Intel® Arria® 10 Avalon® Streaming with SR-IOV IP for PCIe* User Guide

Updated for Intel® Quartus® Prime Design Suite: **20.4**

Online Version
Send Feedback

UG-01161

ID: 683686
Version: 2022.01.11

Contents

1. Datasheet

1.1. Intel® Arria® 10 Avalon®-ST Interface with SR-IOV for PCI Express* Datasheet

Intel® Arria® 10 FPGAs include a configurable, hardened protocol stack for PCI Express* that is compliant with *PCI Express Base Specification 2.1 or 3.0*. The Intel Arria 10 Hard IP for PCI Express with Single Root I/O Virtualization (SR-IOV) IP core consists of this hardened protocol stack and the SR-IOV soft logic. The SR-IOV soft logic uses the Configuration Space Bypass mode of the Hard IP to bypass the internal configuration block and BAR matching logic. These functions are implemented in external soft logic. Soft logic in the SR-IOV Bridge also implements interrupts and error reporting.

The SR-IOV Bridge was redesigned to support up to 8 Physical Functions (PFs) and 2048 Virtual Functions (VFs). The SR-IOV bridge also supports the Address Translation Services (ATS) and TLP Processing Hints (TPH) capabilities.

Figure 1. Intel Arria 10 PCIe Variant with SR-IOV

The following figure shows the high-level modules and connecting interfaces for this variant.

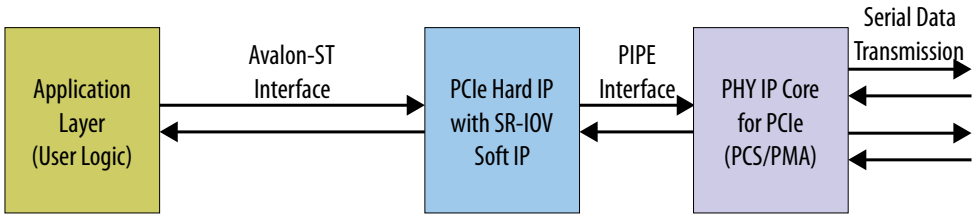
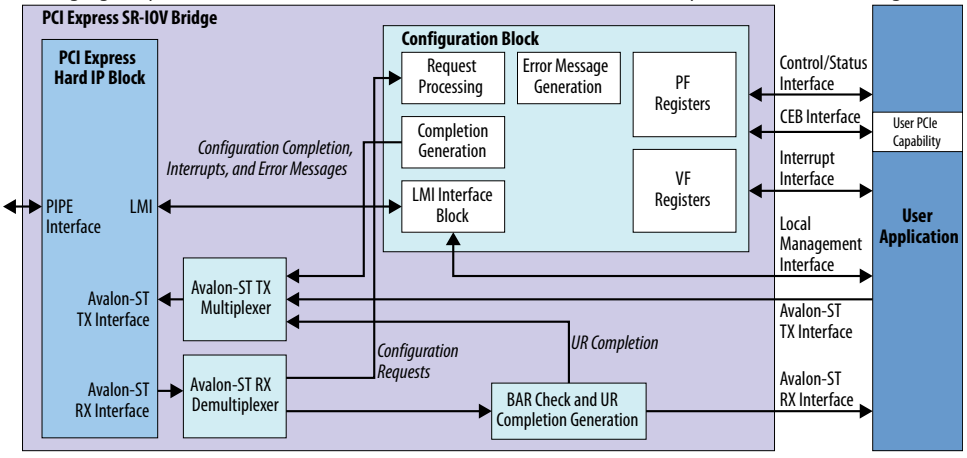


Figure 2. Intel Arria 10 PCIe Variant with SR-IOV

The following figure provides the next level of detail for the modules that comprise the SR-IOV Bridge.



Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Table 2. PCI Express Data Throughput

The following table shows the aggregate bandwidth of a PCI Express link for Gen2 and Gen3 for supported link widths. The protocol specifies 2.5 giga-transfers per second for Gen1, 5.0 giga-transfers per second for Gen2, and 8.0 giga-transfers per second for Gen3. This table provides bandwidths for a single transmit (TX) or receive (RX) channel. The numbers double for duplex operation. Gen1 and Gen2 use 8B/10B encoding which introduces a 20% overhead. In contrast, Gen3 uses 128b/130b encoding which reduces the data throughput lost to encoding to about 1.5%.

	Link Width	
	×4	×8
PCI Express Gen2 (5.0 Gbps) - 256-bit interface	N/A	32
PCI Express Gen3 (8.0 Gbps) - 256-bit interface	31.51	63

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.
- Creating Version-Independent IP and Platform Designer Simulation Scripts
  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.
- Intel Arria 10 Avalon-ST with SR-IOV Interface for PCIe Solutions User Guide Archive on page 138
- Avalon Interface Specifications
  For information about the Avalon-ST interface protocol.
- Arria 10 Avalon-ST Interface for PCIe Solutions User Guide
  For the Avalon-ST interface to the application without SR-IOV.
- PCI Express Base Specification 3.0

1.1.1. SR-IOV Features

New features in the Intel Quartus Prime 17.1 release:

- Added parameter to invert TX polarity.

The Intel Arria 10 Hard IP for PCI Express with SR-IOV supports the following features:

- Support for ×4, and ×8 configurations with Gen2 or Gen3 lane rates for Endpoints
- Configuration Spaces for up to eight PCIe Physical Functions (PFs) and a maximum of 2048 Virtual Functions (VFs) for the PFs
- Base address register (BAR) checking logic
- Dedicated 16 kilobyte (KB) receive buffer
- Platform Designer example designs demonstrating parameterization, design modules, and connectivity
- Extended credit allocation settings to better optimize the RX buffer space based on application type
- Support for Advanced Error Reporting (AER) for PFs

- Support for Address Translation Services (ATS) and TLP Processing Hints (TPH) capabilities
- Support for a Control Shadow Interface to read the current settings for some of the VF Control Register fields in the PCI and PCI Express Configuration Spaces
- Support for Configuration Space Bypass Mode, allowing you to design a custom Configuration Space and support multiple functions
- Support for Function Level Reset (FLR) for PFs and VFs
- Support for Gen3 PIPE simulation
- Support for the following interrupt types:
  - Message signaled interrupts (MSI) for PFs
  - MSI-X for PFs and VFs
  - Legacy interrupts for PFs
- Easy to use:
  - Flexible configuration.
  - Example designs to get started.

The *Intel Arria 10 Avalon-ST Interface with SR-IOV PCIe Solutions User Guide* explains how to use this IP core and not the PCI Express protocol. Although there is inevitable overlap between these two purposes, use this document only in conjunction with an understanding of the *PCI Express Base Specification*.

Note: This release provides separate user guides for the different variants.

Related Information

- Arria 10 Avalon-MM DMA Interface for PCIe Solutions User Guide
  For the Avalon-MM interface and DMA functionality.
- Arria 10 Avalon-MM Interface for PCIe Solutions User Guide
  For the Avalon-MM interface with no DMA.
- Arria 10 Avalon-ST Interface for PCIe Solutions User Guide
  For the Avalon-ST interface.

1.2. Release Information

Table 3. Hard IP for PCI Express Release Information

Item	Description
Version	17.1
Release Date	November 2017
Ordering Codes	Primary: IP-PCIE/SRIOV Renewal: IPR-PCIE/SRIOV
Product IDs	00FB
Vendor ID	6AF7

Intel verifies that the current version of the Quartus Prime 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 Quartus Prime IP Update tool. Intel does not verify compilation with IP core versions older than the previous release.

Related Information

- Intel FPGA IP Release Notes
  Provides release notes for the current and past versions Intel FPGA IP cores.
- Errata for the Intel Arria 10 Hard IP for PCI Express IP Core in the Knowledge Base

1.3. 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 4. Device Family Support

Device Family	Support Level
Intel Arria 10	Final.
Other device families	Refer to the *Intel's PCI Express IP Solutions* web page for support information on other device families.

Related Information

PCI Express Solutions Web Page

1.4. Debug Features

Debug features allow observation and control of the Hard IP for faster debugging of system-level problems.

Related Information

Debugging on page 126

1.5. IP Core Verification

To ensure compliance with the PCI Express specification, Intel performs extensive verification. The simulation environment uses multiple testbenches that consist of industry-standard bus functional models (BFMs) driving the PCI Express link interface. Intel performs the following tests in the simulation environment:

- Directed and pseudorandom stimuli test the Application Layer interface, Configuration Space, and all types and sizes of TLPs
- Error injection tests inject errors in the link, TLPs, and Data Link Layer Packets (DLLPs), and check for the proper responses
- PCI-SIG® Compliance Checklist tests that specifically test the items in the checklist
- Random tests that test a wide range of traffic patterns

Intel provides example designs that you can leverage to test your PCBs and complete compliance base board testing (CBB testing) at PCI-SIG, upon request.

1.5.1. Compatibility Testing Environment

Intel has performed significant hardware testing to ensure a reliable solution. In addition, Intel internally tests every release with motherboards and PCI Express switches from a variety of manufacturers. All PCI-SIG compliance tests are run with each IP core release.

1.6. Performance and Resource Utilization

Because the PCIe protocol stack is implemented in hardened logic, it uses no core device resources (no ALMs and no embedded memory).

The SR-IOV Bridge is implemented is soft logic, requiring FPGA fabric resources. The following table shows the typical device resource utilization for selected configurations using the current version of the Quartus Prime software. With the exception of M20K memory blocks, the numbers of ALMs and logic registers are rounded up to the nearest 50.

Table 5. Performance and Resource Utilization Intel Arria 10 Avalon-ST with SR-IOV

Number of PFs and VFs	ALMs	M20K Memory Blocks	Logic Registers
1 PF, 4 VFs	2350	0	5200
2 PFs, 4 VFs	3600	0	6500
4 PFs, 4 VFs	4650	0	7700
1 PF, 2048 VFs	10350	0	5700
2 PFs, 2048 VFs	11750	0	7500
4 PFs 2048 VFs	14150	0	10650
2 PFs	2300	0	5100
4 PFs	3450	0	6300

Related Information

Running the Fitter
For information on Fitter constraints.

1.7. Recommended Speed Grades for SR-IOV Interface

Table 6. Intel Arria 10 Recommended Speed Grades for All SR-IOV Configurations

Intel recommends setting the Quartus Prime Analysis & Synthesis Settings **Optimization Technique** to **Speed** when the Application Layer clock frequency is 250 MHz. For information about optimizing synthesis, refer to "*Setting Up and Running Analysis and Synthesis* in Quartus Prime Help. For more information about how to effect the **Optimization Technique** settings, refer to *Area and Timing Optimization* in volume 2 of the *Quartus Prime Handbook*. Refer to the *Related Links* below.

Link Rate	Link Width	Interface Width	Application Clock Frequency (MHz)	Recommended Speed Grades
Gen2	×8	256 bits	125	–1, –2, –3
Gen3	×4	256 bits	125	–1, –2, –3
	×8	256 bits	250	–1, –2

Related Information

- Running Synthesis
  For settings that affect timing closure.
- Intel FPGA Software Installation and Licensing Manual
  For comprehensive information for installing and licensing Intel FPGA software.

2. Getting Started with the SR-IOV Design Example

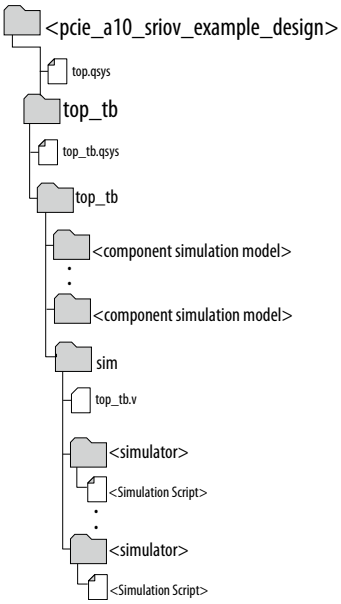
The SR-IOV example design consists of a PCIe Endpoint that includes an SR-IOV bridge configured for one PF and four VFs. The example design also includes a basic application to facilitate host accesses to a target memory. This design example supports simulation. In simulation, the testbench issues downstream memory accesses to the virtual function BAR. The testbench then reads the data written and compares it to the expected result. The test passes if all the comparisons pass.

When you install the Intel Quartus Prime software you also install the IP Library. This installation includes design examples for Hard IP for PCI Express under the `<install_dir>/ip/altera/altera_pcie/` directory. You can copy the design examples from the `<install_dir>/ip/altera/ altera_pcie/ altera_pcie_a10_ed/example_design/a10` directory. This walkthrough uses the `sriov2_target_g3x8_1pf_4vf.qsys` design example.

Note: Starting in the Quartus Prime 16.0 software release, you cannot simulate or compile SR-IOV designs without a license. Contact your local sales representative or email pcie@altera.com to obtain a license.

2.1. Directory Structure for Intel Arria 10 SR-IOV Design Example

Figure 3. Directory Structure for the Generated Example Design



Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

2.2. Design Components for the SR-IOV Design Example

Figure 4. Platform Designer Testbench for Intel Arria 10 Gen1 x8 128-bit SR-IOV Design Example

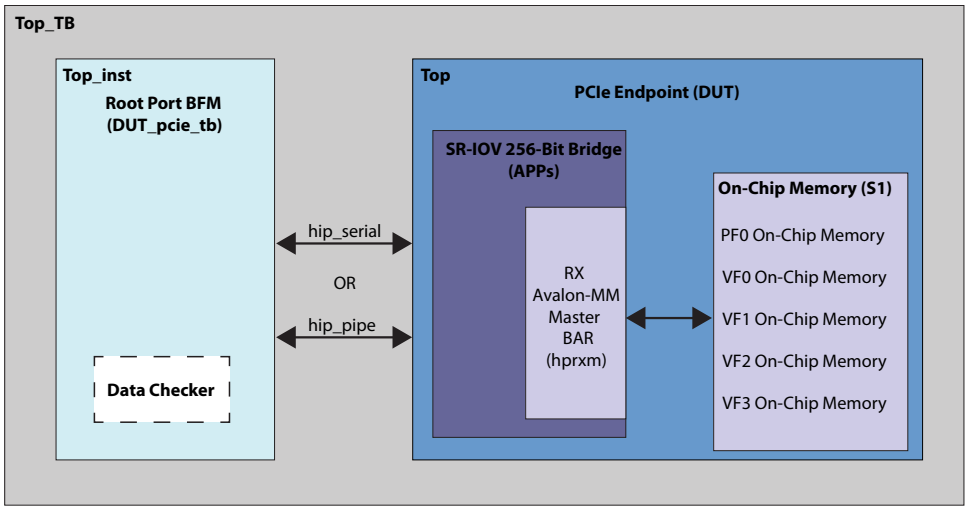
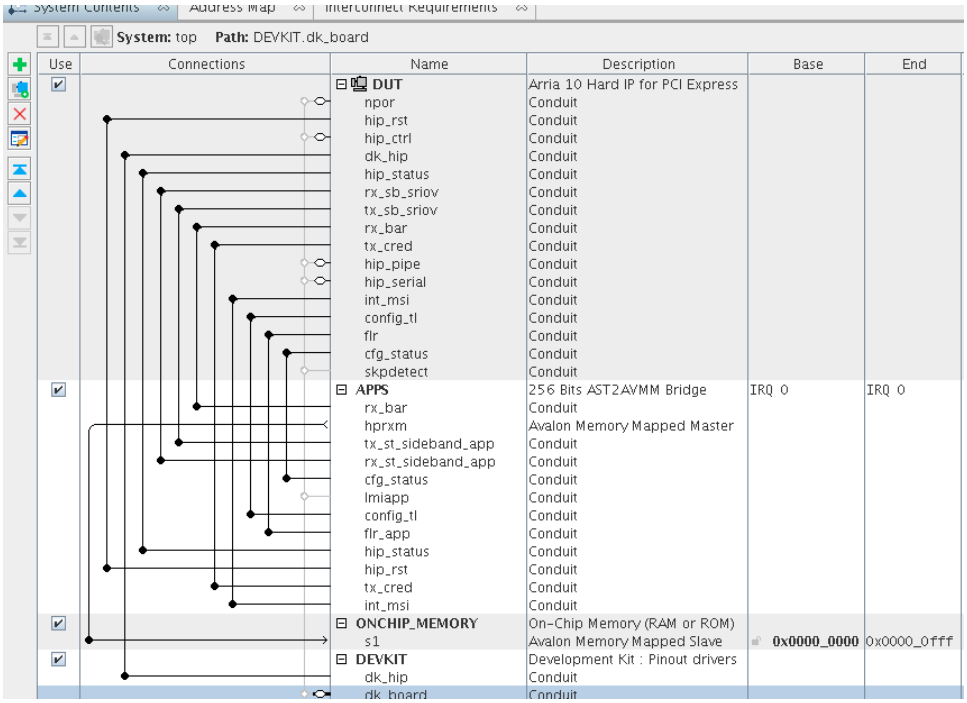


Figure 5. Platform Designer Schematic for Top

This image of the Intel Arria 10 PCI Express DMA Design Example shows only the Avalon-ST, clock, and reset interfaces.



The testbench includes a PCIe Root Port BFM and a PCIe Gen3 x8 Endpoint implemented in hard logic. The SR-IOV bridge, implemented in soft logic, drives memory writes and reads to the four VFs. The simulation includes the following stages:

- Link Training
- Configuration
- Memory writes to each VF
- Memory reads and compares to the expected data

2.3. Generating the SR-IOV Design Example

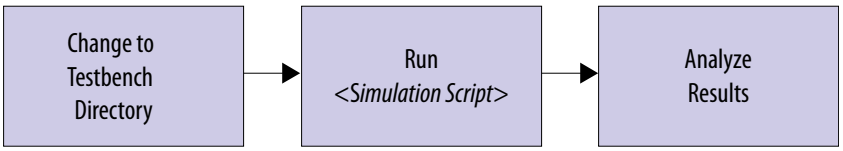
After installing the Quartus Prime software, copy the design examples from the `<install_dir>/ip/altera/ altera_pcie/altera_pcie_a10_ed/example_design/a10` directory. This walkthrough uses the `sriov2_top_target_gen3x8_1pf_4vf.qsys` design example. To run the simulation, you must rename the design example `top.qsys`

1. Launch Platform Designer and open `top.qsys`.
2. On the Generate menu, select **Generate Testbench System**.
3. For **Create testbench Platform Designer system**, select **Standard, BFMs for stand Platform Designer interfaces**.
4. For **Create testbench simulation model**, select either **Verilog** or **VHDL**.
5. For **Output Directory ➤ Testbench**, you can accept the default directory or modify it.
6. Click **Generate**.

   Note: Intel Arria 10 devices do not support the **Create timing and resource estimates for third-party EDA synthesis tools** option on the **Generate ➤ Generate HDL** menu. You can select this menu item, but generation fails.

2.4. Compiling and Simulating the Design for SR-IOV

Figure 6. Procedure



Follow these steps to compile and simulate the design:

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

Table 7. Steps to Run Simulation

Simulator	Working Directory	Instructions
Mentor ModelSim*	*<example_design>*/top_tb/top_tb/sim/mentor/	a. Invoke vsim b. `do msim_setup.tcl` c. `ld_debug` d. `run -all` e. A successful simulation ends with the following message, "Simulation stopped due to successful completion! Simulation passed."
Mentor VCS*	<example_design>/top_tb/top_tb/sim/synopsys/vcs	a. sh vcs_setup.sh USER_DEFINED_SIM_OPTIONS="" b. A successful simulation ends with the following message, "Simulation stopped due to successful completion! Simulation passed."
Cadence NCSim*	*<example_design>*top_tb/top_tb/sim/cadence	a. Create a shell script, my_setup.sh. This script allows you to add additional commands and override the defaults included in `ncsim_setup.sh`. b. Include the following command in `my_setup.sh`: `source ncsim_setup.sh USER_DEFINED_SIM_OPTIONS=""` c. `chmod +x *.sh` d. `./my_setup.sh` e. A successful simulation ends with the following message, "Simulation stopped due to successful completion! Simulation passed."

3. Parameter Settings

3.1. Parameters

This chapter provides a reference for all the parameters of the IP core.

Table 8. 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
Design Environment	**Standalone** **System**	Identifies the environment that the IP is in. • The **Standalone** environment refers to the IP being in a standalone state where all its interfaces are exported. • The **System** environment refers to the IP being instantiated in a Platform Designer system.

Table 9. System Settings

Parameter	Value	Description
Application Interface Type	**Avalon-ST** **Avalon-MM** **Avalon-MM with DMA** **Avalon-ST with SR-IOV**	Selects the interface to the Application Layer. *Note:* When the **Design Environment** parameter is set to **System**, all four **Application Interface Types** are available. However, when **Design Environment** is set to **Standalone**, only **Avalon-ST** and **Avalon-ST with SR-IOV** are available.
Hard IP mode	**Gen3x8, Interface: 256-bit, 250 MHz** **Gen3x4, Interface: 256-bit, 125 MHz** **Gen3x4, Interface: 128-bit, 250 MHz** **Gen3x2, Interface: 128-bit, 125 MHz** **Gen3x2, Interface: 64-bit, 250 MHz** **Gen3x1, Interface: 64-bit, 125 MHz** **Gen2x8, Interface: 256-bit, 125 MHz** **Gen2x8, Interface: 128-bit, 250 MHz** **Gen2x4, Interface: 128-bit, 125 MHz** **Gen2x2, Interface: 64-bit, 125 MHz** **Gen2x4, Interface: 64-bit, 250 MHz** **Gen2x1, Interface: 64-bit, 125 MHz** **Gen1x8, Interface: 128-bit, 125 MHz** **Gen1x8, Interface: 64-bit, 250 MHz** **Gen1x4, Interface: 64-bit, 125 MHz** **Gen1x2, Interface: 64-bit, 125 MHz** **Gen1x1, Interface: 64-bit, 125 MHz** **Gen1x1, Interface: 64-bit, 62.5 MHz**	Selects the following elements: • The lane data rate. Gen1, Gen2, and Gen3 are supported • The width of the data interface between the hard IP Transaction Layer and the Application Layer implemented in the FPGA fabric • The Application Layer interface frequency Intel Cyclone® 10 GX devices support up to Gen2 x4 configurations.
Port type	**Native Endpoint** **Root Port**	Specifies the port type.

continued...

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Parameter	Value	Description
		The Endpoint stores parameters in the Type 0 Configuration Space. The Root Port stores parameters in the Type 1 Configuration Space. The **Avalon-ST with SR-IOV** interface supports only **Native Endpoint** operation. You can enable the Root Port in the current release. Root Port mode only supports the Avalon®-MM interface type, and it only supports basic simulation and compilation. However, the Root Port mode is not fully verified.
RX Buffer credit allocation - performance for received requests	**Minimum** **Low** **Balanced**	Determines the allocation of posted header credits, posted data credits, non-posted header credits, completion header credits, and completion data credits in the 16 KB RX buffer. The settings allow you to adjust the credit allocation to optimize your system. The credit allocation for the selected setting displays in the **Message** pane. The **Message** pane dynamically updates the number of credits for Posted, Non-Posted Headers and Data, and Completion Headers and Data as you change this selection. Refer to the *Throughput Optimization* chapter for more information about optimizing your design. Refer to the *RX Buffer Allocation Selections Available by Interface Type* below for the availability of these settings by interface type. **Minimum**—configures the minimum PCIe specification allowed for non-posted and posted request credits, leaving most of the RX Buffer space for received completion header and data. Select this option for variations where application logic generates many read requests and only infrequently receives single requests from the PCIe link. **Low**—configures a slightly larger amount of RX Buffer space for non-posted and posted request credits, but still dedicates most of the space for received completion header and data. Select this option for variations where application logic generates many read requests and infrequently receives small bursts of requests from the PCIe link. This option is recommended for typical endpoint applications where most of the PCIe traffic is generated by a DMA engine that is located in the endpoint application layer logic. **Balanced**—configures approximately half the RX Buffer space to received requests and the other half of the RX Buffer space to received completions. Select this option for variations where the received requests and received completions are roughly equal.
RX Buffer completion credits	**Header credits** **Data credits**	Displays the number of completion credits in the 16 KB RX buffer resulting from the credit allocation parameter. Each header credit is 16 bytes. Each data credit is 20 bytes.

3.2. Intel Arria 10 Avalon-ST Settings

Table 10. System Settings for PCI Express

Parameter	Value	Description
Enable Avalon-ST reset output port	**On/Off**	When **On**, the generated reset output port has the same functionality that the `reset_status` port included in the Reset and Link Status interface.
Enable byte parity ports on Avalon-ST interface	**On/Off**	When **On**, 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. This parameter is only available for the Avalon-ST Intel Arria 10 Hard IP for PCI Express.
Enable multiple packets per cycle for the 256-bit interface	**On/Off**	When **On**, the 256-bit Avalon-ST interface supports the transmission of TLPs starting at any 128-bit address boundary, allowing support for multiple packets in a single cycle. To support multiple packets per cycle, the Avalon-ST interface includes 2 start of packet and end of packet signals for the 256-bit Avalon-ST interfaces. This is not supported for the Avalon-ST with SR-IOV interface.
Enable credit consumed selection port	**On/Off**	When you turn on this option, the core includes the `tx_cons_cred_sel` port. This parameter does not apply to the Avalon-MM interface.
Enable Configuration bypass (CfgBP)	**On/Off**	When **On**, the Intel Arria 10 Hard IP for PCI Express bypasses the Transaction Layer Configuration Space registers included as part of the Hard IP, allowing you to substitute a custom Configuration Space implemented in soft logic. This parameter is not available for the Avalon-MM IP Cores.
Enable local management interface (LMI)	**On/Off**	When **On**, your variant includes the optional LMI interface. This interface is used to log error descriptor information in the TLP header log registers. The LMI interface provides the same access to Configuration Space registers as Configuration TLP requests.

Related Information

PCI Express Base Specification 3.0

3.3. Intel Arria 10 SR-IOV System Settings

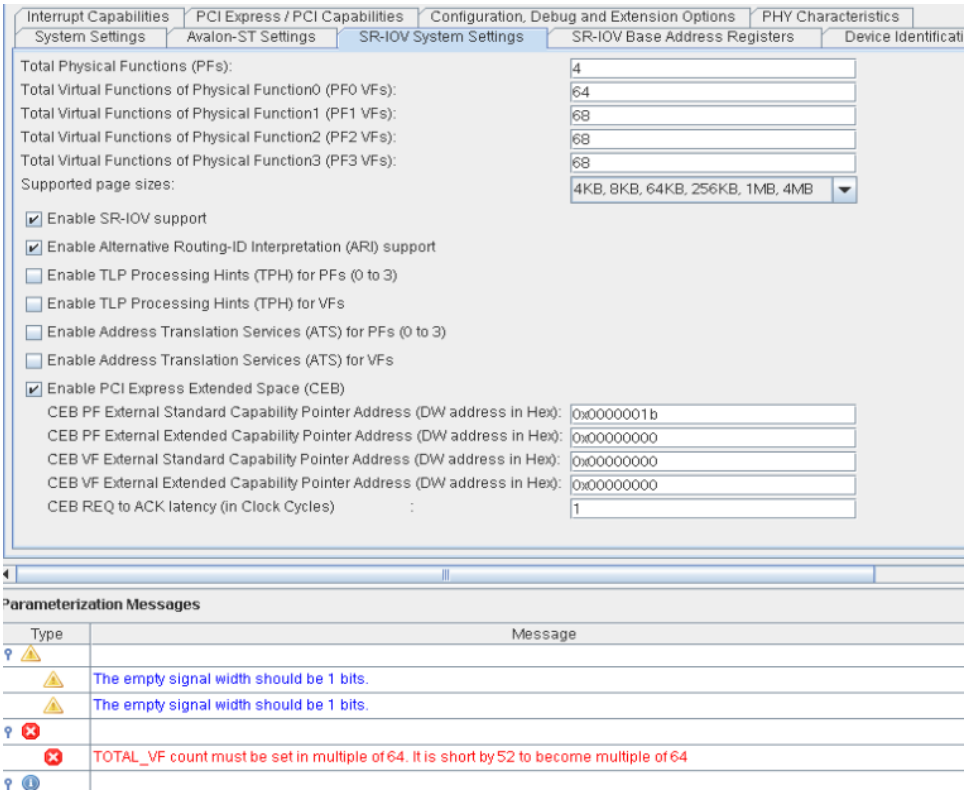
Parameter	Value	Description
Total Physical Functions (PFs) :	`1 - 8`	This core supports 1 - 8 Physical Functions.
Total Virtual Functions of Physical Function0 (PF0 VFs) - Total Virtual Functions of Physical Function7 (PF7 VFs):	`0 - 2048`	Total number of VFs assigned to a PF. You can assign VFs in the following granularities: • Granularity of 1 for 1-8 VFs • Granularity of 4 for 8-256 VFs • Granularity of 64 for 256-1024 VFs • Granularity of 512 for 1024-2048 VFs The sum of VFs assigned to all PFs cannot exceed the 2048 VF total. *Note:* The granularity restriction for assigning VFs applies to both the total VFs of each individual PF as well as the sum of all VFs across all enabled PFs. See the example and snapshot following this table for more details.
System Supported Page Size:	**4KB - 4MB**	Specifies the pages sizes supported. Sets the `Supported Page Sizes` register of the SR-IOV Capability structure.
Enable SR-IOV Support	**On/Off**	When **On**, the variant supports multiple PFs and VFs. When **Off**, .supports PFs only.

continued...

Parameter	Value	Description
Enable Alternative Routing-ID (ARI) support	**On/Off**	When **On**, ARI supports up to 256 functions. Refer to *Section 6.1.3 Alternative Routing-ID Interpretation (ARI)* of the *PCI Express Base Specification* for more information about ARI.
Enable Functional Level Reset (FLR)	**On/Off**	When **On**, each function has its own, individual reset.
Enable TLP Processing Hints (TPH) support for PFs	**On/Off**	When **On**, the variant includes the TPH registers to help you improve latency and traffic congestion.
Enable TLP Processing Hints (TPH) support for VFs		
Enable Address Translation Services (ATS) support for PFs	**On/Off**	When **On**, the variant includes the ATS registers.
Enable Address Translation Services (ATS) support for VFs		
Enable PCI Express Extended Space (CEB)	**On/Off**	When On, the IP core variant includes the optional Configuration Extension Bus (CEB) interface. This interface provides a way to add extra capabilities on top of those available in the internal configuration space of the SR-IOV Bridge.(1)
CEB PF External Standard Capability Pointer Address (DW Address in Hex)	0x0	Specifies the address for the next pointer field of the last capability structure within the SR-IOV bridge for the physical function. It allows the internal PCI Compatible region capability to point to the capability implemented in user logic and establish the link list for the first 256 bytes in the register address space.
CEB PF External Extended Capability Pointer Address (DW Address in Hex)	0x0	Specifies the address for the next pointer field of the last capability structure within the SR-IOV bridge for the physical function. It allows the internal PCI Compatible region capability to point to the capability implemented in user logic and establish the link list for the PCIe extended configuration space. Supports the address range from 0x100 (DW address) or 0x400 (byte address) and beyond.
CEB VF External Standard Capability Pointer Address (DW Address in Hex)	0x0	Specifies the address for the next pointer field of the last capability structure within the SR-IOV bridge for the virtual function. It allows the internal PCI Compatible region capability to point to the capability implemented in user logic and establish the link list for the first 256 bytes in the register address space.
CEB VF External Extended Capability Pointer Address (DW Address in Hex)	0x0	Specifies the address for the next pointer field of the last capability structure within the SR-IOV bridge for the virtual function. It allows the internal PCI Compatible region capability to point to the capability implemented in user logic and establish the link list for the PCIe extended configuration space. Supports the address range from 0x100 (DW address) or 0x400 (byte address) and beyond.
CEB REQ to ACK Latency (in Clock Cycles)	1 - 7	Specifies the timeout value for the request issued on the CEB interface. The SR-IOV bridge will send a completion with all zeros in the data and completion status field back to the host after the timeout. *Note:* A large ACK latency time may result in a bandwidth degradation.

(1) To implement user capabilities registers, Intel recommends that you use an open address space in the configuration space across all available Physical Functions (PFs) and Virtual Functions (VFs). Developing user capabilities registers for individual PFs or VFs is not possible.

The granularity restriction for assigning VFs applies to both the total VFs of each individual PF as well as the sum of all VFs across all enabled PFs. For example, the setting in the snapshot below is invalid since the sum of all VFs assigned to all PFs (i.e, 268 VFs) is not a multiple of 64 and therefore does not meed the granularity restriction even though the VF counts of individual PFs are valid (i.e, they are multiples of 4).



Related Information

PCI Express Base Specification 2.1 or 3.0

3.4. Base Address Register (BAR) Settings

Each function can implement up to six BARs. You can configure up to six 32-bit BARs or three 64-bit BARs for both PFs and VFs. The BAR settings are the same for all VFs associated with a PF.

Table 11. BAR Registers

Parameter	Value	Description
Present (BAR0-BAR5)	**Enabled/Disabled**	Indicates whether or not this BAR is instantiated.
Type	**32-bit address**	Specifies 32- or 64-bit addressing.

continued...

Parameter	Value	Description
	64-bit address	
Prefetchable	**Prefetchable** **Non-prefetchable**	Defining memory as **Prefetchable** allows data in the region to be fetched ahead anticipating that 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: • Reads do not have side effects • Write merging is allowed If you select **64-bit address**, 2 contiguous BARs are combined to form a 64-bit BAR. You must set the higher numbered BAR to **Disabled**. If the BAR TYPE of any even BAR is set to 64-bit memory, the next higher BAR supplies the upper address bits. The supported combinations for 64-bit BARs are {BAR1, BAR0}, {BAR3, BAR2}, {BAR4, BAR5}.
Size	**16 Bytes–2 GB**	Specifies the memory size.

3.5. SR-IOV Device Identification Registers

Table 12. Device ID Registers

The following table lists the default values of the read-only Device ID registers. You can use the parameter editor to change the values of these registers. At run time, you can change the values of these registers using the optional reconfiguration block signals. You can specify Device ID registers for each Physical Function.

Register Name	Default Value	Description
Vendor ID	0x00001172	Sets the read-only value of the `Vendor ID` register. This parameter can not be set to 0xFFFF per the PCI Express Specification. 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 VF Device ID register.
Revision ID	0x00000000	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. Address offset: 0x008.
Subclass code	0x00000000	Sets the read-only value of the `Subclass Code` register. Address offset: 0x008.
Subsystem Vendor ID	0x00000000	Sets the read-only value of the register in the PCI Type 0 Configuration Space. This parameter cannot be set to 0xFFFF per the *PCI Express Base Specification*. This value is assigned by PCI-SIG to the device manufacturer. Address offset: 0x02C.
Subsystem Device ID	0x00000000	Sets the read-only value of the `Subsystem Device ID` register in the PCI Type 0 Configuration Space. Address offset: 0x02C

Related Information

PCI Express Base Specification 2.1 or 3.0

3.6. Intel Arria 10 Interrupt Capabilities

Table 13. MSI and MSI-X Interrupt Settings

Each Physical Function defines its own MSI-X table settings. The VF MSI-X table settings are the same for all the Virtual Functions associated with each Physical Function.

Parameter	Value	Description
MSI Interrupt Settings		
PF0 MSI Requests - PF3 MSI Requests	**1,2,4,8,16,32**	Specifies the maximum number of MSI messages the Application Layer can request. This value is reflected in Multiple Message Capable field of the `Message Control` register, 0x050[31:16]. For MSI Interrupt Settings, if the **PF MSI** option is enabled, all PFs support MSI capability.
MSI-X PF0 - MSI-X PF3 Interrupt Settings		
PF MSI-X	**On/Off**	When **On**, enables the MSI-X functionality. For PF and VF MSI-X Interrupt Settings, if **PF MSI-X** is enabled, all PFs supports MSI-X capability.
VF MSI-X	**On/Off**	
	Bit Range	
MSI-X Table size	[10:0]	System software reads this field to determine the MSI-X Table size <*n*>, which is encoded as <*n–1*>. For example, a returned value of 2047 indicates a table size of 2048. This field is read-only. Legal range is 0–2047 (2^{11}). Address offset: 0x068[26:16]
MSI-X Table Offset	[31:0]	Specifies the offset from the BAR indicated in the**MSI-X Table BAR Indicator**. The lower 3 bits of the table BAR indicator (BIR) are set to zero by software to form a 32-bit qword-aligned offset [(2)]. This field is read-only.
MSI-X Table BAR Indicator	[2:0]	Specifies which one of a function's BAR number. This field is read-only. For 32-bit BARs, the legal range is 0–5. For 64-bit BARs, the legal range is 0, 2, or 4.
MSI-X Pending Bit Array (PBA) Offset	[31:0]	Points to the MSI-X Pending Bit Array table. It is offset from the BAR value indicated in **MSI-X Table BAR Indicator**. 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.
MSI-X PBA BAR Indicator	[2:0]	Specifies which BAR number contains the MSI-X PBA. For 32-bit BARs, the legal range is 0–5. For 64-bit BARs, the legal range is 0, 2, or 4. This field is read-only.
Legacy Interrupts		
PF0 - PF3 Interrupt Pin	**inta–intd**	Applicable for PFs only to support legacy interrupts. When enabled, the core receives interrupt indications from the Application Layer on its `INTA_IN`, `INTB_IN`, `INTC_IN` and `INTD_IN` inputs, and sends out `Assert_INTx` or `Deassert_INTx` messages on the link in response to their activation or deactivation, respectively. You can configure the Physical Functions with separate interrupt pins. Or, both functions can share a common interrupt pin.
PF0 - PF3 Interrupt Line	**0-255**	Defines the input to the interrupt controller (IRQ0 - IRQ15) in the Root Port that is activated by each `Assert_INTx` message.

(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.

Related Information

PCI Express Base Specification Revision 2.1 or 3.0

3.7. Physical Function TLP Processing Hints (TPH)

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 TPH Requestor Capability structure or combine it with the MSI-X Table. 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.

Table 14. TPH Capabilities

The values specified here are common for all PFs.

Parameter	Value	Description
Interrupt Mode	**On/Off**	When **On**, the Steering Tag is selected by an MSI/MSI-X interrupt vector number.
Device Specific Mode	**On/Off**	When **On**, the Steering Tag is selected from Steering Tag Table entry stored in the TPH Requestor Capability structure.
Steering Tag Table location	**0, 1, 2**	When non-zero, specifies the location of the Steering Tag Table. The following encodings are defined: • 0: Steering Tag table not present • 1: Steering Tag table is stored in the TPH Requestor Capability structure • 2: Steering Tag table is located in the MSI-X table.
Steering Tag Table size	**0-2047**	Specifies the number of 2-byte Steering Table entries.

Related Information

PCI Express Base Specification Revision 3.0

3.8. Address Translation Services (ATS)

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 15. ATS Capabilities

ATS must maintain cache coherence between the addresses in the TA and ATC. The TA and associated software ensure that the addresses caches in the ATC are not stale by issuing Invalidate Requests. An Invalidate Request clears a specific subset of the address range from the ATC. For more information about ATS, refer to *Address Translation Services Revision 1.1* The values specified here are common for all PFs.

Parameter	Value	Description
PF0 - PF3 Maximum outstanding Invalidate Requests	0-32	Specifies the maximum number outstanding Invalidate Requests for each PF before putting backpressure on the upstream connection.

Related Information

- Address Translation Services Revision 1.1
- PCI Express Base Specification Revision 3.0

3.9. 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.

3.9.1. PCI Express and PCI Capabilities

Table 16. Capabilities Registers

Parameter	Possible Values	Default Value	Description
Maximum payload size	**128 bytes** **256 bytes** **512 bytes** **1024 bytes** **2048 bytes**	128 bytes	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 (0x084[2:0]). Address: 0x084. *Note:* The SR-IOV bridge supports a single value for the **Maximum payload size** parameter. When the configuration includes 2 or more PFs, you must program all 4 PFs for the SR-IOV bridge to specify a value larger than 128 bytes.
Number of Tags supported	**32** **64**	32	Indicates the number of tags supported for non-posted requests transmitted by the Application Layer. This parameter sets the values in the Device Control register (0x088) of the PCI Express capability structure described in Table 9–9 on page 9–5. The Transaction Layer tracks all outstanding completions for non-posted requests made by the Application Layer. This parameter configures the Transaction Layer for the maximum number of **Tags supported** to track. The Application Layer must set the tag values in all non-posted PCI Express headers to be less than this value. Values greater than 32 also set the extended tag field supported bit in the Configuration Space Device Capabilities register. The Application Layer can only use tag numbers greater than 31 if configuration software sets the Extended Tag Field Enable bit of the Device Control register. *Note:* When more than one physical functions are enabled in the IP core, the non-posted tag pool is shared across all of them.
Completion timeout range	**ABCD** **BCD** **ABC** **AB**	ABCD	Indicates device function support for the optional completion timeout programmability mechanism. This mechanism allows system software to modify the completion timeout value. This field is applicable only to Root Ports and Endpoints that issue requests on their own behalf. Completion timeouts are specified

continued...

Parameter	Possible Values	Default Value	Description
	B **A** **None**		and enabled in the Device Control 2 register (0x0A8) of the *PCI Express Capability Structure Version*. For all other functions this field is reserved and must be hardwired to 0x0000b. Four time value ranges are defined: • Range A: 50 us to 10 ms • Range B: 10 ms to 250 ms • Range C: 250 ms to 4 s • Range D: 4 s to 64 s Bits are set to show timeout value ranges supported. The function must implement a timeout value in the range 50 s to 50 ms. The following values specify the range: • None—Completion timeout programming is not supported • 0001 Range A • 0010 Range B • 0011 Ranges A and B • 0110 Ranges B and C • 0111 Ranges A, B, and C • 1110 Ranges B, C and D • 1111 Ranges A, B, C, and D All other values are reserved. Intel recommends that the completion timeout mechanism expire in no less than 10 ms.
Disable completion timeout	**On/Off**	On	Disables the completion timeout mechanism. When **On**, the core supports the completion timeout disable mechanism via the PCI Express `Device Control Register 2`. The Application Layer logic must implement the actual completion timeout mechanism for the required ranges.

3.9.2. Error Reporting

Table 17. Error Reporting

Parameter	Value	Default Value	Description
Enable Advanced Error Reporting (AER)	**On/Off**	Off	When **On**, enables the Advanced Error Reporting (AER) capability.
Enable ECRC checking	**On/Off**	Off	When **On**, enables ECRC checking. Sets the read-only value of the ECRC check capable bit in the `Advanced Error Capabilities and Control Register`. This parameter requires you to enable the AER capability.
Enable ECRC generation	**On/Off**	Off	When **On**, enables ECRC generation capability. Sets the read-only value of the ECRC generation capable bit in the `Advanced Error Capabilities and Control Register`. This parameter requires you to enable the AER capability.
Enable ECRC forwarding on the Avalon-ST interface	**On/Off**	Off	When **On**, enables ECRC forwarding to the Application Layer. On the Avalon-ST RX path, the incoming TLP contains the ECRC dword(1) and the TD bit is set if an ECRC exists. On the transmit the TLP from the Application Layer must contain the ECRC dword and have the TD bit set.
Track RX completion buffer	**On/Off**	Off	When **On**, the core includes the `rxfc_cplbuf_ovf` output status signal to track the RX posted completion buffer overflow status.

continued...

Parameter	Value	Default Value	Description
overflow on the Avalon-ST interface			

Note:
1. 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.

3.9.3. Link Capabilities

Table 18. 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.
Data link layer active reporting (Root Port only)	**On/Off**	Turn **On** this parameter for a Root Port, if the attached Endpoint supports the optional capability of reporting the DL_Active state of the Data Link Control and Management State Machine. For a hot-plug capable Endpoint (as indicated by the `Hot Plug Capable` field of the `Slot Capabilities` register), this parameter must be turned **On**. For Root Port components that do not support this optional capability, turn **Off** this option.
Surprise down reporting (Root Port only)	**On/Off**	When your turn this option **On**, an Endpoint supports the optional capability of detecting and reporting the surprise down error condition. The error condition is read from the Root Port.
Slot clock configuration	**On/Off**	When you turn this option **On**, indicates that the Endpoint uses the same physical reference clock that the system provides on the connector. When **Off**, 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.

3.9.4. Slot Capabilities

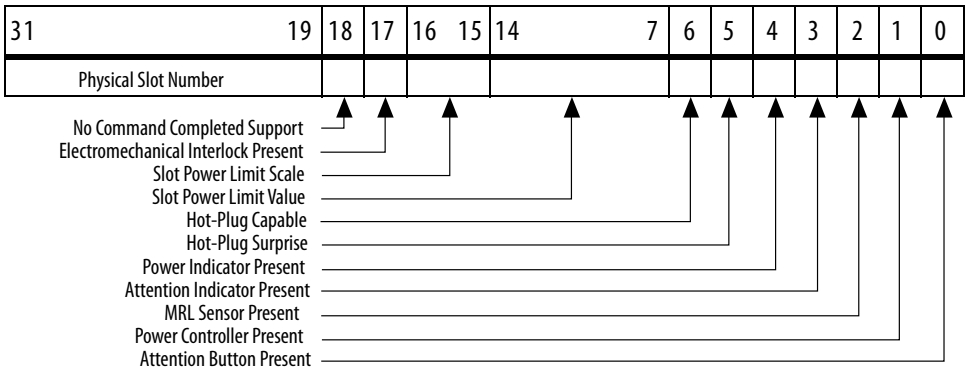
Table 19. Slot Capabilities

Parameter	Value	Description
Use Slot register	**On/Off**	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. Defines the characteristics of the slot. You turn on this option by selecting **Enable slot capability**. Refer to the figure below for bit definitions.
Slot power scale	`0-3`	Specifies the scale used for the **Slot power limit**. The following coefficients are defined: • 0 = 1.0x • 1 = 0.1x • 2 = 0.01x • 3 = 0.001x 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.

continued...

Parameter	Value	Description
		Refer to Section 6.9 of the *PCI Express Base Specification Revision* for more information.
Slot power limit	0–255	In combination with the **Slot power scale value**, specifies the upper limit in watts on power supplied by the slot. Refer to Section 7.8.9 of the *PCI Express Base Specification* for more information.
Slot number	0-8191	Specifies the slot number.

Figure 7. Slot Capability



3.9.5. Power Management

Table 20. Power Management Parameters

Parameter	Value	Description
Endpoint L0s acceptable latency	**Maximum of 64 ns** **Maximum of 128 ns** **Maximum of 256 ns** **Maximum of 512 ns** **Maximum of 1 us** **Maximum of 2 us** **Maximum of 4 us** **No limit**	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). 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. The default value of this parameter is 64 ns. This is a safe setting for most designs.
Endpoint L1 acceptable latency	**Maximum of 1 us** **Maximum of 2 us** **Maximum of 4 us** **Maximum of 8 us** **Maximum of 16 us** **Maximum of 32 us** **Maximum of 64 ns** **No limit**	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`. 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. The default value of this parameter is 1 µs. This is a safe setting for most designs.

These IP cores also do not support the in-band beacon or sideband WAKE# signal, which are mechanisms to signal a wake-up event to the upstream device.

3.10. PHY Characteristics

Table 21. PHY Characteristics

Parameter	Value	Description
Gen2 TX de-emphasis	**3.5dB** **6dB**	Specifies the transmit de-emphasis for Gen2. Intel recommends the following settings: • 3.5dB: Short PCB traces • 6.0dB: Long PCB traces.
Requested equalization far-end TX preset	**Preset0-Preset9**	Specifies the requested TX preset for Phase 2 and 3 far-end transmitter. The default value **Preset8** provides the best signal quality for most designs.
Enable soft DFE controller IP	**On** **Off**	When **On**, the PCIe Hard IP core includes a decision feedback equalization (DFE) soft controller in the FPGA fabric to improve the bit error rate (BER) margin. The default for this option is **Off** because the DFE controller is typically not required. However, short reflective links may benefit from this soft DFE controller IP. This parameter is available only for Gen3 mode. It is not supported when CvP or autonomous modes are enabled.
Enable RX-polarity inversion in soft logic	**On** **Off**	This parameter mitigates the following RX-polarity inversion problem. When the Intel Arria 10 Hard IP core receives TS2 training sequences during the Polling.Config state, when you have not enabled this parameter, automatic lane polarity inversion is not guaranteed. The link may train to a smaller than expected link width or may not train successfully. This problem can affect configurations with any PCIe* speed and width. When you include this parameter, polarity inversion is available for all configurations except Gen1 x1. This fix does not support CvP or autonomous mode.

3.11. Example Designs

The SR-IOV variant does not support the options available on this tab. The *Getting Started with the SR-IOV Design Example* chapter provides an example with simulation and Quartus Prime compilation.

4. Physical Layout

4.1. Hard IP Block Placement In Intel Cyclone 10 GX Devices

Intel Cyclone 10 GX devices include a single hard IP blocks for PCI Express. This hard IP block includes the CvP functionality for flip chip packages.

Figure 8. Intel Cyclone 10 GX Devices with 12 Transceiver Channels and One PCIe Hard IP Block

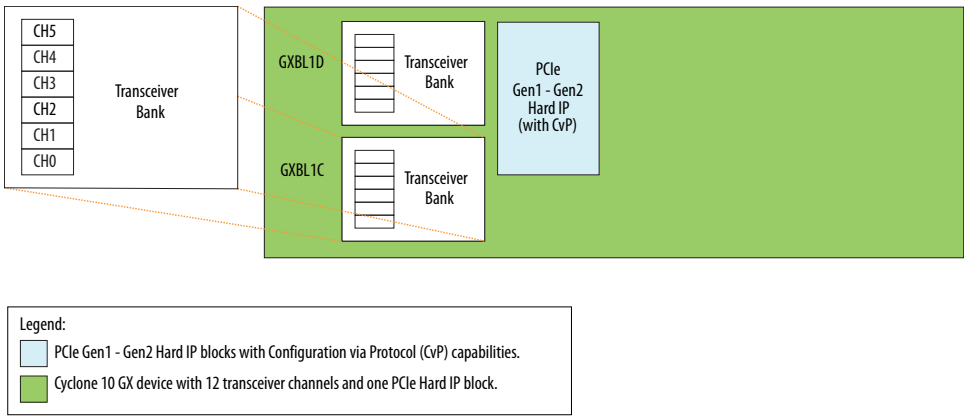
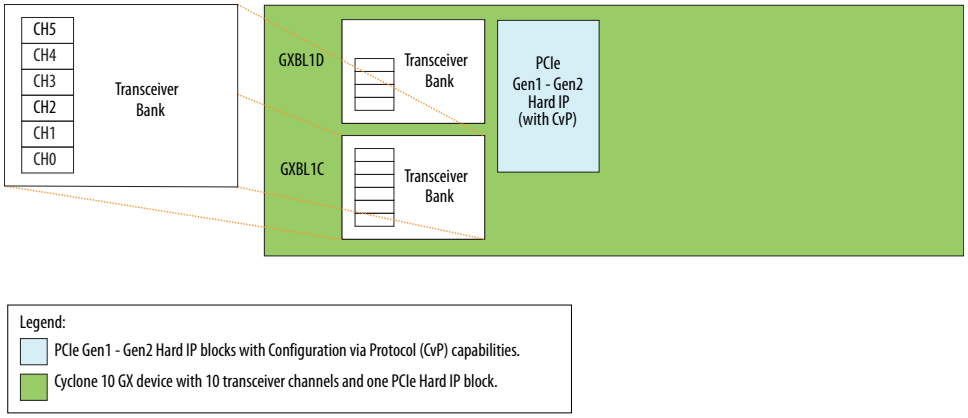


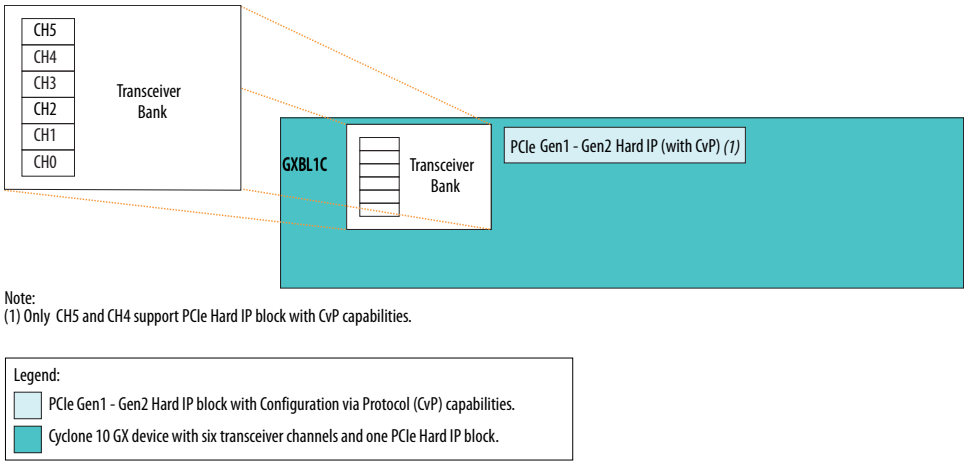
Figure 9. Intel Cyclone 10 GX Devices with 10 Transceiver Channels and One PCIe Hard IP Block



Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Figure 10. Intel Cyclone 10 GX Devices with 6 Transceiver Channels and One PCIe Hard IP Block



Refer to the *Intel Cyclone 10 GX Device Transceiver Layout* in the *Intel Cyclone 10 GX Transceiver PHY User Guide* for comprehensive figures for Intel Cyclone 10 GX devices.

Related Information

- Intel FPGA Arria 10 Transceiver PHY IP Core User Guide
  For information about the transceiver physical (PHY) layer architecture, PLLs, clock networks, and transceiver PHY IP.
- Intel Cyclone 10 GX Transceiver PHY User Guide
  For information about the transceiver PHY layer architecture, PLLs, clock networks, and transceiver PHY IP.

4.2. Hard IP Block Placement In Intel Arria 10 Devices

Intel Arria 10 devices include 1–4 hard IP blocks for PCI Express. The bottom left hard IP block includes the CvP functionality for flip chip packages. For other package types, the CvP functionality is in the bottom right block.

Note: Intel Arria 10 devices do not support configurations that configure a bottom (left or right) hard IP block with a Gen3 x4 or Gen3 x8 IP core and also configure the top hard IP block on the same side with a Gen3 x1 or Gen3 x2 IP core variation.

Figure 11. Intel Arria 10 Devices with 72 Transceiver Channels and Four PCIe Hard IP Blocks

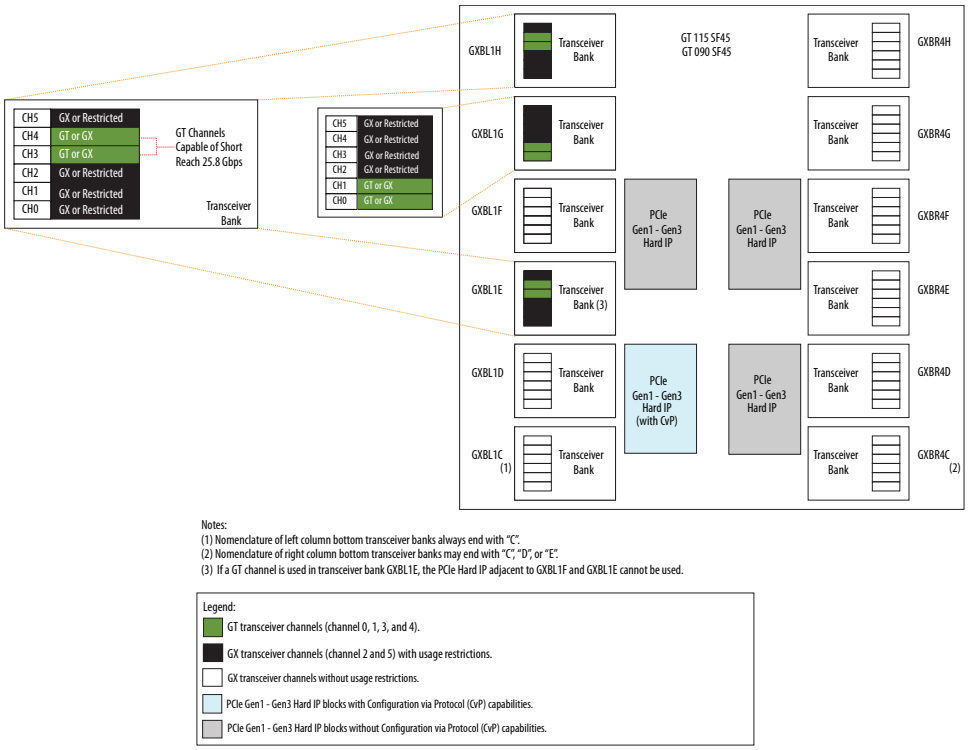
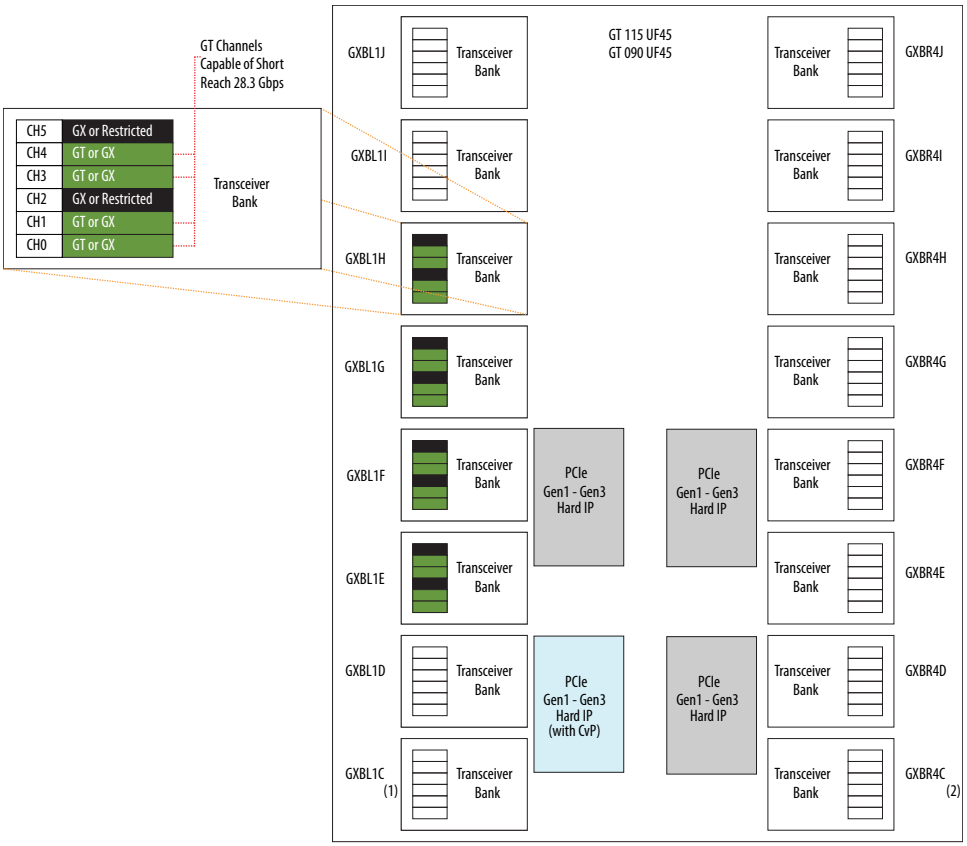


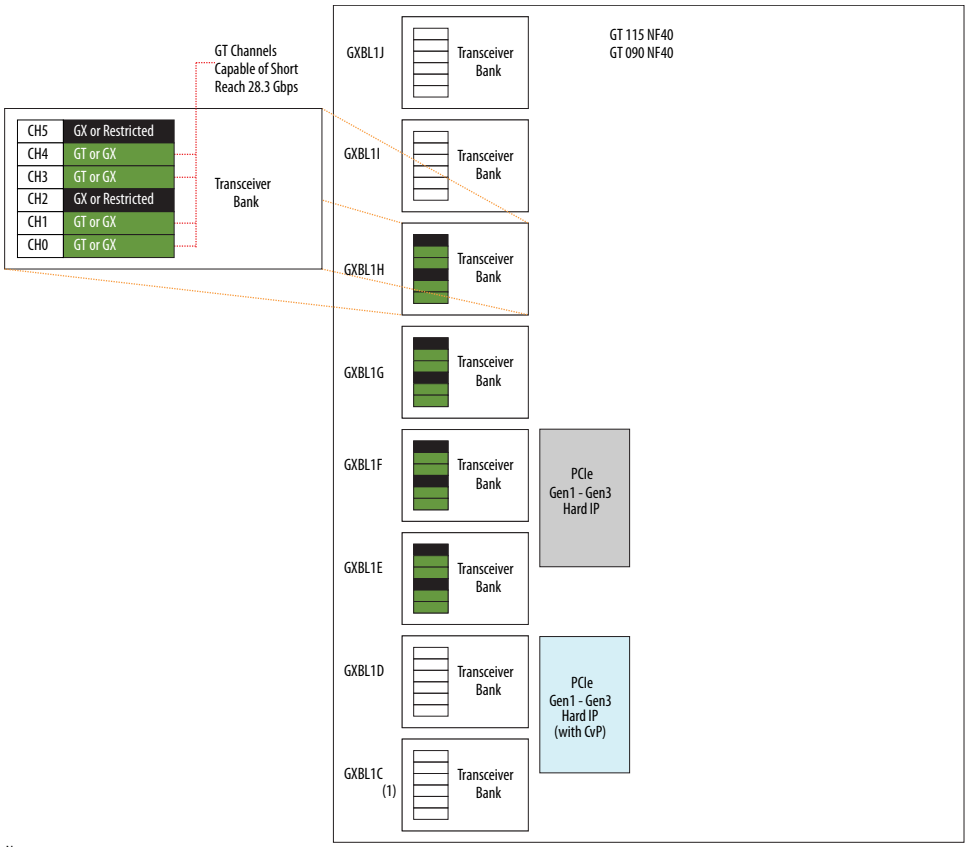
Figure 12. Intel Arria 10 Devices with 96 Transceiver Channels and Four PCIe Hard IP Blocks



Notes:
(1) Nomenclature of left column bottom transceiver banks always ends with "C".
(2) Nomenclature of right column bottom transceiver banks may end with "C", "D", or "E".

Legend:
- GT transceiver channels (channel 0, 1, 3, and 4)
- GX transceiver channels (channel 2 and 5) with usage restrictions.
- GX transceiver channels without usage restrictions.
- PCIe Gen1 - Gen3 Hard IP blocks with Configuration via Protocol (CvP) capabilities.
- PCIe Gen1 - Gen3 Hard IP blocks without Configuration via Protocol (CvP) capabilities.

Figure 13. Intel Arria 10 GT Devices with 48 Transceiver Channels and Two PCIe Hard IP Blocks



Notes:
(1) Nomenclature of left column bottom transceiver banks always end with "C".
(2) These devices have transceivers only on left hand side of the device.

Legend:
- GT transceiver channels (channel 0, 1, 3, and 4).
- GX transceiver channels (channel 2 and 5) with usage restrictions.
- GX transceiver channels without usage restrictions.
- PCIe Gen3 HIP blocks with Configuration via Protocol (CvP) capabilities.
- PCIe Gen3 HIP blocks without Configuration via Protocol (CvP) capabilities.

Refer to the *Intel Arria 10 Transceiver Layout* in the Intel Arria 10 for comprehensive figures for Intel Arria 10 GT, GX, and SX devices.

Related Information

Intel FPGA Arria 10 Transceiver PHY IP Core User Guide
For information about the transceiver physical (PHY) layer architecture, PLLs, clock networks, and transceiver PHY IP.

4.3. Channel and Pin Placement for the Gen1, Gen2, and Gen3 Data Rates

The following figures illustrate pin placements for the Intel Arria 10 Hard IP for PCI Express.

In these figures, channels that are not used for the PCI Express protocol are available for other protocols. Unused channels are shown in gray.

Note: In all configurations, physical channel 4 in the PCS connects to logical channel 0 in the hard IP. You cannot change the channel placements illustrated below.

For the possible values of <*txvr_block_N*> and <*txvr_block_N+1*>, refer to the figures that show the physical location of the Hard IP PCIe blocks in the different types of Intel Arria 10 devices, at the start of this chapter. For each hard IP block, the transceiver block that is adjacent and extends below the hard IP block, is <*txvr_block_N*>, and the transceiver block that is directly above is <*txvr_block_N +1*> . For example, in an Intel Arria 10 device with 96 transceiver channels and four PCIe hard IP blocks, if your design uses the hard IP block that supports CvP, <*txvr_block_N*> is GXB1C and <*txvr_block_N+1*> is GXB1D.

Note: Intel Cyclone 10 GX devices support x1, x2, and x4 at the Gen1 and Gen2 data rates.

Figure 14. Gen1, Gen2, and Gen3 x1 Channel and Pin Placement

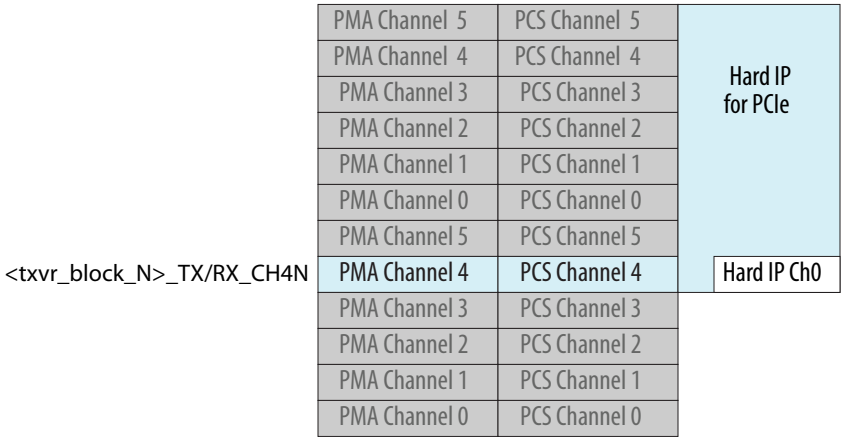


Figure 15. Gen1 Gen2, and Gen3 x2 Channel and Pin Placement

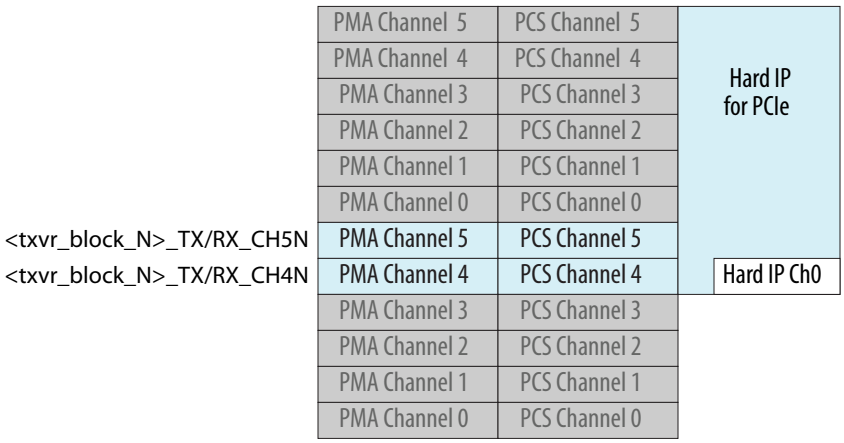


Figure 16. Gen1, Gen2, and Gen3 x4 Channel and Pin Placement

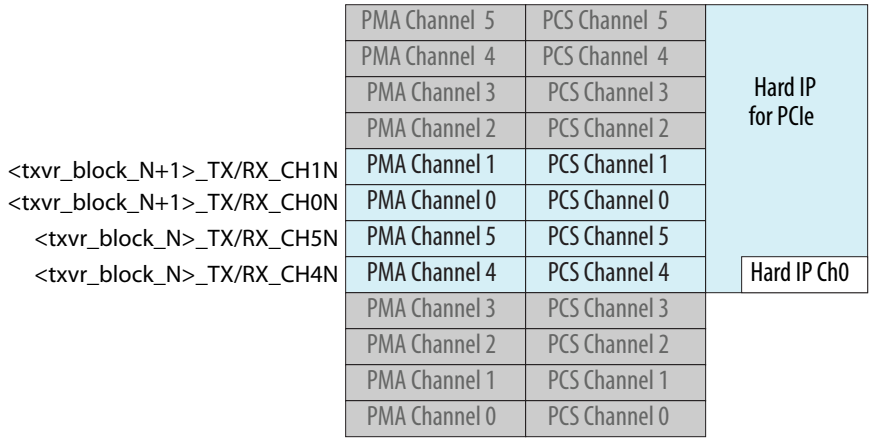
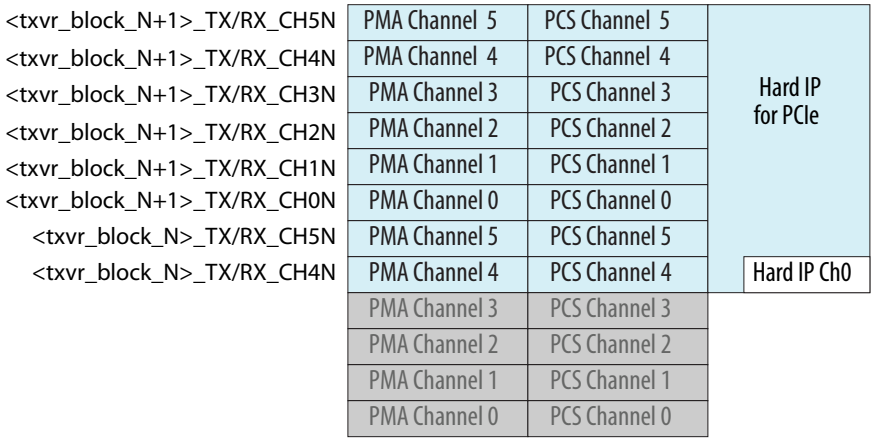


Figure 17. Gen1, Gen2, and Gen3 x8 Channel and Pin Placement



4.4. Channel Placement and fPLL and ATX PLL Usage for the Gen3 Data Rate

The following figures illustrate the channel placement for the Intel Arria 10 Hard IP for PCI Express.

Gen3 variants must initially train at the Gen1 data rate. Consequently, Gen3 variants require an fPLL to generate the 2.5 and 5.0 Gbps clocks, and an ATX PLL to generate the 8.0 Gbps clock. In these figures, channels that are not used for the PCI Express protocol are available for other protocols. Unused channels are shown in gray.

Note: In all configurations, physical channel 4 in the PCS connects to logical channel 0 in the hard IP. You cannot change the channel placements illustrated below.

Figure 18. Intel Arria 10 Gen3 x1 Channel Placement

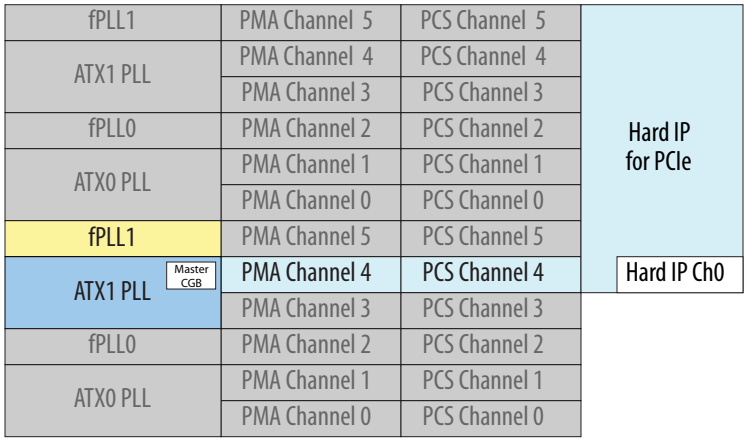


Figure 19. Intel Arria 10 Gen3 x2 Channel Placement

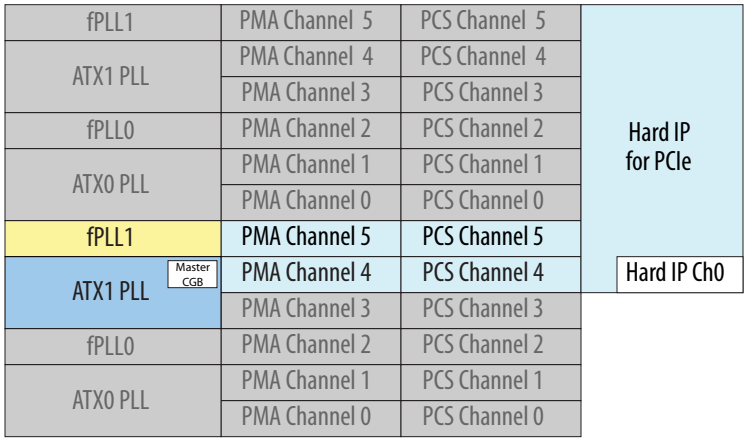


Figure 20. Intel Arria 10 Gen3 x4 Channel Placement

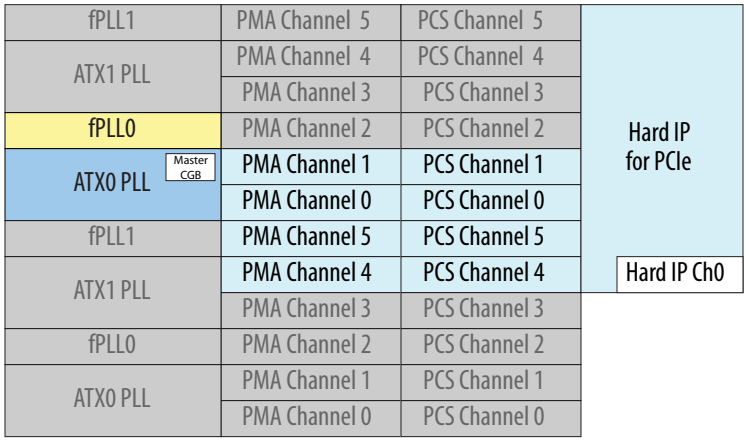
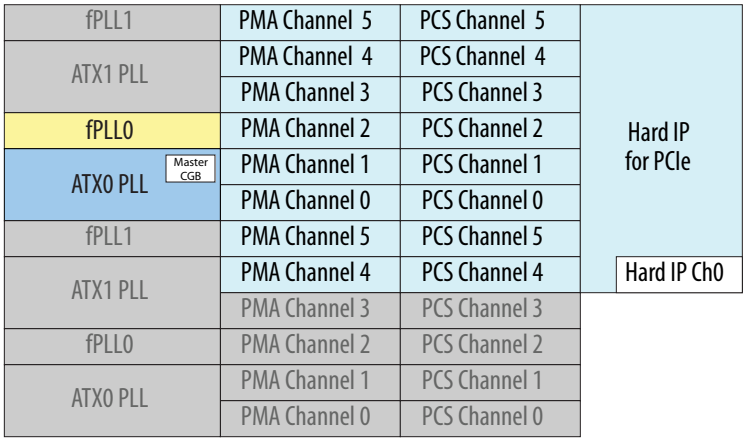


Figure 21. Gen3 x8 Channel Placement



4.5. PCI Express Gen3 Bank Usage Restrictions

Any transceiver channels that share a bank with active PCI Express interfaces that are Gen3 capable have the following restrictions. This includes both Hard IP and Soft IP implementations:

- When VCCR_GXB and VCCT_GXB are set to 1.03 V or 1.12 V, the maximum data rate supported for the non-PCIe channels in those banks is 12.5 Gbps for chip-to-chip applications. These channels cannot be used to drive backplanes or for GT rates.

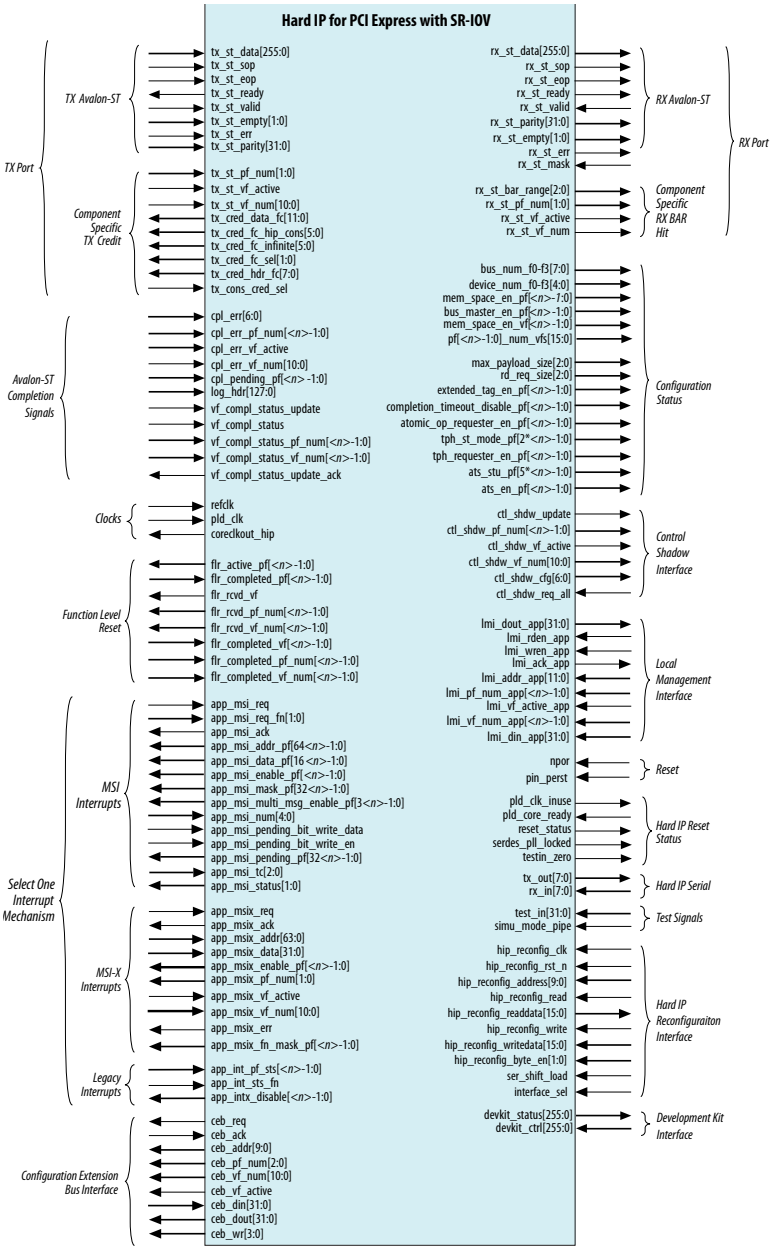
PCI Express interfaces that are only Gen1 or Gen2 capable are not affected.

Status

Affects all Intel Arria 10 ES and production devices. No fix is planned.

5. Interfaces and Signal Descriptions

Figure 22. Intel Arria 10 PCIe with SR-IOV Signals and Interfaces



Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Note: Due to space limitations the figure does not include the PIPE, Reset, or Link Training Signals.

5.1. Avalon-ST TX Interface

User application logic transfers data to the Transaction Layer of the PCIe IP core over the Avalon-ST TX interface.

Table 22. Avalon-ST TX Datapath

Signal	Direction	Description
`tx_st_data[255:0]`	Input	Data for transmission. Transmit data bus. The Application Layer must provide a properly formatted TLP on the TX interface. 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. Refer to the *Qword Alignment* figure in the *Intel Arria 10 Avalon-ST Interface for PCIe Solutions User Guide* for a detailed explanation of qword alignment on the Avalon-ST interface. *Data Alignment and Timing for 256-Bit Avalon-ST TX Interface* in the *Intel Arria 10 Avalon-ST Interface for PCIe Solutions User Guide* for figures showing the mapping of the Transaction Layer's TLP information to `rx_st_data` and examples of the timing of this interface.
`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`(3)	Output	Indicates that the SR-IOV Bridge is ready to accept data for transmission. The SRIOV Bridge 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 Avalon0ST TX interface. The Application Layer can monitor the `reset_status` signal to determine when the IP core has come out of reset. If `tx_st_ready` is asserted by the Transaction Layer on cycle <*n*>, then <*n* + `readyLatency`> is a ready cycle, during which the Application Layer may assert `valid` and transfer data. When `tx_st_ready`, `tx_st_valid` and `tx_st_data` are registered (the typical case), Intel recommends a `readyLatency` of 2 cycles to facilitate timing closure; however, a `readyLatency` of 1 cycle is possible. If no other delays are added to the read-valid latency, the resulting delay corresponds to a `readyLatency` of 2.
`tx_st_valid` (3)	Input	Clocks `tx_st_data` to the core when `tx_st_ready` is also asserted. Between `tx_st_sop` and `tx_st_eop`, `tx_st_valid` must not be deasserted in the middle of a TLP except in response to `tx_st_ready` deassertion. When `tx_st_ready` deasserts, this signal must deassert within 1 or 2 clock cycles. When `tx_st_ready` reasserts, and `tx_st_data` is in mid-TLP, this signal must reassert within 2 cycles. The figure entitled *64-Bit Transaction Layer Backpressures the Application Layer* illustrates the timing of this signal. To facilitate timing closure, Intel recommends that you register both the `tx_st_ready` and `tx_st_valid` signals. If no other delays are added to the ready-valid latency, the resulting delay corresponds to a `readyLatency` of 2.

continued...

(3) To be Avalon-ST compliant, your Application Layer must have a `readyLatency` of 1 or 2 cycles.

Signal	Direction	Description
tx_st_empty[1:0]	Input	Indicates the number of qwords that are empty during cycles that contain the end of a packet. When asserted, the empty dwords are in the high-order bits. Valid only when tx_st_eop is asserted. Indicates the number of upper words that contain data, resulting in the following encodings: • tx_st_empty=0: tx_st_data[255:0] contains valid data • tx_st_empty=1:tx_st_data[191:0] contains valid data • tx_st_empty=2: tx_st_data[127:0] contains valid data • tx_st_empty=3: tx_st_data[63:0]contains valid data
tx_st_err	Input	Indicates an error on transmitted TLP. This signal is used to nullify a packet. It should only be applied to posted and completion TLPs with payload. To nullify a packet, assert this signal for 1 cycle after the SOP and before the EOP. When a packet is nullified, the following packet should not be transmitted until the next clock cycle. tx_st_err is not available for packets that are 1 or 2 cycles long. Note that tx_st_err must be asserted while the valid signal is asserted.
tx_st_parity[31:0]	Input	Byte parity is generated when you turn on **Enable byte parity ports on Avalon ST interface** on the **System Settings** tab of the parameter editor.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.

Related Information

- Qword Alignment
  For an example showing how addresses that are not qword aligned are shifted to create qword alignment.
- Data Alignment and Timing for 256-Bit Avalon-ST TX Interface

5.2. Component-Specific Avalon-ST Interface Signals

Table 23. Component Specific Avalon- ST TX Signals

Signal	Direction	Description
Avalon-ST TX Physical and Virtual Function Identification Signals		
tx_st_pf_num[1:0]	Input	Identifies the Physical Function originating the TLP being transmitted on the TX Stream Interface. The user must provide the originating Function number on this input when transmitting memory requests, Completions, and messages routed by ID. When the originating Function is a VF, this input must be set to the PF Number the VF is attached to. This input is sampled by the SR-IOV bridge when tx_st_sop and tx_st_valid are both high.
tx_st_vf_active	Input	The Application Layer must assert this input when transmitting a TLP driven by a Virtual Function on the TX Stream Interface. The SR-IOV bridge samples this signal when tx_st_sop and tx_st_valid are both asserted.
tx_st_vf_num[10:0]	Input	Identifies the Virtual Function driving the TLP being transmitted on the Avalon-ST TX interface. The Application Layer must provide the VF number offset of the originating Function on this input when transmitting memory requests, Completions, and messages routed by ID. Its value ranges from *0-<n>-1* where *<n>* is the number of VFs in the set of VFs attached to the associated PF. Up to 2048 VFs are supported. The SR-IOV bridge samples this signal when when tx_st_sop, and tx_st_valid, and are asserted.

continued...

Signal	Direction	Description
		Not used when a PF is driving the TLP.
Avalon-ST TX Credit Signals		
tx_cred_data_fc[11:0]	Output	Data credit limit for the received FC completions. Each credit is 16 bytes. There is a latency of two pld_clk clocks between a change on tx_cred_fc_sel and the corresponding data appearing on tx_cred_data_fc and tx_cred_hdr_fc.
tx_cred_fc_hip_cons[5:0]	Output	Asserted for 1 cycle each time the Hard IP consumes a credit. These credits are from messages that the Hard IP for PCIe generates for the following reasons: • To respond to memory read requests • To send error messages This signal is not asserted when an Application Layer credit is consumed. For optimum performance the Application Layer can track of its own consumed credits. (The hard IP also tracks credits and deasserts tx_st_ready if it runs out of credits of any type.) To calculate the total credits consumed, the Application Layer can add its own credits consumed to those consumed by the Hard IP for PCIe. The credit signals are valid after the dlup (data link up) is asserted. The 6 bits of this vector correspond to the following 6 types of credit types: • [5]: posted headers • [4]: posted data • [3]: non-posted header • [2]: non-posted data • [1]: completion header • [0]: completion data During a single cycle, the IP core can consume either a single header credit or both a header and a data credit.
tx_cred_fc_infinite[5:0]	Output	When asserted, indicates that the corresponding credit type has infinite credits available and does not need to calculate credit limits. The 6 bits of this vector correspond to the following 6 types of credit types: • [5]: posted headers • [4]: posted data • [3]: non-posted header • [2]: non-posted data • [1]: completion header • [0]: completion data
tx_cred_fc_sel[1:0]	Input	Signal to select between the tx_cred_hdr_fc and tx_cred_data_fc outputs. There is a latency of two pld_clk clocks between a change on tx_cred_fc_sel and the corresponding data appearing on tx_cred_data_fc and tx_cred_hdr_fc. The following encoding are defined: • 2'b00: Output Posted credits • 2'b01: Output Non-Posted credits • 2'b10: Output Completions
tx_cred_hdr_fc[7:0]	Output	Header credit limit for the FC posted writes. Each credit is 20 bytes. There is a latency of two pld_clk clocks between a change on tx_cred_fc_sel and the corresponding data appearing on tx_cred_data_fc and tx_cred_hdr_fc.
tx_cons_cred_select	Input	When 1, the output tx_cred_data* and tx_cred_hdr* signals specify the value of the hard IP internal credits-consumed counter. When 0, tx_cred_data* and tx_cred_hdr* signal specify the limit value. This signal is present when you turn **On Enable credit consumed selection port** in the parameter editor .

The following table describes the signals that comprise the completion side band signals for the Avalon-ST interface. The Intel Arria 10 Hard IP for PCI Express provides a completion error interface that the Application Layer can use to report errors, such as programming model errors. When the Application Layer detects an error, it can assert the appropriate `cpl_err` bit to indicate what kind of error to log. If separate requests result in two errors, both are logged. The Hard IP sets the appropriate status bits for the errors in the Configuration Space. It also automatically sends error messages in accordance with the *PCI Express Base Specification*. Note that the Application Layer is responsible for sending the completion with the appropriate completion status value for non-posted requests. Refer to Error Handling on page 109 for information on errors that are automatically detected and handled by the Hard IP.

For a description of the completion rules, the completion header format, and completion status field values, refer to Section 2.2.9 of the *PCI Express Base Specification*.

Table 24. Completion Signals for the Avalon-ST Interface

Signal	Direction	Description
`cpl_err[6:0]`	Input	Completion error from a PF or VF This signal reports completion errors to the Configuration Space. The SR-IOV Bridge responds to the assertion of these bits by logging the status in the error reporting registers of the Function and sending error messages when required. When an error occurs, the appropriate signal is asserted for one cycle. The individual bits indicate following error or status conditions: • `cpl_err[0]`: Completion timeout error with recovery. This signal should be asserted when a master-like interface has performed a non-posted request that never receives a corresponding completion transaction after the 50 ms timeout period when the error is correctable. The SR-IOV Bridge sends a Correctable Error message to the Root Complex. • `cpl_err[1]`: Completion timeout error without recovery. This signal should be asserted when a master-like interface has performed a non-posted request that never receives a corresponding completion transaction after the 50 ms time-out period when the error is not correctable. .The SR-IOV Bridge sends a Fatal or Non-Fatal Error message to the Root Complex, The severity of the Completion Timeout error programmed in the AER Uncorrectable Error Severity Register determines the error message. • `cpl_err[2]`: Completer abort error. The Application Layer asserts this signal to respond to a non-posted request with a Completer Abort (CA) completion. The SR-IOV Bridge sends a Fatal or Correctable Error message to the Root Complex. The severity of the Completer Abort Sent error programmed in the AER Uncorrectable Error Severity Register of the Function determines the error message. • `cpl_err[3]`: Unexpected completion error. This signal must be asserted when an Application Layer master block detects an unexpected Completion. The SR-IOV Bridge sends a Fatal or Correctable Error message to the Root Complex. The severity of the Unexpected Completion Received error programmed in the Uncorrectable Error Severity Register of the Function determines the error message..

continued...

Signal	Direction	Description
		• cpl_err[4]: Unsupported Request (UR) error for posted TLP. The Application Layer asserts this signal to treat a posted request as an Unsupported Request. The bridge also sends a Fatal or Non-Fatal Error message to the Root Complex, The severity of the error programmed in the AER Uncorrectable Error Severity Register of the Function determines the message. • cpl_err[5]: Unsupported Request error for non-posted TLP. The Application Layer asserts this signal to respond to a non-posted request with an Request (UR) completion. The bridge also sends a Fatal or Correctable Error message to the Root Complex, The severity of the error programmed in the AER Uncorrectable Error Severity Register of the Function determines the message. • cpl_err[6]: Log header. If header logging is required, this bit must be set in the every cycle in which any of cpl_err[2], cpl_err[3], cpl_err[4], or cpl_err[5]is set. The header must be supplied on the inputs log_hdr[127:0].
cpl_err_pf_num[<*n*>-1:0]	Input	Identifies the Function reporting the error oncpl_err inputs. When the Function is a VF, this input must specify the PF Number to which the VF is attached.
cpl_err_vf_active	Input	Indicates that the Function reporting the error is a Virtual Function. When this input is asserted, the VF number offset of the Function must be provided on cpl_err_vf_num[10:0].
cpl_err_vf_num[10:0]	Input	Whencpl_err_vf_active is asserted, this input identifies the VF number offset of the Function reporting the error. Its value ranges from *0-<n>-1*, where <*n*>is the number of VFs in the set of VFs attached to the associated PF.
cpl_pending_pf[<*n*>-1:0]	Input	Completion pending from the PF. The Application Layer must assert this signal when a PF has issued one of more Non-Posted transactions waiting for completions. For example, when a Non-Posted Request is pending from PF0. cpl_pending_pf[0] records pending completions for PF0. cpl_pending_pf[1] records pending completions for PF1.
log_hdr[127:0]	Input	When any of the bits 2, 3, 4, 5 of cpl_err is asserted, the Application Layer may provide the header of the TLP that caused the error condition. The order of bytes is the same as the order of the header bytes for the Avalon-ST streaming interfaces.
vf_compl_status_update	Input	Completion status update from the VF. The Application Layer must assert vf_compl_status_update whenever the Completion Pending status changes in a VF, The Application Layer must assert vf_compl_status_update until the SR-IOV bridge responds by setting vf_compl_status_update_ack.
vf_compl_status	Input	Must be set to the current Completion Pending status of the associated Function whenvf_compl_status_updateis asserted, The following encodings are defined: • 0: No Completion pending • 1: Completion pending for the Function.

continued...

Signal	Direction	Description
vf_compl_status_pf_num[<n>-1:0]	Input	Must be set to the PF number associated with the signaling Function when vf_compl_status_updateis asserted, <n> is the number of PFs.
vf_compl_status_vf_num[<n>-1:0]	Input	Must be set to the VF offset associated with the signaling Function when vf_compl_status_update is asserted, <n> is the number of VFs.
vf_compl_status_update_ack	Output	The SR-IOV Bridge asserts vf_compl_status_update_ack for 1cycle when it has completed updating its internal state in response to vf_compl_status_update. The Application Layer must assert vf_compl_status_update high until vf_compl_status_update_ack is asserted.

Related Information

PCI Express Base Specification Rev 3.0

5.3. Avalon-ST RX Interface

User application logic receives data from the Transaction Layer of the PCIe IP core over the Avalon-ST RX interface.

Table 25. Avalon-ST RX Datapath

Signal	Direction	Description
rx_st_data[255:0]	Output	Receive data bus. Refer to *Data Alignment and Timing for 256-Bit Avalon-ST RX Interface* in the *Intel Arria 10 Avalon-ST Interface for PCIe Solutions User Guide* for figures showing the mapping of the Transaction Layer's TLP information to rx_st_data and examples of the timing of this interface. Note that the position of the first payload dword depends on whether the TLP address is qword aligned. The mapping of message TLPs is the same as the mapping of TLPs with 4-dword headers. Refer to the *Qword Alignment* figure in the *Intel Arria 10 Avalon-ST Interface for PCIe Solutions User Guide* for a detailed explanation of qword alignment on the Avalon-ST interface. Refer to *Data Alignment and Timing for 256-Bit Avalon-ST RX Interface* in the *Intel Arria 10 Avalon-ST Interface for PCIe Solutions User Guide* for figures showing the mapping of the Transaction Layer's TLP information to rx_st_data and examples of the timing of this interface.
rx_st_sop	Output	Indicates that this is the first cycle of the TLP when rx_st_valid is asserted.
rx_st_eop	Output	Indicates that this is the last cycle of the TLP when rx_st_valid is 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. If rx_st_ready is asserted by the Application Layer on cycle <n> , then <n + > readyLatency is a ready cycle, during which the Transaction Layer may assert valid and transfer data. The RX interface supports a readyLatency of 2 cycles.
rx_st_valid	Output	Clocks rx_st_data into the Application Layer. Deasserts within 2 clocks of rx_st_ready deassertion and reasserts within 2 clocks of rx_st_ready assertion if more data is available to send.
rx_st_parity[31:0]	Output	The IP core generates byte parity when you turn on **Enable byte parity ports on Avalon-ST interface** on the **System Settings** tab of the parameter editor. Each bit represents odd parity of the associated byte of

continued...

Signal	Direction	Description
		the rx_st_data bus. For example, bit[0] corresponds to rx_st_data[7:0], bit[1] corresponds to rx_st_data[15:8], and so on.
rx_st_empty[1:0]	Output	Indicates the number of quadwords that are empty during cycles that contain the end of a packet. Its encodings are defined: • 00: Data on rx_st_data_app[255:0] is valid. If the TLP is ending in this cycle, its last byte is in rx_st_data_ app[255:192]. • 01: Data on rx_st_data_app[191:0] is valid. A TLP is ending in this cycle, and its last byte is in rx_st_data_ app[191:128]. • 10: Data on rx_st_data_app[127:0] is valid. A TLP is ending in this cycle, and its last byte is in rx_st_data_ app[127:64] . • 11: Data on rx_st_data_app[63:0] is valid. A TLP is ending in this cycle, and its last byte is in rx_st_data_ app[63:0] .
rx_st_err	Output	When asserted, indicates an uncorrectable error in the TLP being transferred.

Related Information

- Qword Alignment
  For an example showing how addresses that are not qword aligned are shifted to create qword alignment.
- Data Alignment and Timing for 256-Bit Avalon-ST RX Interface
- Avalon Interface Specifications
  For information about the Avalon-ST interfaces.

5.4. BAR Hit Signals

The IP core contains logic that determines which BAR corresponds to a particular TLP for the following types of transactions: memory reads, memory writes and Atomic Ops. This information is sent out via the rx_st_bar_range[2:0] outputs. User application logic can leverage this information to know what BAR the transactions going across the Avalon-ST RX interface are targeting.

Signal	Direction	Description
rx_st_bar_range[2:0]	output	These outputs identify the matching BAR for the TLP on the Avalon-ST RX interface. They are valid for MRd, MWr and Atomic Op TLPs . These outputs should be ignored for all other TLPs. They are valid in the first cycle of a TLP, when rx_st_valid and rx_st_sop are asserted. The following BAR numbers are defined: • 0: BAR0 when configured as 32-bit BAR or BAR0-1 when configured as 64-bit BAR • 1: BAR1 when configured as 32-bit BAR; reserved when BAR1 combined with BAR0 to form a 64-bit BAR • 2: BAR2 when configured as 32-bit BAR or BAR 2-3 when configured as 64-bit BAR • 3: BAR3 when configured as 32-bit BAR; reserved when BAR2 combined with BAR3 to form a 64-bit BAR • 4: BAR4 when configured as 32-bit BAR, or BAR4-5 when configured as 64-bit BAR • 5: BAR5 when configured as 32-bit BAR; reserved when BAR4 combined with BAR5 to form a 64-bit BAR • 6–7: Reserved

continued...

Signal	Direction	Description
		When rx_st_vf_active is deasserted, the BAR number represents a PF BAR. When rx_st_vf_activeis asserted, the BAR number represents a VF BAR.
rx_st_pf_num[2:0]	output	Identifies the Function targeted by the TLP on the Avalon-ST RX interface. This output is valid for memory requests, Completions, and messages routed by ID. When the targeted Function is a VF, this output provides the PF Number to which the VF is attached.
rx_st_vf_active	output	Indicates that the Function targeted by the TLP is a Virtual Function. When this output is asserted, the VF number offset of the VF is provided on rx_st_vf_num. This output is valid for memory requests, Completions, and messages routed by ID.
rx_st_vf_num[11:0]	output	When rx_st_vf_active is high, this output identifies the VF number offset of the VF targeted by the TLP on the Avalon-ST RX interface. This output is valid for memory requests, Completions, and messages routed by ID. Its value ranges from *0-(<n>-1)*, where *<n>* is the number of VFs associated with a PF.

5.5. Configuration Status Interface

The output signals listed below drive the settings of the various configuration register fields of the Functions. These settings are often needed in designing Application Layer logic.

Table 26. Configuration Status Interface

Signal	Direction	Description
bus_num_f0-f3[7:0]	Output	Bus number assigned to the PF by the Root Complex. Captured from CfgWr transactions. When ARI is in use, the Application Layer uses bus_num_f*<n>-1* to generate requests as a master or respond to requests as a Completer. When ARI is not in use, the application uses bus_num_f*<n>-1* when generating requests from PF*<n>-1* and its associated VF and responding to requests as a Completer. Provided for information only. The SR-IOV Bridge inserts the appropriate bus number in the header of transmitted TLPs
device_num_f0-f3[4:0]	Output	Device number assigned to the PF by the Root Complex. Captured from CfgWr transactions. When ARI is not in use, the Application Layer uses device_num_f*<n>-1* when generating requests from PF*<n>-1* and responding to requests as a Completer. Not used when ARI is enabled. Provided for information only. The SR-IOV Bridge inserts the appropriate device number in the header of transmitted TLPs.
mem_space_en_pf[*<n>*-1:0]	Output	The PF Command Registers drive the Memory Space Enable bit. *<n>* is the number of PFs.
bus_master_en_pf[*<n>*-1:0]	Output	The PF Command Registers drive the Bus Master Enable bit. *<n>* is the number of PFs.
mem_space_en_vf[*<n>*-1:0]	Output	The PF Control Registers drive the SR-IOV Memory Space Enable bit. *<n>* is the number of PFs.
pf[*<n>*-1:0]_num_vfs[15:0]	Output	This output drives the value of the NumVFs register in the PF SR-IOV Capability Structure .

continued...

Signal	Direction	Description
max_payload_size[2:0]	Output	When only PF0 is present, the max payload size field of the PF0 PCI Express Device Control Register drives this output. When more PFs are present, the minimum value of the max payload size field of the PCI Express Device Control Registers drives this output.
rd_req_size[2:0]	Output	When only PF 0 is present, the max read request size field of PF0 PCI Express Device Control Register drives this output. When more PFs are present, the minimum value of the max read request size fields of the PCI Express Device Control Registers drives this output.
extended_tag_en_pf[<*n*>-1:0]	Output	Bit <*n*>of this output reflects the setting of the Extended Tag Enable, bit[8], of the Device Control Register of PF<*n*>.
completion_timeout_disable_pf[<*n*>-1:0]		Bit <*n*> of this output reflects the setting of the Completion Timeout Disable, bit [4], of the Device Control 2 Register of PF<*n*>.
atomic_op_requester_en_pf[<*n*>-1:0]	Output	Bit <*n*> of this output reflects the setting of the Atomic Op Requester Enable bit of the Device Control 2 Register of PF <*n*>.
tph_st_mode_pf[2*<*n*>-1:0]	Output	Bits [1:0] of this output reflect the setting of the TPH ST Mode Select field, bits[1:0] of the TPH Requester Control Register of PF0. Bits [3:2] reflect the setting of the TPH ST Mode Select field bits [1:0] of the TPH Requester Control Register of PF 1, and so on.
tph_requester_en_pf[<*n*>-1:0]	Output	Bit <*n*>of this output reflects the setting of the TPH Requester Enable field, bit[8], of the TPH Requester Control Register of PF <*n*>.
ats_stu_pf[5*<*n*>-1:0]	Output	Bits [4:0] of this output reflect the setting of the Smallest Translation Unit field, bits [4:0], in the ATS Control Register of PF0; bits [9:5] reflect the setting of the Smallest Translation Unit field, bits [4:0], in the ATS Control Register of PF1, and so on.
ats_en_pf[<*n*>-1:0]	Output	Bit <*n*>of this output reflects the setting of the Enable bit, bit [15], in the ATS Control Register of PF <*n*>.

5.6. Clock Signals

Table 27. Clock Signals Hard IP Implementation

Signal	Direction	Description
refclk	Input	Reference clock for the Intel Arria 10 Hard IP for PCI Express. It must have the frequency specified under the **System Settings** heading in the parameter editor.
pld_clk	Input	Clocks the Application Layer. You can drive this clock with coreclkout_hip. If you drive pld_clk with another clock source, it must be equal to or faster than coreclkout. All the interfaces and internal modules of the SR-IOV Bridge use this clock as the reference clock. Its frequency is 125 or 250 MHz.
coreclkout_hip	Output	This is a fixed frequency clock used by the Data Link and Transaction Layers. To meet PCI Express link bandwidth constraints, this clock has minimum frequency requirements as listed in *coreclkout_hip Values for All Parameterizations* in the *Reset and Clocks* chapter .

Refer to *Intel Arria 10 Hard IP for PCI Express Clock Domains* in the *Reset and Clocks* chapter for more information about clocks.

5.7. Function-Level Reset (FLR) Interface

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

Table 28. Function-Level Reset (FLR) Interface

Signal	Width	Direction	Description
flr_active_pf	Number of PFs	Output	The SR-IOV Bridge asserts flr_active_pf when bit 15 of the PCIe Device Control Register is set. Bit 15 is the FLR field. This indicates to the user application that the Physical Function (PF) is undergoing a reset. Among the flr_active_pf bits, bit 0 is for PF0, bit 1 is for PF1, and so on. Once asserted, the flr_active_pf signal remains high until the user application sets flr_completed_pf high for the associated function. The user application must monitor these signals and perform actions necessary to clear any pending transactions associated with the function being reset. The user application must assert flr_completed_pf to indicate it has completed the FLR actions and is ready to re-enable the PF.
flr_rcvd_vf	1	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 a VF. flr_rcvd_pf_num and flr_rcvd_vf_num contain the PF number and the VF offset associated with the Function being reset. The user application responds to a pulse on this output by clearing any pending transactions associated with the VF being reset. It then asserts flr_completed_vf to indicate that it has completed the FLR actions and is ready to re-enable the VF.
flr_rcvd_pf_num	1 - 3	Output	When flr_rcvd_vf is asserted high, this output specifies the PF number associated with the VF undergoing FLR.
flr_rcvd_vf_num	1 - 11	Output	When flr_rcvd_vf is asserted high, this output specifies the VF number offset associated with the VF undergoing FLR.
flr_completed_pf	Number of PFs	Input	The assertion of this input for one or more cycles indicates that the application has completed resetting all the logic associated with the Physical Function. Among the flr_completed_pf bits, bit 0 is for PF0, bit 1 is for PF1, and so on. When the application sees flr_active_pf high, it must assert flr_completed_pf within 100 milliseconds to re-enable the Function.
flr_completed_vf	1	Input	The assertion of this input for one cycle indicates that the user application has completed resetting all the logic associated with the VF identified by the information placed on flr_completed_vf_num and flr_completed_pf_num. The user application must assert flr_completed_vf within 100 milliseconds after receiving the FLR to re-enable the VF.
flr_completed_pf_num	1 - 3	Input	When flr_completed_vf is asserted high, this input specifies the PF number associated with the VF that has completed its FLR.
flr_completed_vf_num	1 - 11	Input	When flr_completed_vf is asserted high, this input specifies the VF number offset associated with the VF that has completed its FLR.

Figure 23. FLR Interface Timing Diagram for Physical Functions

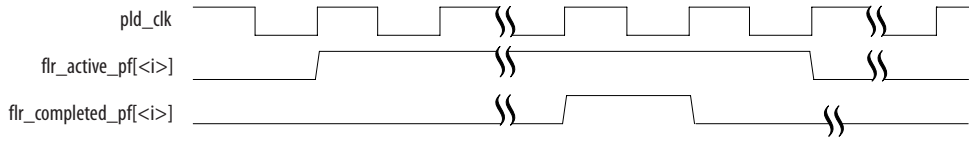
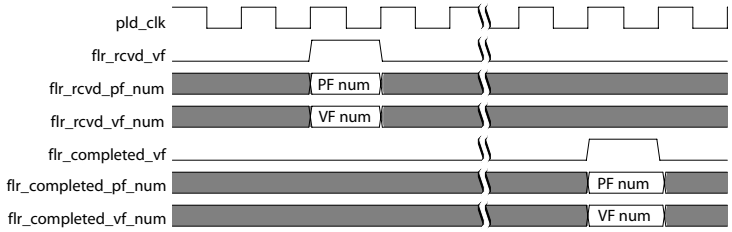


Figure 24. FLR Interface Timing Diagram for Virtual Functions



5.8. SR-IOV Interrupt Interface

The SR-IOV Bridge supports MSI and MSI-X interrupts for both Physical and Virtual Functions. The Application Layer can use this interface to generate MSI or MSI-X interrupts from both PFs and VFs. The SR-IOV Bridge also supports legacy Interrupts for Physical Functions if you configure the core to support only PFs. To support only PFs, turn on **Enable SR-IOV support** on the SR-IOV System Settings tab of the component GUI. The Application Layer should select one of the three types of interrupts, depending on the support provided by the platform and the software drivers. Ground the input pins for the unused interrupt types.

This interface also includes signals to set and clear the individual bits in the MSI Pending Bit Register.

Table 29. MSI Interrupts

Signal	Direction	Description
app_msi_req	Input	When asserted, the Application Layer is requesting that an MSI interrupt be sent. Assertion causes an MSI posted write TLP to be generated based on the MSI configuration register values of the specified Function, and the setting of the app_msi_tc and app_msi_num inputs.
app_msi_req_fn[1:0]	Input	Specifies the PF generating the MSI interrupt. It must be set to the interrupting PF number when asserting app_msi_req.
app_msi_ack	Output	Ack for MSI interrupts. When asserted, indicates that Hard IP has sent an MSI posted write TLP in response app_msi_req . The Application Layer must wait for app_msi_ack after asserting app_msi_req. The Application Layer must de-assert app_msi_req for at least 1 cycle before signaling a new MSI interrupt.
app_msi_addr_pf[64*<*n*>-1:0]	Output	Driven by the MSI address registers of the PFs. <*n*> is the number of PFs.
app_msi_data_pf[16<*n*>-1:0]	Output	Driven by the MSI Data Registers of the PFs, . <*n*>= the number of PFs.
app_msi_enable_pf[<n>-1:0]	Output	Driven by the MSI Enable bit of the MSI Control Registers of the PFs.
app_msi_mask_pf[32<*n*>-1:0]	Output	The MSI Mask Bits of the MSI Capability Structure drive app_msi_mask_pf. This mask allows software to disable or defer message sending on a per-vector basis. app_msi_mask_pf[31:0] mask vectors for PF0.app_msi_mask_pf[63:32] mask vectors for PF1.

continued...

Signal	Direction	Description
app_msi_multi_msg_enable_pf[3*<*n*>-1:0]	Output	Defines the number of interrupt vectors enabled for each PF. The following encodings are defined: • 3'b000: 1 vector • 3'b001: 2 vectors • 3'b010: 4 vectors • 3'b100: 16 vectors • 3'b101: 32 vectors The MSI Multiple Message Enable field of the MSI Control Register of PF0 drives app_msi_multi_msg_enable_pf[2:0]. The MSI Multiple Message Enable field of the MSI Control Register of PF1 drives app_msi_multi_msg_enable_pf[5:3], and so on.
app_msi_num[4:0]	Input	Identifies the MSI interrupt type to be generated. Provides the low-order message data bits to be sent in the message data field of MSI messages. Only bits that are enabled by the apply.
app_msi_pending_bit_write_data	Input	Writes the MSI Pending Bit Register of the specified PF when app_msi_pending_bit_write_en is asserted. app_msi_num[4:0] specifies the bit to be written. For more information about the MSI Pending Bit Array (PBA), refer to *Section 6.8.1.7 Mask Bits for MSI (Optional)* in the *PCI Local Bus Specification, Revision 3.0.* Refer to Figure 26 on page 51 below.
app_msi_pending_bit_write_en	Input	Writes a 0 or 1 into selected bit position in the MSI Pending Bit Register. app_msi_num[4:0] specifies the bit to be written. msi_pending_bit_write_data specifies the data to be written (0 or 1). app_msi_req_fn specifies the function number. This input must be asserted for one cycle to perform the write operation. msi_pending_bit_write_en cannot be asserted when app_msi_req is high. Refer to Figure 26 on page 51 below.
app_msi_pending_pf[32*<*n*>-1:0]	Output	The MSI Data Registers of the PFs drive msi_pending_pf. <*n*> is the number of PFs.
app_msi_tc[2:0]	Input	Specifies the traffic class to be used to send the MSI or MSI-X posted write TLP. Must be valid when app_msi_req is asserted.
app_msi_status[1:0]	Output	Specifies the status of an MSI request. Valid when app_msi_ack is asserted. The following encodings are defined: • 2'b00: MSI message sent • 2'b01: MSI message is pending and not sent upstream because the MSI mask was set. And, the Pending bit was set for the MSI number. • 2'b10: Requested aborted because of invalid parameters. Or, request aborted because the MSI Enabled bit was not set in the function's MSI Capability structure. • 2'b11: Reserved.

Figure 25. Timing Diagram for MSI Interrupt Generation

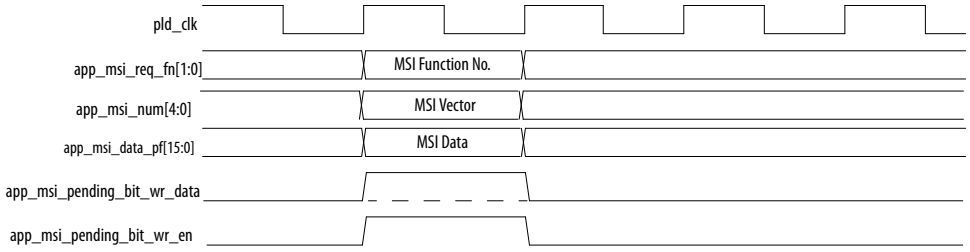


Figure 26. Timing Diagram for MSI Pending Bit Write Operation

The MSI Pending Bit Write Operation aborts the pending MSI interrupt.

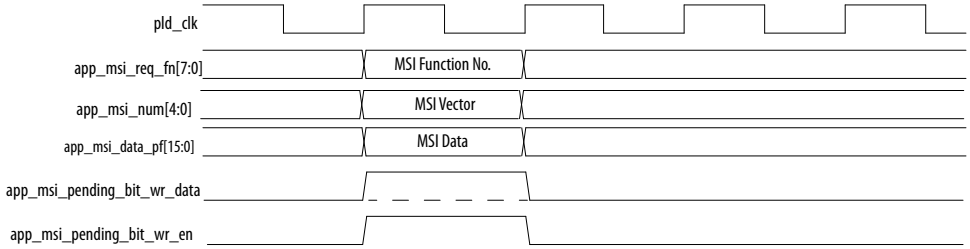


Table 30. MSI-X Interrupts

Signal	Direction	Description
app_msix_req	Input	When asserted, the Application Layer is requesting that an MSI-X interrupt be sent. Assertion causes an MSI-X posted write TLP to be generated. The MSI-X TLP uses data from app_msi_req_pf_num, app_msi_req_vf_num, app_msi_req_vf_active, app_msix_addr, app_msix_data, and app_msi_tc inputs. Refer to Figure 27 on page 52 below.
app_msix_ack	Output	Ack for MSI-X interrupts. When asserted, indicates that Hard IP has sent an MSI-X posted write TLP in response app_msix_req . The Application Layer must wait for after asserting app_msix_req. The Application Layer must de-assert app_msix_req for at least 1 cycle before signaling a new MSI interrupt.
app_msix_addr[63:0]	Input	The Application Layer drives the address for the MSI-X posted write TLP on this input. Driven in the same cycle as app_msix_req.
app_msix_data[31:0]	Input	The Application Layer drives app_msix_data[31:0] for the MSI-X posted write TLP. Driven in the same cycle as app_msix_req.
app_msix_enable_pf[<*n*>-1:0]	Output	Driven by the MSIX Enable bit of the MSIX Control Register of the PFs.
app_msix_pf_num[1:0]	Output	Identifies the Physical Function generating the MSI-X interrupt. It must be set to the interrupting Function number when asserting app_msix_req. When the targeted Function is a VF, this input specifies the PF Number to which the VF is attached.
app_msix_vf_active	Input	Specifies that the Function generating the MSI-X interrupt is a Virtual Function. If this input is asserted, the user must provide the VF number offset of the VF generating the interrupt on app_msix_vf_num.
app_msix_vf_num[10:0]	Input	When app_msix_vf_active is asserted, this input identifies the VF number offset for the VF generating the interrupt. Its value ranges from *0-<n>-1*, where i<*n*>s the number of VFs in the set of VFs attached to the associated PF.
app_msix_tc[2:0]	Input	Specifies the traffic class of the MSI-X posted write TLP. It must be valid when app_msix_req is asserted.
app_msix_ack	Output	Acknowledgment for MSI-X interrupts. A pulse on this output indicates that an MSI-X posted write TLP has been sent in response to the assertion of the app_msix_req input. The user application must wait for the acknowledgment after asserting app_msix_req, and must de-assert app_msix_req for at least one cycle before signaling a new MSI-X interrupt.

continued...

Signal	Direction	Description
app_msix_err	Output	Signals an error during the execution of an MSI-X request. Valid when app_msix_ack is asserted. The following encodings are defined: • 1b'0: MSI-X message sent • 1b'1: Error detected during execution of the MSI-X request. No message sent. The following errors may occur: — The function number is invalid — The MSI-X Enable bit for the function was not set — The MSI-X Function Mask was not set
app_msix_fn_mask_pf[<*n*>-1:0]	Output	Driven by the MSI-X Function Mask bit of the MSI-X Control Register of the PFs.

Figure 27. Timing Diagram for MSI-X Interrupt Generation

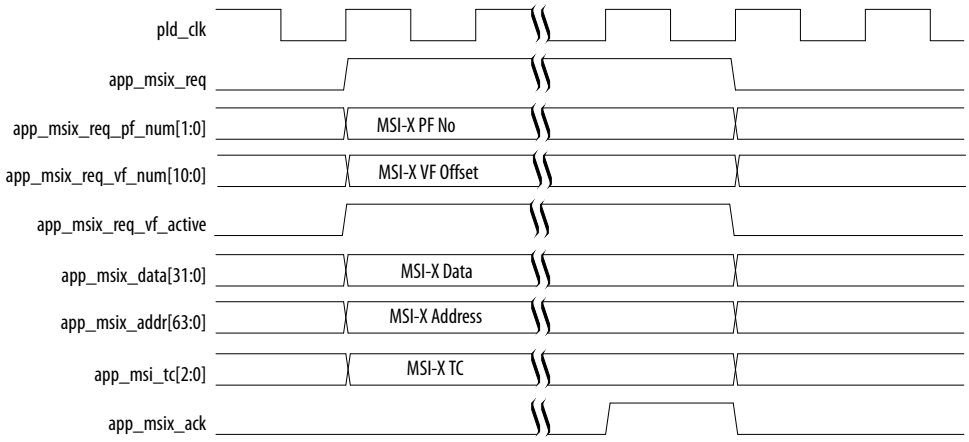


Table 31. Legacy Interrupts

Signal	Direction	Description
app_int_pf_sts[<*n*>-1:0]	Input	The Application Layer uses this signal to generate a legacy INT<*n*>interrupt. PF<*n*> drives app_int_pf_sts[<n>-1:0] . The Hard IP sends a INTx_Assert message upstream to the Root Complex in response to a low-to- high transition. The Hard IP sends a INTX_Deassert in response to a high-to-low transition. The INTX_Deassert message is only sent if a previous INTx_Assert message was sent. When multiple Functions share the same INTx pin, only a single INTx_Assert message is sent if more than one interrupt sharing the same INTx pin are active at the same time. This input has no effect if the INT<*n*>Disable bit in the PCI Command Register of the interrupting function is set to 1.
app_int_sts_fn	Input	Identifies the function generating the legacy interrupt. When app_int_sts_fn = 0, specifies status for PF0. When app_int_sts_fn = 1, specifies status for PF1.
app_intx_disable[<*n*>-1:0]	Output	This output is driven by the INT<*x*>Disable bit of the PCI Command Register of PF.

Related Information

PCI Local Bus Specification, Revision 3.0

5.9. Configuration Extension Bus (CEB) Interface

The Configuration Extension Bus (CEB) interface provides a way to add extra capabilities on top of those available in the internal configuration space of the SR-IOV Bridge. The configuration TLPs with a destination address not matching with internally implemented registers are routed to the CEB interface. When a transaction is presented on this interface, the user application is responsible for acknowledging the request by asserting `ceb_ack` if it is intended or supported. For read requests, `ceb_ack` and `ceb_din` should be driven by the user logic. For write requests, only `ceb_ack` is needed. The bridge will return a Completion with no data upon receiving the acknowledgment.

If the user application does not implement the register at the address targeted by this interface, meaning `ceb_ack` will not be asserted, there will be an acknowledgment timeout, which results in a Completion with all zeros in the data and completion status fields being sent back to the host. This behavior applies for both read and write accesses.

The SR-IOV bridge sends the dword address of the register being accessed on the CEB interface with the additional function number information. This information is held on the address and data busses until an acknowledgement is received from the user application logic.

Note: Intel recommends that you select an open space in the configuration space across all available PFs or VFs by disabling optional Capabilities structures to implement the user capabilities registers. This implies that developing user registers for individual PFs or VFs is not an option.

The user application must return an acknowledgement within the number of clock cycles specified by the **CEB REQ to ACK latency** parameter described in section 3.3.

Table 32. Configuration Extension Bus (CEB) Interface

Signal	Direction	Description
`ceb_req`	Output	When asserted, indicates a valid Configuration Extension Interface access cycle. Deasserted when `ceb_ack` is asserted.
`ceb_ack`	Input	Application asserts this signal for one clock cycle to acknowledge `ceb_req`.
`ceb_addr[9:0]`	Output	Dword address of the register being accessed.
`ceb_pf_num[2:0]`	Output	The Physical Function (PF) number of the register access.
`ceb_vf_num[10:0]`	Output	Indicates the child Virtual Function (VF) number of the parent PF indicated by `ceb_pf_num`.
`ceb_vf_active`	Output	Indicates the access is for a Virtual Function implemented in the slot's Physical Function.
`ceb_din[31:0]`	Input	Application returns data for a read access using this signal bus. The data must be valid when `ceb_ack` is asserted.
`ceb_dout[31:0]`	Output	Write data for a write access.
`ceb_wr[3:0]`	Output	Indicates the configuration register access type (read or write). For writes, `ceb_wr[3:0]` also indicates the byte enables. The following encodings are defined: 4'b0000: Read

continued...

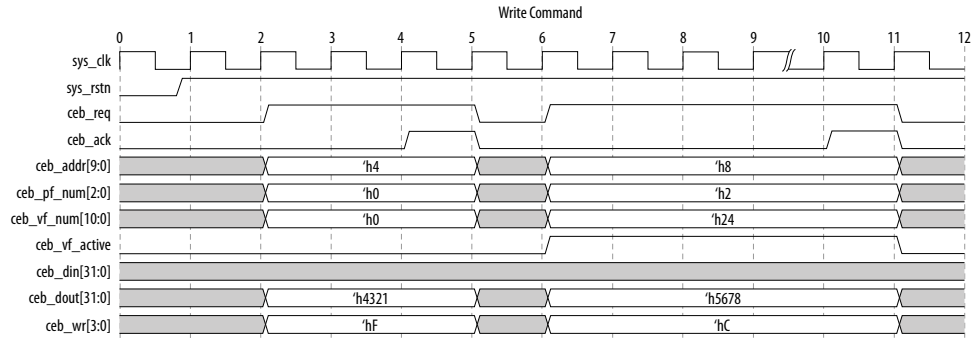
Signal	Direction	Description
		4'b0001: Write byte 0 4'b0010: Write byte 1 4'b0100: Write byte 2 4'b1000: Write byte 3 4'b1111: Write all bytes. Combinations of byte enables (for example,4'b0101) are also valid.

The figure below shows the timing diagram for 2 write commands.

The first command sends a write for all four bytes of the register located at address = 4. `ceb_vf_active` being low indicates this access is for a Physical Function.

The second command sends a write for byte3 and byte2 of the register located at address = 8. `ceb_vf_active` being high indicates this access is for a Virtual Function.

Figure 28. Write for a Physical Function Followed by a Write for a Virtual Function



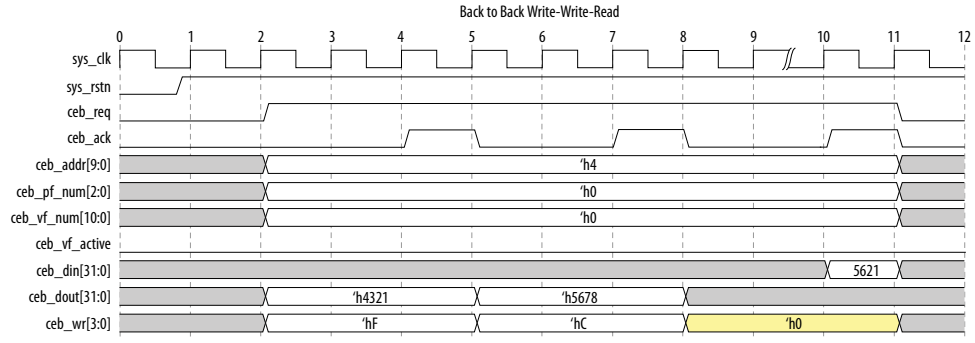
The figure below shows the timing diagram for back-to-back writes followed by a read.

The first command sends a write for all four bytes of the register located at address = 4.

The second command sends a write for byte3 and byte2 of the same register.

The third command sends a read for the same register. Note that the data returned is 5621. The upper two bytes were modified by the second write.

Figure 29. Back-to-back Writes Followed by a Read for a Physical Function

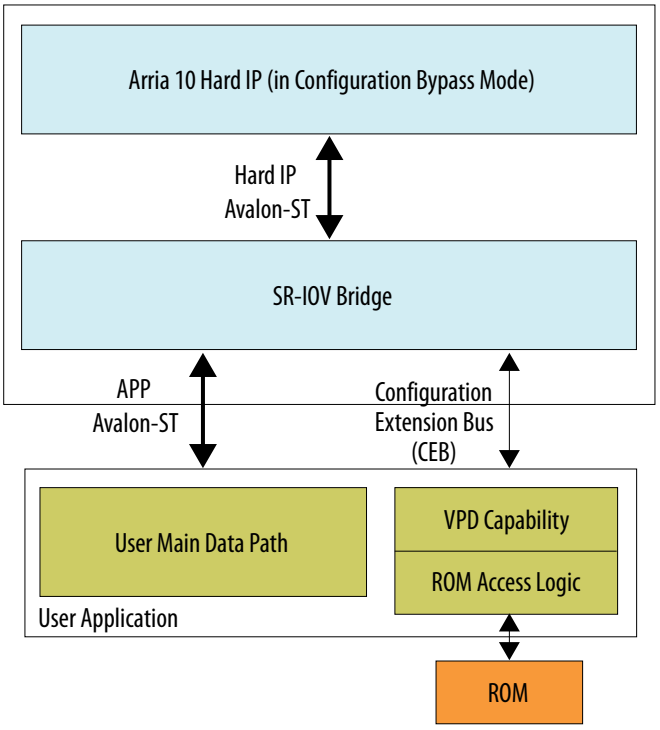


5.9.1. Vital Product Data (VPD) Capability

Vital Product Data (VPD) is information that uniquely identifies the hardware and, potentially, software elements of a system. The objective from a system point of view is to make this information available to the system owner and service personnel. VPD typically resides in a storage device (for example, a serial EEPROM) associated with the Function. Support of the VPD capability is optional. Users can build the VPD capability using the CEB interface. The figure below shows a high-level block diagram of the VPD implementation.

The VPD capability allows the host to access the ROM attached to the device.

Figure 30. VPD Implementation



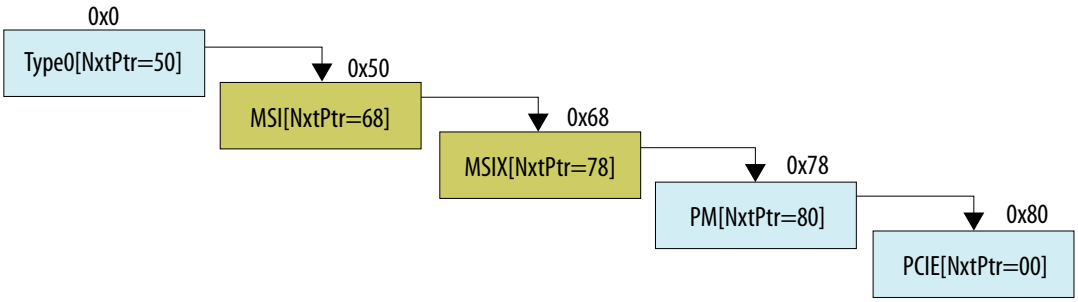
5.9.1.1. Determine the Pointer Address of an External Capability Register

The next pointer field of the last capability structure within the SR-IOV bridge is set by the External Capability Pointer parameter (refer to Table 3.3). Separate parameters are provided for the PCI Compatible region of the physical function (PF) and virtual function (VF) as well as for the PCIe Extended Capability region to point to the next capability in the user logic for the physical function and virtual function.

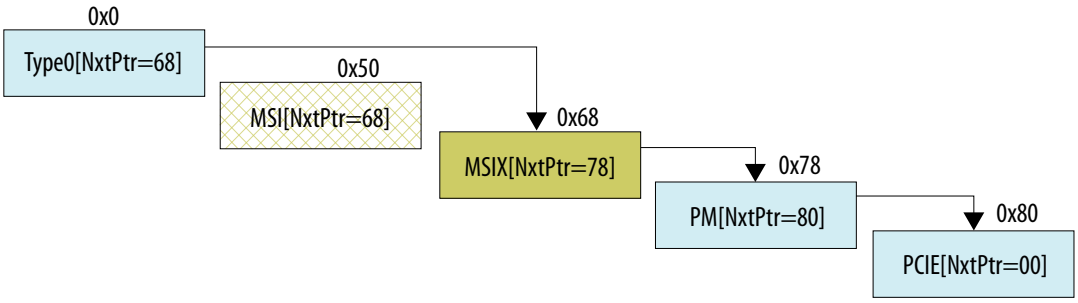
You can select an address from the available address space. Refer to Table 6.2 to determine the available address space to implement additional capabilities.

Most of the address space of the first 256 registers has been taken up by capabilities present in the SR-IOV bridge when all optional capabilities are enabled. If you wish to implement new capabilities in the first 256 registers, some of the optional capabilities in the SR-IOV bridge may need to be disabled.

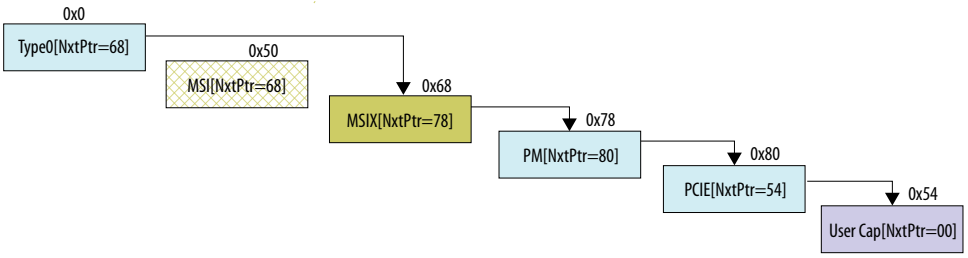
The figure below shows the default link list of the PCI Compatible region inside the SR-IOV bridge core. Capabilities like MSI and MSI-X are optional.



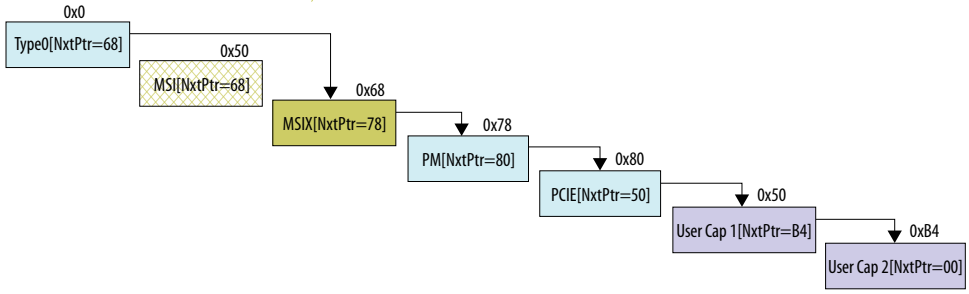
The figure below shows the link list when the MSI capability is disabled. The SR-IOV bridge automatically adjusts its next pointer when you disable any of the optional capabilities.



When the MSI capability is disabled, the address space available from 0x50 to 0x64 can now be used to implement user-specific capabilities using the CEB interface. As shown in the figure above, the last capability in the link is the PCI Express Capability Structure. The external capability pointer parameter will set the NxtPtr field of the PCI Express Capability Structure in this case. If you implement an additional capability at "0x54", it must set the external capability pointer parameter value to "0x54". This allows the PCI Express Capability Structure to point to "0x54" instead of the "null pointer". The link list will appear as shown in the figure below.



The reserve configuration register space 0x0B4:0x0FF can be used to implement additional capabilities as well. If the MSI capability is disabled in the SR-IOV bridge, the user capability can start with address 0x50 or address 0xB4. If you implement an additional capability at "0x50", it must set the external capability pointer parameter value to "0x50". This allows the PCIE capability to point to "0x50" instead of the "null pointer". The link list will appear as shown in the figure below.



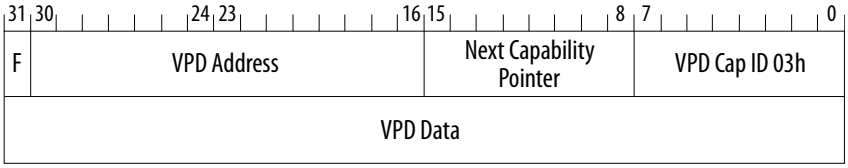
5.9.1.2. VPD Capability Implementation

Here are the required settings if you choose to implement the VPD capability in the MSI capability register space for the physical function:

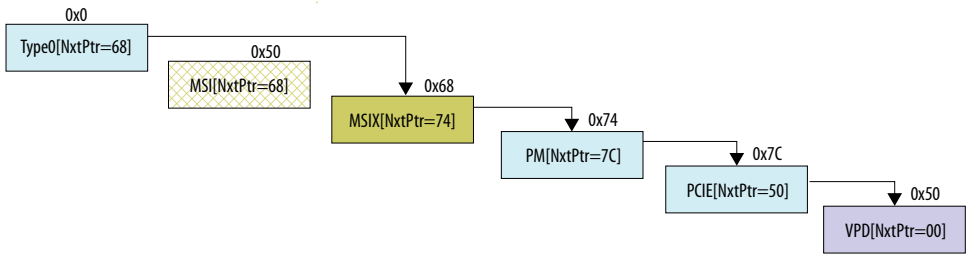
1. Make sure the MSI feature is disabled in the IP Parameter Editor GUI.
2. Check the **Enable PCI Express Extended Space (CEB)** option and specify the **CEB REQ to ACK latency (in Clock Cycles)**.
3. Set the **CEB PF External Standard Capability Pointer Address (DW address in Hex)** value to 0x14 and leave **CEB PF External Extended Capability Pointer Address (DW address in Hex)** in the default setting, 0x0.

Note: Table 6.2 uses byte addresses while the IP Parameter Editor GUI requires the external capability pointer address in dword format. To convert byte addresses to dword addresses, divide the byte address by 4.

Consequently, the VPD Capability register can be accessed by the host system starting from 0x50 (byte address) or 0x14 (dword address). The VPD capability structure is shown in the figure below.



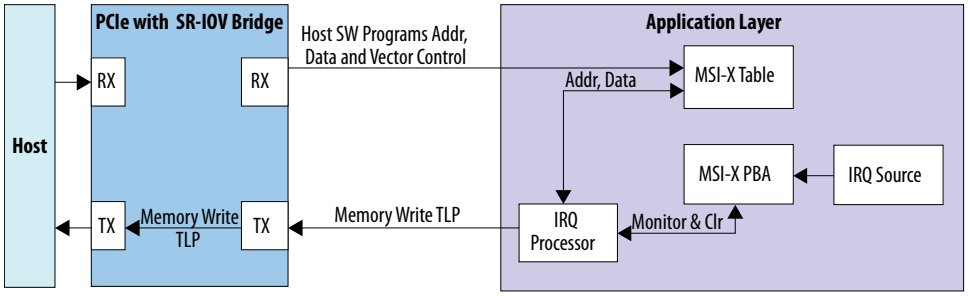
The figure below shows the capability link list for this implementation:



5.10. 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 31. 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 <*value read*> + *1*.

      The maximum table size is 2048 entries. Each 16-byte entry is divided in 4 fields as shown in the figure below. The MSI-X table can be accessed on any BAR configured. 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 32. 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 <n^{th}> entry using the following formula:

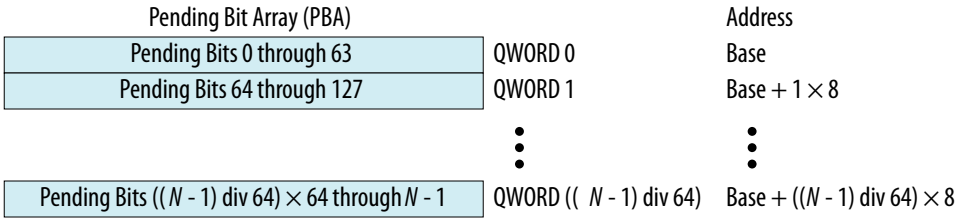
```
nth_address = base address[BAR] + 16<n>
```

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 <m^{th}> bit using the following formulas:

```
qword address = <PBA base addr> + 8(floor(<m>/64))
qword bit = <m> mod 64
```

Figure 33. MSI-X PBA Table



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

5.11. Control Shadow Interface

The Control Shadow interface provides access to the current settings of some of the VF Control Register fields in the PCI and PCI Express Configuration Spaces in the SR-IOV Bridge. Use this interface in one of two ways.

- To monitor specific VF registers for changes: When the Root Port performs a Configuration Write to any of the specified register fields provides the new values and the VF number. The monitor must copy the new register values as they appear on the interface and save them for future use.
- To monitor all VF registers for changes: Assert the `ctl_shdw_req_all` input to request a complete scan of the register fields being monitored for all active VFs. When `ctl_shdw_req_all` is asserted, the SR-IOV Bridge cycles through each VF and provides the current values of the specified register fields. If a Configuration Write occurs during a scan, the SR-IOV Bridge interrupts the scan and outputs the new settings for the targeted VF. It then resumes the scan, continuing sequentially from the VF that was updated.

Table 33. Control Shadow Interface

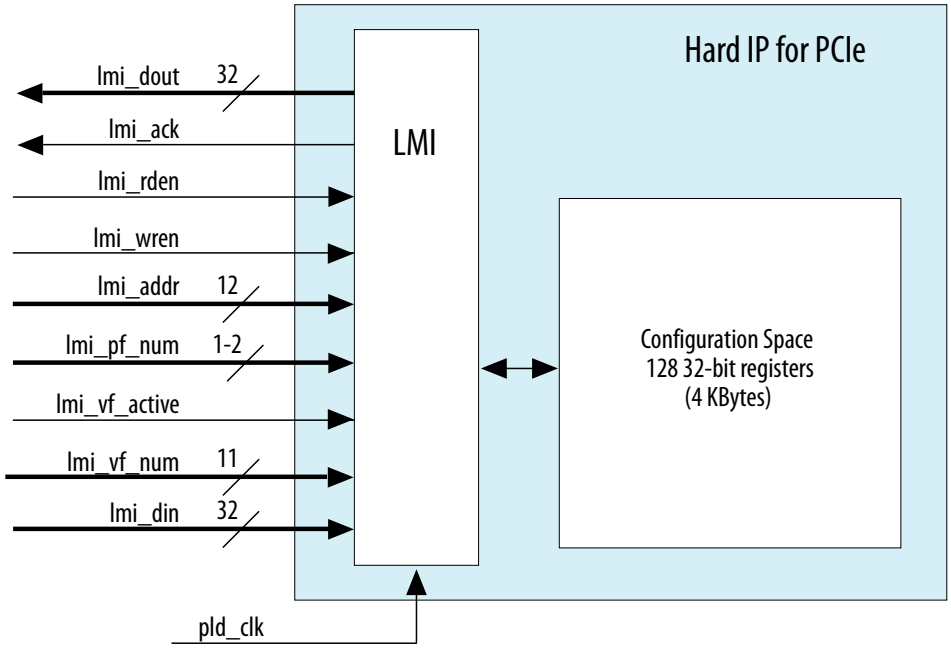
Use this to access the control register fields of the VFs.

Signal	Direction	Description
ctl_shdw_update	Output	The SR-IOV Bridge asserts this output for one clock cycle when there is an update to one or more of the register fields being monitored. The ctl_shdw_cfg outputs drive the new values. ctl_shdw_pf_num, ctl_shdw_vf_num, and ctl_shdw_vf_active identify the VF and its PF.
ctl_shdw_pf_num[<*n*>-1:0]	Output	Identifies the PF whose register settings are on the ctl_shdw_cfg outputs. When the Function is a VF, this input specifies the PF Number to which the VF is attached.
ctl_shdw _vf_active	Output	When asserted, indicates that the Function whose register settings are on the ctl_shdw_cfgoutputs is a VF. ctl_shdw_vf_num drives the VF number offset.
ctl_shdw_vf_num[10:0]	Output	Identifies the VF number offset of the VF whose register settings are on ctl_shdw_cfg outputs when ctl_shdw _vf_active is asserted, Its value ranges from *0*-<*n*>*-1*, where <*n*> is the number of VFs in the set of VFs attached to the associated PF.
ctl_shdw_cfg[6:0]	Output	When ctl_shdw_updateis asserted, this output provides the current settings of the register fields of the associated Function. The bit assignments are as follows: • [0]: Bus Master Enable field, bit[2] of the PCI Command Register • [1]: MSI-X Function Mask field, bit[14] of the MSI-X Message Control Register • [2]: MSIX Enable field, bit[15] of the MSIX Message Control Register • [4:3]: TPH ST Mode Select field, bits[1:0] of TPH Requester Control Register • [5]: TPH Requester Enable field, bit [8] of TPH Requester Control Register • [6]: Enable field, bit [15] of the ATS Control Register
ctl_shdw_req_all	Input	When asserted, requests a complete scan of the register fields being monitored for all active Functions. The SR-IOV Bridge cycles through each Function and provides the current settings of the register fields of each Function in sequence on ctl_shdw_cfg[6:0]. If a Configuration Write occurs during a scan, the SR-IOV Bridge interrupts the scan and outputs the new settings for the targeted VF. It then resumes the scan, continuing sequentially from the VF that was updated. The SR-IOV Bridge checks the state of ctl_shdw_req_all 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.

5.12. Local Management Interface (LMI) Signals

Use the LMI interface to write log error descriptor information in the TLP header log registers. The LMI access to other registers is intended for debugging, not normal operation.

Figure 34. Local Management Interface



The LMI provides access to many of the configuration registers of the Physical and Virtual Functions. You can read all PF and Configuration Registers. You can write fields designated as RW.

Table 34. LMI Interface

Signal	Direction	Description
lmi_dout[31:0]	Output	Data outputs. Valid when lmi_ack_apphas been asserted.
lmi_rden	Input	Read enable input.
lmi_wren	Input	Write enable input.
lmi_ack	Output	Acknowledgment for a read or write operation. The SR-IOV Bridge asserts this output for one cycle after it has completed the read or write operation. For read operations, the assertion of lmi_ack also indicates the presence of valid data on .
lmi_addr[11:0]	Input	Byte address of 32-bit configuration register. Bits [1:0] are not used.
lmi_pf_num_app[<*n*>-1:0]	Input	Specifies the Function number corresponding to the LMI access, Used only when the LMI access is to a Configuration register in the SR-IOV Bridge. When the Function is a VF, this input specifies the PF Number to which the VF is attached. <*n*> is the number of PFs.
lmi _vf_active	Input	Indicates that the Function to be accessed is a Virtual Function. When this input is asserted, the VF number offset of the Function must be provided on lmi_vf_num.
lmi _vf_num[<*n*>-1:0]	Input	When lmi _vf_active is asserted, this input identifies the VF number offset of the Function being accessed. Its value ranges from *0-(<n>-1)*, where <*n*> is the number of VFs in the set of VFs attached to the associated PF. <n> is the number of VFs.
lmi_din[31:0]	Input	Data inputs.

Figure 35. LMI Read

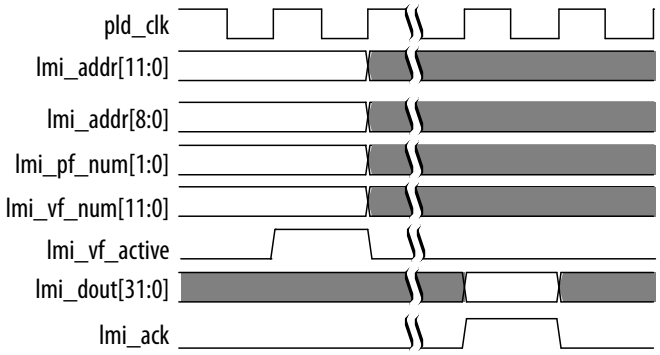
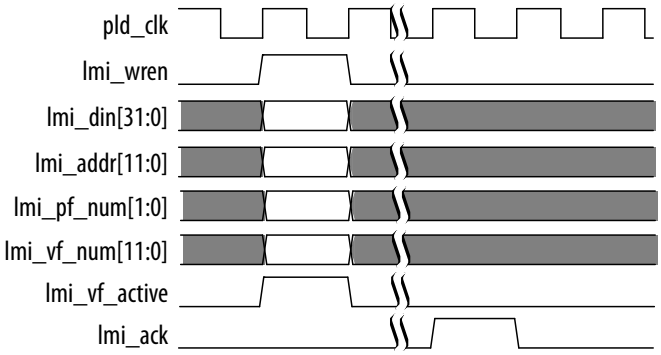


Figure 36. LMI Write

The following figure illustrates the LMI write. Only writable configuration bits are overwritten by this operation. Read-only bits are not affected..



5.13. Reset, Status, and Link Training Signals

Table 35. Reset Signals

Signal	Direction	Description
npor	Input	Active low reset signal. In the Intel hardware example designs, npor is the OR of pin_perst and local_rstn coming from the software Application Layer. If you do not drive a soft reset signal from the Application Layer, this signal must be derived from pin_perst. You cannot disable this signal. Resets the entire Intel Arria 10 Hard IP for PCI Express IP Core and transceiver. Asynchronous. When CvP is enabled, an embedded hard reset controller triggers after the internal status signal indicates that the periphery image is loaded. This embedded reset does not trigger off of pin_perst. In systems that use the hard reset controller, this signal is *edge*, *not level* sensitive; consequently, you cannot use a low value on this signal to hold custom logic in reset. For more information about the hard and soft reset controllers, refer to *Reset*.
pin_perst	Input	Active low reset from the PCIe reset pin of the device. It resets the datapath and control registers. This signal is required for Configuration via Protocol (CvP). For more information about CvP refer to *Configuration via Protocol (CvP)*. Intel Arria 10 devices have up to 4 instances of the Hard IP for PCI Express. Each instance has its own pin_perst signal. Intel Cyclone 10 GX have a signal instance of the Hard IP for PCI Express.

continued...

Signal	Direction	Description
		Every Intel Arria 10 device has 4 nPERST pins, even devices with fewer than 4 instances of the Hard IP for PCI Express. *You must connect the* `pin_perst` *of each Hard IP instance to the corresponding* nPERST *pin of the device.* These pins have the following locations: • `nPERSTL0`: bottom left Hard IP and CvP blocks • `nPERSTL1`: top left Hard IP block • `nPERSTR0`: bottom right Hard IP block • `nPERSTR1`: top right Hard IP block For example, if you are using the Hard IP instance in the bottom left corner of the device, you must connect `pin_perst` to `nPERSL0`. For maximum use of the Intel Arria 10 device, Intel recommends that you use the bottom left Hard IP first. This is the only location that supports CvP over a PCIe link. Refer to the *Intel Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines* or *Intel Cyclone 10 GX Device Family Pin Connection Guidelines*for more detailed information about these pins.

Figure 37. Reset and Link Training Timing Relationships

The following figure illustrates the timing relationship between `npor` and the LTSSM L0 state.

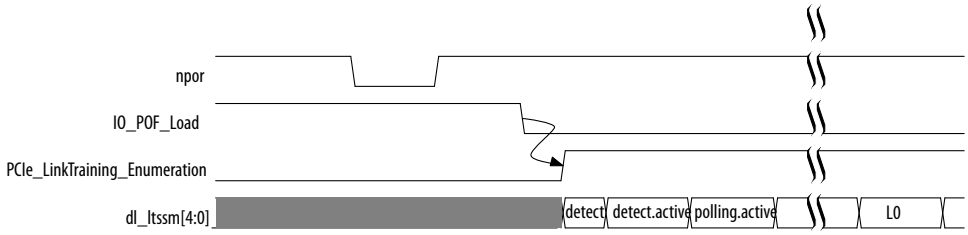


Table 36. Hard IP Reset Status Signals

Signal	Direction	Description
`pld_clk_inuse`	Output	When asserted, indicates that the Hard IP Transaction Layer is using the `pld_clk` as its clock and is ready for operation with the Application Layer. For reliable operation, hold the Application Layer in reset until `pld_clk_inuse` is asserted.
`pld_core_ready`	Input	When asserted, indicates that the Application Layer is ready for operation and is providing a stable clock to the `pld_clk` input. If the `coreclkout_hip` Hard IP output clock is sourcing the `pld_clk` Hard IP input, this input can be connected to the `serdes_pll_locked` output.
`reset_status`	Output	Active high reset status signal. When asserted, this signal indicates that the Hard IP clock is in reset. The `reset_status` signal is synchronous to the `pld_clk` clock and is deasserted only when the `npor` is deasserted and the Hard IP for PCI Express is not in reset (`reset_status_hip` = 0). You should use `reset_status` to drive the reset of your Application Layer. It resets the Hard IP at power-up, for hot reset and link down events.
`serdes_pll_locked`	Output	When asserted, indicates that the PLL that generates the `coreclkout_hip` clock signal is locked. In pipe simulation mode this signal is always asserted.
`testin_zero`	Output	When asserted, indicates accelerated initialization for simulation is active.

Table 37. Status and Link Training Signals

The following table describes additional signals related to the reset function for the including the `ltsssm_state[4:0]` bus that indicates the current link training state. These signals are not top-level signals of the Intel Arria 10 Hard IP for PCI Express IP Core with SR-IOV. They are listed here to assist in debugging link training issues.

Signal	Direction	Description
`cfg_par_err`	Output	Indicates that a parity error in a TLP routed to the internal Configuration Space. You must reset the Hard IP if this error occurs.
`derr_cor_ext_rcv`	Output	Indicates a corrected error in the RX buffer. This signal is for debug only. It is not valid until the RX buffer is filled with data. This is a pulse, not a level, signal. Internally, the pulse is generated with the 500 MHz clock. A pulse extender extends the signal so that the FPGA fabric running at 250 MHz can capture it. Because the error was corrected by the IP core, no Application Layer intervention is required.
`derr_cor_ext_rpl`	Output	Indicates a corrected ECC error in the retry buffer. This signal is for debug only. Because the error was corrected by the IP core, no Application Layer intervention is required. (4)
`derr_rpl`	Output	Indicates an uncorrectable error in the retry buffer. This signal is for debug only. (4)
`dlup`	Output	When asserted, indicates that the Hard IP block is in the Data Link Control and Management State Machine (DLCMSM) DL_Up state.
`dlup_exit`	Output	This signal is asserted low for one `pld_clk` cycle when the IP core exits the DLCMSM DL_Up state, indicating that the Data Link Layer has lost communication with the other end of the PCIe link and left the Up state. When this pulse is asserted, the Application Layer should generate an internal reset signal that is asserted for at least 32 cycles.
`ev128ns`	Output	Asserted every 128 ns to create a time base aligned activity.
`ev1us`	Output	Asserted every 1 µs to create a time base aligned activity.
`hotrst_exit`	Output	Hot reset exit. This signal is asserted for 1 clock cycle when the LTSSM exits the hot reset state. This signal should cause the Application Layer to be reset. This signal is active low. When this pulse is asserted, the Application Layer should generate an internal reset signal that is asserted for at least 32 cycles.
`int_status[3:0]`	Output	These signals drive legacy interrupts to the Application Layer as follows: • int_status[0]: interrupt signal A • int_status[1]: interrupt signal B • int_status[2]: interrupt signal C • int_status[3]: interrupt signal D
`l2_exit`	Output	L2 exit. This signal is active low and otherwise remains high. It is asserted for one cycle (changing value from 1 to 0 and back to 1) after the LTSSM transitions from l2.idle to detect. When this pulse is asserted, the Application Layer should generate an internal reset signal that is asserted for at least 32 cycles.
`lane_act[3:0]`	Output	Lane Active Mode: This signal indicates the number of lanes that configured during link training. The following encodings are defined: • 4'b0001: 1 lane • 4'b0010: 2 lanes • 4'b0100: 4 lanes • 4'b1000: 8 lanes

continued...

(4) Debug signals are not rigorously verified and should only be used to observe behavior. Debug signals should not be used to drive logic custom logic.

Signal	Direction	Description
ltssmstate[4:0]	Output	LTSSM state: The LTSSM state machine encoding defines the following states: • 00000: Detect.Quiet • 00001: Detect.Active • 00010: Polling.Active • 00011: Polling.Compliance • 00100: Polling.Configuration • 00101: Polling.Speed • 00110: config.Linkwidthstart • 00111: Config.Linkaccept • 01000: Config.Lanenumaccept • 01001: Config.Lanenumwait • 01010: Config.Complete • 01011: Config.Idle • 01100: Recovery.Rcvlock • 01101: Recovery.Rcvconfig • 01110: Recovery.Idle • 01111: L0 • 10000: Disable • 10001: Loopback.Entry • 10010: Loopback.Active • 10011: Loopback.Exit • 10100: Hot.Reset • 10101: L0s • 11001: L2.transmit.Wake • 11010: Recovery.Speed • 11011: Recovery.Equalization, Phase 0 • 11100: Recovery.Equalization, Phase 1 • 11101: Recovery.Equalization, Phase 2 • 11110: Recovery.Equalization, Phase 3 • 11111: Recovery.Equalization, Done
rx_par_err	Output	When asserted for a single cycle, indicates that a parity error was detected in a TLP at the input of the RX buffer. The SR-IOV bridge drives this signal to the Application Layer without taking any action. 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[1:0]	Output	When asserted for a single cycle, indicates a parity error during TX TLP transmission. The SR-IOV bridge drives this signal to the Application Layer without taking any action. The following encodings are defined: • 2'b10: A parity error was detected by the TX Transaction Layer. • 2'b01: Some time later, the parity error is detected by the TX Data Link Layer which drives 2'b01 to indicate the error. Intel recommends resetting the Intel Arria 10 Hard IP for PCI Express when this error is detected. Contact Intel if resetting becomes unworkable. *Note:* Not all simulation models assert the Transaction Layer error bit in conjunction with the Data Link Layer error bit.

continued...

Signal	Direction	Description
ko_cpl_spc_data[11:0]	Output	The Application Layer can use this signal to build circuitry to prevent RX buffer overflow for completion data. Endpoints must advertise infinite space for completion data; however, RX buffer space is finite. ko_cpl_spc_data is a static signal that reflects the total number of 16 byte completion data units that can be stored in the completion RX buffer.
ko_cpl_spc_header]7:0]	Output	The Application Layer can use this signal to build circuitry to prevent RX buffer overflow for completion headers. Endpoints must advertise infinite space for completion headers; however, RX buffer space is finite. ko_cpl_spc_header is a static signal that indicates the total number of completion headers that can be stored in the RX buffer.
rxfc_cplbuf_ovf	Output	When asserted, indicates RX Posted Completion buffer overflow.

Related Information

- Reset and Clocks on page 102
- PCI Express Card Electromechanical Specification 2.0
- Intel Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines
  For information about connecting pins on the PCB including required resistor values and voltages.
- Intel Cyclone 10 GX Device Family Pin Connection Guidelines
  For information about connecting pins on the PCB including required resistor values and voltages.

5.14. Hard IP Reconfiguration Interface

The Hard IP reconfiguration interface is an Avalon-MM slave interface with a 10-bit address and 16-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.

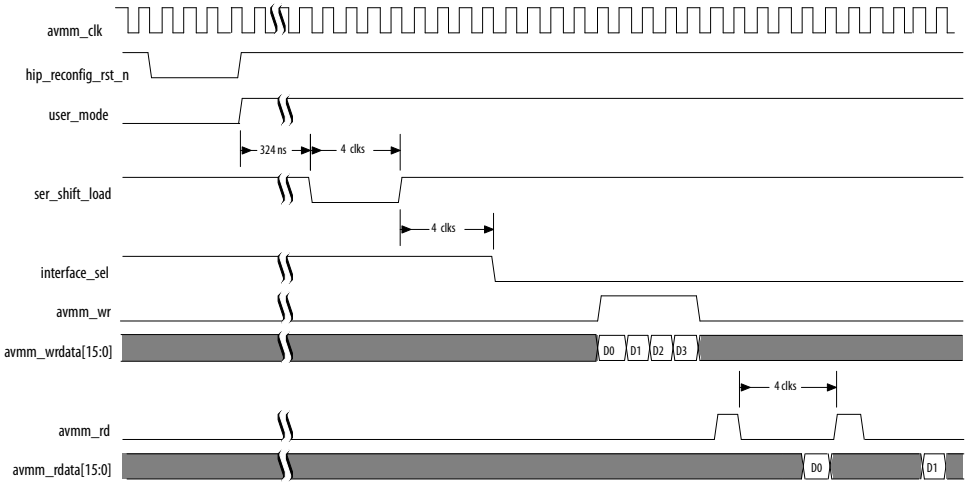
Table 38. Hard IP Reconfiguration Signals

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. Resets all of the dynamic reconfiguration registers to their default values as described in *Hard IP Reconfiguration Registers*.
hip_reconfig_address[9:0]	Input	The 10-bit reconfiguration address.
hip_reconfig_read	Input	Read signal. This interface is not pipelined. You must wait for the return of the hip_reconfig_readdata[15:0] from the current read before starting another read operation.
hip_reconfig_readdata[15:0]	Output	16-bit read data. hip_reconfig_readdata[15:0] is valid on the third cycle after the assertion of hip_reconfig_read.
hip_reconfig_write	Input	Write signal.
hip_reconfig_writedata[15:0]	Input	16-bit write model.

continued...

Signal	Direction	Description
hip_reconfig_byte_en[1:0]	Input	Byte enables, currently unused.
ser_shift_load	Input	You must toggle this signal once after changing to user mode before the first access to read-only registers. This signal should remain asserted for a minimum of 324 ns after switching to user mode.
interface_sel	Input	A selector which must be asserted when performing dynamic reconfiguration. Drive this signal low 4 clock cycles after the release of ser_shif t_load.

Figure 38. Hard IP Reconfiguration Bus Timing of Read-Only Registers



For a detailed description of the Avalon-MM protocol, refer to the *Avalon Memory Mapped Interfaces* chapter in the *Avalon Interface Specifications*.

Related Information

Avalon Interface Specifications
For information about the Avalon-MM interfaces to implement read and write interfaces for master and slave components.

5.15. Serial Data Signals

This differential, serial interface is the physical link between a Root Port and an Endpoint.

The Intel Cyclone 10 GX PCIe IP Core supports 1, 2, or 4 lanes. Each lane includes a TX and RX differential pair. Data is striped across all available lanes.

The Intel Arria 10 PCIe IP Core supports 1, 2, 4 or 8 lanes. Each lane includes a TX and RX differential pair. Data is striped across all available lanes.

Table 39. 1-Bit Interface Signals

In the following table <*n*> is the number of lanes.

Signal	Direction	Description
tx_out[<*n*>-*1*:0]	Output	Transmit output. These signals are the serial outputs of lanes <*n*>-*1*–0.
rx_in[<*n*>-*1*:0]	Input	Receive input. These signals are the serial inputs of lanes <*n*>-*1*–0.

Refer to *Pin-out Files for Intel Devices* for pin-out tables for all Intel devices in **.pdf**, **.txt**, and **.xls** formats.

Transceiver channels are arranged in groups of six. For GX devices, the lowest six channels on the left side of the device are labeled GXB_L0, the next group is GXB_L1, and so on. Channels on the right side of the device are labeled GXB_R0, GXB_R1, and so on. Be sure to connect the Hard IP for PCI Express on the left side of the device to appropriate channels on the left side of the device, as specified in the *Pin-out Files for Intel Devices*.

Related Information

- Hard IP Block Placement In Intel Arria 10 Devices on page 30
- Pin-out Files for Intel Devices

5.16. Test Signals

Table 40. Test Interface Signals

The test_in bus provides run-time control and monitoring of the internal state of the IP core.

Signal	Direction	Description
test_in[31:0]	Input	The bits of the test_in bus have the following definitions: • [0]: Simulation mode. This signal can be set to 1 to accelerate initialization by reducing the value of many initialization counters. • [1]: Reserved. Must be set to 1'b0. • [2]: Descramble mode disable. This signal must be set to 1 during initialization in order to disable data scrambling. You can use this bit in simulation for Gen1 and Gen2 Endpoints and Root Ports to observe descrambled data on the link. Descrambled data cannot be used in open systems because the link partner typically scrambles the data. • [4:3]: Reserved. Must be set to 2'b01. • [5]: Compliance test mode. Disable/force compliance mode. When set, prevents the LTSSM from entering compliance mode. Toggling this bit controls the entry and exit from the compliance state, enabling the transmission of Gen1, Gen2 and Gen3 compliance patterns. • [6]: Forces entry to compliance mode when a timeout is reached in the polling.active state and not all lanes have detected their exit condition. • [7]: Disable low power state negotiation. Intel recommends setting this bit. • [31:8]: Reserved. Set to all 0s.
simu_pipe_mode	Input	When 1'b1, counter values are reduced to speed simulation.

5.17. PIPE Interface Signals

These PIPE signals are available for Gen1, Gen2, and Gen3 variants so that you can simulate using either the serial or the PIPE interface. Note that Intel Arria 10 and Intel Cyclone 10 GX devices do not support the Gen3 PIPE interface. Simulation is faster using the PIPE interface because the PIPE simulation bypasses the SERDES model. By

default, the PIPE interface is 8 bits for Gen1 and Gen2 and 32 bits for Gen3. You can use the PIPE interface for simulation even though your actual design includes a serial interface to the internal transceivers. However, it is not possible to use the Hard IP PIPE interface in hardware, including probing these signals using Signal Tap Embedded Logic Analyzer. These signals are not top-level signals of the Hard IP. They are listed here to assist in debugging link training issues.

Note: The Intel Root Port BFM bypasses Gen3 Phase 2 and Phase 3 Equalization. However, Gen3 variants can perform Phase 2 and Phase 3 equalization if instructed by a third-party BFM.

Table 41. PIPE Interface Signals

In the following table, signals that include lane number 0 also exist for lanes 1-7. These signals are for simulation only. For Quartus Prime software compilation, these pipe signals can be left floating. In Platform Designer, the signals that are part of the PIPE interface have the prefix, *hip_pipe*. The signals which are included to simulate the PIPE interface have the prefix, *hip_pipe_sim_pipe*

Signal	Direction	Description
`currentcoeff0[17:0]`	Output	For Gen3, indicates the coefficients to be used by the transmitter. The 18 bits specify the following coefficients: • [5:0]: C-1 • [11:6]: C0 • [17:12]: C+1
`currentrxpreset0[2:0]`	Output	For Gen3 designs, specifies the current preset.
`eidleinfersel0[2:0]`	Output	Electrical idle entry inference mechanism selection. The following encodings are defined: • 3'b0xx: Electrical Idle Inference not required in current LTSSM state • 3'b100: Absence of COM/SKP Ordered Set the in 128 us window for Gen1 or Gen2 • 3'b101: Absence of TS1/TS2 Ordered Set in a 1280 UI interval for Gen1 or Gen2 • 3'b110: Absence of Electrical Idle Exit in 2000 UI interval for Gen1 and 16000 UI interval for Gen2 • 3'b111: Absence of Electrical idle exit in 128 us window for Gen1
`phystatus0`	Input	PHY status <*n*>. This signal communicates completion of several PHY requests.
`powerdown0[1:0]`	Output	Power down <*n*>. This signal requests the PHY to change its power state to the specified state (P0, P0s, P1, or P2).
`rate[1:0]`	Output	Controls the link signaling rate. The following encodings are defined: • 2'b00: Gen1 • 2'b01: Gen2 • 2'b10: Gen3 • 2'b11: Reserved
`rxblkst0`	Input	For Gen3 operation, indicates the start of a block in the receive direction.
`rxdata0[31:0]`	Input	Receive data. This bus receives data on lane <*n*>.
`rxdatak0[3:0]`	Input	Data/Control bits for the symbols of receive data. 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.
`rxelecidle0`	Input	Receive electrical idle <*n*>. When asserted, indicates detection of an electrical idle.
`rxpolarity0`	Output	Receive polarity <*n*>. This signal instructs the PHY layer to invert the polarity of the 8B/10B receiver decoding block.

continued...

Signal	Direction	Description
rxstatus0[2:0]	Input	Receive status <*n*>. This signal encodes receive status and error codes for the receive data stream and receiver detection.
rxvalid0	Input	Receive valid <*n*>. This symbol indicates symbol lock and valid data on rxdata<*n*> and rxdatak <*n*>.
sim_pipe_ltssmstate0[4:0]	Input and Output	LTSSM state: The LTSSM state machine encoding defines the following states: • 5'b00000: Detect.Quiet • 5'b 00001: Detect.Active • 5'b00010: Polling.Active • 5'b 00011: Polling.Compliance • 5'b 00100: Polling.Configuration • 5'b00101: Polling.Speed • 5'b00110: Config.LinkwidthsStart • 5'b 00111: Config.Linkaccept • 5'b 01000: Config.Lanenumaccept • 5'b01001: Config.Lanenumwait • 5'b01010: Config.Complete • 5'b 01011: Config.Idle • 5'b01100: Recovery.Rcvlock • 5'b01101: Recovery.Rcvconfig • 5'b01110: Recovery.Idle • 5'b 01111: L0 • 5'b10000: Disable • 5'b10001: Loopback.Entry • 5'b10010: Loopback.Active • 5'b10011: Loopback.Exit • 5'b10100: Hot.Reset • 5'b10101: L0s • 5'b11001: L2.transmit.Wake • 5'b11010: Recovery.Speed • 5'b11011: Recovery.Equalization, Phase 0 • 5'b11100: Recovery.Equalization, Phase 1 • 5'b11101: Recovery.Equalization, Phase 2 • 5'b11110: Recovery.Equalization, Phase 3 • 5'b11111: Recovery.Equalization, Done
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]	Input	Specifies the data rate. The 2-bit encodings have the following meanings: • 2'b00: Gen1 rate (2.5 Gbps) • 2'b01: Gen2 rate (5.0 Gbps) • 2'b1X: Gen3 rate (8.0 Gbps)
txblkst		For Gen3 operation, indicates the start of a block in the transmit direction.
txcompl0	Output	Transmit compliance <*n*>. This signal forces the running disparity to negative in compliance mode (negative COM character).
txdata0[31:0]	Output	Transmit data. This bus transmits data on lane <*n*>.
txdatak0[3:0]	Output	Transmit data control <*n*>. This signal serves as the control bit for txdata <*n*>. 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.

continued...

Signal	Direction	Description
txdataskip0	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: • 1'b0: TX data is invalid • 1'b1: TX data is valid
txdeemph0	Output	Transmit de-emphasis selection. The value for this signal is set 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.
txdetectrx0	Output	Transmit detect receive <*n*>. This signal tells the PHY layer to start a receive detection operation or to begin loopback.
txelecidle0	Output	Transmit electrical idle <*n*>. This signal forces the TX output to electrical idle.
txmargin0[2:0]	Output	Transmit V_{OD} margin selection. The value for this signal is based on the value from the `Link Control 2 Register`. Available for simulation only.
txswing0	Output	When asserted, indicates full swing for the transmitter voltage. When deasserted indicates half swing.
txsynchd0[1:0]	Output	For Gen3 operation, specifies the block type. The following encodings are defined: • 2'b01: Ordered Set Block • 2'b10: Data Block

5.18. Intel Arria 10 Development Kit Conduit Interface

The Intel Arria 10 Development Kit conduit interface signals are optional signals that allow you to connect your design to the Intel Arria 10 FPGA Development Kit. Enable this interface by selecting **Enable Intel Arria 10 FPGA Development Kit connection** on the **Configuration, Debug, and Extension Options** tab of the component GUI. The `devkit_status` output port includes signals useful for debugging.

Table 42. The Intel Arria 10 Development Kit Conduit Interface

Signal Name	Direction	Description
devkit_status[255:0]	Output	The `devkit_status[255:0]` bus comprises the following status signals : • `devkit_status[1:0]`: `current_speed` • `devkit_status[2]`: `derr_cor_ext_rcv` • `devkit_status[3]`: `derr_cor_ext_rpl` • `devkit_status[4]`: `derr_err` • `devkit_status[5]`: `rx_par_err` • `devkit_status[7:6]`: `tx_par_err` • `devkit_status[8]`: `cfg_par_err` • `devkit_status[9]`: `dlup` • `devkit_status[10]`: `dlup_exit` • `devkit_status[11]`: `ev128ns` • `devkit_status[12]`: `ev1us` • `devkit_status[13]`: `hotrst_exit` • `devkit_status[17:14]`: `int_status[3:0]` • `devkit_status[18]`: `l2_exit` • `devkit_status[22:19]`: `lane_act[3:0]`

continued...

Signal Name	Direction	Description
		• devkit_status[27:23]: ltssmstate[4:0] • devkit_status[35:28]: ko_cpl_spc_header[7:0] • devkit_status[47:36]: ko_cpl_spc_data[11:0] • devkit_status[48]: rxfc_cplbuf_ovf • devkit_status[49]: reset_status • devkit_status[255:50]: Reserved
devkit_ctrl[255:0]	Input	The devkit_ctrl[255:0] bus comprises the following status signals. You can optionally connect these pins to an on-board switch for PCI-SIG compliance testing, such as bypass compliance testing. • devkit_ctrl[0]:test_in[0] is typically set to 1'b0 • devkit_ctrl[4:1]:test_in[4:1] is typically set to 4'b0100 • devkit_ctrl[6:5]:test_in[6:5] is typically set to 2'b01 • devkit_ctrl[31:7]:test_in[31:7] is typically set to 25'h3 • devkit_ctrl[63:32]:is typically set to 32'b0 • devkit_ctrl[255:64]:is typically set to 192'b0

6. Registers

6.1. Addresses for Physical and Virtual Functions

The SR-IOV Bridge implements the PCI and PCI Express Configuration Spaces for up to eight Physical Functions and up to 2048 Virtual Functions in soft logic.

The SR-IOV bridge creates a static address map to match the number of PF and VFs specified in the component GUI. For systems using multiple PFs and ARI, the PFs are stored sequentially so that the offset values are contiguous. For systems including both PFs and VFs, the PFs are stored sequentially, followed by the VFs. The offset for the VFs associated with each PF are also contiguous. You cannot change the stride and offset values. For example, in a system with 4 PFs, the VFs for PF0, start at address 4 and continue to N_0 *+3*, where N_0 is the number of VFs attached to PF0. The VFs for PF1 begin at N_0 *+4* and continue to *(*N_0 + N_1*+3)*, and so on.

The SR-IOV bridge provides component-specific Avalon-ST interface input signals to identify the PF and VF for TX TLPs. When the requestor is a PF, the Application Layer should drive the PF number on `tx_st_pf_num`. When the requestor is a VF, the Application Layer should drive the PF number on `tx_st_pf_num` and VF number index on `tx_st_vf_num`. The SR-IOV bridge samples these numbers when `tx_st_sop` and `tx_st_valid` are both asserted. The SR-IOV bridge combines this information with the captured bus numbers and VF offset and stride settings of the PF, to construct the RID and insert it TLP header.

In the following tables, N_0, N_1, N_2, and N_3 are the number of VFs attached to PF 0, 1, 2, and 3, respectively.

Figure 39. Function Address Map With no SR-IOV and ARI, 1 – 4 PFs

Address (Decimal)	Function
0	Physical Function 0
1	Physical Function 1
2	Physical Function 2
3	Physical Function 3

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Figure 40. Function Address Map With Single PF and SR-IOV

Address (Decimal)	Function
0	Physical Function 0
1	PF 0, VF 0
2	PF 0, VF 1
• • •	
N_0	PF 0, VF N_0 - 1

Figure 41. Function Address Map With 2 PFs and SR-IOV

Address (Decimal)	Function
0	Physical Function 0
1	Physical Function 1
2	PF 0, VF 0
3	PF 0, VF 1
• • •	
$N_0 + 1$	PF 0, VF N_0 - 1
$N_0 + 2$	PF 1, VF 0
$N_0 + 3$	PF 1, VF 1
• • •	
$N_0 + N_1 + 1$	PF 1, VF N_1 - 1

Figure 42. Function Address Map With 3 PFs and SR-IOV

Address (Decimal)	Function
0	Physical Function 0
1	Physical Function 1
2	Physical Function 2
3	PF 0, VF 0
4	PF 0, VF 1
• • •	
$N_0 + 2$	PF 0, VF N_0 - 1
$N_0 + 3$	PF 1, VF 0
$N_0 + 4$	PF 1, VF 1
• • •	
$N_0 + N_1 + 2$	PF 1, VF N_1 - 1
$N_0 + N_1 + 3$	PF 2, VF 0
$N_0 + N_1 + 4$	PF 2, VF 1
• • •	
$N_0 + N_1 + N_2 + 2$	PF 2, VF N_2 - 1

Figure 43. Function Address Map With 4 PFs and SR-IOV

Address (Decimal)	Function
0	Physical Function 0
1	Physical Function 1
2	Physical Function 2
3	Physical Function 3
4	PF 0, VF 0
5	PF 0, VF 1
•••	
$N_0 + 3$	PF 0, VF N_0 - 1
$N_0 + 4$	PF 1, VF 0
$N_0 + 5$	PF 1, VF 1
•••	
$N_0 + N_1 + 3$	PF 1, VF N_1 - 1
$N_0 + N_1 + 4$	PF 2, VF 0
$N_0 + N_1 + 5$	PF 2, VF 1
•••	
$N_0 + N_1 + N_2 + 3$	PF 2, VF N_2 - 1
$N_0 + N_1 + N_2 + 4$	PF 3, VF 0
$N_0 + N_1 + N_2 + 5$	PF 3, VF 1
•••	
$N_0 + N_1 + N_2 + N_3 + 3$	PF 3, VF N_3 - 1

6.2. Correspondence between Configuration Space Registers and the PCIe Specification

Table 43. Configuration Space Registers for a Physical Function

Byte Address	SR-IOV Bridge Configuration Space Register	Corresponding Section in PCIe Specification
0x000:0x03C	PCI Header Type 0 Configuration Registers	Type 0 Configuration Space Header
0x040:0x04C	Reserved	N/A
0x050:0x064	MSI Capability Structure	MSI Capability Structure
0x068:0x070	MSI-X Capability Structure	MSI-X Capability Structure
0x074	Reserved	N/A
0x078:0x07C	Power Management Capability Structure	PCI Power Management Capability Structure
0x080:0x0B0	PCI Express Capability Structure	PCI Express Capability Structure
0x0B4:0x0FF	Reserved	N/A
0x100	ARI Enhanced Capability Header	PCI Express Extended Capability ID for AER and next capability pointer.
0x104:0x128	Advanced Error Reporting AER	Advanced Error Reporting Capability
0x12C:0x15C	Reserved	N/A
0x160:0x164	Alternative RID (ARI) Capability Structure	PCI Express Extended Capability ID for ARI and next capability pointer
0x168:0x19C	Reserved	N/A
0x200:0x23C	Single-Root I/O Virtualization (SR-IOV) Capability Structure	SR-IOV Extended Capability Header in *Single Root I/O Virtualization and Sharing Specification, Rev. 1.1*
0x240:0x2FC	Reserved	N/A
0x300:0x308	Transaction Processing Hints (TPH) Requester Capability Structure	TLP Processing Hints (TPH)
0x30C:0x2BC	Reserved.	N/A
0x3C0:0x3C4	Address Translation Services (ATS) Capability Structure	Address Translation Services Extended Capability (ATS) in *Single Root I/O Virtualization and Sharing Specification, Rev. 1.1*
0x3C8:0xFFF	Reserved	N/A

Related Information

- PCI Express Base Specification 3.0
- Single Root I/O Virtualization and Sharing Specification Revision 1.1

6.3. PCI and PCI Express Configuration Space Registers

6.3.1. Type 0 Configuration Space Registers

Figure 44. Type 0 Configuration Space Registers - Byte Address Offsets and Layout

Endpoints store configuration data in the Type 0 Configuration Space.

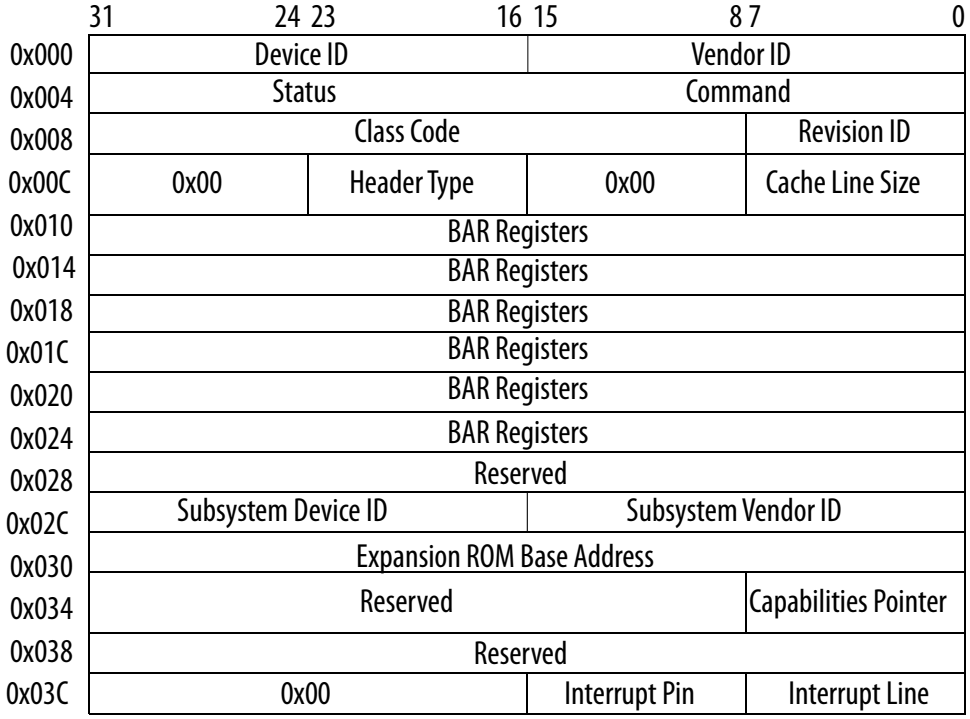


Table 44. Correspondence Configuration Space Capability Structures and PCIe Base Specification Description

The following table lists the appropriate section of the *PCI Express Base Specification* that describes these registers. Refer to the *PCI Express Base Specification* for more information.

Byte Address		
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	0x00 Header Type 0x00 Cache Line Size	Type 0 Configuration Space Header
0x010	Base Address 0	Base Address Registers (Offset 10h - 24h)
0x014	Base Address 1	Base Address Registers (Offset 10h - 24h)
0x018	Base Address 2	Base Address Registers (Offset 10h - 24h)
0x01C	Base Address 3	Base Address Registers (Offset 10h - 24h)
0x020	Base Address 4	Base Address Registers (Offset 10h - 24h)
0x024	Base Address 5	Base Address Registers (Offset 10h - 24h)
0x028	Reserved	

continued...

Byte Address		
0x02C	Subsystem Device ID Subsystem Vendor ID	Type 0 Configuration Space Header
0x030	Reserved	
0x034	Capabilities PTR	Type 0 Configuration Space Header
0x038	Reserved	Type 0 Configuration Space Header
0x03C	0x00 Interrupt Pin Interrupt Line	Type 0 Configuration Space Header

6.3.2. PCI and PCI Express Configuration Space Register Content

For comprehensive information about these registers, refer to Chapter 7 of the *PCI Express Base Specification Revision 3.0*.

Related Information

PCI Express Base Specification Revision 3.0

6.3.3. Interrupt Line and Interrupt Pin Register

These registers are used only when you configure the Physical Function (PF) to support PCI legacy interrupts. The following sequence of events implements a legacy interrupt:

1. A rising edge on `app_intx_req` indicates the assertion of the corresponding legacy interrupt from the client.
2. In response, the PF drives `Assert_INTx` to activate a legacy interrupt.
3. A falling edge on `app_int_sts_x` indicates the deassertion of the corresponding legacy interrupt from the client.
4. In response, the PF sends `Deassert_INTx` to deactivate the legacy interrupt.

The `Interrupt Pin` register specifies the interrupt input used to signal interrupts. The PFs may be configured with separate interrupt pins. Or, both PFs may share a common interrupt pin. You configure the `Interrupt Pin` register in Platform Designer.

The `Interrupt Line` register specifies the interrupt controller (IRQ0–IRQ15) input of the in the Root Port activated by each `Assert_INTx` message. You configure the `Interrupt Line` register in Platform Designer.

Table 45. Interrupt Line and Interrupt Pin Register -0x03C

Bit Location	Description	Default Value	Access
[15:11]	Not implemented	0	RO
[10:8]	Interrupt Pin register. When legacy interrupts are enabled, specifies the pin this function uses to signal an interrupt . The following encodings are defined: • 3'b001: INTA_IN • 3'b010: INTB_IN • 3'b011: INTC_IN • 3'b100: INTD_IN	Set in Platform Designer	RO
[7:0]	Interrupt Line register. Identifies the interrupt controller IRQx input of the Root Port that is activated by this function's interrupt. The following encodings are defined: • 6'h000000: IRQ0 • 6'h000001: IRQ1 • 6'h000002: IRQ2 • ... • 6'h0000FF: unknown or not connected	Set in Platform Designer	RO

Related Information

PCI Local Bus Specification 3.0

6.4. MSI Registers

Figure 45. MSI Register Byte Address Offsets and Layout

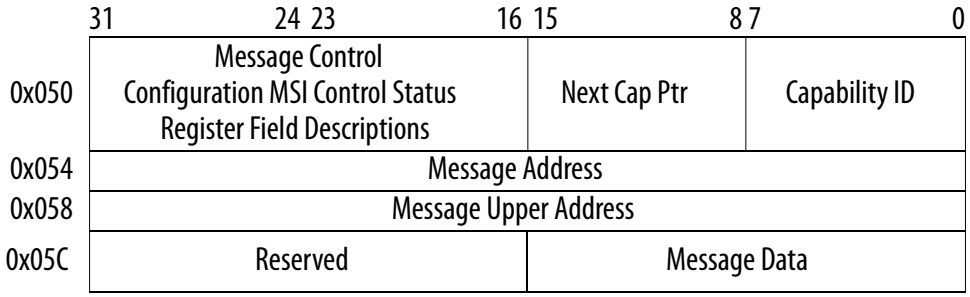


Table 46. MSI Control Register - 0x050

Bits	Register Description	Default Value	Access
[31:25]	Not implemented	0	RO
[24]	Per-Vector Masking Capable. This bit is hardwired to 1. The design always supports per-vector masking of MSI interrupts.	1	RO
[23]	64-bit Addressing Capable. When set, the device is capable of using 64-bit addresses for MSI interrupts.	Set in Platform Designer	RO
[22:20]	Multiple Message Enable. This field defines the number of interrupt vectors for this function. The following encodings are defined: • 3'b000: 1 vector • 3'b001: 2 vectors • 3'b010: 4 vectors	0	RW

continued...

Bits	Register Description	Default Value	Access
	• 3'b011: 8 vectors • 3'b100: 16 vectors • 3'b101: 32 vectors The `Multiple Message Capable` field specifies the maximum value allowed.		
[19:17]	Multiple Message Capable. Defines the maximum number of interrupt vectors the function is capable of supporting. The following encodings are defined: • 3'b000: 1 vector • 3'b001: 2 vectors • 3'b010: 4 vectors • 3'b011: 8 vectors • 3'b100: 16 vectors • 3'b101: 32 vectors	Set inPlatform Designer	RO
[16]	MSI Enable. This bit must be set to enable the MSI interrupt generation.	0	RW
[15:8]	Next Capability Pointer. Points to either MSI-X or Power Management Capability.	0x68 or 0x78	RO
[7:0]	Capability ID. PCI-SIG assigns this value.	0x05	RO

Table 47. MSI Message Address Registers - 0x054 and 0x058

Bits	Register Description	Default Value	Access
[1:0]	The two least significant bits of the memory address. These are hardwired to 0 to align the memory address on a Dword boundary.	0	RO
[31:2]	Lower address for the MSI interrupt.	0	RW
[31:0]	Upper 32 bits of the 64-bit address to be used for the MSI interrupt. If the `64-bit Addressing Capable` bit in the `MSI Control` register is set to 1, this value is concatenated with the lower 32-bits to form the memory address for the MSI interrupt. When the `64-bit Addressing Capable` bit is 0, this register always reads as 0.	0	RW

Table 48. MSI Message Data Register - 0x058 or 0x05C Register

If the 64-bit Addressing Capable bit in the `MSI Control Register` is set to 1, this register is at address 0x05C. Otherwise, it is at address 0x058.

Bits	Register Description	Default Value	Access
[15:0]	Data for MSI Interrupts generated by this function. This base value is written to Root Port memory to signal an MSI interrupt. When one MSI vector is allowed, this value is used directly. When 2 MSI vectors are allowed, the upper 15 bits are used. And, the least significant bit indicates the interrupt number. When 4 MSI vectors are allowed, the lower 2 bits indicate the interrupt number, and so on.	0	RW
[31:16]	Reserved	0	RO

Table 49. MSI Mask Register - 0x05C (32-bit addressing) or 0x060 (64-bit addressing)

If the 64-bit Addressing Capable bit in the `MSI Control Register` is set to 1, this register is at address 0x060. Otherwise, it is at 0x05C.

Bits	Register Description	Default Value	Access
31:0	Mask bits for MSI interrupts. The number of implemented bits depends on the number of MSI vectors configured. When one MSI vectors is used , only bit 0 is RW. The other bits read as zeros. When two MSI vectors are used, bits [1:0] are RW, and so on. A one in a bit position masks the corresponding MSI interrupt.	See description	0

Table 50. Pending Bits for MSI Interrupts Register - 0x060 (32-bit addressing) or 0x064 (64-bit addressing)

If the 64-bit Addressing Capable bit in the `MSI Control Register` is set to 1, this register is at address 0x064. Otherwise, it is at 0x060.

Bits	Register Description	Default Value	Access
31:0	Pending bits for MSI interrupts. A 1 in a bit position indicated the corresponding MSI interrupt is pending in the core. The number of implemented bits depends on the number of MSI vectors configured. When 1 MSI vectors is used, only bit 0 is RW. The other bits read as zeros. When 2 MSI vectors are used, bits [1:0] are RW, and so on.	RO	0

Related Information

PCI Local Bus Specification 3.0

6.5. MSI-X Capability Structure

Figure 46. MSI-X Capability Registers - Byte Address Offsets and Layout

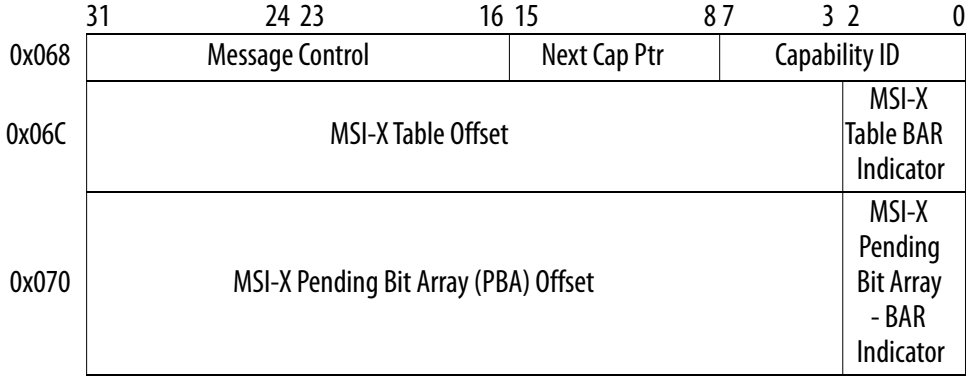


Table 51. MSI-X Capability ID, Capability Pointer and Mask Register - 0x068

Bits	Register Description	Default Value	Access
[31]	MSI-X Enable. When set, enables MSI-X interrupt generation.	0	RW
[30]	MSI-X Function Mask. When set, masks all MSI-X interrupts from this function.	0	RW
[29:27]	Reserved.	0	RO

continued...

Bits	Register Description	Default Value	Access
[26:16]	Size of the MSI-X Table. The value in this field is 1 less than the size of the table set up for this function. The maximum value is 0x7FF, or 4096 interrupt vectors.	Set in Platform Designer	RO
[15:8]	Next Capability Pointer. Points to Power Management Capability.	0x80	RO
[7:0]	Capability ID. PCI-SIG assigns this ID.	0x11	RO

Table 52. MSI-X Table Offset BAR Indicator Register - 0x06C

Bits	Register Description	Default Value	Access
[2:0]	MSI-X Table BAR Indicator. Specifies the BAR number whose address range contains the MSI-X Table. • 3'b000: BAR0 • 3'b001: BAR1 • 3'b010: BAR2 • 3'b011: BAR3 • 3'b100: BAR4 • 3'b101: BAR5	Set in Platform Designer	RO
[31:3]	Specifies the memory address offset for the MSI-X Table relative to the BAR base address value of the BAR number specified in MSI-X Table BAR Indicator,[2:0] above. The address is extended by appending 3 zeros to create quad-word alignment.	Set in Platform Designer	RO

Table 53. MSI-X Pending Bit Array (PBA) Offset Register - 0x070

Bits	Register Description	Default Value	Access
[2:0]	MSI-X Pending Bit Array BAR Indicator. Specifies the BAR number whose address range contains the Pending Bit Array (PBA) table for this function. The following encodings are defined: • 3'b000: BAR0 • 3'b001: BAR1 • 3'b010: BAR2 • 3'b011: BAR3 • 3'b100: BAR4 • 3'b101: BAR5	Set in Platform Designer	RO
[31:3]	Specifies the memory address offset for the PBA relative to the specified base address value of the BAR number specified in MSI-X Pending Bit Array BAR Indicator, at [2:0] above. The address is extended by appending 3 zeros to create quad-word alignment.	Set in Platform Designer	RO

Related Information

PCI Local Bus Specification 3.0

6.6. Power Management Capability Structure

Figure 47. Power Management Capability Structure - Byte Address Offsets and Layout

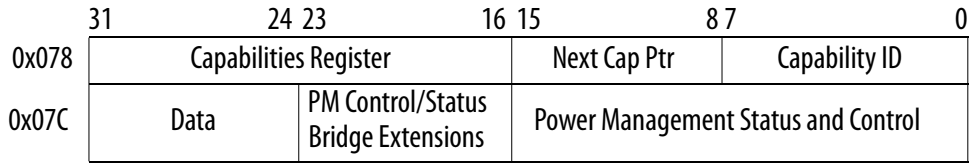


Table 54. Power Management Capabilities Register - 0x078

Bits	Register Description	Default Value	Access
[31:19]	Not Implemented	RO	0
[18:16]	Version ID: Version of Power Management Capability	RO	0x3
[15:8]	Next Capability Pointer: Points to the PCI Express Capability.	RO	0x80
[7:0]	Capability ID assigned by PCI-SIG.	RO	0x01

Table 55. Power Management Control and Status Registers - 0x07C

Bits	Register Description	Default Value	Access
[31:4]	Not implemented.	RO	0
[3]	No Soft Reset: If set, the Function maintains its internal state when in the $D3_{hot}$ state. Software is not required to re-initialize the registers when the Function returns from $D3_{hot}$ to D0.	RO	Set inPlatform Designer
[2]	Reserved.	RO	0
[1:0]	Indicates the power state of this Function. The only allowed settings are 2'b00 (D0) and 2b'11 .	RW	0

6.7. PCI Express Capability Structure

Figure 48. 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
0x080	PCI Express Capabilities Register		Next Cap Pointer	PCI Express Capabilities ID
0x084	Device Capabilities			
0x088	Device Status		Device Control	
0x08C	Link Capabilities			
0x090	Link Status		Link Control	
0x094	Slot Capabilities			
0x098	Slot Status		Slot Control	
0x09C	Root Capabilities		Root Control	
0x0A0	Root Status			
0x0A4	Device Compatibilities 2			
0x0A8	Device Status 2		Device Control 2	
0x0AC	Link Capabilities 2			
0x0B0	Link Status 2		Link Control 2	

Table 56. PCI Express Capability Register - 0x080

Bits	Description	Default Value	Access
[31:19]	Reserved	0	RO
[18:16]	Version ID: Version of Power Management Capability.	0x3	RO
[15:8]	Next Capability Pointer: Points to the PCI Express Capability.	0x80	RO
[7:0]	Capability ID assigned by PCI-SIG.	0x01	RO

Table 57. PCI Express Device Capabilities Register -0x084

Bits	Description	Default Value	Access
[2:0]	Maximum Payload Size supported by the Function. Can be configured as 000 (128 bytes) or 001 (256 bytes)	Set in Platform Designer	RO
[4:3]	Reserved	0	RO
[5]	Extended tags supported	Set in Platform Designer	RO
[8:6]	Acceptable L0S latency	Set in Platform Designer	RO
[11:9]	Acceptable L1 latency	Set in Platform Designer	RO
[14:12]	Reserved	0	RO
[15]	Role-Based error reporting supported	1	RO
[17:16]	Reserved	0	RO
[27:18]	Captured Slot Power Limit Value and Scale: Not implemented	0	RO
[28]	FLR Capable. Indicates that the device has FLR capability	Set in Platform Designer	RO
[31:29]	Reserved	0	RO

Table 58. PCI Express Device Control and Status Register - 0x088

Bits	Description	Default Value	Access
[0]	Enable Correctable Error Reporting.	0	RW
[1]	Enable Non-Fatal Error Reporting.	0	RW
[2]	Enable Fatal Error Reporting.	0	RW
[3]	Enable Unsupported Request (UR) Reporting.	0	RW
[4]	Enable Relaxed Ordering.	Set in Platform Designer	RW
[7:5]	Maximum Payload Size.	0 (128 bytes)	RW
[8]	Extended Tag Field Enable.	0	RW
[10:9]	Reserved.	0	RO
[11]	Enable No-Snoop.	1	RW
[14:12]	Maximum Read Request Size.	2 (512 bytes)	RW
[15]	Function-Level Reset. Writing a 1 generates a Function-Level Reset for this Function if the FLR Capable bit of the Device Capabilities Register is set. This bit always reads as 0.	0	RW

continued...

Bits	Description	Default Value	Access
[16]	Correctable Error detected.	0	RW1C
[17]	Non-Fatal Error detected.	0	RW1C
[18]	Fatal Error detected.	0	RW1C
[19]	Unsupported Request detected.	0	RW1C
[20]	Reserved.	0	RO
[21]	Transaction Pending: Indicates that a Non- Posted request issued by this Function is still pending.	0	RO
[31:22]	Reserved.	0	RO

Table 59. Link Capabilities Register - 0x08C

Bits	Description	Default Value	Access
[3:0]	Maximum Link Speed	1: 2.5 GT/s 2: 5.0 GT/s 3: 8.0 GT/s	RO
[9:4]	Maximum Link Width	1, 2, 4 or 8	RO
[10]	ASPM Support for L0S state	Set in Platform Designer	RO
[11]	ASPM Support for L1 state	Set in Platform Designer	RO
[14:12]	L0S Exit Latency	Set in Platform Designer, 0x6	RO
[17:15]	L1 Exit Latency	Set in Platform Designer, 0x0	RO
[21:18]	Reserved	0	RO
[22]	ASPM Optionality Compliance	1	RO
[31:23]	Reserved	0	RO

Table 60. Link Control and Status Register - 0x090

Bits	Description	Default Value	Access
[1:0]	ASPM Control	0	RW
[2]	Reserved	0	R O
[3]	Read Completion Boundary	0	RW
[5:4]	Reserved	0	RO
[6]	Common Clock Configuration	0	RW
[7]	Extended Synch	0	RW
[15:8]	Reserved	0	RO
[19:16]	Negotiated Link Speed	0	RO
[25:20]	Negotiated Link Width	0	RO

continued...

Bits	Description	Default Value	Access
[27:26]	Reserved	0	RO
[28]	Slot Clock Configuration	Set in Platform Designer	RO
[31:29]	Reserved	0	RO

Table 61. PCI Express Device Capabilities 2 Register - 0x0A4

Bits	Description	Default Value	Access
[3:0]	Completion Timeout ranges	Set in Platform Designer	RO
[4]	Completion Timeout disable supported	Set in Platform Designer	RO
[31:5]	Reserved	0	RO

Table 62. PCI Express Device Control 2 and Status 2 Register - 0x0A8

Bits	Description	Default Value	Access
[3:0]	Completion Timeout value	0xF	RW
[4]	Completion Timeout disable	1	RW
[5]	Reserved	0	RO
[6]	Atomic Operation Requester Enable	0	RW
[31:7]	Reserved	0	RO

Table 63. Link Capabilities 2 Register - 0x0AC

Bits	Description	Default Value	Access
[0]	Reserved	0	RO
[3:1]	Link speeds supported	1 (2.5 GT/s) 3 (5.0 GT/s) 7 (8.0 GT/s) Set in Platform Designer	RO
[31:4]	Reserved	0	RO

Table 64. Link Control 2 and Status 2 Register - 0x0B0

Bits	Description	Default Value	Access
[3:0]	Target Link Speed	1: Gen1 2: Gen2 3: Gen3	RWS
[4]	Enter Compliance	0	RWS
[5]	Hardware Autonomous Speed Disable	0	RW
[6]	Selectable De-emphasis	0	RO
[9:7]	Transmit Margin	0	RWS
[10]	Enter Modified Compliance	0	RWS

continued...

Bits	Description	Default Value	Access
[11]	Compliance SOS	0	RWS
[15:12]	Compliance Preset/De-emphasis	0	RWS
[16]	Current De-emphasis Level	0	RO
[17]	Equalization Complete	0	RO
[18]	Equalization Phase 1 Successful	0	RO
[19]	Equalization Phase 2 Successful	0	RO
[20]	Equalization Phase 3 Successful	0	RO
[21]	Link Equalization Request	0	RW1C
[31:22]	Reserved	0	RO

6.8. Advanced Error Reporting (AER) Enhanced Capability Header Register

Table 65. AER Enhanced Capability Header Register -0x100

Bits	Register Description Default Value	Default Value	Access
[15:0]	PCI Express Extended Capability ID.	0x0001	RO
[19:16]	Capability Version.	2	RO
[31:20]	Next Capability Pointer: If ARI is supported, points to the ARI Capability, 0x160. Otherwise, the following values are possible: • If PF0 has a maximum link speed of 8.0 GT/s: Next Capability = Secondary PCIe, 0x280. • if PF 0 has a maximum link speed of 2.5 or 5.0 GT/s and TPH Requester Capability: Next Capability = TPH Requester, 0x300. • If 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, 0x3C0 • PFs 1–3 with TPH Requester Capability: Next Capability = TPH Requester, 0x300. • PFs 1–3 with ATS Capability: Next Capability = ATS, 0x3C0. • All other cases: Next Capability = 0.	See description	RO

6.9. Uncorrectable Error Status Register

This register controls which errors are forwarded as internal uncorrectable errors. All of the errors are severe and may place the device or PCIe link in an inconsistent state.

Table 66. Uncorrectable Error Status Register - 0x104

Bits	Register Description	Default Value	Access
[31:21]	Reserved.	0	RO
[20]	When set, indicates an Unsupported Request Received	0	RW1C
[19]	When set, indicates an ECRC Error Detected	0	RW1C
[18]	When set, indicates a Malformed TLP Received	0	RW1C
[17]	When set, indicates Receiver Overflow	0	RW1C

continued...

Bits	Register Description	Default Value	Access
[16]	When set, indicates an unexpected Completion was received	0	RW1C
[15]	When set, indicates a Completer Abort (CA) was transmitted	0	RW1C
[14]	When set, indicates a Completion Timeout	0	RW1C
[13]	When set, indicates a Flow Control protocol error	0	RW1C
[12]	When set, indicates that a poisoned TLP was received	0	RW1C
[11:5]	Reserved	0	RO
[4]	When set, indicates a Data Link Protocol error	0	RW1C
[3:0]	Reserved	0	RO

Related Information

PCI Express Base Specification 3.0

6.10. Uncorrectable Error Mask Register

Table 67. Uncorrectable Error Mask Register - 0x108

Bits	Register Description	Default Value	Access
[31:21]	Reserved	0	RO
[20]	When set, masks an Unsupported Request Received	0	RW
[19]	When set, masks an ECRC Error Detected	0	RW
[18]	When set, masks a Malformed TLP Received	0	RW
[17]	When set, masks Receiver Overflow	0	RW
[16]	When set, masks an unexpected Completion was received	0	RW
[15]	When set, masks a Completer Abort (CA) was transmitted	0	RW
[14]	When set, masks a Completion Timeout	0	RW
[13]	When set, masks a Flow Control protocol error	0	RW
[12]	When set, masks that a poisoned TLP was received	0	RW
[11:5]	Reserved	0	RO
[4]	When set, masks a Data Link Protocol error	0	RW
[3:0]	Reserved	0	RO

6.11. Uncorrectable Error Severity Register

If a severity bit is 0, the core reports a Fatal error to the Root Port. If a severity bit is 1, the core reports a Non-Fatal error to the Root Port.

Table 68. Uncorrectable Error Severity Register - 0x10C

Bits	Register Description	Default Value	Access
[31:21]	Reserved	0	RO
[20]	Unsupported Request Received	0	RW

continued...

Bits	Register Description	Default Value	Access
[19]	ECRC Error Detected	0	RW
[18]	Malformed TLP Received	1	RW
[17]	Receiver Overflow	1	RW
[16]	Unexpected Completion was received	0	RW
[15]	Completer Abort (CA) was transmitted	0	RW
[14]	Completion Timeout	0	RW
[13]	Flow Control protocol error	1	RW
[12]	Poisoned TLP	0	RW
[11:6]	Reserved	0	RO
[5]	Surprise Down Error	0	RO
[4]	Data Link Protocol error	1	RW
[3:0]	Reserved	0	RO

Related Information

PCI Express Base Specification 3.0

6.12. Correctable Error Status Register

Table 69. Correctable Error Status Register -0x110

Bits	Register Description	Default Value	Access
[31:14]	Reserved	0	RO
[13]	When set, indicates an Advisory Non-Fatal Error	0	RW1C
[12]	When set, indicates a Replay Timeout	0	RW1C
[11:9]	Reserved	0	RO
[8]	When set, indicates a Replay Number Rollover	0	RW1C
[7]	When set, indicates a Bad DLLP received	0	RW1C
[6]	When set, indicates a Bad TLP received	0	RW1C
[5:1]	Reserved	0	RO
[0]	When set, indicates a Receiver Error	0	RW1C

Related Information

PCI Express Base Specification 3.0

6.13. Correctable Error Mask Register

Table 70. Correctable Error Mask Register - 0x114

Bits	Register Description	Default Value	Access
[31:14]	Reserved	0	RO
[13]	When set, masks an Advisory Non-Fatal Error	0	RW

continued...

Bits	Register Description	Default Value	Access
[12]	When set, masks a Replay Timeout	0	RW
[11:9]	Reserved	0	RO
[8]	When set, masks a Replay Number Rollover	0	RW
[7]	When set, masks a Bad DLLP received	0	RW
[6]	When set, masks a Bad TLP received	0	RW
[5:1]	Reserved	0	RO
[0]	When set, masks a Receiver Error	0	RW

Related Information

PCI Express Base Specification 3.0

6.14. Advanced Error Capabilities and Control Register

Table 71. Advanced Error Capabilities and Control Register - 0x118

Bits	Register Description	Default Value	Access
[4:0]	First Error Pointer	0	ROS
[5]	ECRC Generation Capable	Set in Platform Designer	RO
[6]	ECRC Generation Enable	0	RW
[7]	ECRC Check Capable	Set in Platform Designer	RO
[8]	ECRC Check Enable	0	RW
[31:9]	Reserved	0	RO

6.15. Header Log Registers 0-3

Table 72. Header Log Registers 0-3 - 0x11C - 0x128

This register contains the first 4 bytes of a TLP header captured from the link on detection of an uncorrectable error. Byte 0 (Type/Format field) of the header is stored in bit positions [31:24].

Bits	Register Description	Default Value	Access
[31:0]	First 4 bytes of captured TLP header	0	ROS

6.16. SR-IOV Virtualization Extended Capabilities Registers

6.16.1. SR-IOV Virtualization Extended Capabilities Registers Address Map

Figure 49. SR-IOV Virtualization Extended Capabilities Registers

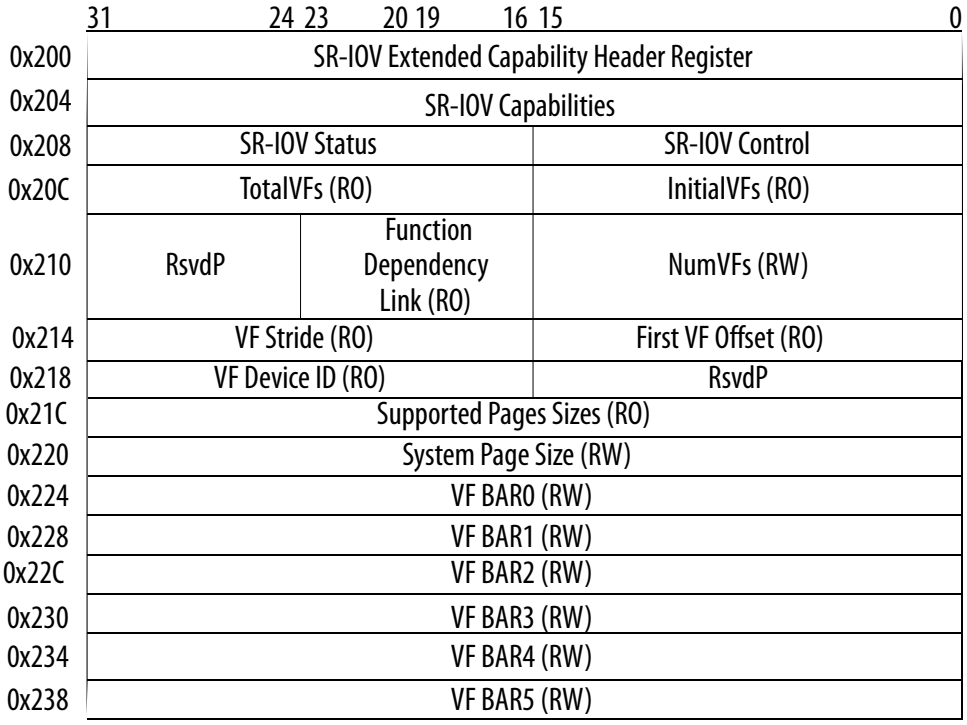


Table 73. SR-IOV Virtualization Extended Capabilities Registers

Byte Address Offset	Name	Description
Alternative RID (ARI) Capability Structure		
0x160	ARI Enhanced Capability Header	PCI Express Extended Capability ID for ARI and next capability pointer.
0x0164	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.
Single-Root I/O Virtualization (SR-IOV) Capability Structure		
0x200	SR-IOV Extended Capability Header	PCI Express Extended Capability ID for SR-IOV and next capability pointer.
0x204	SR-IOV Capabilities Register	Lists supported capabilities of the SR-IOV implementation.
0x208	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.
0x20C	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.

continued...

Byte Address Offset	Name	Description
0x210	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.
0x214	VF Offset/Stride	Specifies the offset and stride values used to assign routing IDs to the VFs.
0x258	VF Device ID	Specifies VF Device ID assigned to the device.
0x21C	Supported Page Sizes	Specifies all page sizes supported by the device.
0x220	System Page Size	Stores the page size currently selected.
0x224	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.
0x228	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.
0x22C	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.
0x230	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.
0x234	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.
0x238	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.
0x23C	VF Migration State Array Offset	Not implemented.
Secondary PCI Express Extended Capability Structure (Gen3, PF 0 only)		
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.
Transaction Processing Hints (TPH) Requester Capability Structure		
0x300	TPH Requester Extended Capability Header	PCI Express Extended Capability ID for TPH Requester Capability, and next capability pointer.
0x304	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.
0x308	TPH Requester Control Register	This register contains enable and mode select bits for the TPH Requester Capability.

continued...

Byte Address Offset	Name	Description
Address Translation Services (ATS) Capability Structure		
0x3C0	ATS Extended Capability Header	PCI Express Extended Capability ID for ATS Capability, and next capability pointer.
0x3C4	ATS Capability Register and ATS Control Register	This location contains the 16-bit ATS Capability Register and the 16-bit ATS Control Register.

6.16.2. ARI Enhanced Capability Header

Table 74. ARI Enhanced Capability ID - 0x1600x178

Bits	Register Description Default Value	Default Value	Access
[15:0]	PCI Express Extended Capability ID for ARI.	0x000E	RO
[19:16]	Capability Version.	0x1	RO
[31:20]	Next Capability Pointer: The following values are possible: • 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: — PF0 with maximum link speed of 8.0 GT/s: Next Capability = Secondary PCIe, 0x280. — PF0 with maximum link speed of 2.5 or 5.0 GT/s and TPH Requester Capability: Next Capability = TPH Requester, 0x300. — 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. — PFs 1–3 with TPH Requester Capability: Next Capability = TPH Requester, 0x300. — PFs 1–3 with ATS Capability: Next Capability = ATS, 0x3C0. — All other cases: Next Capability = 0.	See description	RO

Table 75. ARI Enhanced Capability Header and Control Register -0x164

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

6.16.3. SR-IOV Enhanced Capability Registers

Table 76. SR-IOV Extended Capability Header Register - 0x200

Bits	Register Description	Default Value	Access
[15:0]	PCI Express Extended Capability ID	0x0010	RO
[19:16]	Capability Version	1	RO
[31:16]	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: • PF0 has a maximum link speed of 8.0 GT/s: Next Capability = Secondary PCIe, 0x280. • PF0 has a maximum link speed of 2.5 or 5.0 GT/s and TPH Requester Capability: Next Capability = TPH Requester, 0x300. • 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, 0x3C0. • PFs 1–3 with TPH Requester Capability: Next Capability = TPH Requester,0x300. • PFs 1–3 with ATS Capability: Next Capability = ATS, 0x3C0. • All other cases: Next Capability = 0.	Set in Platform Designer	RO

Table 77. SR-IOV Capabilities Register - 0x204

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 Default Value	RO

Table 78. SR-IOV Control and Status Registers - 0x208

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

6.16.4. Initial VFs and Total VFs Registers

Table 79. Initial VFs and Total VFs Registers - 0x20C

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 80. Function Dependency Link and NumVFs Registers -0x210

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 81. VF Offset and Stride Registers -0x214

Bits	Register Description	Default Value	Access
[15:0]	VF Offset (offset of first VF's Routing ID with respect to the Routing ID of its PF). In a system with 4 PFs, PF0 has a routing ID of 0, PF1 has a routing ID of 1, and so on. The following calculations determine the routing IDs for the VFs: • VF Offset for PF0 = TOTAL_PF_COUNT. • VF Offset for PF1 = (PF0_VF_COUNT + (TOTAL_PF1_COUNT -1)) • VF Offset for PF2 = (PF0_VF_COUNT + PF1_VF_COUNT + (TOTAL_PF_COUNT – 2)) • VF Offset for PF3 = (PF0_VF_COUNT + PF1_VF_COUNT + PF2_VF_COUNT + (TOTAL_PF_COUNT – 3))	Refer to description	RO
[31:16]	VF Stride	1	RO

6.16.5. VF Device ID Register

Table 82. VF Device ID Register - 0x218

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

6.16.6. Page Size Registers

Table 83. Supported Page Size Register -0x21C

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 84. System Page Size Register - 0x220

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

6.16.7. 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 85. VF BARs 0-5 - 0x224-0x238

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: • 1'b0: 32-bit BAR • 1'b1: 64-bit BAR created by pairing BAR0 with BAR1, BAR2 with BAR3, or BAR4 with BAR5	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 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

6.16.8. Secondary PCI Express Extended Capability Header

Table 86. Secondary PCI Express Extended Capability Header - 0x280

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: • If TPH Requester Capability is supported by the Function, its Next Capability = TPH Requester, 0x300. • Otherwise, if the ATS Capability is supported by the Function, its Next Capability = ATS, 0x3C0. • In all other cases: Next Capability = 0.	0x240, 0x280, or 0	RO

6.16.9. Lane Status Registers

Table 87. Lane Error Status Register - 0x288

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 88. Lane Equalization Control Registers 0–3 - 0x28C-0x298

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 0 when link width = 1	RO
[30:28]	Upstream Port Lane 1 Receiver Preset Hint	0x7 when link width > 1 0 when link width = 1	RO
[31]	Reserved	0	RO

6.16.10. Transaction Processing Hints (TPH) Requester Enhanced Capability Header

Table 89. Transaction Processing Hints (TPH) Requester Enhanced Capability Header Register - 0x300

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.	0x280 or 0	RO

6.17. Virtual Function Registers

The SR-IOV Bridge implements the PCI and PCI Express Configuration Spaces for a maximum of 2048 Virtual Functions. The VF registers available are a subset of the PF registers. For example, the VFs do not implement the Link Capabilities 2 register. The definitions of VF registers are the same as PF registers. For additional details, refer to the *PCI Express Base Specification 3.0*.

Note: The following Virtual Function register map is applicable to all revisions of the Intel Arria 10 Avalon Streaming with SR-IOV IP for PCIe from the 18.1 release of Intel Quartus Prime onward.

Table 94. Virtual Function Registers - Differences from PF

Address (hex)	Name	Description
0x000	Vendor ID and Device ID Register	Vendor ID Register and Device ID Registers defined in *PCI Express Base Specification 3.0* . These registers are hardwired to all 1s.
0x004	Command and Status Register	PCI Command and Status Registers. Refer to *Command and Status Register for VFs* for descriptions of the implemented fields.
0x008	Revision ID and Class Code Register	PCI Revision ID and Class Code Registers defined in *PCI Express Base Specification 3.0* . The VF has the same settings and access as PF0.
0x00C	BIST, Header Type, Latency Timer and Cache Line Size Registers	Contains the following registers defined in the *PCI Express Base Specification 3.0* : BIST Register, Header Type Register, Latency Timer, Cache Line Size Register. These registers are hardwired to all 0s for VFs.
0x010: 0x028	Reserved	N/A
0x02C	Subsystem Vendor ID and Subsystem ID Registers	PCI Subsystem Vendor ID and Subsystem ID Registers. The VF has the same settings and access as PF0.
0x030	Reserved	N/A
0x034	Capabilities Pointer	This register points to the first Capability Structure in the PCI Configuration Space. For VFs, it points to the MSI-X capability.
0x038: 0x03C	Reserved	N/A
MSI-X Capability Structure		
0x07C	MSI-X Control Register	Contains the MSI-X Message Control Register, Capability ID for MSI-X, and the next capability pointer. The VF has the same fields and access as the parent PF.
0x080	MSI-X Table Offset	Points to the MSI-X Table in memory. Also specifies the BAR corresponding to the memory segment where the MSI-X Table resides. The VF has the same fields and access as the PF.
0x084	MSI-X PBA Offset	Points to the MSI-X Pending Bit Array in memory. Also, specifies the BAR corresponding to the memory segment where the PBA Array resides. The VF has the same fields and access as the parent PF.
PCI Express Capability Structure		
0x040	PCI Express Capability List Register	Capability ID, PCI Express Capabilities Register, and the next capability pointer. Refer to cite="PCI Express Capability List Register for VFs" for descriptions of the implemented fields.
0x044	PCI Express Device Capabilities Register	PCI Express Device Capabilities Register. The VF Device Capabilities Register supports the same fields as the PF Device Capabilities Register.
0x048	PCI Express Device Control and Status Registers	The lower 16 bits implement the PCI Express Device Control Register. The upper 16 bits implement the Device Status Register. Refer to *PCI Express Devices Control and Status Registers for VFs* for descriptions of the implemented fields.
0x04C	Link Capabilities Register	A read to any VF with this address returns the Link Capabilities Register settings of the parent PF.
0x050	Link Control and Status Registers	This register is not implemented for VFs, and reads as all 0s.

continued...

Address (hex)	Name	Description
0x054	Device Capabilities 2 Registers	A read to any VF with this address returns the Device Capabilities 2 Register settings of the parent PF.
0x058	Device Control 2 and Status 2 Registers	This register is not implemented for VFs. A read to this address returns all 0s.
0x05C	Link Capabilities 2 Register	This register is not implemented for VFs. A read to this address returns all 0s.
0x060	Link Control 2 and Status 2 Registers	This register contains control and status bits for the PCIe link. For VFs, bit[16] stores the current de-emphasis level setting for the parent PF. All other bits are reserved.
Alternate RID (ARI) Capability Structure		
0x100	ARI Enhanced Capability Header	PCI Express Extended Capability ID for ARI and Next Capability pointer. The Next Capability pointer points to NULL.
0x104	ARI Capability Register, ARI Control Register	This register is not implemented for VFs. A read to this address returns all 0s.
Transaction Processing Hints (TPH) Requester Capability Structure		
0x300	TPH Requester Extended Capability Header	PCI Express Extended Capability ID for TPH Requester Capability, and next capability pointer.
0x304	TPH Requester Capability Register	This register contains the advertised parameters for the TPH Requester Capability.
0x308	TPH Requester Control Register	This register contains enable and mode select bits for the TPH Requester Capability.
Address Translation Services (ATS) Capability Structure		
0x3C0	ATS Extended Capability Header	PCI Express Extended Capability ID for ATS Capability, and next capability pointer.
0x3C4	ATS Capability Register and ATS Control Register	This location contains the 16-bit ATS Capability Register and the 16-bit ATS Control Register.

Table 95. Command and Status Register for VFs - 0x004

Bits	Register Description	Default Value	Access
[1:0]	Reserved.	0	RO
[2]	Bus Master enable. When set, the VF can generate transactions as a bus master.	0	RW
[19:3]	Reserved.	0	RO
[20]	Indicates the presence of PCI Extended Capabilities. This bit is hardwired to 1.	1	RO
[23:21]	Reserved.	0	RO
[24]	Master Data Parity Error: The device sets this bit when when the following occurs: • It has received a poisoned Completion directed at this VF • This VF has sent a poisoned memory write request on the link This bit can only be set if the `Parity Error Response Enable` bit of the `PCI Command Register` of the parent PF is 1. This bit is cleared by writing a 1.	0	RW1C
[26:25]	Reserved.	0	RO

continued...

Bits	Register Description	Default Value	Access
[27]	Signaled Target Abort: The device sets this bit when this VF has sent a Completion with the Completer Abort (CA) status to the link. This bit is cleared by writing a 1	0	RW1C
[28]	Received Target Abort: The device sets this bit when it has received a Completion with the Completer Abort (CA) status targeting t this VF. This bit is cleared by writing a 1	0	RW1C
[29]	Received Master Abort: The device sets this bit when it has received a Completion with the Unsupported Request (UR) status targeting this VF. This bit is cleared by writing a 1	0	RW1C
[30]	Signaled System Error: The VF sets this bit when it has sent Fatal or Non-Fatal error message to the Root Complex. This bit can only be set if the `SERR Enable` bit of the `PCI Command Register` of the parent PF is enabled. This bit is cleared by writing a 1.	0	RW1C
[31]	Received Master Abort: The VF sets this bit when it has received a Completion with the Unsupported Request (UR) status. This bit is cleared by writing a 1	0	RW1C

Table 96. PCI Express Capability List Register for VFs - 0x080

Bits	Register Description	Default Value	Access
[31:19]	Hardwired to 0.	2	RO
[18:16]	Version ID: Version of PCI Express Capability.	0	RW
[15:8]	Next Capability Pointer: Points to NULL.	0	RO
[7:0]	Capability ID assigned by PCI-SIG.	0x10	RO

Table 97. PCI Express Device Control and Status Registers for VFs - 0x088

Bits	Register Description	Default Value	Access
Control Register			
[14:0]	Reserved.	0	RO
[15]	Function-Level Reset. Writing a 1 to this bit generates a Function-Level Reset for this VF. Only functional when the PF Device Capabilities Register FLR Capable bit is set. This bit always reads as 0.	0	RW
Status Register			
[16]	Correctable Error Detected.	0	RW1C
[17]	Non-Fatal Error Detected.	0	RW1C
[18]	Fatal Error Detected.	0	RW1C
[19]	Unsupported Request Detected.	0	RW1C
[20]	Not implemented.	0	RO
[21]	Transaction Pending. When set, indicates that a Non-Posted request issued by this VF is still pending.	0	RO
[31:22]	Reserved.	0	RO

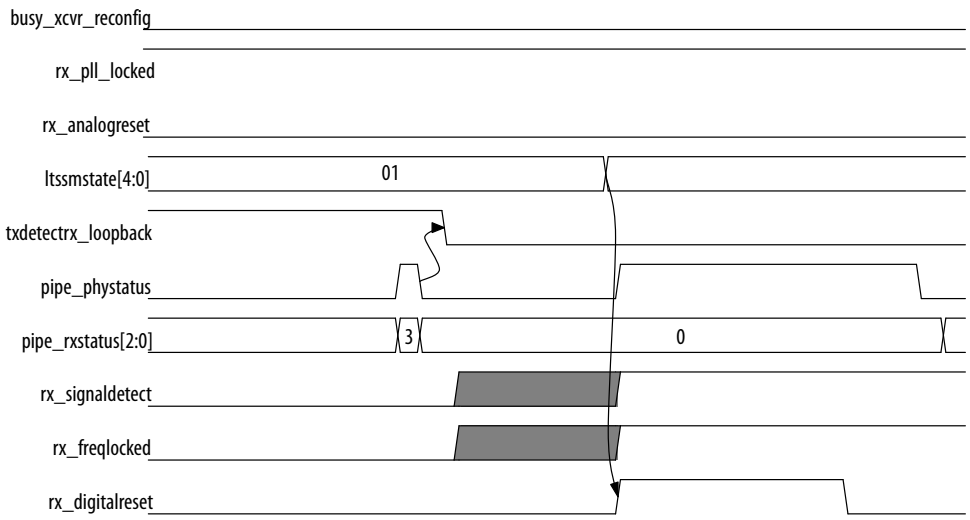
7. Reset and Clocks

The following figure shows the hard reset controller that is embedded inside the Hard IP for PCI Express. This controller takes in the `npor` and `pin_perst` inputs and generates the internal reset signals for other modules in the Hard IP.

7.1. Reset Sequence for Hard IP for PCI Express IP Core and Application Layer

After `pin_perst` or `npor` is released, the Hard IP reset controller deasserts `reset_status`. Your Application Layer logic can then come out of reset and become operational.

Figure 50. RX Transceiver Reset Sequence



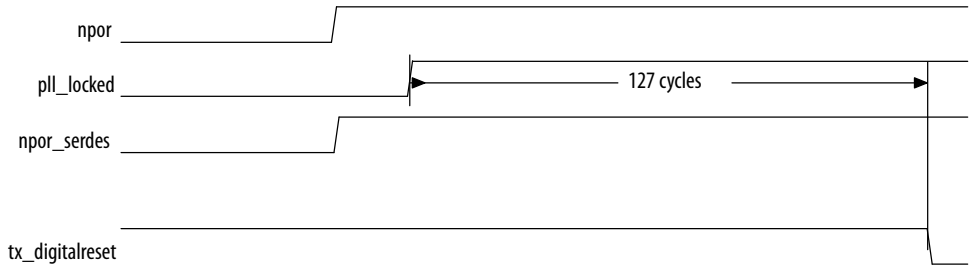
The RX transceiver reset sequence includes the following steps:

1. After `rx_pll_locked` is asserted, the LTSSM state machine transitions from the Detect.Quiet to the Detect.Active state.
2. When the `pipe_phystatus` pulse is asserted and `pipe_rxstatus[2:0] = 3`, the receiver detect operation has completed.
3. The LTSSM state machine transitions from the Detect.Active state to the Polling.Active state.
4. The Hard IP for PCI Express asserts `rx_digitalreset`. The `rx_digitalreset` signal is deasserted after `rx_signaldetect` is stable for a minimum of 3 ms.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Figure 51. TX Transceiver Reset Sequence



The TX transceiver reset sequence includes the following steps:

1. After `npor` is deasserted, the IP core deasserts the `npor_serdes` input to the TX transceiver.
2. The SERDES reset controller waits for `pll_locked` to be stable for a minimum of 127 `pld_clk` cycles before deasserting `tx_digitalreset`.

For descriptions of the available reset signals, refer to *Reset Signals, Status, and Link Training Signals*.

7.2. Function Level Reset (FLR)

The following sequence of events occurs after a FLR to a Physical Function:

1. The host stops all traffic from and to the Function.
2. The host writes the FLR bit in the `Device Control` Register to trigger the FLR reset.
3. The SR-IOV Bridge resets R/W non-sticky control bits in the Configuration Space of the Function. It notifies the Application Layer via `flr_active_*` signals.
4. The Application Layer cleans up all state related to the Function. It asserts FLR Completed via `flr_completed_*` signal. The Application Layer should either discard all pending requests from the Function, or send Completions. If the Application Layer sends Completions, the host drops them without checking for errors.
5. The SR-IOV Bridge re-enables the Function by deasserting the `flr_active_*` signal associated with this function.
6. The host re-enumerates the Function.

This handshake ensures that the Completion for a request issued before the FLR does not return after the FLR is complete.

7.3. Clocks

The Hard IP contains a clock domain crossing (CDC) synchronizer at the interface between the PHY/MAC and the DLL layers. The synchronizer allows the Data Link and Transaction Layers to run at frequencies independent of the PHY/MAC. The CDC synchronizer provides more flexibility for the user clock interface. Depending on parameters you specify, the core selects the appropriate `coreclkout_hip`. You can use these parameters to enhance performance by running at a higher frequency for latency optimization or at a lower frequency to save power.

In accordance with the *PCI Express Base Specification*, you must provide a 100 MHz reference clock that is connected directly to the transceiver.

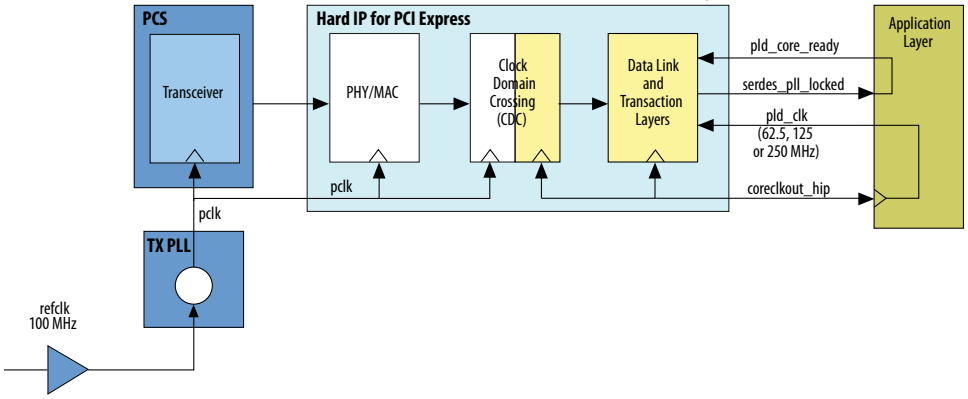
Related Information

PCI Express Base Specification 3.0

7.3.1. Clock Domains

Figure 52. Clock Domains and Clock Generation for the Application Layer

The following illustrates the clock domains when using `coreclkout_hip` to drive the Application Layer and the `pld_clk` of the IP core. The Intel-provided example design connects `coreclkout_hip` to the `pld_clk`. However, this connection is not mandatory. Inside the Hard IP for PCI Express, the blocks shown in white are in the `pclk` domain, while the blocks shown in yellow are in the `coreclkout_hip` domain.



As this figure indicates, the IP core includes the following clock domains: `pclk`, `coreclkout_hip` and `pld_clk`.

7.3.1.1. coreclkout_hip

Table 98. Application Layer Clock Frequency for All Combinations of Link Width, Data Rate and Application Layer Interface Widths

The `coreclkout_hip` signal is derived from `pclk`. The following table lists frequencies for `coreclkout_hip`, which are a function of the link width, data rate, and the width of the Application Layer to Transaction Layer interface. The frequencies and widths specified in this table are maintained throughout operation. If the link downtrains to a lesser link width or changes to a different maximum link rate, it maintains the frequencies it was originally configured for as specified in this table. (The Hard IP throttles the interface to achieve a lower throughput.)

Link Width	Max Link Rate	Avalon Interface Width	coreclkout_hip
×8	Gen1	128	125 MHz
×4	Gen2	128	125 MHz
×8	Gen2	128	250 MHz
×8	Gen2	256	125 MHz
×2	Gen3	128	125 MHz
×4	Gen3	128	250 MHz
×4	Gen3	256	125 MHz
×8	Gen3	256	250 MHz

7.3.1.2. pld_clk

`coreclkout_hip` can drive the Application Layer clock along with the `pld_clk` input to the IP core. The `pld_clk` can optionally be sourced by a different clock than `coreclkout_hip`. The `pld_clk` minimum frequency cannot be lower than the `coreclkout_hip` frequency. Based on specific Application Layer constraints, a PLL can be used to derive the desired frequency.

7.3.2. Clock Summary

Table 99. Clock Summary

Name	Frequency	Clock Domain
`coreclkout_hip`	62.5, 125 or 250 MHz	Avalon-ST interface between the Transaction and Application Layers.
`pld_clk`	125 or 250 MHz	Application and Transaction Layers.
`refclk`	100 MHz	SERDES (transceiver). Dedicated free running input clock to the SERDES block.

8. Programming and Testing SR-IOV Bridge MSI Interrupts

8.1. Setting Up and Verifying MSI Interrupts

The following procedure specifies and tests MSI interrupts. Perform the first five steps once, during or after enumeration.

1. Disable legacy interrupts by setting `Interrupt Disable` bit of the `Command` register using a Configuration Write Request. The `Interrupt Disable` bit is bit 10 of the `Command` register.
2. Enable MSI interrupts by setting the `MSI enable` of the `MSI Control` register using a Configuration Write Request. The `MSI enable` is bit 16 of 0x050.
3. Set up the MSI Address and MSI Data using a Configuration Write Request.
4. Specify the number of MSI vectors in the `Multiple Message Enable` field of the `MSI Control` register using Configuration Write Request.
5. Unmask the bits associated with MSI vectors in the previous step register using Configuration Write Request..
6. Send MSI requests via the `app_msi*` interface.
7. Verify that `app_msi_status[1:0]=0` when `app_msi_ack=1`.
8. Expect a Memory Write TLP request with the address and data matching those previously specified.

You can build on this procedure to verify that the Message TLP is dropped and `app_msi_status = 0x2` if either of the following conditions are true:

- The MSI capability is present, but the `MSI enable` bit is not set.
- The MSI capability is disabled, but the application sends an MSI request.

Related Information

SR-IOV Interrupt Interface on page 49

8.2. Masking MSI Interrupts

If Application Layer sends MSI interrupt when the corresponding mask bit is set, the bridge does not send this MSI interrupt to the host. Instead, the bridge sets the corresponding pending bit internally. The core sends this interrupt if its corresponding mask bit is cleared and the previous pending bit is set. The following procedure illustrates how to mask and unmask interrupts. The first four steps are the same as for

Setting Up and Verifying MSI Interrupts. Perform them once, during or after enumeration.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

1. Disable legacy interrupts by setting `Interrupt Disable` bit of the `Command` register using a Configuration Write Request. The `Interrupt Disable` bit is bit 10 of the `Command` register.
2. Enable MSI interrupts by setting the `MSI enable` of the `MSI Control` register using a Configuration Write Request. The `MSI enable` bit is bit 16 of 0x050.
3. Specify the MSI Address and MSI Data using a Configuration Write Request.
4. Specify the number of MSI vectors in the `Multiple Message Enable` field of the `MSI Control` register using a Configuration Write Request.
5. Select a function and interrupt number using a Configuration Write Request.
6. Set the MSI mask bit for the selected function and interrupt number using a Configuration Write Request.
7. Generate an MSI interrupt request for the selected function and interrupt number using the `app_msi*` interface using a Configuration Write Request. You should receive the MSI Ack. No MSI interrupt message is sent to the host.
8. Verify that `app_msi_status[1:0]=2'b01` when `app_msi_ack=1`.
9. Read the `Pending Bit` register for the function specified using a Configuration Read Request. Verify that the pending bit for the interrupt specified is set to 1.
10. Clear the pending bit using the MSI interrupt interface using a Configuration Write Request.
11. Clear the MSI mask bit for the selected function and interrupt number using a Configuration Write Request..
12. Verify that the SR-IOV Bridge sends the Message TLP to the host.
13. Read the `Pending Bit` register of the function specified using a Configuration Read Request. Verify that the pending bit for the interrupt specified is now 0.

8.3. Dropping a Pending MSI Interrupt

The following procedure shows how to drop a pending MSI interrupt. The first four steps are the same as for *Setting Up and Verifying MSI Interrupts* Perform them once, during or after enumeration. .

1. Disable legacy interrupts by setting `Interrupt Disable` bit of the `Command` register using a Configuration Write Request. The `Interrupt Disable` bit is bit 10 of the `Command` register.
2. Enable MSI interrupts by setting the `MSI enable` of the `MSI Control` register using a Configuration Write Request. The `MSI enable` bit is bit 16 of 0x050.
3. Set up the MSI Address and MSI Data using a Configuration Write Request.
4. Specify the number of MSI vectors in the `Multiple Message Enable` field of the `MSI Control` register using a Configuration Write Request.
5. Select a function and interrupt number using a Configuration Write Request.
6. Set the `MSI mask` bit for the selected function and interrupt number using a Configuration Write Request.
7. Use the MSI interrupt interface (`app_msi*`) to generate an MSI interrupt request for the selected Function and interrupt number. You should receive the MSI Ack. No MSI interrupt message is sent to the host.

8. Verify that `app_msi_status[1:0]=2'b01`when `app_msi_ack=1`.
9. Read the `Pending Bit` register for the function specified using a Configuration Read Request. Verify that the pending bit corresponding to the interrupt specified is set to 1.
10. Send a Configuration Write Request to clear the pending bit using the MSI interrupt interface.
11. Send a Configuration Write Request to clear the `MSI mask` bit for the selected function and interrupt number.
12. Verify that the SR-IOV bridge does not send the Message TLP on the Avalon-ST interface.
13. Read the `Pending Bit` register of the function specified using a Configuration Read Request. Verify that the pending bit for the interrupt specified is now 0.
14. Repeat this sequence for all MSI numbers and functions.

9. Error Handling

Each PCI Express compliant device must implement a basic level of error management and can optionally implement advanced error management. The IP core implements both basic and advanced error reporting. Error handling for a Root Port is more complex than that of an Endpoint.

Table 100. Error Classification

The *PCI Express Base Specification* defines three types of errors, outlined in the following table.

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

Related Information

PCI Express Base Specification 3.0

9.1. Physical Layer Errors

Table 101. Errors Detected by the Physical Layer

The following table describes errors detected by the Physical Layer. Physical Layer error reporting is optional in the *PCI Express Base Specification*.

Error	Type	Description
Receive port error	Correctable	This error has the following 3 potential causes: • Physical coding sublayer error when a lane is in L0 state. These errors are reported to the Hard IP block via the per lane PIPE interface input receive status signals, `rxstatus<lane_number>[2:0]` using the following encodings: — 3'b100: 8B/10B Decode Error — 3'b101: Elastic Buffer Overflow — 3'b110: Elastic Buffer Underflow — 3'b111: Disparity Error • Deskew error caused by overflow of the multilane deskew FIFO. • Control symbol received in wrong lane.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

9.2. Data Link Layer Errors

Table 102. Errors Detected by the Data Link Layer

Error	Type	Description
Bad TLP	Correctable	This error occurs when a LCRC verification fails or when a sequence number error occurs.
Bad DLLP	Correctable	This error occurs when a CRC verification fails.
Replay timer	Correctable	This error occurs when the replay timer times out.
Replay num rollover	Correctable	This error occurs when the replay number rolls over.
Data Link Layer protocol	Uncorrectable(fatal)	This error occurs when a sequence number specified by the Ack/Nak block in the Data Link Layer (`AckNak_Seq_Num`) does not correspond to an unacknowledged TLP.

9.3. Transaction Layer Errors

Table 103. Errors Detected by the Transaction Layer

Error	Type	Description
Poisoned TLP received	Uncorrectable (non-fatal)	This error occurs if a received Transaction Layer Packet has the EP poison bit set. The received TLP is passed to the Application Layer and the Application Layer logic must take appropriate action in response to the poisoned TLP. Refer to "2.7.2.2 Rules for Use of Data Poisoning" in the *PCI Express Base Specification* for more information about poisoned TLPs.
Unsupported Request for Endpoints	Uncorrectable (non-fatal)	This error occurs whenever a component receives any of the following Unsupported Requests: • Type 0 Configuration Requests for a non-existing function. • Completion transaction for which the Requester ID does not match the bus, device and function number. • Unsupported message. • A Type 1 Configuration Request TLP for the TLP from the PCIe link. • A locked memory read (MEMRDLK) on native Endpoint. • A locked completion transaction. • A 64-bit memory transaction in which the 32 MSBs of an address are set to 0. • A memory or I/O transaction for which there is no BAR match. • A memory transaction when the Memory Space Enable bit (bit [1] of the PCI Command register at Configuration Space offset 0x4) is set to 0. • A poisoned configuration write request (`CfgWr0`) In all cases the TLP is deleted in the Hard IP block and not presented to the Application Layer. If the TLP is a non-posted request, the Hard IP block generates a completion with Unsupported Request status.
Completion timeout	Uncorrectable (non-fatal)	This error occurs when a request originating from the Application Layer does not generate a corresponding completion TLP within the established time. It is the responsibility of the Application Layer logic to provide the completion timeout mechanism. The completion timeout should be reported from the Transaction Layer using the `cpl_err[0]` signal.

continued...

Error	Type	Description
Completer abort (1)	Uncorrectable (non-fatal)	The Application Layer reports this error using the `cpl_err[2]`signal when it aborts receipt of a TLP.
Unexpected completion	Uncorrectable (non-fatal)	This error is caused by an unexpected completion transaction. The Hard IP block handles the following conditions: • The Requester ID in the completion packet does not match the Configured ID of the Endpoint. • The completion packet has an invalid tag number. (Typically, the tag used in the completion packet exceeds the number of tags specified.) • The completion packet has a tag that does not match an outstanding request. • The completion packet for a request that was to I/O or Configuration Space has a length greater than 1 dword. • The completion status is Configuration Retry Status (CRS) in response to a request that was not to Configuration Space. In all of the above cases, the TLP is not presented to the Application Layer; the Hard IP block deletes it. The Application Layer can detect and report other unexpected completion conditions using the `cpl_err[2]` signal. For example, the Application Layer can report cases where the total length of the received successful completions do not match the original read request length.
Receiver overflow (1)	Uncorrectable (fatal)	This error occurs when a component receives a TLP that violates the FC credits allocated for this type of TLP. In all cases the hard IP block deletes the TLP and it is not presented to the Application Layer.
Flow control protocol error (FCPE) (1)	Uncorrectable (fatal)	This error occurs when a component does not receive update flow control credits with the 200 µs limit.
Malformed TLP	Uncorrectable (fatal)	This error is caused by any of the following conditions: • The data payload of a received TLP exceeds the maximum payload size. • The TD field is asserted but no TLP digest exists, or a TLP digest exists but the TD bit of the PCI Express request header packet is not asserted. • A TLP violates a byte enable rule. The Hard IP block checks for this violation, which is considered optional by the PCI Express specifications. • A TLP in which the `type` and `length` fields do not correspond with the total length of the TLP. • A TLP in which the combination of format and type is not specified by the PCI Express specification. • A request specifies an address/length combination that causes a memory space access to exceed a 4 KB boundary. The Hard IP block checks for this violation, which is considered optional by the PCI Express specification. • Messages, such as Assert_INTX, Power Management, Error Signaling, Unlock, and Set Power Slot Limit, must be transmitted across the default traffic class. The Hard IP block deletes the malformed TLP; it is not presented to the Application Layer.
Note: 1. Considered optional by the *PCI Express Base Specification Revision*.		

9.4. Error Reporting and Data Poisoning

How the Endpoint handles a particular error depends on the configuration registers of the device.

Refer to the *PCI Express Base Specification 3.0* for a description of the device signaling and logging for an Endpoint.

The Hard IP block implements data poisoning, a mechanism for indicating that the data associated with a transaction is corrupted. Poisoned TLPs have the error/poisoned bit of the header set to 1 and observe the following rules:

- Received poisoned TLPs are sent to the Application Layer and status bits are automatically updated in the Configuration Space.
- Received poisoned Configuration Write TLPs are not written in the Configuration Space.
- The Configuration Space never generates a poisoned TLP; the error/poisoned bit of the header is always set to 0.

Poisoned TLPs can also set the parity error bits in the PCI Configuration Space Status register.

Table 104. Parity Error Conditions

Status Bit	Conditions
Detected parity error (status register bit 15)	Set when any received TLP is poisoned.
Master data parity error (status register bit 8)	This bit is set when the command register parity enable bit is set and one of the following conditions is true: • The poisoned bit is set during the transmission of a Write Request TLP. • The poisoned bit is set on a received completion TLP.

Poisoned packets received by the Hard IP block are passed to the Application Layer. Poisoned transmit TLPs are similarly sent to the link.

Related Information

PCI Express Base Specification 3.0

9.5. Uncorrectable and Correctable Error Status Bits

The following section is reprinted with the permission of PCI-SIG. Copyright 2010 PCI-SIG.

Figure 53. Uncorrectable Error Status Register

The default value of all the bits of this register is 0. An error status bit that is set indicates that the error condition it represents has been detected. Software may clear the error status by writing a 1 to the appropriate bit.

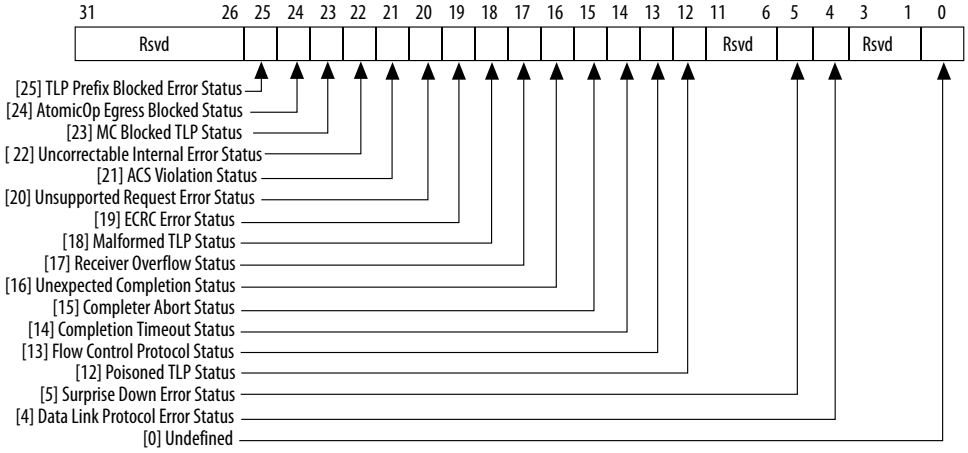
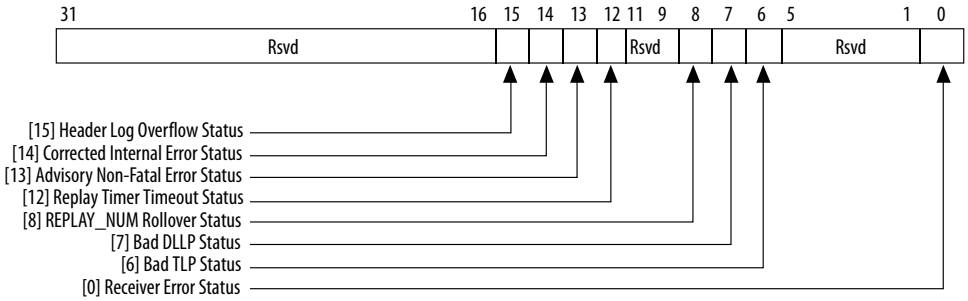


Figure 54. Correctable Error Status Register

The default value of all the bits of this register is 0. An error status bit that is set indicates that the error condition it represents has been detected. Software may clear the error status by writing a 1 to the appropriate bit.



10. IP Core Architecture

10.1. PCI Express Protocol Stack

The Intel Arria 10 Hard IP for PCI Express with SR-IOV 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 cyclical redundancy code (CRC) values and checks all CRCs 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 55. Intel Arria 10 Hard IP for PCI Express with SR-IOV

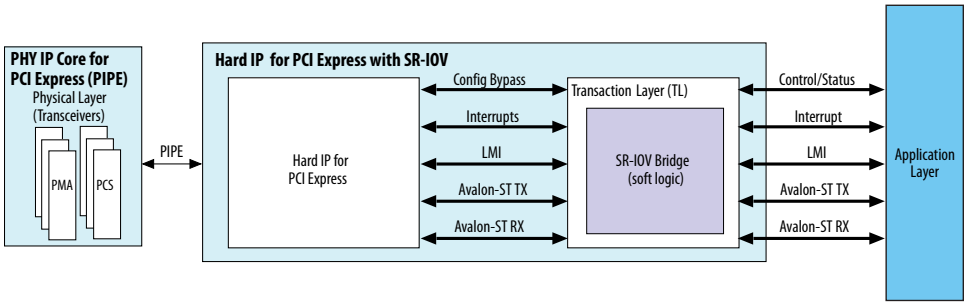


Table 105. Application Layer Clock Frequencies

Lanes	Gen1	Gen2	Gen3
×4	N/A	N/A	125 MHz @ 256 bits
×8	N/A	125 MHz @ 256 bits	250 MHz @ 256 bits

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Related Information

PCI Express Base Specification 3.0

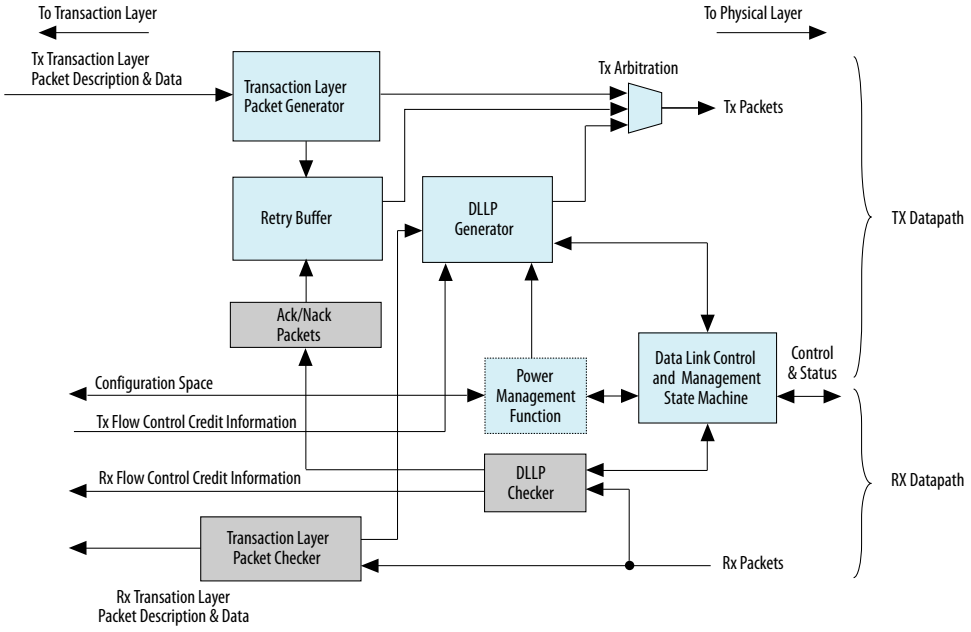
10.2. Data Link Layer

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 56. 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 Arria 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)

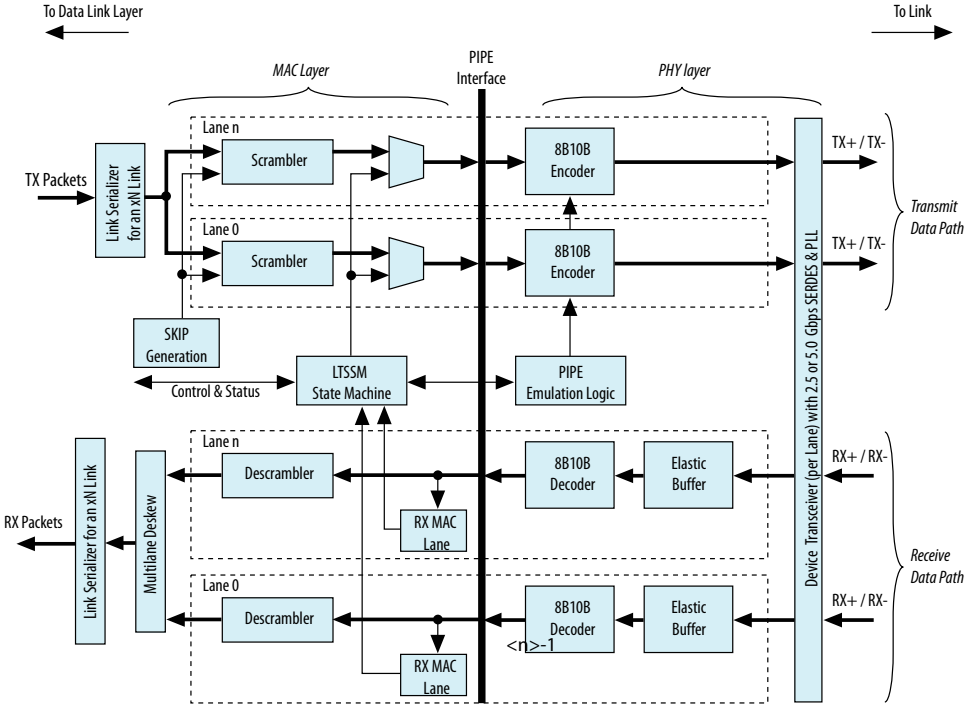
10.3. Physical Layer

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
- Scrambling/descrambling and 8B/10B encoding/decoding for 2.5 Gbps (Gen1) and 5.0 Gbps (Gen2) per lane
- Serializing and deserializing data
- Equalization (Gen3)
- Operating the PIPE 3.0 Interface
- Implementing auto speed negotiation (Gen2 and Gen3)
- Implementing auto speed negotiation (Gen2)
- Transmitting and decoding the training sequence
- Providing hardware autonomous speed control
- Implementing auto lane reversal

Figure 57. Physical Layer Architecture



PHY Layer—The PHY layer includes the 8B/10B encode and decode functions for Gen1 and Gen2. The PHY also includes elastic buffering and serialization/deserialization functions.

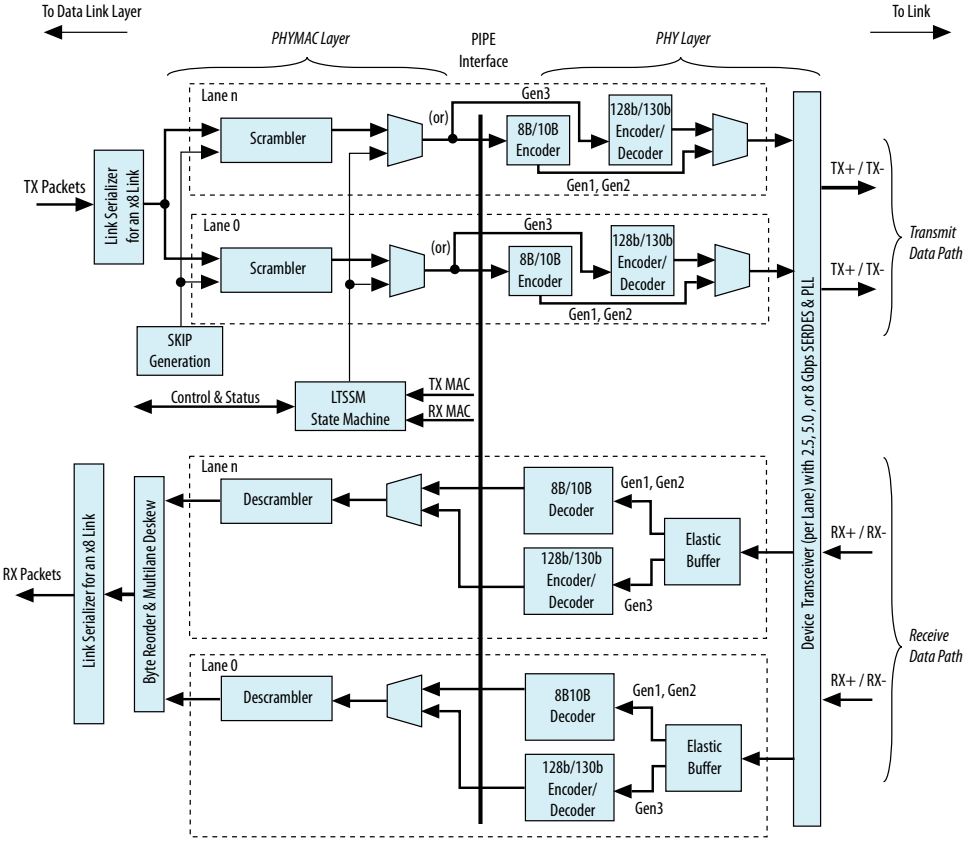
The Physical Layer is subdivided by the PIPE Interface Specification into two layers (bracketed horizontally in above figure):

- PHYMAC—The MAC layer includes the LTSSM and the scrambling/descrambling. byte reordering, and multilane deskew functions.
- Media Access Controller (MAC) Layer—The MAC layer includes the LTSSM and the scrambling and descrambling 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.
- PHY Layer—The PHY layer includes the 8B/10B encode and decode functions for Gen1 and Gen2. 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 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 58. 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.

10.4. Top-Level Interfaces

10.4.1. Avalon-ST Interface

An Avalon-ST interface connects the Application Layer and the Transaction Layer. This is a point-to-point, streaming interface designed for high throughput applications. The Avalon-ST interface includes the RX and TX datapaths.

For more information about the Avalon-ST interface, including timing diagrams, refer to the *Avalon Interface Specifications*.

RX Datapath

The RX datapath transports data from the Transaction Layer to the Application Layer's Avalon-ST interface. Masking of non-posted requests is partially supported. Refer to the description of the `rx_st_mask` signal for further information about masking.

TX Datapath

The TX datapath transports data from the Application Layer's Avalon-ST interface to the Transaction Layer. The Hard IP provides credit information to the Application Layer for posted headers, posted data, non-posted headers, non-posted data, completion headers and completion data.

The Application Layer may track credits consumed and use the credit limit information to calculate the number of credits available. However, to enforce the PCI Express Flow Control (FC) protocol, the Hard IP also checks the available credits before sending a request to the link, and if the Application Layer violates the available credits for a TLP it transmits, the Hard IP blocks that TLP and all future TLPs until credits become available. By tracking the credit consumed information and calculating the credits available, the Application Layer can optimize performance by selecting for transmission only the TLPs that have credits available.

10.4.2. Clocks and Reset

The *PCI Express Base Specification* requires an input reference clock, which is called `refclk` in this design. The *PCI Express Base Specification* stipulates that the frequency of this clock be 100 MHz.

The *PCI Express Base Specification* also requires a system configuration time of 100 ms. To meet this specification, IP core includes an embedded hard reset controller. This reset controller exits the reset state after the periphery of the device is initialized.

Related Information

Reset, Status, and Link Training Signals on page 62

10.4.3. Interrupts

The Hard IP for PCI Express offers the following interrupt mechanisms:

- Message Signaled Interrupts (MSI)— MSI uses the TLP single dword memory writes to to implement interrupts. This interrupt mechanism conserves pins because it does not use separate wires for interrupts. In addition, the single dword provides flexibility in data presented in the interrupt message. The MSI Capability structure is stored in the Configuration Space and is programmed using Configuration Space accesses. MSI interrupts are only supported for Physical Functions.
- MSI-X—The Transaction Layer 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 in contrast to the MSI capability structure, which contains all of the control and status information for the interrupt vectors. MSI-X interrupts are supported for Physical and Virtual Functions.
- Legacy interrupts—The `app_int_sts` port controls legacy interrupt generation. When `app_int_sts` is asserted, the Hard IP generates an Assert_INT<*n*> message TLP.
- MSI interrupts are only supported for Physical Functions.

10.4.4. PIPE

The PIPE interface implements the Intel-designed PIPE interface specification. You can use this parallel interface to speed simulation; however, you cannot use the PIPE interface in actual hardware.

- The simulation models support PIPE and serial simulation.
- For Gen3, the Intel BFM bypasses Gen3 Phase 2 and Phase 3 Equalization. However, Gen3 variants can perform Phase 2 and Phase 3 equalization if instructed by a third-party BFM.

Related Information

PIPE Interface Signals on page 68

10.5. Intel Arria 10 Hard IP for PCI Express with Single-Root I/O Virtualization (SR-IOV)

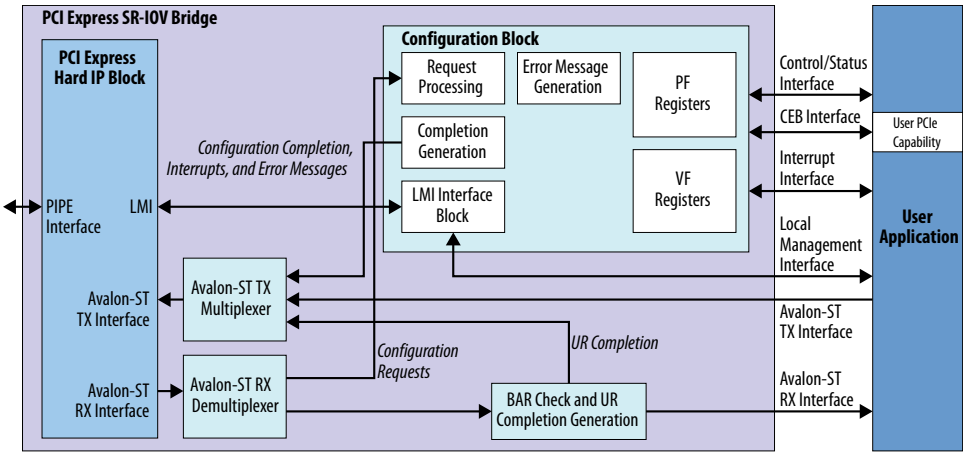
The Intel Arria 10 Hard IP for PCI Express with SR-IOV bypasses the Configuration Space and base address register (BAR) matching logic of the Hard IP. The SR-IOV bridge implements the following functions in soft logic:

- Configuration spaces for eight PCIe Physical Functions and 2048 Virtual Functions
- BAR checking logic
- Interrupt generation
- Error messages for Advanced Error Reporting (AER)
- Address Translation Services (ATS)
- TLP Processing Hints (TPH)
- Control Shadow Interface to read the current settings for some of the VF Control Register fields in the PCI and PCI Express Configuration Spaces

The SR-IOV processes memory requests, Completions and messages received from the link. It passes them to the Application Layer unmodified, using the Avalon-ST RX interface. The SR-IOV Bridge does not maintain any state for requests outstanding on the Master or Target sides. The RX interface delivers Completion TLPs to the Application Layer in the same order as received from the link. It does not match Completion TLPs with the outstanding requests from the Application Layer.

The following figure illustrates the SR-IOV Bridge logic and its interfaces to the Intel Arria 10 Hard IP for PCI Express and Application Layer.

Figure 59. Block Diagram of the Intel Arria 10 Hard IP for PCI Express IP with the SR-IOV Bridge



SR-IOV Bridge Logic Details

Soft logic in the SR-IOV Bridge decodes Configuration Space transactions on Avalon-ST interface via Configuration Bypass mode and forwards them to the internal Configuration Block. The Configuration Block implements the following logic:

- Processes incoming Configuration Space TLPs and generates Completions
- Includes Configuration Space registers of eight Physical Functions and 2048 Virtual Functions.
- Generates MSI, MSI-X, and Legacy Interrupts (`INTx Assert and Deassert`)
- Generates error messages for AER
- Multiplexes the following data sources to on the Avalon-ST TX interface:
  - Master-side requests and Target-side Completions generated by the Application Layer
  - UR Completions from the BAR Check block for Memory Read Requests
- Transmits Memory Read and Memory Write received on TX Avalon-ST interface from the Application Layer. The core forwards these requests to the host *as is*, without any checking for errors. Application Layer logic must make sure the transmitted memory requests satisfy all PCI Express requirements.

BAR Logic Details

The BAR block includes the following functions:

- Compares the addresses of received memory transactions to the BAR settings for the targeted function
- Identifies PF and VF BAR accesses
- Discards all memory transactions that are not in the address range of any of the configured BARs
- Generates Unsupported Request (UR) Completions for requests that fail the BAR check

Local Management Interface (LMI)

SR-IOV LMI logic accesses Configuration Space Registers of all Physical and Virtual Functions. The LMI logic accepts read and write requests from the Application Layer and directs requests to either the LMI interface of the Hard IP or the Configuration Registers in the Configuration Block.

Related Information

- Local Management Interface (LMI) Signals on page 60
- PCI Express Base Specification Revision 3.0

11. Design Implementation

Completing your design includes additional steps to specify analog properties, pin assignments, and timing constraints.

11.1. Making Pin Assignments to Assign I/O Standard to Serial Data Pins

Before running Quartus Prime compilation, use the **Pin Planner** to assign I/O standards to the pins of the device.

1. On the Quartus Prime **Assignments** menu, select **Pin Planner**. The **Pin Planner** appears.
2. In the **Node Name** column, locate the PCIe serial data pins.
3. In the **I/O Standard** column, double-click the right-hand corner of the box to bring up a list of available I/O standards.
4. Select the appropriate standard from the following table.

Table 106. I/O Standards for PCIe Pins

Pin Type	I/O Standard
PCIE REFCLK	**Current Mode Logic (CML)**, **HCSL**
PCIE RX	**Current Mode Logic (CML)** (5)
PCIE TX	**High Speed Differential I/O** (6)

The Quartus Prime software adds instance assignments to your Quartus Prime Settings File (`*.qsf`). The assignment is in the form `set_instance_assignment -name IO_STANDARD <"IO_STANDARD_NAME"> -to <signal_name>`. The `*.qsf` is in your synthesis directory.

Related Information

- Intel Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines
  For information about connecting pins on the PCB including required resistor values and voltages.
- Intel Cyclone 10 GX Device Family Pin Connection Guidelines
  For information about connecting pins on the PCB including required resistor values and voltages.

(5) AC coupling is required at the transmitter for the PCIE RX signals.

(6) AC coupling is required at the transmitter for the PCIE TX signals.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

11.2. Recommended Reset Sequence to Avoid Link Training Issues

Successful link training can only occur after the FPGA is configured. Designs using CvP for configuration initially load the I/O ring and periphery image. Intel Arria 10 devices include a Nios II Hard Calibration IP core that automatically calibrates the transceivers to optimize signal quality after CvP completes and before entering user mode. Link training occurs after calibration. Refer to *Reset Sequence for Hard IP for PCI Express IP Core and Application Layer* for a description of the key signals that reset, control dynamic reconfiguration, and link training.

Related Information

- Intel FPGA Intel Arria 10 Transceiver PHY IP Core User Guide
  For information about requirements for the CLKUSR pin used during automatic calibration.
- Intel FPGA Intel Cyclone 10 GX Transceiver PHY IP Core User Guide
  For information about requirements for the CLKUSR pin used during automatic calibration.

11.3. SDC Timing Constraints

Your top-level Synopsys Design Constraints file (`.sdc`) must include the following timing constraint macro for the Intel Arria 10 Hard IP for PCIe IP core.

Example 1. SDC Timing Constraints Required for the Intel Arria 10 Hard IP for PCIe and Design Example

```
# Constraints required for the Hard IP for PCI Express
# derive_pll_clock is used to calculate all clock derived
# from PCIe refclk. It must be applied once across all
# of the SDC files used in a project
derive_pll_clocks -create_base_clocks
```

You should only include this constraint in one location across all of the SDC files in your project. Differences between Fitter timing analysis and Timing Analyzer timing analysis arise if these constraints are applied multiple times.

Related Information

What assignments do I need for a PCIe Gen1, Gen2 or Gen3 design that targets an Arria 10 ES2, ES3 or production device?
Starting with the Quartus Prime Software Release 17.0, these assignments are automatically included in the design. You do not have to add them.

12. Debugging

As you bring up your PCI Express system, you may face a number of issues related to FPGA configuration, link training, BIOS enumeration, data transfer, and so on. This chapter suggests some strategies to resolve the common issues that occur during hardware bring-up.

12.1. Setting Up Simulation

Changing the simulation parameters reduces simulation time and provides greater visibility.

12.1.1. Changing Between Serial and PIPE Simulation

By default, the Intel testbench runs a serial simulation. You can change between serial and PIPE simulation by editing the top-level testbench file.

For Endpoint designs, the top-level testbench file is *<working_dir>*/*<instantiation_name>*_tb/*<instantiation_name>*_tb/sim/*<instantiation_name>*_tb.v

The `serial_sim_hwtcl` and `enable_pipe32_sim_hwtcl` parameters control serial mode or PIPE simulation mode. To change to PIPE mode, change `enable_pipe32_sim_hwtcl` to `1'b1` and `serial_sim_hwtcl` to `1'b0`.

Table 107. Controlling Serial and PIPE Simulations

Data Rates	Parameter Settings	
	serial_sim_hwtcl	enable_pipe32_sim_hwtcl
Serial simulation	1	0
PIPE simulation	0	1

12.1.2. Using the PIPE Interface for Gen1 and Gen2 Variants

Running the simulation in PIPE mode reduces simulation time and provides greater visibility.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Complete the following steps to simulate using the PIPE interface:

1. Go to the directory, <*testbench_dir*>/pcie_ed_tb/ip/pcie_ed_tb/DUT_pcie_tb_ip/sim/
2. Open DUT_pcie_tb_ip.v.
3. Search for the string, serial_sim_hwtcl. Set the value of this parameter to 0 if it is 1.
4. Save DUT_pcie_tb_ip.v.

12.1.3. Viewing the Important PIPE Interface Signals

You can view the most important PIPE interface signals, txdata, txdatak, rxdata, and rxdatak at the following level of the design hierarchy: altpcie_<*device*>_hip_pipen1b|twentynm_hssi_<*gen*>_<*lanes*>_pcie_hip.

12.1.4. Disabling the Scrambler for Gen1 and Gen2 Simulations

The encoding scheme implemented by the scrambler applies a binary polynomial to the data stream to ensure enough data transitions between 0 and 1 to prevent clock drift. The data is decoded at the other end of the link by running the inverse polynomial.

Complete the following steps to disable the scrambler:

1. Open <*work_dir*>/<*variant*>/testbench/<*variant*>_tb/simulation/submodules/altpcie_tbed_<*dev*>_hwtcl.v.
2. Search for the string, test_in.
3. To disable the scrambler, set test_in[2] = 1.
4. Save altpcie_tbed_sv_hwtcl.v.

12.1.5. Disabling 8B/10B Encoding and Decoding for Gen1 and Gen2 Simulations

You can disable 8B/10B encoding and decoding to facilitate debugging.

For Gen1 and Gen2 variants, you can disable 8B/10B encoding and decoding by setting test_in[2] in <*testbench_dir*>/pcie_<*dev*>_hip_avmm_bridge_example_design_tb/ip/pcie_example_design_tb/DUT_pcie_tb_ip/altera_pcie_<*dev*>_tbed_<*ver*>/sim/altpcietb_bfm)top_rp.v.

12.2. 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.

Make the changes shown in the following table for the 32-bit PIPE interface.

Table 108. Changes for 32-Bit PIPE Interface

8-Bit PIPE Interface	32-Bit PIPE Interface
`output wire [7:0] pcie_a10_hip_0_hip_pipe_txdata0`	`output wire [31:0] pcie_a10_hip_0_hip_pipe_txdata0`
`input wire [7:0] pcie_a10_hip_0_hip_pipe_rxdata0`	`input wire [31:0] pcie_a10_hip_0_hip_pipe_rxdata0`
`output wire pcie_a10_simulation_inst_pcie_a10_hip_0_hip_pipe_txdatak0`	`output wire [3:0] pcie_a10_simulation_inst_pcie_a10_hip_0_hip_pipe_txdatak0`
`input wire pcie_a10_simulation_inst_pcie_a10_hip_0_hip_pipe_rxdatak0`	`input wire [3:0] pcie_a10_simulation_inst_pcie_a10_hip_0_hip_pipe_rxdatak0`

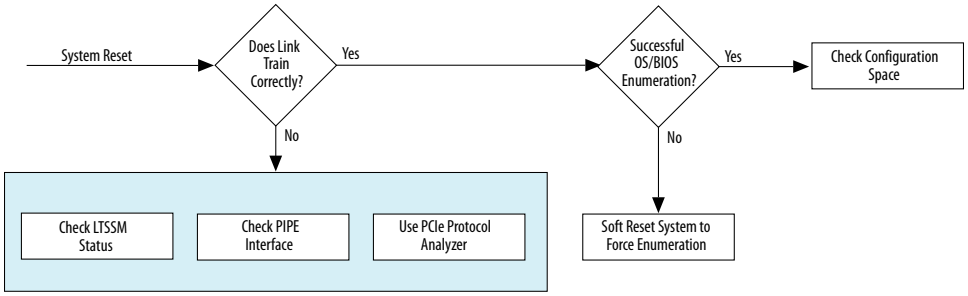
12.3. Hardware Bring-Up Issues

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 60. Debugging Link Training Issues



12.4. 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.

Related Information

Reset, Status, and Link Training Signals on page 62

12.5. Creating a Signal Tap Debug File to Match Your Design Hierarchy

For Intel Arria 10 and Intel Cyclone 10 GX devices, the Intel Quartus Prime software generates two files, `build_stp.tcl` and `<ip_core_name>.xml`. You can use these files to generate a Signal Tap file with probe points matching your design hierarchy.

The Intel Quartus Prime software stores these files in the `<IP core directory>/synth/debug/stp/` directory.

Synthesize your design using the Intel Quartus Prime software.

1. To open the Tcl console, click **View ➤ Utility Windows ➤ Tcl Console**.
2. Type the following command in the Tcl console:
   `source <IP core directory>/synth/debug/stp/build_stp.tcl`
3. To generate the STP file, type the following command:
   `main -stp_file <output stp file name>.stp -xml_file <input xml_file name>.xml -mode build`
4. To add this Signal Tap file (**.stp**) to your project, select **Project ➤ Add/Remove Files in Project**. Then, compile your design.
5. To program the FPGA, click **Tools ➤ Programmer**.
6. To start the Signal Tap Logic Analyzer, click **Quartus Prime ➤ Tools ➤ Signal Tap Logic Analyzer**.

   The software generation script may not assign the Signal Tap acquisition clock in `<output stp file name>.stp`. Consequently, the Intel Quartus Prime software automatically creates a clock pin called `auto_stp_external_clock`. You may need to manually substitute the appropriate clock signal as the Signal Tap sampling clock for each STP instance.
7. Recompile your design.
8. To observe the state of your IP core, click **Run Analysis**.

   You may see signals or Signal Tap instances that are red, indicating they are not available in your design. In most cases, you can safely ignore these signals and instances. They are present because software generates wider buses and some instances that your design does not include.

12.6. 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.

12.7. 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.

13. Document Revision History

13.1. Document Revision History for the Intel Arria 10 Avalon Streaming with SR-IOV IP for PCIe User Guide

Date	Version	Changes Made
2022.01.11	20.4	Corrected a bit value in *Figure 63. Memory Read Request, 64-Bit Addressing*.
2020.12.14	20.4	Added notes to the *SR-IOV System Settings* section in the *Parameter Settings* chapter to tie the VF assignment granularity restriction and user capabilities registers implementation recommendation to the appropriate parameters.
2020.11.30	20.3	Removed Tables for the PCI Express data throughput and recommended speed grades for Gen2 x4 mode from the *Datasheet* chapter because this mode is not supported by the Intel Arria 10 Avalon Streaming with SR-IOV IP for PCI Express.
2020.10.19	20.3	Added a note clarifying the granularity restriction for assigning VFs to the *Intel Arria 10 SR-IOV System Settings* section. Added a note that includes the recommendation on how to implement user capabilities registers to the *Configuration Extension Bus (CEB) Interface* section.
2020.10.05	20.3	Added the *Configuration Extension Bus (CEB) Interface* section to the *Interfaces and Signal Descriptions* chapter. Updated the VF configuration space register mapping in the *Virtual Function Registers* section.
2019.12.19	18.0.1	Updated the signal descriptions and timing diagrams for the Function-Level Reset (FLR) Interface.
2019.09.20	18.0.1	Updated the maximum number of Physical Functions (PFs) supported from four to eight.
2019.04.08	18.0	Removed the duplicate *PCI Express Protocol Stack* section.
2018.12.28	18.0	Added the note stating that the non-posted tag pool is shared across all enabled physical functions.
2018.08.13	18.0	Added the step to invoke Vsim to the instructions for running ModelSim simulations.
2017.12.12	17.1	Made the following changes: • Corrected *Feature Comparison for all Hard IP for PCI Express IP Core* table: The Avalon-MM DMA interface does not automatically handle out-of-order completions. • Added **Enable TX-polarity inversion soft logic** parameter. • Changed the design example in the *Intel Arria 10 SR-IOV Quick Start Guide* to `sriov2_top_target_gen3x8_1pf_4vf.qsys`. The previous example is no longer available. • Removed reference to the PFs in the description of the Control Shadow interface. This interface is only for VFs. • Corrected the description of VF offsets in the *VF Offset and Stride Registers* table.
2017.05.30	17.0	Made the following changes to the user guide:

continued...

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Date	Version	Changes Made
		• Removed links to the *What assignments do I need for a PCIe Gen1, Gen2 or Gen3 design that targets an Intel Arria 10 ES2, ES3 or production device?* answer. These assignment are already included in the 17.0 release.
2017.05.15	17.0	Made the following changes to the IP core: • Added option soft DFE Controller IP on the **PHY** tab of the parameter editor to improve BER margin. The default for this option is off because it is typically not required. Short reflective links may benefit from this soft DFE controller IP. This parameter is available only for Gen3 configurations. Made the following changes to the user guide: • Updated *PCI Express Gen3 Bank Usage Restrictions* status. These restrictions affect all Aria 10 ES and production devices. • Added statement that Intel Arria 10 devices do not support the **Create timing and resource estimates for third-party EDA synthesis tools** option on the **Generate ➤ Generate HDL** menu. • Added cross references to data alignment and timing diagrams for the Avalon-ST interface. • Corrected definitions of `rx_par_err`, `tx_par_err`, and `cfg_par_err`. These errors are not recorded in the VSEC register. The VSEC register is not supported. • Corrected minor errors and typos.
2017.03.15	17.0	Rebranded as Intel.
2016.10.31	16.1	Made the following changes to the IP core: • Added final support for the 128-bit interface to the user application. • Added support for the Intel FPGA IP Evaluation Mode. • Added licensing requirement. • Changed timing model support from preliminary to final for most Intel Arria 10 device packages. Exceptions include some military and automotive speed grades with the extended temperature range. Made the following changes to the user guide: • Corrected the number of tags supported in the *Feature Comparison for all Hard IP for PCI Express IP Cores* table. • Removed recommendations about connecting `pin_perst`. These recommendations do not apply to Arria 10 devices. • Changed the recommended value of `test_in[31:0]` from 0xa8 to 0x188. • Added a note saying that all 4 PFs must be programmed to support a **Maximum payload size** value greater than 128 bytes. • Added -3 to recommended speed grades for the 125 MHz interface.
2016.05.02	16.0	Redesigned the SR-IOV bridge. The 16.0 SR-IOV bridge includes the following changes: • Changed number of supported PFs from 2 to 4. • Changed number of supported VFs from 128 to 2048. • Redesigned BAR matching logic, interrupt generation, and error reporting functions • Added support for ATS, TPH, including register definitions • Added Control Shadow interface to read the current settings for some of the VF Control Register fields in the PCI and PCI Express Configuration Spaces • Removed 128-bit interface to the Application Layer and Gen1 support. • Redesigned interface to the Application Layer. • Updated address map to reflect the fact that ARI is required, TPH, and ATS registers. • Added register definitions for TLP Processing Hints (TPH) and Address • Updated bit definitions for the following registers: — ARI Enhanced Capability (0x100) — VF Offset and Stride Registers (0x194) to reflect 4 PFs

continued...

Date	Version	Changes Made
		Added the following new interfaces: • Avalon-ST sideband interface to distinguish VF and PF traffic • Function Number and BAR Identification Signals • Updated SR-IOV Parameter Settings chapter to include the new parameters necessary to support the new functionality. • Updated figures in *Physical Layout of Hard IP in Intel Arria 10 Devices* to include more detail about transceiver banks and channel restrictions. • Updated default value of many registers to state that they can be set in Platform Designer. • Restored the *Transaction Layer Packet (TLP) Header Formats* appendix. • Added transceiver bank usage placement restrictions for Gen3 ES3 devices. • Removed support for -3 speed grade devices. • Corrected minor errors and typos.
2015.11.02	15.1	Made the following changes: • Improved component GUI that simplifies parameterization. Among the changes is a new single parameter, HIP mode that combines all supported data rates, interface widths and frequencies as a single parameter. • Added Development Kit interface signal definitions. • Removed **Legacy Endpoint** from **Port type** parameter options. The Legacy Endpoint is no longer supported for Intel Arria 10 devices. • Revised discussion on possible conflict between LMI writes and Host writes to the Configuration Space. • Replaced previous *Getting Started* chapter with a simplified *Intel Arria 10 SR-IOV Quick Start Guide* chapter. • Corrected *Component Specific TX Credit* signal definitions. • Added definitions for missing signals and corrected *Intel Arria 10 PCIe with SR-IOV Signals and Interfaces* figure: — Removed `rx_st_err` — Added `tx_cred_cons_sel` to TX credit interface — Added Hard IP Status interface signals to figure • Fixed minor errors and typos.
2015.06.05	15.0	Added note in Physical Layout of Hard IP in Intel Arria 10 Devices to explain Intel Arria 10 design constraint that requires that if the lower HIP on one side of the device is configured with a Gen3 x4 or Gen3 x8 IP core, and the upper HIP on the same side of the device is also configured with a Gen3 IP core, then the upper HIP must be configured with a x4 or x8 IP core.
2015.05.04	15.0	• Added **Enable Intel Debug Master Endpoint (ADME)** parameter to support optional Native PHY register programming with the Intel System Console. • Enhanced descriptions of channel placement, added fPLL placement for Gen1 and Gen2 data rates, and added master CGB location, in Hard IP Block Placement In Intel Arria 10 Devices on page 30. • Removed Migration and TLP Format appendices, and added new appendix Frequently Asked Questions on page 0 . • Added column for Avalon-ST Interface with SR-IOV variations in Feature Comparison for all Hard IP for PCI Express IP Cores table in *Features* section. • Reorganized sections in Debugging on page 126. • Updated information in SDC Timing Constraints on page 125. • Removed erroneous mention of `tl_cfg_ctl` signal, which is not available in this IP core. • Removed list of static example designs from *Design Examples*. You can derive the list from the installation directory where example designs are available. • Fixed minor errors and typos.
2014.12.15	14.1	Initial release.

A. Transaction Layer Packet (TLP) Header Formats

The following sections show the TLP header formats for TLPs without a data payload, and for those with a data payload.

A.1. TLP Packet Formats without Data Payload

The following figures show the header format for TLPs without a data payload.

Figure 61. Memory Read Request, 32-Bit Addressing

Memory Read Request, 32-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[31:2]																														0	0
Byte 12	Reserved																															

Figure 62. Memory Read Request, Locked 32-Bit Addressing

Memory Read Request, Locked 32-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	0	1	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[31:2]																														0	0
Byte 12	Reserved																															

Figure 63. Memory Read Request, 64-Bit Addressing

Memory Read Request, 64-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	1	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[63:32]																															
Byte 12	Address[31:2]																														0	0

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Figure 64. Memory Read Request, Locked 64-Bit Addressing

Memory Read Request, Locked 64-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	1	0	0	0	0	1	0	TC			0	0	0	0	T	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[63:32]																															
Byte 12	Address[31:2]																														0	0

Figure 65. Configuration Read Request Root Port (Type 1)

Configuration Read Request Root Port (Type 1)

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Bus Number								Device No					Func			0	0	0	0	Ext Reg				Register No						0	0
Byte 12	Reserved																															

Figure 66. I/O Read Request

I/O Read Request

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Address[31:2]																														0	0
Byte 12	Reserved																															

Figure 67. Message without Data

Message without Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	1	1	0	r 2	r 1	r 0	0	TC			0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Byte 4	Requester ID																Tag								Message Code							
Byte 8	Vendor defined or all zeros																															
Byte 12	Vendor defined or all zeros																															

Note:

(1) Not supported in Avalon-MM.

Figure 68. Completion without Data

Completion without Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	1	0	1	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Completer ID																Status			B	Byte Count											
Byte 8	Requester ID																Tag								0	Lower Address						
Byte 12	Reserved																															

Figure 69. Completion Locked without Data

Completion Locked without Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	1	0	1	1	0	TC				0	0	0	0	TD	EP	Attr			0	0	Length							
Byte 4	Completer ID																Status				B	Byte Count										
Byte 8	Requester ID																Tag								0	Lower Address						
Byte 12	Reserved																															

A.2. TLP Packet Formats with Data Payload

Figure 70. Memory Write Request, 32-Bit Addressing

Memory Write Request, 32-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	0	0	0	0	0	TC				0	0	0	0	TD	EP	Attr			0	0	Length							
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[31:2]																														0	0
Byte 12	Reserved																															

Figure 71. Memory Write Request, 64-Bit Addressing

Memory Write Request, 64-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	1	0	0	0	0	0	0	TC				0	0	0	0	TD	EP	Attr			0	0	Length							
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[63:32]																															
Byte 12	Address[31:2]																														0	0

Figure 72. Configuration Write Request Root Port (Type 1)

Configuration Write Request Root Port (Type 1)

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Bus Number								Device No								0	0	0	0	Ext Reg				Register No						0	0
Byte 12	Reserved																															

Figure 73. I/O Write Request

I/O Write Request

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Address[31:2]																														0	0
Byte 12	Reserved																															

Figure 74. Completion with Data

Completion with Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	1	0	1	0	0	TC			0	0	0	0	TD	EP	Attr			0	0	Length								
Byte 4	Completer ID																Status			B	Byte Count											
Byte 8	Requester ID																Tag								0	Lower Address						
Byte 12	Reserved																															

Figure 75. Completion Locked with Data

Completion Locked with Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	1	0	1	1	0	TC			0	0	0	0	TD	EP	Attr			0	0	Length								
Byte 4	Completer ID																Status			B	Byte Count											
Byte 8	Requester ID																Tag								0	Lower Address						
Byte 12	Reserved																															

Figure 76. Message with Data

Message with Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	1	1	0	r 2	r 1	r 0	0	TC			0	0	0	0	TD	EP	0	0	0	0	Length									
Byte 4	Requester ID																Tag								Message Code							
Byte 8	Vendor defined or all zeros for Slot Power Limit																															
Byte 12	Vendor defined or all zeros for Slots Power Limit																															

B. Intel Arria 10 Avalon-ST with SR-IOV Interface for PCIe Solutions User Guide Archive

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
17.0	Intel Arria 10 Avalon-ST with SR-IOV Interface for PCIe Solutions User Guide
16.1.1	Intel Arria 10 Avalon-ST with SR-IOV Interface for PCIe Solutions User Guide
16.1	Intel Arria 10 Avalon-ST with SR-IOV Interface for PCIe Solutions User Guide
16.0	Intel Arria 10 Avalon-ST with SR-IOV Interface for PCIe Solutions User Guide
15.1	Intel Arria 10 Avalon-ST with SR-IOV Interface for PCIe Solutions User Guide
15.0	Intel Arria 10 Avalon-ST with SR-IOV Interface for PCIe Solutions User Guide
14.1	Intel Arria 10 Avalon-ST with SR-IOV Interface for PCIe Solutions User Guide

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Development Software ***category:***

Click to view products by Intel ***manufacturer:***

Other Similar products are found below :

RAPPID-567XFSW SRP004001-01 SW163052 SYSWINEV21 Core429-SA WS01NCTF1E W128E13 SW89CN0-ZCC IPS-EMBEDDED IP-UART-16550 MPROG-PRO535E AFLCF-08-LX-CE060-R21 WS02-CFSC1-EV3-UP SYSMAC-STUDIO-EIPCPLR LIB-PL-PC-N-1YR-DISKID LIB-PL-A-F SW006026-COV 1120270005 1120270006 MIKROBASIC PRO FOR FT90X (USB DONGLE) MIKROC PRO FOR FT90X (USB DONGLE) MIKROC PRO FOR PIC (USB DONGLE LICENSE) MIKROBASIC PRO FOR AVR (USB DONGLE LICEN MIKROBASIC PRO FOR FT90X MIKROC PRO FOR DSPIC30/33 (USB DONGLE LI MIKROPASCAL PRO FOR ARM (USB DONGLE LICE MIKROPASCAL PRO FOR FT90X MIKROPASCAL PRO FOR FT90X (USB DONGLE) MIKROPASCAL PRO FOR PIC32 (USB DONGLE LI SW006021-2H ATATMELSTUDIO 2400573 2702579 2988609 2702546 SW006022-DGL 2400303 2701356 VDSP-21XX-PCFLOAT VDSP-BLKFN-PC-FULL 88970111 DG-ACC-NET-CD 55195101-102 SW1A-W1C MDK-ARM PCI-EXP1-E3-US PCI-T32-E3-US SW006021-2NH SW006021-1H SW006021-2