

However, if the `addr_map_4GB_limit` input is set to 1, the address map is limited to 4 GB. The `ebfm_cfg_rp_ep` procedure assigns 32-bit and 64-bit prefetchable memory BARs largest to smallest, starting at the top of the 32-bit memory space and continuing as needed down to the ending address of the last 32-bit non-prefetchable BAR.

- d. If the `addr_map_4GB_limit` input to the `ebfm_cfg_rp_ep` procedure is set to 0, then the `ebfm_cfg_rp_ep` procedure assigns the 64-bit prefetchable memory BARs smallest to largest starting at the 4 GB address assigning memory ascending above the 4 GB limit throughout the full 64-bit memory space.

If the `addr_map_4GB_limit` input to the `ebfm_cfg_rp_ep` procedure is set to 1, the `ebfm_cfg_rp_ep` procedure assigns the 32-bit and the 64-bit prefetchable memory BARs largest to smallest starting at the 4 GB address and assigning memory by descending below the 4 GB address to memory addresses as needed down to the ending address of the last 32-bit non-prefetchable BAR.

The above algorithm cannot always assign values to all BARs when there are a few very large (1 GB or greater) 32-bit BARs. Although assigning addresses to all BARs may be possible, a more complex algorithm would be required to effectively assign these addresses. However, such a configuration is unlikely to be useful in real systems. If the procedure is unable to assign the BARs, it displays an error message and stops the simulation.

4. Based on the above BAR assignments, the `ebfm_cfg_rp_ep` procedure assigns the Root Port Configuration Space address windows to encompass the valid BAR address ranges.
5. The `ebfm_cfg_rp_ep` procedure enables master transactions, memory address decoding, and I/O address decoding in the Endpoint PCIe control register.

The `ebfm_cfg_rp_ep` procedure also sets up a `bar_table` data structure in BFM shared memory that lists the sizes and assigned addresses of all Endpoint BARs. This area of BFM shared memory is write-protected. Consequently, any application logic write accesses to this area cause a fatal simulation error.

BFM procedure calls to generate full PCIe addresses for read and write requests to particular offsets from a BAR use this data structure. . This procedure allows the testbench code that accesses the Endpoint application logic to use offsets from a BAR and avoid tracking specific addresses assigned to the BAR. The following table shows how to use those offsets.

**Table 87. BAR Table Structure**

| Offset (Bytes) | Description                              |
|----------------|------------------------------------------|
| +0             | PCI Express address in BAR0              |
| +4             | PCI Express address in BAR1              |
| +8             | PCI Express address in BAR2              |
| +12            | PCI Express address in BAR3              |
| +16            | PCI Express address in BAR4              |
| +20            | PCI Express address in BAR5              |
| +24            | PCI Express address in Expansion ROM BAR |
| continued...   |                                          |

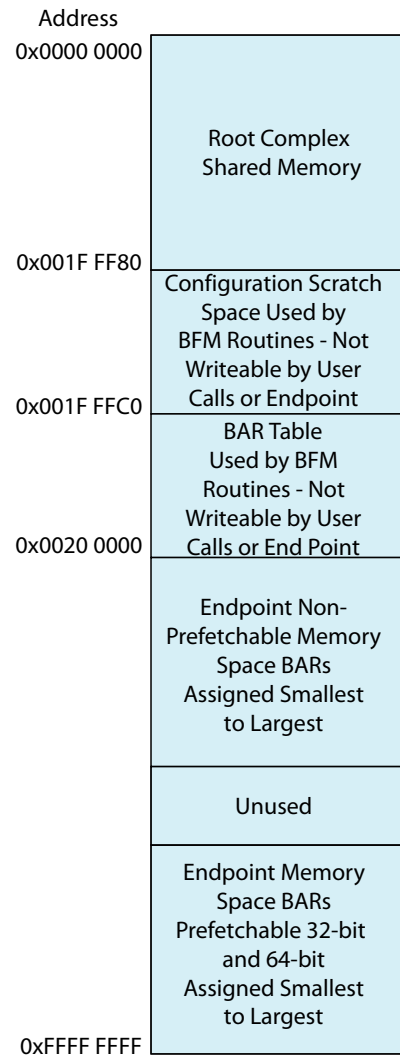


| Offset (Bytes) | Description                                                                  |
|----------------|------------------------------------------------------------------------------|
| +28            | Reserved                                                                     |
| +32            | BAR0 read back value after being written with all 1's (used to compute size) |
| +36            | BAR1 read back value after being written with all 1's                        |
| +40            | BAR2 read back value after being written with all 1's                        |
| +44            | BAR3 read back value after being written with all 1's                        |
| +48            | BAR4 read back value after being written with all 1's                        |
| +52            | BAR5 read back value after being written with all 1's                        |
| +56            | Expansion ROM BAR read back value after being written with all 1's           |
| +60            | Reserved                                                                     |

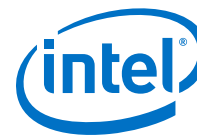
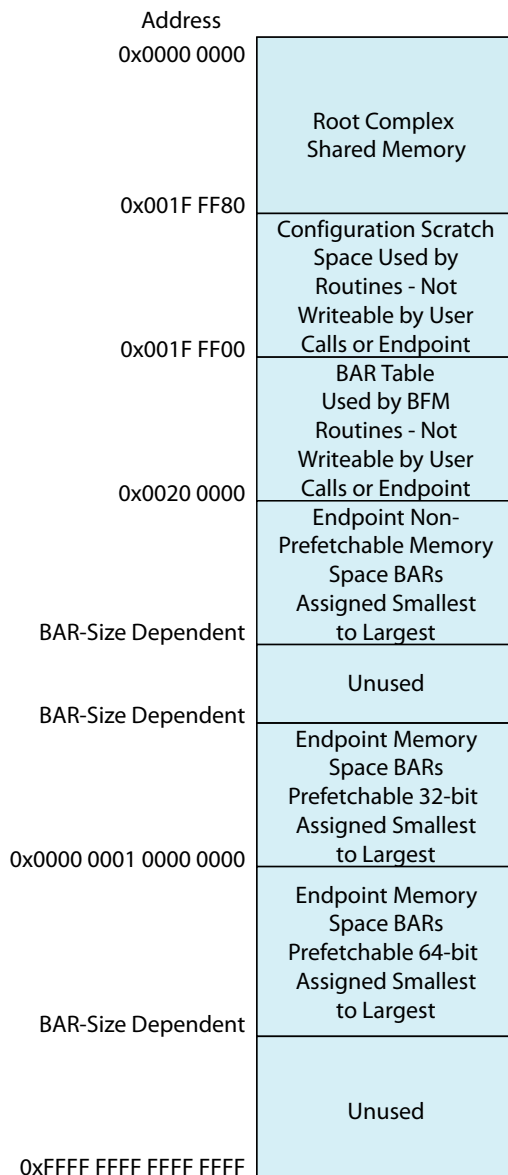
The configuration routine does not configure any advanced PCI Express capabilities such as the AER capability.

Besides the `ebfm_cfg_rp_ep` procedure in `altpcieth_bfm_rp_gen3_x8.sv`, routines to read and write Endpoint Configuration Space registers directly are available in the Verilog HDL include file. After the `ebfm_cfg_rp_ep` procedure runs the PCI Express I/O and Memory Spaces have the layout shown in the following three figures. The memory space layout depends on the value of the **`addr_map_4GB_limit`** input parameter. The following figure shows the resulting memory space map when the **`addr_map_4GB_limit`** is 1.

**Figure 74. Memory Space Layout—4 GB Limit**

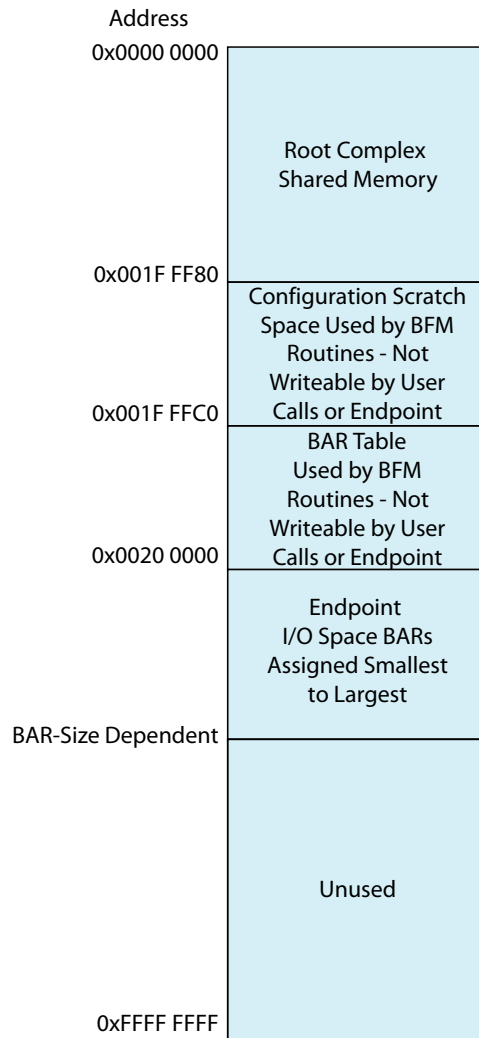


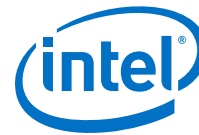
The following figure shows the resulting memory space map when the **addr\_map\_4GB\_limit** is 0.

**Figure 75. Memory Space Layout—No Limit**

The following figure shows the I/O address space.

**Figure 76. I/O Address Space**





### 9.3.4. Issuing Read and Write Transactions to the Application Layer

The Endpoint Application Layer issues read and write transactions by calling one of the `ebfm_bar` procedures in `altpcieth_g3bfm_rdwr.v`. The procedures and functions listed below are available in the Verilog HDL include file `altpcieth_g3bfm_rdwr.v`. The complete list of available procedures and functions is as follows:

- `ebfm_barwr`: writes data from BFM shared memory to an offset from a specific Endpoint BAR. This procedure returns as soon as the request has been passed to the VC interface module for transmission.
- `ebfm_barwr_imm`: writes a maximum of four bytes of immediate data (passed in a procedure call) to an offset from a specific Endpoint BAR. This procedure returns as soon as the request has been passed to the VC interface module for transmission.
- `ebfm_barrrd_wait`: reads data from an offset of a specific Endpoint BAR and stores it in BFM shared memory. This procedure blocks waiting for the completion data to be returned before returning control to the caller.
- `ebfm_barrrd_nowt`: reads data from an offset of a specific Endpoint BAR and stores it in the BFM shared memory. This procedure returns as soon as the request has been passed to the VC interface module for transmission, allowing subsequent reads to be issued in the interim.

These routines take as parameters a BAR number to access the memory space and the BFM shared memory address of the `bar_table` data structure that was set up by the `ebfm_cfg_rp_ep` procedure. (Refer to *Configuration of Root Port and Endpoint*.) Using these parameters simplifies the BFM test driver routines that access an offset from a specific BAR and eliminates calculating the addresses assigned to the specified BAR.

The Root Port BFM does not support accesses to Endpoint I/O space BARs.

#### Related Information

[Configuration of Root Port and Endpoint](#) on page 132

## 9.4. BFM Procedures and Functions

The BFM includes procedures, functions, and tasks to drive Endpoint application testing. It also includes procedures to run the chaining DMA design example.

The BFM read and write procedures read and write data to BFM shared memory, Endpoint BARs, and specified configuration registers. The procedures and functions are available in the Verilog HDL. These procedures and functions support issuing memory and configuration transactions on the PCI Express link.

### 9.4.1. `ebfm_barwr` Procedure

The `ebfm_barwr` procedure writes a block of data from BFM shared memory to an offset from the specified Endpoint BAR. The length can be longer than the configured `MAXIMUM_PAYLOAD_SIZE`. The procedure breaks the request up into multiple transactions as needed. This routine returns as soon as the last transaction has been accepted by the VC interface module.

| Location  | altpcieth_g3bfm_rdwr.v                                                 |                                                                                                                                                                                                                                                                          |
|-----------|------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | ebfm_barwr(bar_table, bar_num, pcie_offset, lcladdr, byte_len, tclass) |                                                                                                                                                                                                                                                                          |
| Arguments | bar_table                                                              | Address of the Endpoint bar_table structure in BFM shared memory. The bar_table structure stores the address assigned to each BAR so that the driver code does not need to be aware of the actual assigned addresses only the application specific offsets from the BAR. |
|           | bar_num                                                                | Number of the BAR used with pcie_offset to determine PCI Express address.                                                                                                                                                                                                |
|           | pcie_offset                                                            | Address offset from the BAR base.                                                                                                                                                                                                                                        |
|           | lcladdr                                                                | BFM shared memory address of the data to be written.                                                                                                                                                                                                                     |
|           | byte_len                                                               | Length, in bytes, of the data written. Can be 1 to the minimum of the bytes remaining in the BAR space or BFM shared memory.                                                                                                                                             |
|           | tclass                                                                 | Traffic class used for the PCI Express transaction.                                                                                                                                                                                                                      |

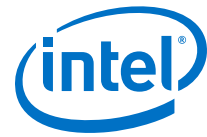
### 9.4.2. ebfm\_barwr\_imm Procedure

The ebfm\_barwr\_imm procedure writes up to four bytes of data to an offset from the specified Endpoint BAR.

| Location  | altpcieth_g3bfm_rdwr.v                                                      |                                                                                                                                                                                                                                                                                                         |
|-----------|-----------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | ebfm_barwr_imm(bar_table, bar_num, pcie_offset, imm_data, byte_len, tclass) |                                                                                                                                                                                                                                                                                                         |
| Arguments | bar_table                                                                   | Address of the Endpoint bar_table structure in BFM shared memory. The bar_table structure stores the address assigned to each BAR so that the driver code does not need to be aware of the actual assigned addresses only the application specific offsets from the BAR.                                |
|           | bar_num                                                                     | Number of the BAR used with pcie_offset to determine PCI Express address.                                                                                                                                                                                                                               |
|           | pcie_offset                                                                 | Address offset from the BAR base.                                                                                                                                                                                                                                                                       |
|           | imm_data                                                                    | Data to be written. In Verilog HDL, this argument is reg [31:0].In both languages, the bits written depend on the length as follows:<br>Length Bits Written <ul style="list-style-type: none"> <li>4: 31 down to 0</li> <li>3: 23 down to 0</li> <li>2: 15 down to 0</li> <li>1: 7 down to 0</li> </ul> |
|           | byte_len                                                                    | Length of the data to be written in bytes. Maximum length is 4 bytes.                                                                                                                                                                                                                                   |
|           | tclass                                                                      | Traffic class to be used for the PCI Express transaction.                                                                                                                                                                                                                                               |

### 9.4.3. ebfm\_barrrd\_wait Procedure

The ebfm\_barrrd\_wait procedure reads a block of data from the offset of the specified Endpoint BAR and stores it in BFM shared memory. The length can be longer than the configured maximum read request size; the procedure breaks the request up into multiple transactions as needed. This procedure waits until all of the completion data is returned and places it in shared memory.



| Location  | altpcieth_g3bfm_rdwr.v                                                       |                                                                                                                                                                                                                                                                          |
|-----------|------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | ebfm_barrrd_wait(bar_table, bar_num, pcie_offset, lcladdr, byte_len, tclass) |                                                                                                                                                                                                                                                                          |
| Arguments | bar_table                                                                    | Address of the Endpoint bar_table structure in BFM shared memory. The bar_table structure stores the address assigned to each BAR so that the driver code does not need to be aware of the actual assigned addresses only the application specific offsets from the BAR. |
|           | bar_num                                                                      | Number of the BAR used with pcie_offset to determine PCI Express address.                                                                                                                                                                                                |
|           | pcie_offset                                                                  | Address offset from the BAR base.                                                                                                                                                                                                                                        |
|           | lcladdr                                                                      | BFM shared memory address where the read data is stored.                                                                                                                                                                                                                 |
|           | byte_len                                                                     | Length, in bytes, of the data to be read. Can be 1 to the minimum of the bytes remaining in the BAR space or BFM shared memory.                                                                                                                                          |
|           | tclass                                                                       | Traffic class used for the PCI Express transaction.                                                                                                                                                                                                                      |

#### 9.4.4. ebfm\_barrrd\_nowt Procedure

The ebfm\_barrrd\_nowt procedure reads a block of data from the offset of the specified Endpoint BAR and stores the data in BFM shared memory. The length can be longer than the configured maximum read request size; the procedure breaks the request up into multiple transactions as needed. This routine returns as soon as the last read transaction has been accepted by the VC interface module, allowing subsequent reads to be issued immediately.

| Location  | altpcieth_g3bfm_rdwr.v                                                       |                                                                                                                                 |
|-----------|------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | ebfm_barrrd_nowt(bar_table, bar_num, pcie_offset, lcladdr, byte_len, tclass) |                                                                                                                                 |
| Arguments | bar_table                                                                    | Address of the Endpoint bar_table structure in BFM shared memory.                                                               |
|           | bar_num                                                                      | Number of the BAR used with pcie_offset to determine PCI Express address.                                                       |
|           | pcie_offset                                                                  | Address offset from the BAR base.                                                                                               |
|           | lcladdr                                                                      | BFM shared memory address where the read data is stored.                                                                        |
|           | byte_len                                                                     | Length, in bytes, of the data to be read. Can be 1 to the minimum of the bytes remaining in the BAR space or BFM shared memory. |
|           | tclass                                                                       | Traffic Class to be used for the PCI Express transaction.                                                                       |

#### 9.4.5. ebfm\_cfgwr\_imm\_wait Procedure

The ebfm\_cfgwr\_imm\_wait procedure writes up to four bytes of data to the specified configuration register. This procedure waits until the write completion has been returned.

| Location     | altpcieth_g3bfm_rdwr.v                                                                       |                                                      |
|--------------|----------------------------------------------------------------------------------------------|------------------------------------------------------|
| Syntax       | ebfm_cfgwr_imm_wait(bus_num, dev_num, fnc_num, imm_regb_ad, regb_ln, imm_data, compl_status) |                                                      |
| Arguments    | bus_num                                                                                      | PCI Express bus number of the target device.         |
|              | dev_num                                                                                      | PCI Express device number of the target device.      |
|              | fnc_num                                                                                      | Function number in the target device to be accessed. |
| continued... |                                                                                              |                                                      |

| Location | altpcieth_g3bfm_rdwr.v |                                                                                                                                                                                                                                                                                                                                                                                                |
|----------|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          | regb_ad                | Byte-specific address of the register to be written.                                                                                                                                                                                                                                                                                                                                           |
|          | regb_ln                | Length, in bytes, of the data written. Maximum length is four bytes. The regb_ln and the regb_ad arguments cannot cross a DWORD boundary.                                                                                                                                                                                                                                                      |
|          | imm_data               | Data to be written.<br>This argument is reg [31:0].<br>The bits written depend on the length: <ul style="list-style-type: none"> <li>• 4: 31 down to 0</li> <li>• 3: 23 down to 0</li> <li>• 2: 15 down to 0</li> <li>• 1: 7 down to 0</li> </ul>                                                                                                                                              |
|          | compl_status           | This argument is reg [2:0].<br>This argument is the completion status as specified in the PCI Express specification. The following encodings are defined: <ul style="list-style-type: none"> <li>• 3'b000: SC— Successful completion</li> <li>• 3'b001: UR— Unsupported Request</li> <li>• 3'b010: CRS — Configuration Request Retry Status</li> <li>• 3'b100: CA — Completer Abort</li> </ul> |

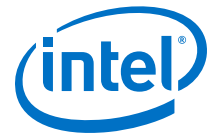
#### 9.4.6. ebfm\_cfgwr\_imm\_nowt Procedure

The ebfm\_cfgwr\_imm\_nowt procedure writes up to four bytes of data to the specified configuration register. This procedure returns as soon as the VC interface module accepts the transaction, allowing other writes to be issued in the interim. Use this procedure only when successful completion status is expected.

| Location  | altpcieth_g3bfm_rdwr.v                                                           |                                                                                                                                                                                                                                                                                |
|-----------|----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | ebfm_cfgwr_imm_nowt(bus_num, dev_num, fnc_num, imm_regb_adr, regb_len, imm_data) |                                                                                                                                                                                                                                                                                |
| Arguments | bus_num                                                                          | PCI Express bus number of the target device.                                                                                                                                                                                                                                   |
|           | dev_num                                                                          | PCI Express device number of the target device.                                                                                                                                                                                                                                |
|           | fnc_num                                                                          | Function number in the target device to be accessed.                                                                                                                                                                                                                           |
|           | regb_ad                                                                          | Byte-specific address of the register to be written.                                                                                                                                                                                                                           |
|           | regb_ln                                                                          | Length, in bytes, of the data written. Maximum length is four bytes. The regb_ln the regb_ad arguments cannot cross a DWORD boundary.                                                                                                                                          |
|           | imm_data                                                                         | Data to be written<br>This argument is reg [31:0].<br>In both languages, the bits written depend on the length. The following encodes are defined. <ul style="list-style-type: none"> <li>• 4: [31:0]</li> <li>• 3: [23:0]</li> <li>• 2: [15:0]</li> <li>• 1: [7:0]</li> </ul> |

#### 9.4.7. ebfm\_cfgrd\_wait Procedure

The ebfm\_cfgrd\_wait procedure reads up to four bytes of data from the specified configuration register and stores the data in BFM shared memory. This procedure waits until the read completion has been returned.



| Location  | altpcieth_g3bfm_rdwr.v                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------|-------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | ebfm_cfgrd_wait(bus_num, dev_num, fnc_num, regb_ad, regb_ln, lcladdr, compl_status) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Arguments | bus_num                                                                             | PCI Express bus number of the target device.                                                                                                                                                                                                                                                                                                                                                                                                                     |
|           | dev_num                                                                             | PCI Express device number of the target device.                                                                                                                                                                                                                                                                                                                                                                                                                  |
|           | fnc_num                                                                             | Function number in the target device to be accessed.                                                                                                                                                                                                                                                                                                                                                                                                             |
|           | regb_ad                                                                             | Byte-specific address of the register to be written.                                                                                                                                                                                                                                                                                                                                                                                                             |
|           | regb_ln                                                                             | Length, in bytes, of the data read. Maximum length is four bytes. The regb_ln and the regb_ad arguments cannot cross a DWORD boundary.                                                                                                                                                                                                                                                                                                                           |
|           | lcladdr                                                                             | BFM shared memory address of where the read data should be placed.                                                                                                                                                                                                                                                                                                                                                                                               |
|           | compl_status                                                                        | Completion status for the configuration transaction.<br>This argument is reg [2:0].<br>In both languages, this is the completion status as specified in the PCI Express specification. The following encodings are defined. <ul style="list-style-type: none"> <li>• 3'b000: SC— Successful completion</li> <li>• 3'b001: UR— Unsupported Request</li> <li>• 3'b010: CRS — Configuration Request Retry Status</li> <li>• 3'b100: CA — Completer Abort</li> </ul> |

#### 9.4.8. ebfm\_cfgrd\_nowt Procedure

The ebfm\_cfgrd\_nowt procedure reads up to four bytes of data from the specified configuration register and stores the data in the BFM shared memory. This procedure returns as soon as the VC interface module has accepted the transaction, allowing other reads to be issued in the interim. Use this procedure only when successful completion status is expected and a subsequent read or write with a wait can be used to guarantee the completion of this operation.

| Location  | altpcieth_g3bfm_rdwr.v                                                |                                                                                                                                       |
|-----------|-----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | ebfm_cfgrd_nowt(bus_num, dev_num, fnc_num, regb_ad, regb_ln, lcladdr) |                                                                                                                                       |
| Arguments | bus_num                                                               | PCI Express bus number of the target device.                                                                                          |
|           | dev_num                                                               | PCI Express device number of the target device.                                                                                       |
|           | fnc_num                                                               | Function number in the target device to be accessed.                                                                                  |
|           | regb_ad                                                               | Byte-specific address of the register to be written.                                                                                  |
|           | regb_ln                                                               | Length, in bytes, of the data written. Maximum length is four bytes. The regb_ln and regb_ad arguments cannot cross a DWORD boundary. |
|           | lcladdr                                                               | BFM shared memory address where the read data should be placed.                                                                       |

#### 9.4.9. BFM Configuration Procedures

The BFM configuration procedures are available in altpcieth\_bfm\_rp\_gen3\_x8.sv. These procedures support configuration of the Root Port and Endpoint Configuration Space registers.

All Verilog HDL arguments are type integer and are input-only unless specified otherwise.

### 9.4.9.1. ebfm\_cfg\_rp\_ep Procedure

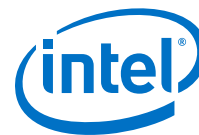
The ebfm\_cfg\_rp\_ep procedure configures the Root Port and Endpoint Configuration Space registers for operation.

| Location  | altpcieth_g3bfm_configure.v                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------|--------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | ebfm_cfg_rp_ep(bar_table, ep_bus_num, ep_dev_num, rp_max_rd_req_size, display_ep_config, addr_map_4GB_limit) |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Arguments | bar_table                                                                                                    | Address of the Endpoint bar_table structure in BFM shared memory. This routine populates the bar_table structure. The bar_table structure stores the size of each BAR and the address values assigned to each BAR. The address of the bar_table structure is passed to all subsequent read and write procedure calls that access an offset from a particular BAR.                                                 |
|           | ep_bus_num                                                                                                   | PCI Express bus number of the target device. This number can be any value greater than 0. The Root Port uses this as the secondary bus number.                                                                                                                                                                                                                                                                    |
|           | ep_dev_num                                                                                                   | PCI Express device number of the target device. This number can be any value. The Endpoint is automatically assigned this value when it receives the first configuration transaction.                                                                                                                                                                                                                             |
|           | rp_max_rd_req_size                                                                                           | Maximum read request size in bytes for reads issued by the Root Port. This parameter must be set to the maximum value supported by the Endpoint Application Layer. If the Application Layer only supports reads of the MAXIMUM_PAYLOAD_SIZE, then this can be set to 0 and the read request size is set to the maximum payload size. Valid values for this argument are 0, 128, 256, 512, 1,024, 2,048 and 4,096. |
|           | display_ep_config                                                                                            | When set to 1 many of the Endpoint Configuration Space registers are displayed after they have been initialized, causing some additional reads of registers that are not normally accessed during the configuration process such as the Device ID and Vendor ID.                                                                                                                                                  |
|           | addr_map_4GB_limit                                                                                           | When set to 1 the address map of the simulation system is limited to 4 GB. Any 64-bit BARs are assigned below the 4 GB limit.                                                                                                                                                                                                                                                                                     |

### 9.4.9.2. ebfm\_cfg\_decode\_bar Procedure

The ebfm\_cfg\_decode\_bar procedure analyzes the information in the BAR table for the specified BAR and returns details about the BAR attributes.

| Location  | altpcieth_bfm_configure.v                                                   |                                                                                                                                                              |
|-----------|-----------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | ebfm_cfg_decode_bar(bar_table, bar_num, log2_size, is_mem, is_pref, is_64b) |                                                                                                                                                              |
| Arguments | bar_table                                                                   | Address of the Endpoint bar_table structure in BFM shared memory.                                                                                            |
|           | bar_num                                                                     | BAR number to analyze.                                                                                                                                       |
|           | log2_size                                                                   | This argument is set by the procedure to the log base 2 of the size of the BAR. If the BAR is not enabled, this argument is set to 0.                        |
|           | is_mem                                                                      | The procedure sets this argument to indicate if the BAR is a memory space BAR (1) or I/O Space BAR (0).                                                      |
|           | is_pref                                                                     | The procedure sets this argument to indicate if the BAR is a prefetchable BAR (1) or non-prefetchable BAR (0).                                               |
|           | is_64b                                                                      | The procedure sets this argument to indicate if the BAR is a 64-bit BAR (1) or 32-bit BAR (0). This is set to 1 only for the lower numbered BAR of the pair. |



## 9.4.10. BFM Shared Memory Access Procedures

These procedures and functions support accessing the BFM shared memory.

### 9.4.10.1. Shared Memory Constants

The following constants are defined in `altpciethb_g3bfm_shmem.v`. They select a data pattern for the `shmem_fill` and `shmem_chk_ok` routines. These shared memory constants are all Verilog HDL type integer.

**Table 88. Constants: Verilog HDL Type INTEGER**

| Constant             | Description                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------|
| SHMEM_FILL_ZEROS     | Specifies a data pattern of all zeros                                                                                     |
| SHMEM_FILL_BYTE_INC  | Specifies a data pattern of incrementing 8-bit bytes (0x00, 0x01, 0x02, etc.)                                             |
| SHMEM_FILL_WORD_INC  | Specifies a data pattern of incrementing 16-bit words (0x0000, 0x0001, 0x0002, etc.)                                      |
| SHMEM_FILL_DWORD_INC | Specifies a data pattern of incrementing 32-bit DWORDs (0x00000000, 0x00000001, 0x00000002, etc.)                         |
| SHMEM_FILL_QWORD_INC | Specifies a data pattern of incrementing 64-bit qwords (0x0000000000000000, 0x0000000000000001, 0x0000000000000002, etc.) |
| SHMEM_FILL_ONE       | Specifies a data pattern of all ones                                                                                      |

### 9.4.10.2. shmem\_write Task

The `shmem_write` procedure writes data to the BFM shared memory.

| Location  | altpciethb_g3bfm_shmem.v                   |                                                                                                                                                                                                                                                                                |
|-----------|--------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | <code>shmem_write(addr, data, leng)</code> |                                                                                                                                                                                                                                                                                |
| Arguments | addr                                       | BFM shared memory starting address for writing data                                                                                                                                                                                                                            |
|           | data                                       | Data to write to BFM shared memory.<br>This parameter is implemented as a 64-bit vector. <code>leng</code> is 1–8 bytes. Bits 7 down to 0 are written to the location specified by <code>addr</code> ; bits 15 down to 8 are written to the <code>addr+1</code> location, etc. |
|           | length                                     | Length, in bytes, of data written                                                                                                                                                                                                                                              |

### 9.4.10.3. shmem\_read Function

The `shmem_read` function reads data to the BFM shared memory.

| Location     | altpciethb_g3bfm_shmem.v                   |                                                     |
|--------------|--------------------------------------------|-----------------------------------------------------|
| Syntax       | <code>data:= shmem_read(addr, leng)</code> |                                                     |
| Arguments    | addr                                       | BFM shared memory starting address for reading data |
|              | leng                                       | Length, in bytes, of data read                      |
| Return       | data                                       | Data read from BFM shared memory.                   |
| continued... |                                            |                                                     |

| Location | altpcieth_g3bfm_shmem.v |                                                                                                                                                                                                                                                                                                                          |
|----------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          |                         | <p>This parameter is implemented as a 64-bit vector. leng is 1- 8 bytes. If leng is less than 8 bytes, only the corresponding least significant bits of the returned data are valid.</p> <p>Bits 7 down to 0 are read from the location specified by addr; bits 15 down to 8 are read from the addr+1 location, etc.</p> |

#### 9.4.10.4. shmem\_display Verilog HDL Function

The shmem\_display Verilog HDL function displays a block of data from the BFM shared memory.

| Location  | altpcieth_g3bfm_shmem.v                                                               |                                                                                                                                                                                                                                          |
|-----------|---------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | Verilog HDL: dummy_return:=shmem_display(addr, leng, word_size, flag_addr, msg_type); |                                                                                                                                                                                                                                          |
| Arguments | addr                                                                                  | BFM shared memory starting address for displaying data.                                                                                                                                                                                  |
|           | leng                                                                                  | Length, in bytes, of data to display.                                                                                                                                                                                                    |
|           | word_size                                                                             | Size of the words to display. Groups individual bytes into words. Valid values are 1, 2, 4, and 8.                                                                                                                                       |
|           | flag_addr                                                                             | Adds a <== flag to the end of the display line containing this address. Useful for marking specific data. Set to a value greater than 2**21 (size of BFM shared memory) to suppress the flag.                                            |
|           | msg_type                                                                              | Specifies the message type to be displayed at the beginning of each line. See "BFM Log and Message Procedures" on page 18-37 for more information about message types. Set to one of the constants defined in Table 18-36 on page 18-41. |

#### 9.4.10.5. shmem\_fill Procedure

The shmem\_fill procedure fills a block of BFM shared memory with a specified data pattern.

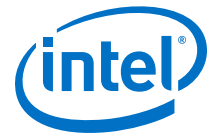
| Location  | altpcieth_g3bfm_shmem.v            |                                                                                                                                                                                                 |
|-----------|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | shmem_fill(addr, mode, leng, init) |                                                                                                                                                                                                 |
| Arguments | addr                               | BFM shared memory starting address for filling data.                                                                                                                                            |
|           | mode                               | Data pattern used for filling the data. Should be one of the constants defined in section <i>Shared Memory Constants</i> .                                                                      |
|           | leng                               | Length, in bytes, of data to fill. If the length is not a multiple of the incrementing data pattern width, then the last data pattern is truncated to fit.                                      |
|           | init                               | Initial data value used for incrementing data pattern modes. This argument is reg [63:0].<br>The necessary least significant bits are used for the data patterns that are smaller than 64 bits. |

#### Related Information

[Shared Memory Constants](#) on page 145

#### 9.4.10.6. shmem\_chk\_ok Function

The shmem\_chk\_ok function checks a block of BFM shared memory against a specified data pattern.



| Location  | altrpciethb_g3bfm_shmem.v                                    |                                                                                                                                                                             |
|-----------|--------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax    | result:= shmem_chk_ok(addr, mode, leng, init, display_error) |                                                                                                                                                                             |
| Arguments | addr                                                         | BFM shared memory starting address for checking data.                                                                                                                       |
|           | mode                                                         | Data pattern used for checking the data. Should be one of the constants defined in section "Shared Memory Constants" on page 18–35.                                         |
|           | leng                                                         | Length, in bytes, of data to check.                                                                                                                                         |
|           | init                                                         | This argument is reg [63:0].The necessary least significant bits are used for the data patterns that are smaller than 64-bits.                                              |
|           | display_error                                                | When set to 1, this argument displays the data failing comparison on the simulator standard output.                                                                         |
| Return    | Result                                                       | Result is 1-bit.<br><ul style="list-style-type: none"> <li>1'b1 — Data patterns compared successfully</li> <li>1'b0 — Data patterns did not compare successfully</li> </ul> |

### 9.4.11. BFM Log and Message Procedures

The following procedures and functions are available in the Verilog HDL include file altrpciethb\_bfm\_log.v

These procedures provide support for displaying messages in a common format, suppressing informational messages, and stopping simulation on specific message types.

The following constants define the type of message and their values determine whether a message is displayed or simulation is stopped after a specific message. Each displayed message has a specific prefix, based on the message type in the following table.

You can suppress the display of certain message types. The default values determining whether a message type is displayed are defined in the following table. To change the default message display, modify the display default value with a procedure call to ebfm\_log\_set\_suppressed\_msg\_mask.

Certain message types also stop simulation after the message is displayed. The following table shows the default value determining whether a message type stops simulation. You can specify whether simulation stops for particular messages with the procedure ebfm\_log\_set\_stop\_on\_msg\_mask.

All of these log message constants type integer.

**Table 89. Log Messages**

| Constant (Message Type) | Description                                                                                            | Mask Bit No | Display by Default | Simulation Stops by Default | Message Prefix |
|-------------------------|--------------------------------------------------------------------------------------------------------|-------------|--------------------|-----------------------------|----------------|
| EBFM_MSG_DEBUG          | Specifies debug messages.                                                                              | 0           | No                 | No                          | DEBUG:         |
| EBFM_MSG_INFO           | Specifies informational messages, such as configuration register values, starting and ending of tests. | 1           | Yes                | No                          | INFO:          |
| continued...            |                                                                                                        |             |                    |                             |                |

| Constant<br>(Message Type)  | Description                                                                                                                                                                                                                                                                                                                                                                        | Mask Bit No | Display by Default     | Simulation Stops by Default | Message Prefix |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|------------------------|-----------------------------|----------------|
| EBFM_MSG_WARNING            | Specifies warning messages, such as tests being skipped due to the specific configuration.                                                                                                                                                                                                                                                                                         | 2           | Yes                    | No                          | WARNING :      |
| EBFM_MSG_ERROR_INFO         | Specifies additional information for an error. Use this message to display preliminary information before an error message that stops simulation.                                                                                                                                                                                                                                  | 3           | Yes                    | No                          | ERROR :        |
| EBFM_MSG_ERROR_CONTINUE     | Specifies a recoverable error that allows simulation to continue. Use this error for data comparison failures.                                                                                                                                                                                                                                                                     | 4           | Yes                    | No                          | ERROR :        |
| EBFM_MSG_ERROR_FATAL        | Specifies an error that stops simulation because the error leaves the testbench in a state where further simulation is not possible.                                                                                                                                                                                                                                               | N/A         | Yes<br>Cannot suppress | Yes<br>Cannot suppress      | FATAL :        |
| EBFM_MSG_ERROR_FATAL_TB_ERR | Used for BFM test driver or Root Port BFM fatal errors. Specifies an error that stops simulation because the error leaves the testbench in a state where further simulation is not possible. Use this error message for errors that occur due to a problem in the BFM test driver module or the Root Port BFM, that are not caused by the Endpoint Application Layer being tested. | N/A         | Y<br>Cannot suppress   | Y<br>Cannot suppress        | FATAL :        |

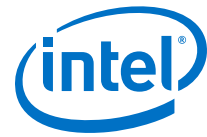
#### 9.4.11.1. ebfm\_display Verilog HDL Function

The `ebfm_display` procedure or function displays a message of the specified type to the simulation standard output and also the log file if `ebfm_log_open` is called.

A message can be suppressed, simulation can be stopped or both based on the default settings of the message type and the value of the bit mask when each of the procedures listed below is called. You can call one or both of these procedures based on what messages you want displayed and whether or not you want simulation to stop for specific messages.

- When `ebfm_log_set_suppressed_msg_mask` is called, the display of the message might be suppressed based on the value of the bit mask.
- When `ebfm_log_set_stop_on_msg_mask` is called, the simulation can be stopped after the message is displayed, based on the value of the bit mask.

| Location | altrpcietb_g3bfm_log.v                                                   |                                                                                                                                                                                                                    |
|----------|--------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax   | Verilog HDL: <code>dummy_return:=ebfm_display(msg_type, message);</code> |                                                                                                                                                                                                                    |
| Argument | msg_type                                                                 | Message type for the message. Should be one of the constants defined in <a href="#">Table 88</a> on page 145.                                                                                                      |
|          | message                                                                  | The message string is limited to a maximum of 100 characters. Also, because Verilog HDL does not allow variable length strings, this routine strips off leading characters of 8'h00 before displaying the message. |
| Return   | always 0                                                                 | Applies only to the Verilog HDL routine.                                                                                                                                                                           |



#### 9.4.11.2. ebfm\_log\_stop\_sim Verilog HDL Function

The ebfm\_log\_stop\_sim procedure stops the simulation.

| Location | altrpcietb_bfm_log.v                             |                                                                                                                                                                                                                                                                                    |
|----------|--------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax   | Verilog HDL: return:=ebfm_log_stop_sim(success); |                                                                                                                                                                                                                                                                                    |
| Argument | success                                          | When set to a 1, this process stops the simulation with a message indicating successful completion. The message is prefixed with SUCCESS.<br>Otherwise, this process stops the simulation with a message indicating unsuccessful completion. The message is prefixed with FAILURE. |
| Return   | Always 0                                         | This value applies only to the Verilog HDL function.                                                                                                                                                                                                                               |

#### 9.4.11.3. ebfm\_log\_set\_suppressed\_msg\_mask Task

The ebfm\_log\_set\_suppressed\_msg\_mask procedure controls which message types are suppressed.

| Location | altrpcietb_bfm_log.v                        |                                                                                                                                                                                                    |
|----------|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax   | ebfm_log_set_suppressed_msg_mask (msg_mask) |                                                                                                                                                                                                    |
| Argument | msg_mask                                    | This argument is reg [EBFM_MSG_ERROR_CONTINUE: EBFM_MSG_DEBUG] .<br>A 1 in a specific bit position of the msg_mask causes messages of the type corresponding to the bit position to be suppressed. |

#### 9.4.11.4. ebfm\_log\_set\_stop\_on\_msg\_mask Verilog HDL Task

The ebfm\_log\_set\_stop\_on\_msg\_mask procedure controls which message types stop simulation. This procedure alters the default behavior of the simulation when errors occur as described in the *BFM Log and Message Procedures*.

| Location | altrpcietb_bfm_log.v                     |                                                                                                                                                                                                                                         |
|----------|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax   | ebfm_log_set_stop_on_msg_mask (msg_mask) |                                                                                                                                                                                                                                         |
| Argument | msg_mask                                 | This argument is reg [EBFM_MSG_ERROR_CONTINUE: EBFM_MSG_DEBUG] .<br>A 1 in a specific bit position of the msg_mask causes messages of the type corresponding to the bit position to stop the simulation after the message is displayed. |

#### Related Information

[BFM Log and Message Procedures](#) on page 147

#### 9.4.11.5. ebfm\_log\_open Verilog HDL Function

The ebfm\_log\_open procedure opens a log file of the specified name. All displayed messages are called by ebfm\_display and are written to this log file as simulator standard output.

| Location | altrpcietb_bfm_log.v |                                                                                   |
|----------|----------------------|-----------------------------------------------------------------------------------|
| Syntax   | ebfm_log_open (fn)   |                                                                                   |
| Argument | fn                   | This argument is type string and provides the file name of log file to be opened. |

#### 9.4.11.6. ebfm\_log\_close Verilog HDL Function

The ebfm\_log\_close procedure closes the log file opened by a previous call to ebfm\_log\_open.

| Location | altrpcietb_bfm_log.v |
|----------|----------------------|
| Syntax   | ebfm_log_close       |
| Argument | NONE                 |

#### 9.4.12. Verilog HDL Formatting Functions

The Verilog HDL Formatting procedures and functions are available in the altrpcietb\_bfm\_log.v. The formatting functions are only used by Verilog HDL. All these functions take one argument of a specified length and return a vector of a specified length.

##### 9.4.12.1. himage1

This function creates a one-digit hexadecimal string representation of the input argument that can be concatenated into a larger message string and passed to ebfm\_display.

| Location     | altrpcietb_bfm_log.v |                                                                                                                 |
|--------------|----------------------|-----------------------------------------------------------------------------------------------------------------|
| Syntax       | string:= himage(vec) |                                                                                                                 |
| Argument     | vec                  | Input data type reg with a range of 3:0.                                                                        |
| Return range | string               | Returns a 1-digit hexadecimal representation of the input argument. Return data is type reg with a range of 8:1 |

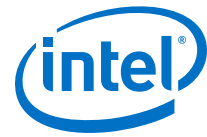
##### 9.4.12.2. himage2

This function creates a two-digit hexadecimal string representation of the input argument that can be concatenated into a larger message string and passed to ebfm\_display.

| Location       | altrpcietb_bfm_log.v |                                                                                                                                                            |
|----------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax         | string:= himage(vec) |                                                                                                                                                            |
| Argument range | vec                  | Input data type reg with a range of 7:0.                                                                                                                   |
| Return range   | string               | Returns a 2-digit hexadecimal presentation of the input argument, padded with leading 0s, if they are needed. Return data is type reg with a range of 16:1 |

##### 9.4.12.3. himage4

This function creates a four-digit hexadecimal string representation of the input argument can be concatenated into a larger message string and passed to ebfm\_display.



| Location       | altpcieth_bfm_log.v                                                                                                                                              |                                           |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|
| Syntax         | string:= himage(vec)                                                                                                                                             |                                           |
| Argument range | vec                                                                                                                                                              | Input data type reg with a range of 15:0. |
| Return range   | Returns a four-digit hexadecimal representation of the input argument, padded with leading 0s, if they are needed. Return data is type reg with a range of 32:1. |                                           |

#### 9.4.12.4. himage8

This function creates an 8-digit hexadecimal string representation of the input argument that can be concatenated into a larger message string and passed to ebfm\_display.

| Location       | altpcieth_bfm_log.v  |                                                                                                                                                                |
|----------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax         | string:= himage(vec) |                                                                                                                                                                |
| Argument range | vec                  | Input data type reg with a range of 31:0.                                                                                                                      |
| Return range   | string               | Returns an 8-digit hexadecimal representation of the input argument, padded with leading 0s, if they are needed. Return data is type reg with a range of 64:1. |

#### 9.4.12.5. himage16

This function creates a 16-digit hexadecimal string representation of the input argument that can be concatenated into a larger message string and passed to ebfm\_display.

| Location       | altpcieth_bfm_log.v  |                                                                                                                                                                 |
|----------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax         | string:= himage(vec) |                                                                                                                                                                 |
| Argument range | vec                  | Input data type reg with a range of 63:0.                                                                                                                       |
| Return range   | string               | Returns a 16-digit hexadecimal representation of the input argument, padded with leading 0s, if they are needed. Return data is type reg with a range of 128:1. |

#### 9.4.12.6. dimage1

This function creates a one-digit decimal string representation of the input argument that can be concatenated into a larger message string and passed to ebfm\_display.

| Location       | altpcieth_bfm_log.v  |                                                                                                                                                                                                                      |
|----------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax         | string:= dimage(vec) |                                                                                                                                                                                                                      |
| Argument range | vec                  | Input data type reg with a range of 31:0.                                                                                                                                                                            |
| Return range   | string               | Returns a 1-digit decimal representation of the input argument that is padded with leading 0s if necessary. Return data is type reg with a range of 8:1.<br>Returns the letter U if the value cannot be represented. |

#### 9.4.12.7. dimage2

This function creates a two-digit decimal string representation of the input argument that can be concatenated into a larger message string and passed to ebfm\_display.

| Location       | altpcieth_bfm_log.v  |                                                                                                                                                                                                                              |
|----------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax         | string:= dimage(vec) |                                                                                                                                                                                                                              |
| Argument range | vec                  | Input data type reg with a range of 31:0.                                                                                                                                                                                    |
| Return range   | string               | Returns a 2-digit decimal representation of the input argument that is padded with leading 0s if necessary. Return data is type reg with a range of 16:1.<br>Returns the letter <i>U</i> if the value cannot be represented. |

#### 9.4.12.8. dimage3

This function creates a three-digit decimal string representation of the input argument that can be concatenated into a larger message string and passed to ebfm\_display.

| Location       | altpcieth_bfm_log.v  |                                                                                                                                                                                                                              |
|----------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax         | string:= dimage(vec) |                                                                                                                                                                                                                              |
| Argument range | vec                  | Input data type reg with a range of 31:0.                                                                                                                                                                                    |
| Return range   | string               | Returns a 3-digit decimal representation of the input argument that is padded with leading 0s if necessary. Return data is type reg with a range of 24:1.<br>Returns the letter <i>U</i> if the value cannot be represented. |

#### 9.4.12.9. dimage4

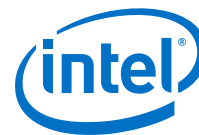
This function creates a four-digit decimal string representation of the input argument that can be concatenated into a larger message string and passed to ebfm\_display.

| Location       | altpcieth_bfm_log.v  |                                                                                                                                                                                                                              |
|----------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax         | string:= dimage(vec) |                                                                                                                                                                                                                              |
| Argument range | vec                  | Input data type reg with a range of 31:0.                                                                                                                                                                                    |
| Return range   | string               | Returns a 4-digit decimal representation of the input argument that is padded with leading 0s if necessary. Return data is type reg with a range of 32:1.<br>Returns the letter <i>U</i> if the value cannot be represented. |

#### 9.4.12.10. dimage5

This function creates a five-digit decimal string representation of the input argument that can be concatenated into a larger message string and passed to ebfm\_display.

| Location       | altpcieth_bfm_log.v  |                                                                                                                                                                                                                              |
|----------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax         | string:= dimage(vec) |                                                                                                                                                                                                                              |
| Argument range | vec                  | Input data type reg with a range of 31:0.                                                                                                                                                                                    |
| Return range   | string               | Returns a 5-digit decimal representation of the input argument that is padded with leading 0s if necessary. Return data is type reg with a range of 40:1.<br>Returns the letter <i>U</i> if the value cannot be represented. |



#### 9.4.12.11. dimage6

This function creates a six-digit decimal string representation of the input argument that can be concatenated into a larger message string and passed to `ebfm_display`.

| Location       | altpcieth_bfm_log.v  |                                                                                                                                                                                                                              |
|----------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax         | string:= dimage(vec) |                                                                                                                                                                                                                              |
| Argument range | vec                  | Input data type reg with a range of 31:0.                                                                                                                                                                                    |
| Return range   | string               | Returns a 6-digit decimal representation of the input argument that is padded with leading 0s if necessary. Return data is type reg with a range of 48:1.<br>Returns the letter <i>U</i> if the value cannot be represented. |

#### 9.4.12.12. dimage7

This function creates a seven-digit decimal string representation of the input argument that can be concatenated into a larger message string and passed to `ebfm_display`.

| Location       | altpcieth_bfm_log.v  |                                                                                                                                                                                                                                      |
|----------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax         | string:= dimage(vec) |                                                                                                                                                                                                                                      |
| Argument range | vec                  | Input data type reg with a range of 31:0.                                                                                                                                                                                            |
| Return range   | string               | Returns a 7-digit decimal representation of the input argument that is padded with leading 0s if necessary. Return data is type reg with a range of 56:1.<br>Returns the letter <i>&lt;U&gt;</i> if the value cannot be represented. |

## 10. Troubleshooting and Observing the Link

### 10.1. Troubleshooting

#### 10.1.1. Simulation Fails To Progress Beyond Polling.Active State

If your PIPE simulation cycles between the Detect.Quiet, Detect.Active, and Polling.Active LTSSM states, the PIPE interface width may be incorrect. The width of the DUT top-level PIPE interface is 32 bits for Intel Stratix 10 devices.

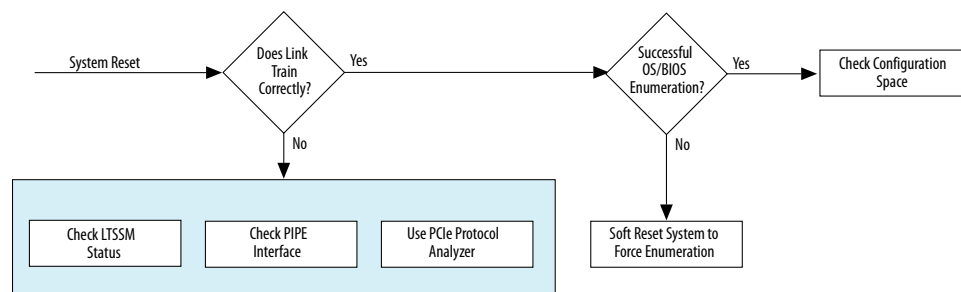
#### 10.1.2. Hardware Bring-Up Issues

Typically, PCI Express hardware bring-up involves the following steps:

1. System reset
2. Link training
3. BIOS enumeration

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

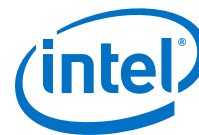
**Figure 77. Debugging Link Training Issues**



#### 10.1.3. Link Training

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. If you encounter link training issues, viewing the actual data in hardware should help you determine the root cause. You can use the following tools to provide hardware visibility:

- Signal Tap Embedded Logic Analyzer
- Third-party PCIe protocol analyzer



You can use Signal Tap Embedded Logic Analyzer to diagnose the LTSSM state transitions that are occurring on the PIPE interface. The `ltssmstate` bus encodes the status of LTSSM. The LTSSM state machine reflects the Physical Layer's progress through the link training process. For a complete description of the states these signals encode, refer to *Reset, Status, and Link Training Signals*. When link training completes successfully and the link is up, the LTSSM should remain stable in the L0 state. When link issues occur, you can monitor `ltssmstate` to determine the cause.

#### 10.1.4. Link Hangs in L0 State

There are many reasons that link may stop transmitting data. The following table lists some possible causes.

**Table 90. Link Hangs in L0**

| Possible Causes                                 | Symptoms and Root Causes                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Workarounds and Solutions                                                                                                                                                                                                                                                                               |
|-------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Avalon-ST signaling violates Avalon-ST protocol | Avalon-ST protocol violations include the following errors: <ul style="list-style-type: none"> <li>More than one <code>tx_st_sop</code> per <code>tx_st_eop</code>.</li> <li>Two or more <code>tx_st_eop</code>'s without a corresponding <code>tx_st_sop</code>.</li> <li><code>rx_st_valid</code> is not asserted with <code>tx_st_sop</code> or <code>tx_st_eop</code>.</li> </ul> These errors are applicable to both simulation and hardware.                                                     | Add logic to detect situations where <code>tx_st_ready</code> remains deasserted for more than 100 cycles. Set post-triggering conditions to check for the Avalon-ST signaling of last two TLPs to verify correct <code>tx_st_sop</code> and <code>tx_st_eop</code> signaling.                          |
| Incorrect payload size                          | Determine if the length field of the last TLP transmitted by End Point is greater than the InitFC credit advertised by the link partner. For simulation, refer to the log file and simulation dump. For hardware, use a third-party logic analyzer trace to capture PCIe transactions.                                                                                                                                                                                                                 | If the payload is greater than the initFC credit advertised, you must either increase the InitFC of the posted request to be greater than the <b>max payload size</b> or reduce the payload size of the requested TLP to be less than the InitFC value.                                                 |
| Flow control credit overflows                   | Determine if the credit field associated with the current TLP type in the <code>tx_cred</code> bus is less than the requested credit value. When insufficient credits are available, the core waits for the link partner to release the correct credit type. Sufficient credits may be unavailable if the link partner increments credits more than expected, creating a situation where the Intel Stratix 10 Hard IP for PCI Express IP Core credit calculation is out-of-sync with the link partner. | Add logic to detect conditions where the <code>tx_st_ready</code> signal remains deasserted for more than 100 cycles. Set post-triggering conditions to check the value of the <code>tx_cred_*</code> and <code>tx_st_*</code> interfaces. Add a FIFO status signal to determine if the TXFIFO is full. |
| Malformed TLP is transmitted                    | Refer to the error log file to find the last good packet transmitted on the link. Correlate this packet with TLP sent on Avalon-ST                                                                                                                                                                                                                                                                                                                                                                     | Revise the Application Layer logic to correct the error condition.                                                                                                                                                                                                                                      |
| continued...                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                         |

| Possible Causes                                   | Symptoms and Root Causes                                                                                                                                                                                                                                                                                                                                                                                                                        | Workarounds and Solutions                                                                                                                                |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                   | <p>interface. Determine if the last TLP sent has any of the following errors:</p> <ul style="list-style-type: none"> <li>The actual payload sent does not match the length field.</li> <li>The format and type fields are incorrectly specified.</li> <li>TD field is asserted, indicating the presence of a TLP digest (ECRC), but the ECRC DWORD is not present at the end of TLP.</li> <li>The payload crosses a 4KByte boundary.</li> </ul> |                                                                                                                                                          |
| Insufficient Posted credits released by Root Port | <p>If a Memory Write TLP is transmitted with a payload greater than the <b>maximum payload size</b>, the Root Port may release an incorrect posted data credit to the Endpoint in simulation. As a result, the Endpoint does not have enough credits to send additional Memory Write Requests.</p>                                                                                                                                              | <p>Make sure Application Layer sends Memory Write Requests with a payload less than or equal the value specified by the <b>maximum payload size</b>.</p> |
| Missing completion packets or dropped packets     | <p>The RX Completion TLP might cause the RX FIFO to overflow. Make sure that the total outstanding read data of all pending Memory Read Requests is smaller than the allocated completion credits in RX buffer.</p>                                                                                                                                                                                                                             | <p>You must ensure that the data for all outstanding read requests does not exceed the completion credits in the RX buffer.</p>                          |

#### Related Information

- [Design Debugging with the Using Signal Tap Logic Analyzer](#)
- [PCI Express Base Specification 3.0](#)

### 10.1.5. Use Third-Party PCIe Analyzer

A third-party protocol analyzer for PCI Express records the traffic on the physical link and decodes traffic, saving you the trouble of translating the symbols yourself. A third-party protocol analyzer can show the two-way traffic at different levels for different requirements. For high-level diagnostics, the analyzer shows the LTSSM flows for devices on both side of the link side-by-side. This display can help you see the link training handshake behavior and identify where the traffic gets stuck. A traffic analyzer can display the contents of packets so that you can verify the contents. For complete details, refer to the third-party documentation.

### 10.1.6. BIOS Enumeration Issues

Both FPGA programming (configuration) and the initialization of a PCIe link require time. Potentially, an Intel FPGA including a Hard IP block for PCI Express may not be ready when the OS/BIOS begins enumeration of the device tree. If the FPGA is not fully programmed when the OS/BIOS begins enumeration, the OS does not include the Hard IP for PCI Express in its device map.

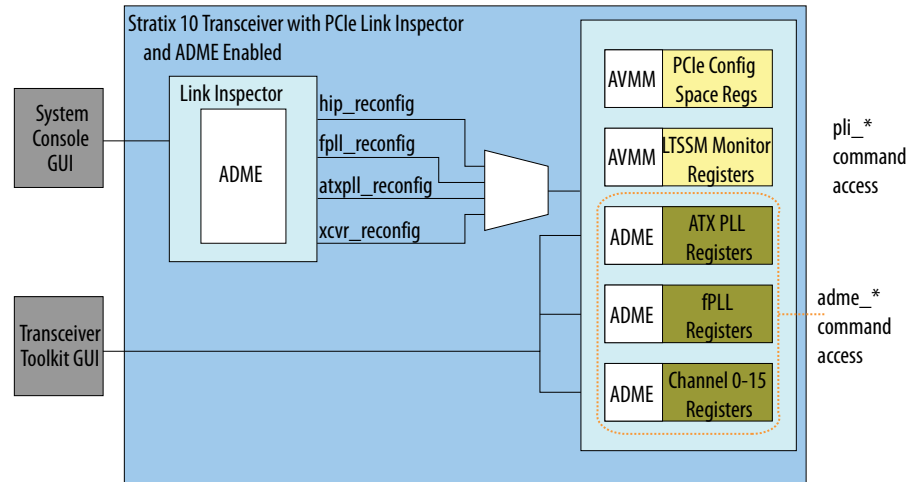
To eliminate this issue, you can perform a soft reset of the system to retain the FPGA programming while forcing the OS/BIOS to repeat enumeration.

## 10.2. PCIe Link Inspector Overview

Use the PCIe Link Inspector to monitor the PCIe link at the Physical, Data Link and Transaction Layers.

The following figure provides an overview of the debug capability available when you enable all of the options on the **Configuration, Debug and Extension Option** tab of the Intel Stratix 10 Hard IP for PCI Express IP component GUI.

**Figure 78. Overview of PCIe Link Inspector Hardware**



As this figure illustrates, the PCIe Link (pli\*) commands provide access to the following registers:

- The PCI Express Configuration Space registers
- LTSSM monitor registers
- ATX PLL dynamic partial reconfiguration I/O (DPRIO) registers from the dynamic reconfiguration interface
- fPLL DPRIO registers from the dynamic reconfiguration interface
- Native PHY DPRIO registers from the dynamic reconfiguration interface

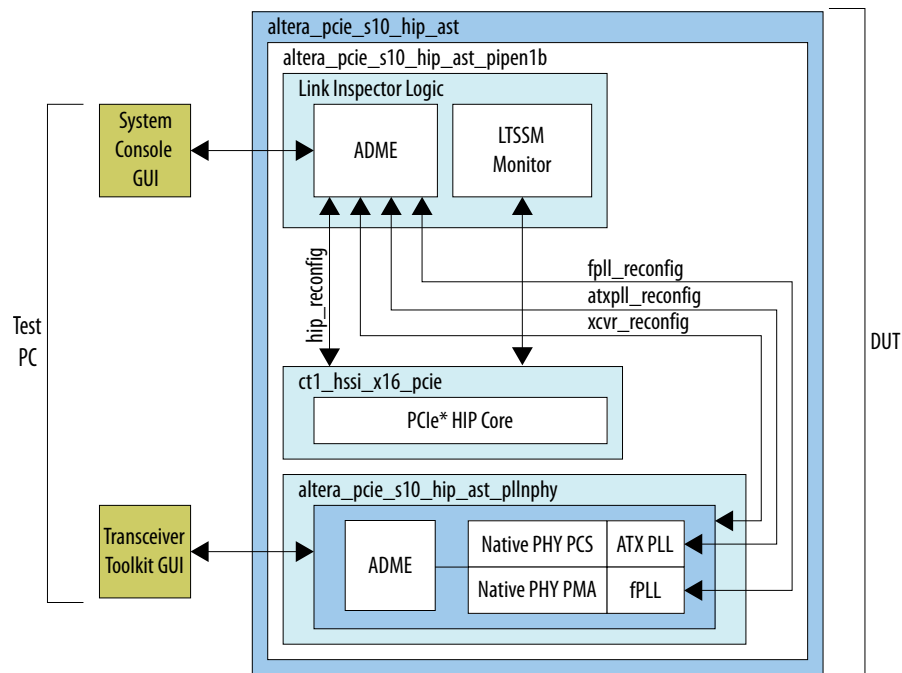
The ADME commands (adme\*\_) provide access to the following registers

- ATX PLL DPRIO registers from the ATX PLL ADME interface
- fPLL DPRIO registers from the fPLL ADME interface
- Native PHY DPRIO registers from the Native PHY ADME interface

### 10.2.1. PCIe Link Inspector Hardware

When you enable, the PCIe Link Inspector, the `altera_pcie_s10_hip_ast_pipen1b` module of the generated IP includes the PCIe Link Inspector as shown in the figure below.

**Figure 79. Intel Stratix 10 Avalon-ST and SR-IOV Hard IP for PCIe IP with the PCIe Link Inspector**



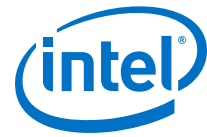
You drive the PCIe Link Inspector from a System Console running on a separate test PC. The System Console connects to the PCIe Link Inspector via an Altera Debug Master Endpoint (ADME). An Intel FPGA Download Cable makes this connection.

**Note:** When you enable the PCIe Link Inspector, the PCIe IP has a clock, `hip_reconfig_clk`, and a reset, `hip_reconfig_rst_n`, brought out at the top level. These signals provide the clock and reset to the following interfaces:

- The ADME module
- FPLL reconfiguration interface (`fpll_reconfig`)
- ATXPLL reconfiguration interface (`atxpll_reconfig`)
- Transceiver reconfiguration interface (`xcvr_reconfig`)
- Hard IP reconfiguration interface (`hip_reconfig`)

You must provide a clock source of up to 100 MHz to drive the `hip_reconfig_clk` clock. When you run a dynamically-generated design example on the Intel Stratix 10-GX Development Kit, these signals are automatically connected.

If you run the PCIe Link Inspector on your own hardware, be sure to connect the `hip_reconfig_clk` to a 100 MHz clock source and the `hip_reconfig_rst_n` to the appropriate reset signal.



When you generate a PCIe design example (with a PCIe IP instantiated) without enabling the Link Inspector, the following interfaces are not exposed at the top level of the PCIe IP:

- fpll\_reconfig interface
- atxpll\_reconfig interface
- xcvr\_reconfig interface
- hip\_reconfig interface

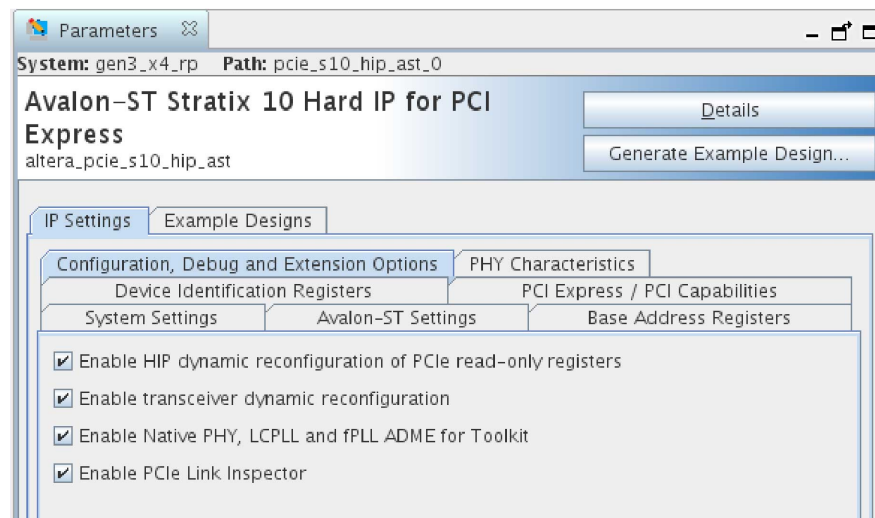
If you later want to enable the Link Inspector using the same design, you need to provide a free-running clock and a reset to drive these interfaces at the top level of the PCIe IP. Intel recommends that you generate a new design example with the Link Inspector enabled. When you do so, the design example will include a free-running clock and a reset for all reconfiguration interfaces.

#### 10.2.1.1. Enabling the PCIe Link Inspector

You enable the PCIe Link Inspector on the **Configuration Debug and Extension Options** tab of the parameter editor. You must also turn on the following parameters to use the **PCIe Link Inspector**:

- **Enable HIP (hard IP) transceiver dynamic reconfiguration**
- **Enable dynamic reconfiguration of PCIe read-only registers**
- **Enable Native PHY, ATX PLL, and fPLL ADME for Transceiver Toolkit**

**Figure 80. Enable the Link Inspector in the Avalon-ST Intel Stratix 10 Hard IP for PCI Express IP**



By default, all of these parameters are disabled.

### 10.2.1.2. Launching the PCIe Link Inspector

Use the design example you compiled in the *Quick Start Guide*, to familiarize yourself with the PCIe Link Inspector. Follow the steps in the *Generating the Avalon-ST Design* or *Generating the Avalon-MM Design* and *Compiling the Design* to generate the SRAM Object File, (.sof) for this design example.

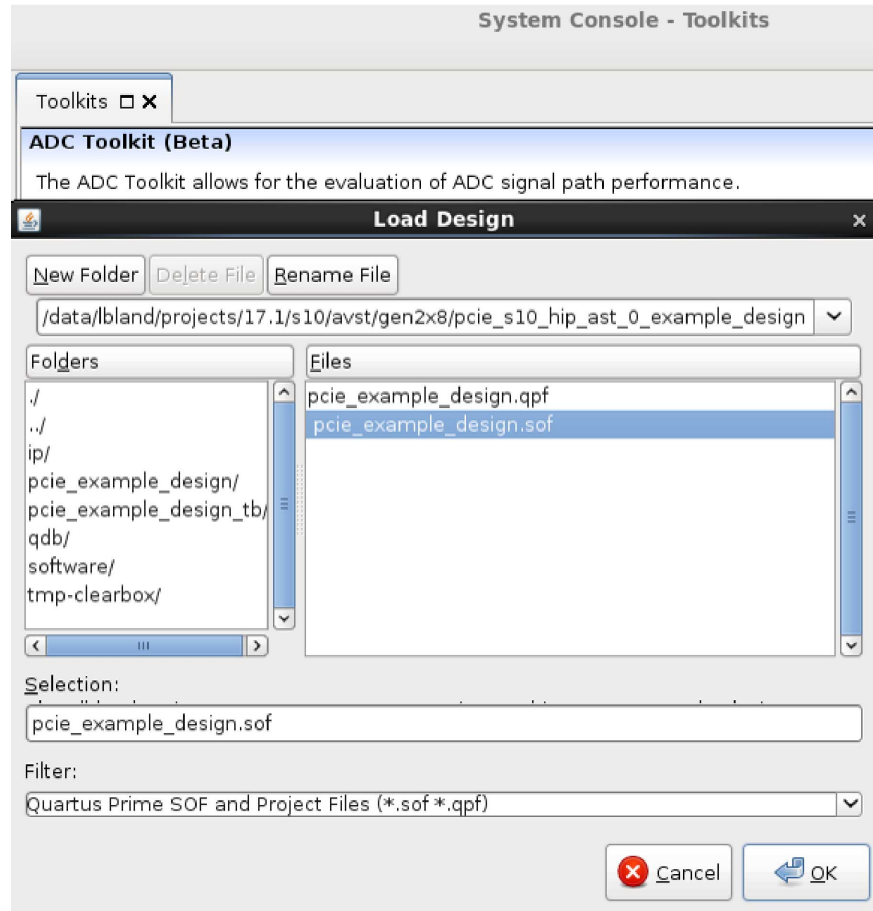
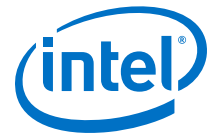
To use the PCIe Link Inspector, download the .sof to the Intel Stratix 10 Development Kit. Then, open the System Console on the test PC and load the design to the System Console as well. Loading the .sof to the System Console allows the System Console to communicate with the design using ADME. ADME is a JTAG-based Avalon-MM master. It drives an Avalon-MM slave interfaces in the PCIe design. When using ADME, the Intel Quartus Prime software inserts the debug interconnect fabric to connect with JTAG.

Here are the steps to complete these tasks:

1. Use the Intel Quartus Prime Programmer to download the .sof to the Intel Stratix 10 FPGA Development Kit.

*Note:* To ensure that you have the correct operation, you must use the same version of the Intel Quartus Prime Programmer and Intel Quartus Prime Pro Edition software that you used to generate the .sof.

2. To load the design to the System Console:
  - a. Launch the Intel Quartus Prime Pro Edition software on the test PC.
  - b. Start the System Console, **Tools > System Debugging Tools > System Console**.
  - c. On the System Console File menu, select **Load design** and browse to the .sof file.



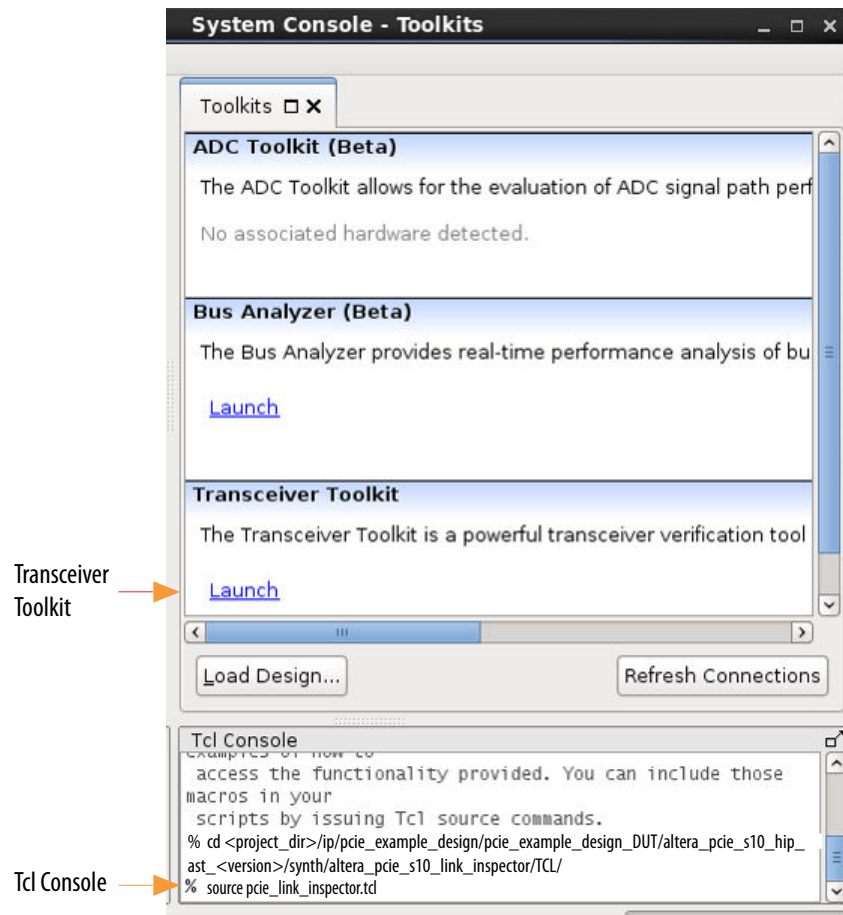
- d. Select the .sof and click **OK**.  
The .sof loads to the System Console.

3. In the System Console Tcl console, type the following commands:

```
% cd <project_dir>/ip/pcie_example_design/
pcie_example_design_DUT/altera_pcie_s10_hip_ast_<version>/synth/
altera_pcie_s10_link_inspector
% source TCL/setup_adme.tcl
% source TCL/xcvr_pll_test_suite.tcl
% source TCL/pcie_link_inspector.tcl
```

The source `TCL/pcie_link_inspector.tcl` command automatically outputs the current status of the PCIe link to the Tcl Console. The command also loads all the PCIe Link Inspector functionality.

**Figure 81. Using the Tcl Console to Access the PCIe Link Inspector**



You can also start the Transceiver Toolkit from the System Console. The Transceiver Toolkit provides real-time monitoring and reconfiguration of the PMA and PCS. For example, you can use the Transceiver Toolkit Eye Viewer to get an estimate of the eye-opening at the receiver serial data sampling point. The PCIe Link Inspector extends the functionality of the Transceiver Toolkit.

#### Related Information

- [Simulating the Design Example](#) on page 22
- [Compiling the Design Example and Programming the Device](#) on page 23

### 10.2.1.3. The PCIe Link Inspector LTSSM Monitor

The LTSSM monitor stores up to 1024 LTSSM states and additional status information in a FIFO. When the FIFO is full, it stops storing. Reading the LTSSM offset at address 0x02 empties the FIFO. The `ltssm_state_monitor.tcl` script implements the LTSSM monitor commands.

#### 10.2.1.3.1. LTSSM Monitor Registers

You can program the LTSSM monitor registers to change the default behavior.

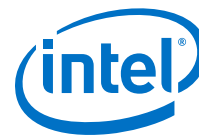
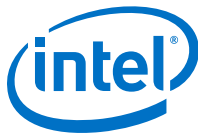


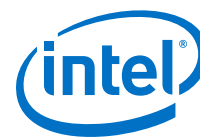
Table 91. LTSSM Registers

| Base Address | LTSSM Address | Access | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|---------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x20000      | 0x00          | RW     | <p>LTSSM Monitor Control register. The LTSSM Monitor Control includes the following fields:</p> <ul style="list-style-type: none"> <li>[1:0]: Timer Resolution Control. Specifies the number of hip_reconfig_clk the PCIe link remains in each LTSSM state. The following encodings are defined: <ul style="list-style-type: none"> <li>2'b00: The main timer increments each hip_reconfig_clk cycle. This is the default value.</li> <li>2'b01: The main timer increments each 16 hip_reconfig_clk cycles.</li> <li>2'b10: The main timer increments each 256 hip_reconfig_clk cycles.</li> <li>2'b11: The main timer increments each &lt;n&gt; hip_reconfig_clk cycles. The Timer Resolution Step field defines &lt;n&gt;.</li> </ul> </li> <li>[17:2]: Timer Resolution Step. Specifies the value of &lt;n&gt; when Timer Resolution Control = 2'b11.</li> <li>[18]: LTSSM FIFO reset. The following encodings are defined: <ul style="list-style-type: none"> <li>1'b0: The LTSSM FIFO operates normally.</li> <li>1'b1: The LTSSM FIFO is in reset.</li> </ul> </li> <li>[19]: Reserved.</li> <li>[20]: LTSSM State Match Enable. The following encodings are defined: <ul style="list-style-type: none"> <li>1'b0: The LTSSM State match function is disabled</li> <li>1'b1: The LTSSM State match function is enabled. When the current LTSSM state matches a state stored in the LTSSM State Match register, the State Match Flag asserts.</li> </ul> </li> <li>[27:22] LTSSM State Match. When enabled, the LTSSM monitor compares the value in this register against each LTSSM state. If the values match, the LTSSM state match flag (offset address 0x01, bit 29) is set to 1.</li> <li>[31:28]: Reserved.</li> </ul> |
| 0x20000      | 0x01          | RO     | <p>LTSSM Quick Debug Status register. The LTSSM Quick Debug Status register includes the following fields:</p> <ul style="list-style-type: none"> <li>[9:0]: Number LTSSM States. Specifies the number of states currently stored in the FIFO.</li> <li>[10]: LTSSM FIFO Full Flag. When asserted, the LTSSM FIFO is full.</li> <li>[11]: LTSSM FIFO Empty Flag. When asserted, the LTSSM FIFO is empty.</li> <li>[12]: Current PERSTN Status. Stores the current PERSTN value.</li> <li>[13]: Current SERDES PLL Locked. The following encodings are defined: <ul style="list-style-type: none"> <li>1'b0: The SERDES PLL is not locked.</li> <li>1'b1: The SERDES PLL is locked.</li> </ul> </li> <li>[14]: PCIe Link Status. The following encodings are defined: <ul style="list-style-type: none"> <li>1'b0: The link is down.</li> <li>1'b1: The link is up.</li> </ul> </li> <li>[16:15] Current PCIe Data Rate. The following encodings are defined: <ul style="list-style-type: none"> <li>2'b00: Reserved.</li> <li>2'b01=Gen1.</li> <li>2'b10=Gen2.</li> <li>2'b11=Gen3.</li> </ul> </li> <li>[17]: Native PHY Channel Locked to Data. The following encodings are defined: <ul style="list-style-type: none"> <li>1'b0: At least one CDR channel is not locked to data.</li> <li>1'b1: All CDR channels are locked to data.</li> </ul> </li> <li>[21:18]: Current Number of PCIe Active Lanes.</li> <li>[22]: Reserved.</li> </ul>                                                                                                                                                                                                                                                                                     |

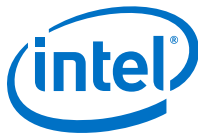
*continued...*



| Base Address | LTSSM Address | Access | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|---------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |               |        | <ul style="list-style-type: none"><li>• [28:23]: Current LTSSM State.</li><li>• [29]: LTSSM State Match Flag. Asserts when the current state matches the state you specify in LTSSM State Match.</li><li>• [31:30]: Reserved.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 0x20000      | 0x02          | RO     | <p>LTSSM FIFO Output.</p> <p>Reading this register is equivalent to reading one entry from the LTSSM FIFO. Reading this register also updates the LTSSM FIFO, 0x03. The following fields are defined:</p> <ul style="list-style-type: none"><li>• [5:0]: LTSSM State.</li><li>• [7:6]: PCIe Current Speed.</li><li>• [12:8]: PCIe Lane Act.</li><li>• [13]: SerDes PLL Locked.</li><li>• [14]: Link Up.</li><li>• [15]: PERSTN.</li><li>• [16]: Native PHY Channel 0. When asserted, the CDR is locked to data.</li><li>• [17]: Native PHY Channel 1. When asserted, the CDR is locked to data.</li><li>• [18]: Native PHY Channel 2. When asserted, the CDR is locked to data.</li><li>• [19]: Native PHY Channel 3. When asserted, the CDR is locked to data.</li><li>• [20]: Native PHY Channel 4. When asserted, the CDR is locked to data.</li><li>• [21]: Native PHY Channel 5. When asserted, the CDR is locked to data.</li><li>• [22]: Native PHY Channel 6. When asserted, the CDR is locked to data.</li><li>• [23]: Native PHY Channel 7. When asserted, the CDR is locked to data.</li><li>• [24]: Native PHY Channel 8. When asserted, the CDR is locked to data.</li><li>• [25]: Native PHY Channel 9. When asserted, the CDR is locked to data.</li><li>• [26]: Native PHY Channel 10. When asserted, the CDR is locked to data.</li><li>• [27]: Native PHY Channel 11. When asserted, the CDR is locked to data.</li><li>• [29]: Native PHY Channel 12. When asserted, the CDR is locked to data.</li><li>• [28]: Native PHY Channel 13. When asserted, the CDR is locked to data.</li><li>• [30]: Native PHY Channel 14. When asserted, the CDR is locked to data.</li><li>• [31]: Native PHY Channel 15. When asserted, the CDR is locked to data.</li></ul> |
| 0x20000      | 0x03          | RO     | <p>LTSSM FIFO Output [63:32]</p> <p>[29:0] Main Timer. The timer resets to 0 on each LTSSM transition. The value in this register indicates how long the PCIe link remains in each LTSSM state.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 0x20000      | 0x04          | RW     | <p>LTSSM Skip State Storage Control register. Use this register to specify a maximum of 4 LTSSM states. When LTSSM State Skip Enable is on, the LTSSM FIFO does not store the specified state or states.</p> <p>Refer to <a href="#">Table 92</a> on page 165 for the state encodings.</p> <p>[5:0]: LTSSM State 1.</p> <p>[6]: LTSSM State 1 Skip Enable.</p> <p>[12:7]: LTSSM State 2.</p> <p>[13]: LTSSM State 2 Skip Enable.</p> <p>[19:14]: LTSSM State 3.</p> <p>[20]: LTSSM State 3 Skip Enable.</p> <p>[26:21]: LTSSM State 4.</p> <p>[27]: LTSSM State 4 Skip Enable.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

**Table 92. LTSSM State Encodings for the LTSSM Skip Field**

| State                          | Encoding |
|--------------------------------|----------|
| Detect.Quiet                   | 6'h00    |
| Detect.Active                  | 6'h01    |
| Polling.Active                 | 6'h02    |
| Polling.Compliance             | 6'h03    |
| Polling.Configuration          | 6'h04    |
| PreDetect.Quiet                | 6'h05    |
| Detect.Wait                    | 6'h06    |
| Configuration.Linkwidth.Start  | 6'h07    |
| Configuration.Linkwidth.Accept | 6'h08    |
| Configuration.Lanenum.Wait     | 6'h09    |
| Configuration.Lanenum.Accept   | 6'h0A    |
| Configuration.Complete         | 6'h0B    |
| Configuration.Idle             | 6'h0C    |
| Recovery.RcvrLock              | 6'h0D    |
| Recovery.Speed                 | 6'h0E    |
| Recovery.RcvrCfg               | 6'h0F    |
| Recovery.Idle                  | 6'h10    |
| Recovery.Equalization Phase 0  | 6'h20    |
| Recovery.Equalization Phase 1  | 6'h21    |
| Recovery.Equalization Phase 2  | 6'h22    |
| Recovery.Equalization Phase 3  | 6'h23    |
| L0                             | 6'h11    |
| L0s                            | 6'h12    |
| L123.SendEIdle                 | 6'h13    |
| L1.Idle                        | 6'h14    |
| L2.Idle                        | 6'h15    |
| L2.TransmitWake                | 6'h16    |
| Disabled.Entry                 | 6'h17    |
| Disabled.Idle                  | 6'h18    |
| Disabled                       | 6'h19    |
| Loopback.Entry                 | 6'h1A    |
| Loopback.Active                | 6'h1B    |
| Loopback.Exit                  | 6'h1C    |
| <i>continued...</i>            |          |



| State                 | Encoding |
|-----------------------|----------|
| Loopback.Exit.Timeout | 6'h1D    |
| HotReset.Entry        | 6'h1E    |
| Hot.Reset             | 6'h1F    |

### 10.2.1.3.2. ltssm\_save2file <file\_name>

The `ltssm_save2file <file_name>` command empties and LTSSM FIFO and saves all states to the file name you specify.

```
% ltssm_save2file <file_name>
```

### 10.2.1.3.3. ltssm\_debug\_check

The `ltssm_debug_check` command returns the current PCIe link status. It provides general information about the health of the link. Because this command returns real-time data, an unstable link returns different states every time you run it.

```
% ltssm_debug_check
```

Sample output:

```
#####
LTSSM Quick Debugs Check Status
#####
This PCIe is GEN1X8
Pin_Perstn signal is High
SerDes PLL is Locked
Link is Up
NOT all Channel CDRs are Locked to Data
LTSSM FIFO is Full
LTSSM FIFO is NOT Empty
LTSSM FIFO stored 1024 data
Current LTSSM State is 010001 L0
```

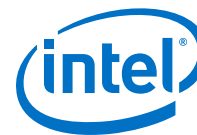
### 10.2.1.3.4. ltssm\_display <num\_states>

The `ltssm_display <num_states>` command reads data from the LTSSM FIFO and outputs <num\_states> LTSSM states and associated status to the System Console.

```
% ltssm_display <num_states>
```

Sample output:

| LTSSM[4:0]              | Perstn | Locktodata | Link Up | Active Lanes | Current Speed | Timer    | PLL Locked |
|-------------------------|--------|------------|---------|--------------|---------------|----------|------------|
| 0x00 Detect.Quiet       | 0      | 0x0000     | 0       | 0            | 00            | 0        | 0          |
| 0x00 Detect.Quiet       | 1      | 0x00FF     | 1       | 8            | 00            | 183237   | 1          |
| 0x11 L0                 | 1      | 0x00FF     | 1       | 8            | 01            | 26607988 | 1          |
| 0x0D Recovery.Rcvlock   | 1      | 0x00FF     | 1       | 8            | 01            | 252      | 1          |
| 0x0F Recovery.Rcvconfig | 1      | 0x00FF     | 1       | 8            | 01            | 786      | 1          |
| continued...            |        |            |         |              |               |          |            |



|                         |   |        |   |   |      |         |   |
|-------------------------|---|--------|---|---|------|---------|---|
| 0x0E Recovery.Speed     | 1 | 0x00FF | 1 | 8 | 02   | 175242  | 1 |
| 0x0D Recovery.Rcvlock   | 1 | 0x00FF | 1 | 8 | 02   | 6291458 | 1 |
| 0x0E Recovery.Speed     | 1 | 0x00FF | 1 | 8 | 01   | 175296  | 1 |
| 0x0D Recovery.Rcvlock   | 1 | 0x00FF | 1 | 8 | 01   | 2628    | 1 |
| 0x0F Recovery.Rcvconfig | 1 | 0x00FF | 1 | 8 | 0001 | 602     | 1 |
| 0x00 Recovery.Idle      | 1 | 0x00FF |   |   | 0001 | 36      | 1 |

You can use this command to empty the FIFO by setting <num\_states> to 1024:

```
% ltssm_display 1024
```

## 10.2.1.4. Accessing the Configuration Space and Transceiver Registers

### 10.2.1.4.1. PCIe Link Inspector Commands

These commands use the PCIe Link Inspector connection to read and write registers in the Configuration Space, LTSSM monitor, PLLs, and Native PHY channels.

**Table 93. PCIe Link Inspector (PLI) Commands**

These commands are available in the link\_insp\_test\_suite.tcl script.

| Command                                                                               | Description                                                                                                                                                                                                                                                                                                                |
|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pli_read32 <slave_if> <pli_base_addr><br><pli_reg_addr>                               | Performs a 32-bit read from the slave interface at the base address and register address specified.                                                                                                                                                                                                                        |
| pli_read8 <slave_if> <base_addr> <reg_addr>                                           | Performs an 8-bit read from the slave interface at the base address and register address specified.                                                                                                                                                                                                                        |
| pli_write32 <slave_if> <pli_base_addr><br><pli_reg_addr> <value>                      | Performs a 32-bit write of the value specified to the slave interface at the base address and the register address specified.                                                                                                                                                                                              |
| pli_write8 <slave_if> <base_addr> <reg_addr><br><value>                               | Performs an 8-bit write of the value specified to the slave interface at the base address and the register address specified.                                                                                                                                                                                              |
| pli_rmw32 <slave_if> <base_addr> <reg_addr><br><bit_mask> <value>                     | Performs a 32-bit read-modify-write of the value specified to the slave interface to the slave interface at the base address and register address using the bit mask specified.                                                                                                                                            |
| pli_rmw8 <slave_if> <base_addr> <reg_addr><br><bit_mask> <value>                      | Performs an 8-bit read-modify-write to the slave interface at the base address and register address using the bit mask specified.                                                                                                                                                                                          |
| pli_dump_to_file <slave_if> <filename><br><base_addr> <start_reg_addr> <end_reg_addr> | Writes the contents of the slave interface to the file specified. The base address and the start and end register addresses specify range of the write.<br>The <slave_if> argument can have the following values: <ul style="list-style-type: none"> <li>\$atxp11</li> <li>\$fp11</li> <li>\$channel(&lt;n&gt;)</li> </ul> |

### PCIe Link Inspector Command Examples

The following commands use the addresses specified below in the *Register Address Map*.

Use the following command to read register 0x480 from the ATX PLL:

```
% pli_read8 $pli_adme $atxp11_base_addr 0x480
```

Use the following command to write 0xFF to the fPLL register at address 0x4E0:

```
% pli_write8 $pli_adme $fp11_base_addr 0x4E0 0xFF
```

Use the following command to perform a read-modify-write to write 0x02 to channel 3 with a bit mask of 0x03 for the write:

```
% pli_rmw8 $pli_adme $xcvr_ch3_base_addr 0x481 0x03 0x02
```

Use the following command to instruct the LTSSM monitor to skip recording of the Recovery.Rcvlock state:

```
$pli_write $pli_adme $ltssm_base_addr 0x04 0x0000000D
```

### Related Information

[Register Address Map](#) on page 169

#### 10.2.1.4.2. ADME PLL and Channel Commands

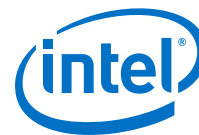
These commands use the Native PHY and channel PLL ADME master ports to read and write registers in the ATX PLL, fPLL, and Native PHY transceiver channels.

**Table 94. ADME Commands To Access PLLs and Channels**

These commands are available in the `xcvr_pll_test_suite.tcl`

| Command                                                                                              | Description                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>adme_read32 &lt;slave_if&gt; &lt;reg_addr&gt;</code>                                           | Performs a 32-bit read from the slave interface of the register address specified.                                                                                                                                                                                                                                                                                  |
| <code>adme_read8 &lt;slave_if&gt; &lt;reg_addr&gt;</code>                                            | Performs an 8-bit read from the slave interface of the register address specified.                                                                                                                                                                                                                                                                                  |
| <code>adme_write32 &lt;slave_if&gt; &lt;reg_addr&gt; &lt;value&gt;</code>                            | Performs a 32-bit write of the value specified to the slave interface and register specified                                                                                                                                                                                                                                                                        |
| <code>adme_write8 &lt;slave_if&gt; &lt;reg_addr&gt; &lt;value&gt;</code>                             | Performs an 8-bit write of the value specified to the slave interface and register specified                                                                                                                                                                                                                                                                        |
| <code>adme_rmw32 &lt;slave_if&gt; &lt;reg_addr&gt; &lt;bit_mask&gt; &lt;value&gt;</code>             | Performs a 32-bit read-modify-write to the slave interface at the register address using the bit mask specified.                                                                                                                                                                                                                                                    |
| <code>adme_rmw8 &lt;slave_if&gt; &lt;reg_addr&gt; &lt;bit_mask&gt; &lt;value&gt;</code>              | Performs an 8-bit read-modify-write to the slave interface at the register address using the bit mask specified.                                                                                                                                                                                                                                                    |
| <code>adme_dump_to_file &lt;slave_if&gt; &lt;filename&gt; &lt;start_addr&gt; &lt;end_addr&gt;</code> | Writes the contents of the slave interface to the file specified. The start and end register addresses specify the range of the write.<br>The <code>&lt;slave_if&gt;</code> argument can have the following values: <ul style="list-style-type: none"> <li><code>\$atxp11</code></li> <li><code>\$fp11</code></li> <li><code>\$channel(&lt;n&gt;)</code></li> </ul> |

*continued...*



| Command       | Description                                                                                    |
|---------------|------------------------------------------------------------------------------------------------|
| atxpll_check  | Checks the ATX PLL lock and calibration status                                                 |
| fpll_check    | Checks the fPLL lock and calibration status                                                    |
| channel_check | Checks each channel clock data recovery (CDR) lock status and the TX and RX calibration status |

### ADME Command Examples

The following PLL commands use the addresses specified below in the *Register Address Map*.

Use the following command to read the value register address 0x480 in the ATX PLL:

```
% adme_read8 $atxpll_adme 0x480
```

Use the following command to write 0xFF to register address 0x4E0 in the fPLL:

```
% adme_write8 $fpll_adme 0x4E0 0xFF
```

Use the following command to perform a read-modify-write to write register address 0x02 in channel 3:

```
% adme_rmw8 $channel_adme(3) 0x03 0x02
```

Use the following command to save the register values from 0x100-0x200 from the ATX PLL to a file:

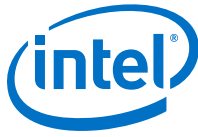
```
% adme_dump_to_file $atxpll <directory_path>atx_regs.txt 0x100 0x200
```

#### 10.2.1.4.3. Register Address Map

Here are the base addresses when you run the as defined in `link_insp_test_suite.tcl` and `ltssm_state_monitor.tcl`.

**Table 95. PCIe Link Inspector and LTSSM Monitor Register Addresses**

| Base Address | Functional Block     | Access |
|--------------|----------------------|--------|
| 0x00000      | fPLL                 | RW     |
| 0x10000      | ATX PLL              | RW     |
| 0x20000      | LTSSM Monitor        | RW     |
| 0x40000      | Native PHY Channel 0 | RW     |
| 0x42000      | Native PHY Channel 1 | RW     |
| 0x44000      | Native PHY Channel 2 | RW     |
| 0x46000      | Native PHY Channel 3 | RW     |
| 0x48000      | Native PHY Channel 4 | RW     |
| 0x4A000      | Native PHY Channel 5 | RW     |
| 0x4C000      | Native PHY Channel 6 | RW     |
| continued... |                      |        |



| Base Address | Functional Block         | Access |
|--------------|--------------------------|--------|
| 0x4E000      | Native PHY Channel 7     | RW     |
| 0x50000      | Native PHY Channel 8     | RW     |
| 0x52000      | Native PHY Channel 9     | RW     |
| 0x54000      | Native PHY Channel 10    | RW     |
| 0x56000      | Native PHY Channel 11    | RW     |
| 0x58000      | Native PHY Channel 12    | RW     |
| 0x5A000      | Native PHY Channel 13    | RW     |
| 0x5C000      | Native PHY Channel 14    |        |
| 0x5E000      | Native PHY Channel 15    | RW     |
| 0x80000      | PCIe Configuration Space | RW     |

**Related Information**

- [Configuration Space Registers](#) on page 106
- [Logical View of the L-Tile/H-Tile Transceiver Registers](#)  
For detailed register descriptions for the ATX PLL, fPLL, PCS, and PMA registers.

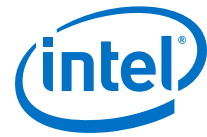
**10.2.1.5. Additional Status Commands****10.2.1.5.1. Displaying PLL Lock and Calibration Status Registers**

1. Run the following System Console Tcl commands to display the lock and the calibration status for the PLLs and channels.

```
% source TCL/setup_adme.tcl
% source TCL/xcvr_pll_suite.tcl
```

2. Here are sample transcripts:

```
#####
ATXPLL Status
#####
ATXPLL is Locked
ATXPLL Calibration is Done
#####
FPLL Status
#####
FPLL is Locked
FPLL Calibration is Done
#####
Channel# 0 Status
#####
Channel#0 CDR is Locked to Data
Channel#0 CDR is Locked to Reference Clock
Channel#0 TX Calibration is Done
Channel#0 RX Calibration is Done
#####
...
#####
Channel#7 CDR is Locked to Data
Channel#7 CDR is Locked to Reference Clock
Channel#7 TX Calibration is Done
Channel#7 RX Calibration is Done
```



## A. PCI Express Core Architecture

---

### A.1. Transaction Layer

The Transaction Layer is located between the Application Layer and the Data Link Layer. It generates and receives Transaction Layer Packets. The following illustrates the Transaction Layer. The Transaction Layer includes three sub-blocks: the TX datapath, Configuration Space, and RX datapath.

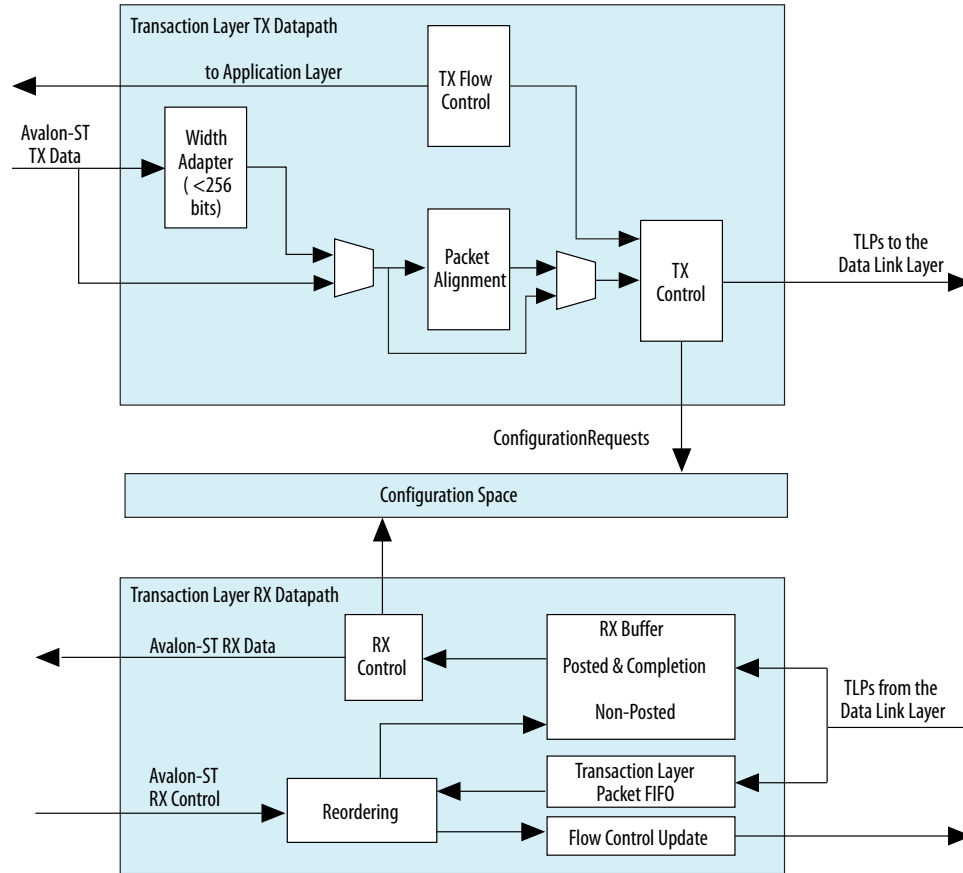
Tracing a transaction through the RX datapath includes the following steps:

1. The Transaction Layer receives a TLP from the Data Link Layer.
2. The Configuration Space determines whether the TLP is well formed and directs the packet based on traffic class (TC).
3. TLPs are stored in a specific part of the RX buffer depending on the type of transaction (posted, non-posted, and completion).
4. The receive reordering block reorders the queue of TLPs as needed, fetches the address of the highest priority TLP from the TLP FIFO block, and initiates the transfer of the TLP to the Application Layer.

Tracing a transaction through the TX datapath involves the following steps:

1. The Transaction Layer informs the Application Layer that sufficient flow control credits exist for a particular type of transaction using the TX credit signals. The Application Layer may choose to ignore this information.
2. The Application Layer requests permission to transmit a TLP. The Application Layer must provide the transaction and must be prepared to provide the entire data payload in consecutive cycles.
3. The Transaction Layer verifies that sufficient flow control credits exist and acknowledges or postpones the request. If there is insufficient space in the retry buffer, the Transaction Layer does not accept the TLP.
4. The Transaction Layer forwards the TLP to the Data Link Layer.

**Figure 82. Architecture of the Transaction Layer: Dedicated Receive Buffer**

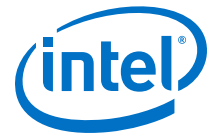
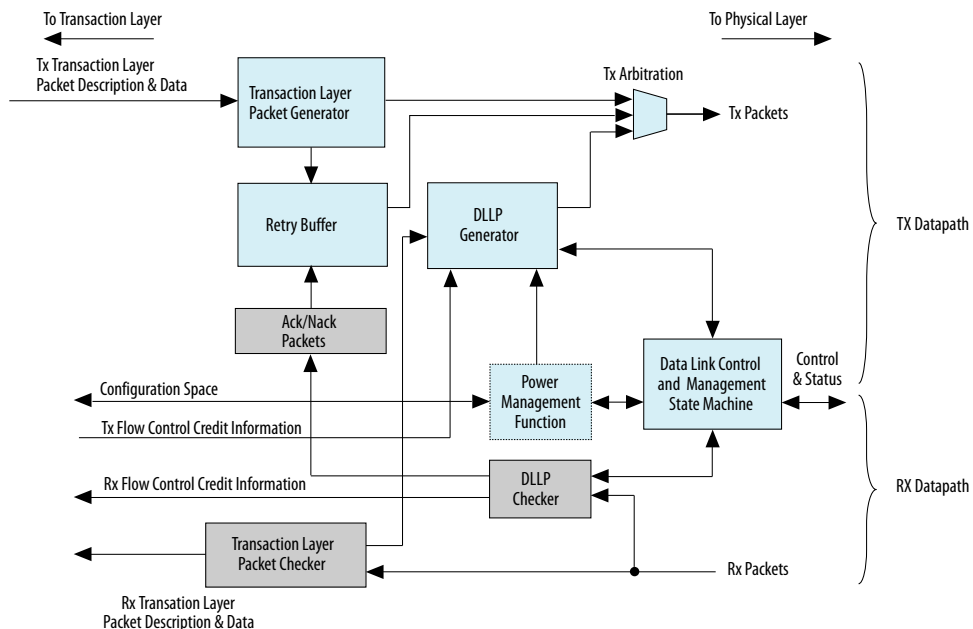


## A.2. Data Link Layer

The Data Link Layer is located between the Transaction Layer and the Physical Layer. It maintains packet integrity and communicates (by DLL packet transmission) at the PCI Express link level.

The DLL implements the following functions:

- Link management through the reception and transmission of DLL Packets (DLLP), which are used for the following functions:
  - Power management of DLLP reception and transmission
  - To transmit and receive ACK/NAK packets
  - Data integrity through generation and checking of CRCs for TLPs and DLLPs
  - TLP retransmission in case of NAK DLLP reception or replay timeout, using the retry (replay) buffer
  - Management of the retry buffer
  - Link retraining requests in case of error through the Link Training and Status State Machine (LTSSM) of the Physical Layer

**Figure 83. Data Link Layer**

The DLL has the following sub-blocks:

- **Data Link Control and Management State Machine**—This state machine connects to both the Physical Layer's LTSSM state machine and the Transaction Layer. It initializes the link and flow control credits and reports status to the Transaction Layer.
- **Power Management**—This function handles the handshake to enter low power mode. Such a transition is based on register values in the Configuration Space and received Power Management (PM) DLLPs. All of the Intel Stratix 10 Hard IP for PCIe IP core variants do not support low power modes.
- **Data Link Layer Packet Generator and Checker**—This block is associated with the DLLP's 16-bit CRC and maintains the integrity of transmitted packets.
- **Transaction Layer Packet Generator**—This block generates transmit packets, including a sequence number and a 32-bit Link CRC (LCRC). The packets are also sent to the retry buffer for internal storage. In retry mode, the TLP generator receives the packets from the retry buffer and generates the CRC for the transmit packet.
- **Retry Buffer**—The retry buffer stores TLPs and retransmits all unacknowledged packets in the case of NAK DLLP reception. In case of ACK DLLP reception, the retry buffer discards all acknowledged packets.

- ACK/NAK Packets—The ACK/NAK block handles ACK/NAK DLLPs and generates the sequence number of transmitted packets.
- Transaction Layer Packet Checker—This block checks the integrity of the received TLP and generates a request for transmission of an ACK/NAK DLLP.
- TX Arbitration—This block arbitrates transactions, prioritizing in the following order:
  - Initialize FC Data Link Layer packet
  - ACK/NAK DLLP (high priority)
  - Update FC DLLP (high priority)
  - PM DLLP
  - Retry buffer TLP
  - TLP
  - Update FC DLLP (low priority)
  - ACK/NAK FC DLLP (low priority)

### A.3. Physical Layer

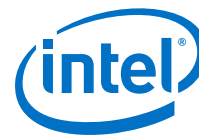
The Physical Layer is the lowest level of the PCI Express protocol stack. It is the layer closest to the serial link. It encodes and transmits packets across a link and accepts and decodes received packets. The Physical Layer connects to the link through a high-speed SERDES interface running at 2.5 Gbps for Gen1 implementations, at 2.5 or 5.0 Gbps for Gen2 implementations, and at 2.5, 5.0 or 8.0 Gbps for Gen3 implementations.

The Physical Layer is responsible for the following actions:

- Training the link
- Scrambling/descrambling and 8B/10B encoding/decoding for 2.5 Gbps (Gen1), 5.0 Gbps (Gen2), or 128b/130b encoding/decoding of 8.0 Gbps (Gen3) per lane
- Serializing and deserializing data
- Equalization (Gen3)
- Operating the PIPE 3.0 Interface
- Implementing auto speed negotiation (Gen2 and Gen3)
- Transmitting and decoding the training sequence
- Providing hardware autonomous speed control
- Implementing auto lane reversal

The Physical Layer is subdivided by the PIPE Interface Specification into two layers (bracketed horizontally in above figure):

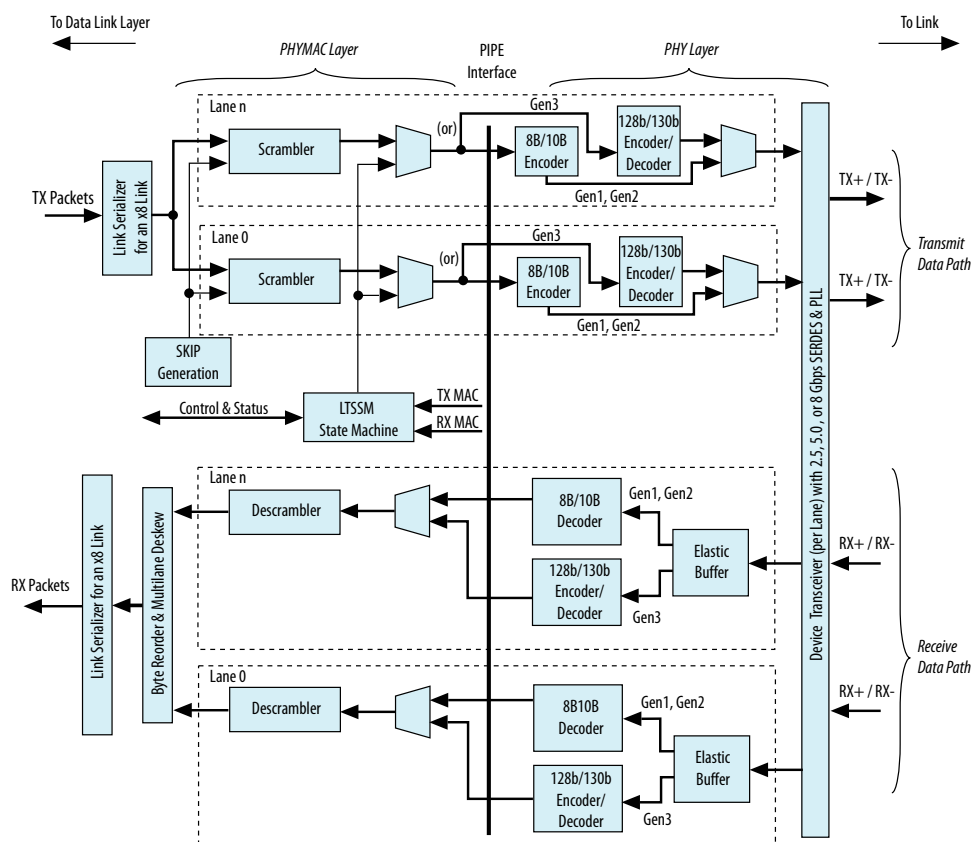
- PHYMAC—The MAC layer includes the LTSSM and the scrambling/descrambling, byte reordering, and multilane deskew functions.
- PHY Layer—The PHY layer includes the 8B/10B encode and decode functions for Gen1 and Gen2. It includes 128b/130b encode and decode functions for Gen3. The PHY also includes elastic buffering and serialization/deserialization functions.



The Physical Layer integrates both digital and analog elements. Intel designed the PIPE interface to separate the PHYMAC from the PHY. The Intel Stratix 10 Hard IP for PCI Express complies with the PIPE interface specification.

**Note:** The internal PIPE interface is visible for simulation. It is not available for debugging in hardware using a logic analyzer such as Signal Tap. If you try to connect Signal Tap to this interface the design fails compilation.

**Figure 84. Physical Layer Architecture**



The PHYMAC block comprises four main sub-blocks:

- MAC Lane—Both the RX and the TX path use this block.
  - On the RX side, the block decodes the Physical Layer packet and reports to the LTSSM the type and number of TS1/TS2 ordered sets received.
  - On the TX side, the block multiplexes data from the DLL and the Ordered Set and SKP sub-block (LTSTX). It also adds lane specific information, including the lane number and the force PAD value when the LTSSM disables the lane during initialization.
- LTSSM—This block implements the LTSSM and logic that tracks TX and RX training sequences on each lane.
- For transmission, it interacts with each MAC lane sub-block and with the LTSTX sub-block by asserting both global and per-lane control bits to generate specific Physical Layer packets.
  - On the receive path, it receives the Physical Layer packets reported by each MAC lane sub-block. It also enables the multilane deskew block. This block reports the Physical Layer status to higher layers.
  - LTSTX (Ordered Set and SKP Generation)—This sub-block generates the Physical Layer packet. It receives control signals from the LTSSM block and generates Physical Layer packet for each lane. It generates the same Physical Layer Packet for all lanes and PAD symbols for the link or lane number in the corresponding TS1/TS2 fields. The block also handles the receiver detection operation to the PCS sub-layer by asserting predefined PIPE signals and waiting for the result. It also generates a SKP Ordered Set at every predefined timeslot and interacts with the TX alignment block to prevent the insertion of a SKP Ordered Set in the middle of packet.
  - Deskew—This sub-block performs the multilane deskew function and the RX alignment between the initialized lanes and the datapath. The multilane deskew implements an eight-word FIFO buffer for each lane to store symbols. Each symbol includes eight data bits, one disparity bit, and one control bit. The FIFO discards the FTS, COM, and SKP symbols and replaces PAD and IDL with D0.0 data. When all eight FIFOs contain data, a read can occur. When the multilane lane deskew block is first enabled, each FIFO begins writing after the first COM is detected. If all lanes have not detected a COM symbol after seven clock cycles, they are reset and the resynchronization process restarts, or else the RX alignment function recreates a 64-bit data word which is sent to the DLL.

## B. TX Credit Adjustment Sample Code

This sample Verilog HDL code computes the available credits for non-posted TLPs. It provides the updated credit information from the remote device on the tx\_nph\_cdts and tx\_npd\_cdts buses. The tx\_nph\_cdts and tx\_npd\_cdts buses drive the actual available credit space in the link partner's RX buffer. The credit information contained in these buses is difficult to use because, due to the EMIB (Embedded Multi-die Interconnect Bridge) latency, the values that you can observe on these buses are not real-time values.

The following Verilog RTL restores the credit limit for non-posted TLPs that can be used by application logic before it sends a TLP.

```
module nph_credit_limit_gen (
 input clk,
 input rst_n,
 input [7:0] tx_nph_cdts,
 input tx_hdr_cdts_consumed,
 input [1:0] tx_cdts_type,
 output [7:0] tx_nph_cdts_limit
);

 reg tx_nph_credit_consume_1r;
 reg tx_nph_credit_consume_2r;
 reg [7:0] tx_nph_credit_consume_count_hip_3r;

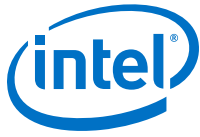
 always @(posedge clk) begin
 if (!rst_n) begin
 tx_nph_credit_consume_1r <= 1'b0;
 tx_nph_credit_consume_2r <= 1'b0;
 tx_nph_credit_consume_count_hip_3r <= 8'h0;
 end else begin
 tx_nph_credit_consume_1r <= (tx_cdts_type == 2'b01) ?
 tx_hdr_cdts_consumed : 1'b0;
 tx_nph_credit_consume_2r <= tx_nph_credit_consume_1r;
 tx_nph_credit_consume_count_hip_3r <=
 tx_nph_credit_consume_count_hip_3r + tx_nph_credit_consume_2r;
 end
 end

 assign tx_nph_cdts_limit = tx_nph_cdts +
 tx_nph_credit_consume_count_hip_3r;
endmodule
```

The following pseudo-code explains the Verilog RTL above.

```
// reset credit_consume_count initially
tx_nph_credit_consume_count = 0;
tx_nph_credit_consume_count_hip = 0;

if (tx_nph_credit_limit_count - (tx_nph_credit_consume_count +
tx_nph_credit_required) <= (2^8)/2) {
 send NPH packet
 tx_nph_credit_consume_count += tx_nph_credit_required;
```



```
}

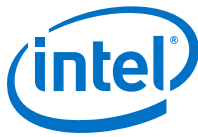
where
tx_nph_credit_required: the number of credits required to send given NP TLP.
For NP, tx_nph_credit_required is 1.
tx_nph_credit_consume_count_hip += (tx_cdts_type == "01") ?
tx_hdr_cdts_consumed : 1'b0;
tx_nph_credit_consume_count_hip_delayed : three pld_clk cycle delayed
tx_nph_credit_consume_count_hip
tx_nph_credit_limit_count = tx_nph_cdts +
tx_nph_credit_consume_count_hip_delayed;
```

## C. Document Revision History

### C.1. Document Revision History for the Intel Stratix 10 Avalon-ST and Single-Root I/O Virtualization (SR-IOV) Interfaces for PCI Express Solutions User Guide

| Document Version | Intel Quartus Prime Version | Changes                                                                                     |
|------------------|-----------------------------|---------------------------------------------------------------------------------------------|
| 2018.09.04       | 18.0                        | Changed the flr_pf_active signal name to flr_pf_active_o, and flr_pf_done to flr_pf_done_i. |
| 2018.08.29       | 18.0                        | Added the step to invoke vsim to the instructions for running a ModelSim simulation.        |

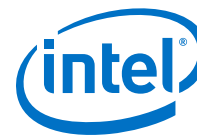
| Date          | Version | Changes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| May 2018      | 18.0    | <p>Made the following changes to the user guide:</p> <ul style="list-style-type: none"> <li>Updated <i>Section 1.2: Features</i> to state that AER is always enabled for Stratix 10 devices.</li> <li>Updated <i>Section 2.3: Generating the Design Example</i> to describe the recommended_pinassignments_s10.txt file and its function, and to update some steps in the generation flow.</li> <li>Changed the latency of tx_st_ready from 14 to 3 cycles in <i>Section 3.2: Avalon-ST TX Interface</i>. Also added a timing diagram showing the proper assertion and deassertion of tx_st_valid.</li> <li>Added a note to <i>Section 4.2: Multifunction and SR-IOV System Settings</i> to state that different functions cannot use the same tag for their memory read requests.</li> <li>Updated <i>Section 6.1.1: TLP Header and Data Alignment for the Avalon-ST RX and TX Interfaces</i> to show the correct byte ordering for the Header dwords.</li> <li>Changed the rx_st_ready_i to rx_st_valid_o[1:0] latency from 6 to 14 cycles in <i>Section 6.1.3: Avalon-ST 512-Bit RX Interface</i>.</li> <li>Updated <i>Section 6.1.3: Avalon-ST 512-bit RX Interface</i> and <i>Section 6.1.5: Avalon-ST 512-bit TX Interface</i> to note that these interfaces are not strictly Avalon-compliant because they support 2 SOPs and 2 EOPs.</li> <li>Added the requirement for a free-running refclk to meet the PCIe 100 ms wake-up time specification to <i>Section 6.1.8: Clocks</i>.</li> <li>Updated <i>Section 6.1.15: Configuration Extension Bus Interface</i> to state that this interface is not available when SR-IOV is enabled.</li> </ul> |
| December 2017 | 17.1    | <p>Made the following changes to the user guide:</p> <ul style="list-style-type: none"> <li>Added <i>H-Tile Multiplexed Configuration Register Information Available on tl_cfg_ctl</i> table.</li> <li>Corrected command sequences for the PCIe Link Inspector in the <i>Launching the Link Inspector</i> and <i>Displaying PLL Lock and Calibration Status Registers</i> topics. All the .tcl scripts are in the TCL directory which must be included in the source command.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| November 2017 | 17.1    | Removed <b>Enable RX-polarity inversion in soft logic</b> parameter. This parameter is not required for Intel Stratix 10 devices.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| continued...  |         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |



| Date          | Version                                                 | Changes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------|---------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| November 2017 | 17.1                                                    | <p>Added descriptions for the following new features of the Intel Stratix 10 Avalon-ST and SR-IOV Interfaces for PCI Express IP core :</p> <ul style="list-style-type: none"><li>• SR-IOV support, including the following:<ul style="list-style-type: none"><li>— SR-IOV Virtualization Extended Capabilities Registers</li><li>— Virtual Function Registers</li><li>— Control Shadow Interface for SR-IOV</li><li>— Intel Stratix 10 SR-IOV System Settings Parameters</li></ul></li><li>• Function-Level Reset (FLR) support</li><li>• Configuration Status Interface</li><li>• Interrupt signals to support multiple functions</li><li>• Physical Function TLP Processing Hints (TPH) support</li><li>• Address Translation Services (ATS) support</li><li>• A 512-bit interface to the Application Layer for Gen3 x16 configurations.</li><li>• PCIe Link Inspector support to monitor the PCIe link at the Physical, Data Link and Transaction Layers.</li><li>• The following topics to describe the 512-bit interface:<ul style="list-style-type: none"><li>— <i>Avalon-ST 512-Bit RX Interface</i></li><li>— <i>Avalon-ST 512-Bit TX Interface</i></li><li>— <i>Transmitting PCIe TLPs Using the 512-Bit Interface</i></li></ul></li><li>• Bit encoding for rx_st_bar_range for the Expansion ROM which is supported in this release.</li><li>• Compilation support for dynamically-generated design example.</li><li>• Linux driver to run the dynamically-generated design example.</li></ul> <p>Made the following changes to the user guide:</p> <ul style="list-style-type: none"><li>• Revised <i>Generating the Avalon-ST Design</i> to generate the example design from the .ip file.</li><li>• Removed the tx_st_empty signal from <i>Avalon-ST TX Interface Cycle Definition for Three-Dword Header TLPs</i> and <i>Avalon-ST TX Interface Cycle Definition for Four-Dword Header TLPs</i>. The tx_st_empty signal is not available for the 256-bit interface.</li><li>• Removed <i>Chaining DMA Design Examples</i> from the <i>Testbench and Design Example</i> chapter. This design example is not supported for Intel Stratix 10 devices.</li><li>• Updated for latest Intel branding conventions.</li><li>• Revised <i>Avalon-ST Stratix 10 Hard IP for PCI Express Top-Level Signals</i> to show signals that are only available in for L-Tile or H-Tile devices.</li><li>• Added fields to the <i>Multiplexed Configuration Register Information Available on tl_cfg_ctl</i> for L-Tile devices.</li><li>• Added separate <i>Multiplexed Configuration Register Information Available on tl_cfg_ctl</i> table for H-Tile devices.</li><li>• Removed description of testin_zero. This signal is not a top-level signal of the IP.</li><li>• Rebranded as Intel</li><li>• Corrected minor errors and typos.</li></ul> |
| May 2017      | Quartus Prime Pro v17.1 Stratix 10 ES Editions Software | <p>Made the following changes to the IP core:</p> <ul style="list-style-type: none"><li>• Added support for the H-Tile transceiver including example designs available in the installation directory.</li><li>• Added support for a Gen3x16 simulation testbench model that you can use in an Avery testbench.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| continued...  |                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

### C. Document Revision History

UG-20032 | 2018.09.04



| Date         | Version                                     | Changes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |                                             | <p>Made the following changes to the user guide:</p> <ul style="list-style-type: none"><li>• Added definitions for <b>Advance</b>, <b>Preliminary</b>, and <b>Final</b> timing models.</li><li>• Added <i>Testbench and Design Example for the Avalon-ST Interface</i> chapter.</li><li>• Added figures showing the connections between the Avalon-ST and user application and between Stratix 10 Hard IP for PCI Express IP Core, system interfaces, and the user application.</li><li>• Revised <i>Generation</i> discussion to match the The Quartus Prime Pro v17.1 Stratix 10 ES Editions Software design flow.</li><li>• Changed TX credit interface pseudo code.<br/>tx_nph_credit_consume_count_hip_delayed is 3 pld_clk cycle delayed, not 2.</li></ul> |
| October 2016 | Quartus Prime Pro – Stratix 10 Edition Beta | Initial release                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |