PCIe Controller and IPS for AXI4

User Guide

Version 1.09

07 November 2023

CADENCE CONFIDENTIAL

© 2023 Cadence Design Systems, Inc. All rights reserved.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This document is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this document, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this document may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This document contains the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only in accordance with, a written agreement between Cadence and its customer.

Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this document subject to the following conditions:

1. This document may not be modified in any way.

2. Any authorized copy of this document or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.

3. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND DOES NOT REPRESENT A COMMITMENT ON THE PART OF CADENCE. EXCEPT AS MAY BE EXPLICITLY SET FORTH IN A WRITTEN AGREEMENT BETWEEN CADENCE AND ITS CUSTOMER, CADENCE DOES NOT MAKE, AND EXPRESSLY DISCLAIMS, ANY REPRESENTATIONS OR WARRANTIES AS TO THE COMPLETENESS, ACCURACY OR USEFULNESS OF THE INFORMATION CONTAINED IN THIS DOCUMENT. CADENCE DOES NOT WARRANT THAT USE OF SUCH INFORMATION WILL NOT INFRINGE ANY THIRD PARTY RIGHTS, AND CADENCE DISCLAIMS ALL IMPLIED WARRANTIES, INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. CADENCE DOES NOT ASSUME ANY LIABILITY FOR DAMAGES OR COSTS OF ANY KIND THAT MAY RESULT FROM USE OF SUCH INFORMATION. CADENCE CUSTOMER HAS COMPLETE CONTROL AND FINAL DECISION-MAKING AUTHORITY OVER ALL ASPECTS OF THE DEVELOPMENT, MANUFACTURE, SALE AND USE OF CUSTOMER'S PRODUCT, INCLUDING, BUT NOT LIMITED TO, ALL DECISIONS WITH REGARD TO DESIGN, PRODUCTION, TESTING, ASSEMBLY, QUALIFICATION, CERTIFICATION, INTEGRATION OF CADENCE PRODUCTS, INSTRUCTIONS FOR USE, LABELING AND DISTRIBUTION, AND CADENCE EXPRESSLY DISAVOWS ANY RESPONSIBILITY WITH REGARD TO ANY SUCH DECISIONS REGARDING CUSTOMER'S PRODUCT.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

About this Document....viii

1 Overview of the IP....12

1.1 Overview....12

1.2 System Overview of the Controller IP....12

1.3 Core Configuration....13

2 Architecture Overview....15

2.1 Architectural Block Diagram....15

2.2 Controller PCS....16

2.3 PHY Logical Layer....16

2.4 Data Link Layer....17

2.5 Transaction Layer....17

2.5.1 TL Receive Side Operation....18

2.5.2 TL Transmit Side Operation....18

2.6 Host Adaptation Layer Operation....18

2.7 Application Layer....18

3 Signal List and Description....20

3.1 Pin List Description....20

4 Clocks and Reset....21

4.1 IP Clocks....21

4.1.1 Fixed Core Clock (FCC)....21

4.1.2 PM Clock Frequency....22

4.1.3 Reg Clock Usage....22

4.2 Reset Signals....23

4.2.1 Reset Outputs....23

4.2.2 *link_down_handling_in_progress*....23

4.2.3 Hot Reset Out....23

5 IP Register Interface....25

5.1 Overview....25

5.2 Register Address Map....26

5.2.1 IP Controller Registers....27

5.3 IP Register Access....27

5.3.1 AXI-Lite Interface (B Path)....27

5.3.2 APB Interface (B Path)....28

5.4 IP Registers Access through the PCIe Link....29

5.5 Config Snoop Interface....29

5.5.1 Inbound CFG Write Request Flow (A Path)....30

5.5.2 Inbound CFG Read Request Flow (A Path) ... 31

6 Registers ... 32
6.1 Register Description ... 32

7 Controller and PHY Integration Solution ... 33
7.1 Controller IP Integrated Protocol Stack (IPS) ... 33

8 Application Interface AXI4 ... 34
8.1 AXI Overview ... 34
8.2 AXI Subordinate ... 34
8.2.1 AXI Subordinate Operation Types ... 34
8.2.2 AXI Subordinate Write Operations ... 35
8.2.3 AXI Subordinate Read operations ... 38
8.2.4 Tag Management for Non-Posted Transactions ... 40
8.2.5 Application Layer Read Size ... 40
8.2.6 PF/VF Encoding ... 41
8.2.7 AXI Subordinate Access ... 44
8.2.8 AXI Subordinate Access through Regions ... 44
8.2.9 AXI Subordinate Access through Sideband Descriptor ... 46
8.2.10 AXI Subordinate Access Note ... 46
8.2.11 AXI Subordinate Access by Examples ... 47
8.2.12 AXI Subordinate (Error) Response ... 52
8.3 AXI Manager ... 54
8.3.1 PCIe to AXI Address Translation ... 54
8.3.2 AXI Manager Write Access ... 56
8.3.3 AXI Manager Read Access ... 60
8.4 AXI Link-Down Handling ... 66
8.5 Atomic Operations ... 67
8.5.1 PCIe Atomic Operations ... 67
8.5.2 AXI-5 Atomic Transactions ... 67
8.5.3 AtomicOp on AXI Subordinate ... 70
8.5.4 AtomicOp on AXI Manager ... 71
8.6 Credit Availability Interface ... 71
8.6.1 Overview ... 71
8.6.2 Credits Scaling ... 72
8.6.3 Credit Threshold Register ... 72
8.6.4 Link Down and Low Power Mode ... 72
8.6.5 DMA Credit-Based Flow ... 72

9 Miscellaneous Interfaces ... 73

9.1 Interrupt Interface....73
9.1.1 INTx or Legacy Interrupt Interface....73
9.1.2 MSI-X Interrupts....73
9.1.3 MSI Interface....73
9.2 FLR Handling....74
9.2.1 HLS FLR Handling....74

10 Power Management....76
10.1 Overview....76
10.2 Function Power States....76
10.2.1 Function Power State Change Handshake Interface....76
10.3 PCIe Link Power States....77
10.3.1 L0s Link Power State....77
10.3.2 L1 Link Power State....77
10.3.3 Debug Registers for L1....79
10.3.4 L1 Substate....80
10.3.5 L1.1 Operation....82
10.3.6 L1.2 Operation....83
10.3.7 Register Details for L1-Substate....85
10.3.8 General Integration Details for L1 Substates....87
10.3.9 L2 and L3....92

11 Synthesis and Physical Design Guidelines....94
11.1 Synthesis Setup....94
11.2 Technology Specific Model....94
11.3 Clock MUX for CPCS Module....94
11.4 Handling Asynchronous FIFOS in Physical design....94
11.5 Synthesis App notes....95

12 Integration Guide....97
12.1 Start-Up Sequence....97
12.2 Reset Sequence....97
12.2.1 Reset Reference Block Diagram....98
12.3 Precision Time Measurement....98
12.3.1 Features....99
12.3.2 Functional Description....99
12.4 SRAM Integration....102
12.5 Vital Product Data (VPD) Integration....103
12.5.1 VPD Read Example....103
12.5.2 VPD Write Example....104
12.6 Hot-Plug Integration....104

12.6.1 Overview....104
12.6.2 Signal Description....106
12.6.3 Cadence PCIe Controller Hot Plug Implementation....108
12.6.4 Register Description....110
12.7 PF/VF Encoding at IP Interfaces....121
12.7.1 HLS PF/VF Encoding....121

13 Programming Guide....124
13.1 Programming IP Registers....124
13.2 Software Driver Setup....124
13.3 Link Equalization Setup....124
13.3.1 Gen3 Link Equalization Setup in RP Mode....124
13.3.2 Gen4 Link Equalization Setup in RP Mode....125
13.3.3 **Gen5 Link Equalization Setup in RP Mode**....125
13.3.4 Gen3 Link Equalization Setup in EP Mode....125
13.3.5 Gen4 Link Equalization Setup in EP Mode....125
13.3.6 Gen5 Link Equalization Setup in EP Mode....126
13.4 Error Handling....126
13.4.1 Receive Side Errors (Inbound)....126
13.4.2 Malformed Errors....128
13.4.3 HLS Outbound Errors....128

14 Debugging Guide....129
14.1 Link Training Debug....129
14.1.1 Link Training Status Registers....129
14.1.2 Link Error Status Registers....132
14.1.3 Deskew Status Registers....143
14.1.4 Link Equalization Control Setup Register....144
14.1.5 Link Equalization Debug Status Registers....163
14.1.6 LTSSM Timeout Control Registers....170
14.1.7 LTSSM Transition Freeze, Interrupt Control Status Registers....172
14.1.8 LTSSM Transition Cause Status Registers....178
14.1.9 7-bit LTSSM State Encoding....180
14.1.10 L0->Recovery LTSSM State Transition Cause Encoding....183
14.1.11 LTSSM Transition Encoding....184
14.1.12 LTSSM State Encoding....202

15 Verification....207
15.1 Verification Environment....207
15.2 Verification Overview....207

Contents

16 Appendix C: PCIe Configuration Capabilities Linked List....209
16.1 PCIe Controller Capabilities Linked List Implementation.... 209

17 Appendix G....211
17.1 Differences between the Released Versions.... 211

About this Document

This User Guide describes the interface and operational details for integration of the Cadence PCIe Controller IP (HPA) into a target design. The document assumes the readers have a good understanding of the PCI Express Specifications . Refer to the *References and Related Documents*.

Important: *"Cadence is committed to using respectful language in our code and communications. We are also active in the removal and/or replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related product/protocol terminology can be updated without end-user impact."*

Intended Audience

This document is written to help architects and procurement/engineering management who are involved in the System-on-a-Chip (SoC) design.

Delivery Package Information

- For information on the Signal List and Description, refer to the `<package_path>/docs/cdnhpa_ips_IO_descriptions.html` file of the delivery package.
- For information on the SRAM, refer to the `<package_path>/docs/cdnhpa_ips_sram_requirements.html` file of the delivery package.
- For information on the Register Description, refer to the `<package_path>/models/ipxact/*` directory of the delivery package.
- For information on the AXI ID Pool, refer to the `<package_path>/docs/cdnhpa_ips_axi_m_axid.html` file of the delivery package

To open the html files from linux, type:

```
<web_browser_name> <filename>.html
```

References and Related Documents

- PCI Express® Base Specifications Revision 1.0, 2.0, 3.0, 4.0.
- PCI Express® Base Specifications Revision 5.0 Version 1.0.
- PHY Interface for the PCI Express Architecture (PIPE) Specification (Intel Corporation), Version 5.2.
- AMBA® AXI and ACE Protocol Specification Issue E.
- AMBA® DTI (*Distributed Translation Interface*) Protocol Specification Version 1.0.
- AMBA® APB Protocol Specification Issue C.

Acronyms and Abbreviations

The following acronyms and abbreviations are used in this document.

Table 1: Acronyms and Abbreviations

Term	Definition
AER	Advanced Error Reporting
AES	Advanced Encryption Standard
ASPM	Active State Power Management
CRC	Cyclic Redundancy Check
DLLP	Data Link Layer Packet
EOP	End Of Packet
FLR	Function-Level Reset

Table 1: Acronyms and Abbreviations

Term	Definition
HAL	Host Adaption Layer. Application interface
HLS	HAL Stream interface (CXS Based)
HPA	High Performance Architecture (Second generation of cadence PCIe IP)
IDO	ID-based Ordering
LCRC	Link Cyclic Redundancy Check
LGA	Legacy Architecture (First generation of cadence PCIe IP)
LTR	Latency Tolerance Reporting
LTSSM	Link Training and Status State Machine
OBFF	Optimized Buffer Flush/Fill
PCIe	Peripheral Component Interface Express
PNP	Posted/Non-Posted
RCD	Restricted CXL Device mode. Formerly known as CXL1.1 Device mode. Limitations include lack of hot-plug support and only supporting 68B flits. Enumerates as an RCiEP.
RCH	Restricted CXL Host mode. Formerly known as CXL1.1 Host mode.
SOP	Start Of Packet
TD	TLP Digest field
TLP	Transaction Layer Packet
TS	Trailer Size field
UR	Unsupported request

Getting Help

If you have a problem using this product or the documentation, you can submit a SR (*Service Request*) to Cadence Support. When doing so, you must provide:

- Enough information about the problem so that it can be investigated efficiently.
- Describe the problem in full, give the version of the IP you are using, and state the exact circumstances in which the problem occurs.

Note:

For queries on how to use Cadence products, refer to the product documentation that is installed on your network.

Service Requests

SRs are your way of giving feedback, asking questions, getting solutions, and reporting problems. Unless told otherwise,Cadence support staff will respond to your service request. If Cadence Support cannot answer your question, Cadence Research and Development personnel will get involved.

Important:

Ensure that you specify the severity level of the service request as accurately as possible.

There are three levels of severity:

- **Critical**: You cannot proceed without a solution to the issue.
- **Important**: You can proceed, but you need a solution to the issue.
- **Minor**: You prefer to have a solution, but you can wait for it.

Note:

Note:

You can request support to increase the severity level of an issue. Therefore, do not use **Critical** unless immediate resolution of an issue is absolutely necessary and urgently required.

Using Cadence Online Support

Cadence encourages you to submit requests using *Cadence Online Support*. With Cadence Online Support you can also track your open service requests.

To use *Cadence Online Support* to submit a service request:

1. If you do not have a Cadence Online Support account, go to http://support.cadence.com and under ***New User?***, click ***Register Now***.

On the ***Registration for a New Cadence Account page***, read the instructions clearly and enter your details to register.

2. Log in to Cadence Online Support, and on the ***Support Home*** page, under ***My Requests***, click ***CREATE CASE***.

Additional information

For details on the other cores in the Cadence PCIe family, contact the Cadence customer support. For more information on Cadence and its products, visit: http://www.cadence.com.

Chapter 1: Overview of the IP

1.1 Overview

The Cadence PCIe family consists of a set of configurable PCIe controller cores designed for use in ASIC devices to provide an industry standard interface to the PCIe link. This document describes the second generation of PCIe IP Controller, which is subsequently also referred to in this document as HPA *(High Performance Architecture)* IP. This IP can be configured for a range of bandwidths from a single lane to 16 lanes for regular PCIe. Controllers can be configured in the following modes selectable by a strap pin:

- EP *(Endpoint)*
- RP *(Root Port).*

The Controller is compliant with the latest *PCIe Specification*. Refer to the section *References and Related Documents*.

- At the PCIe Physical Layer, the controller provides a PIPE compatible interface to SERDES devices.
- At the Application interface, the controller provides with the following interface choice/choices:
  - AXI4 bus interface with optional DMA feature.

  The width of the client interface is based on the configuration.

The PIPE interface operates on a fixed data PIPE width.

The controller operates with:

- A FCC *(Fixed Core Clock)*, that is, maximum PIPE interface clock frequency

The Cadence ® PCIe Controller supports various features that can be configured. Refer to *Core Configuration* for additional information.

1.2 System Overview of the Controller IP

The following figures show the basic overview of system implemented using the EP and RP PCIe Controller IP. Each figure shows:

- The key components that are used to connect the IP either in EP or RP mode
- PCIe usage of the IP as EP or RP modes.

This document describes the Cadence ® PCIe Controller including the Controller PCS. It does not describe the PMA and its associated PCS.

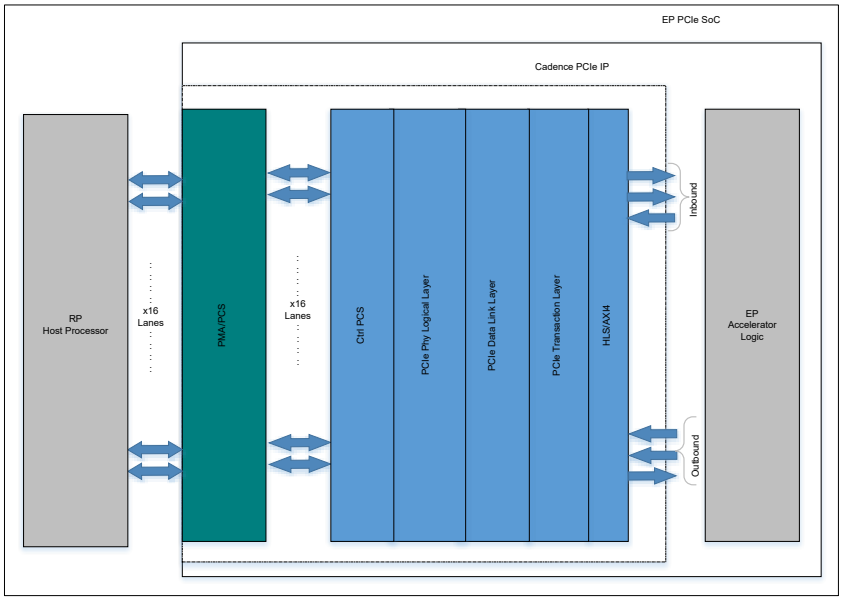


Figure 1.1: EP PCIe System-Level Block Diagram Example

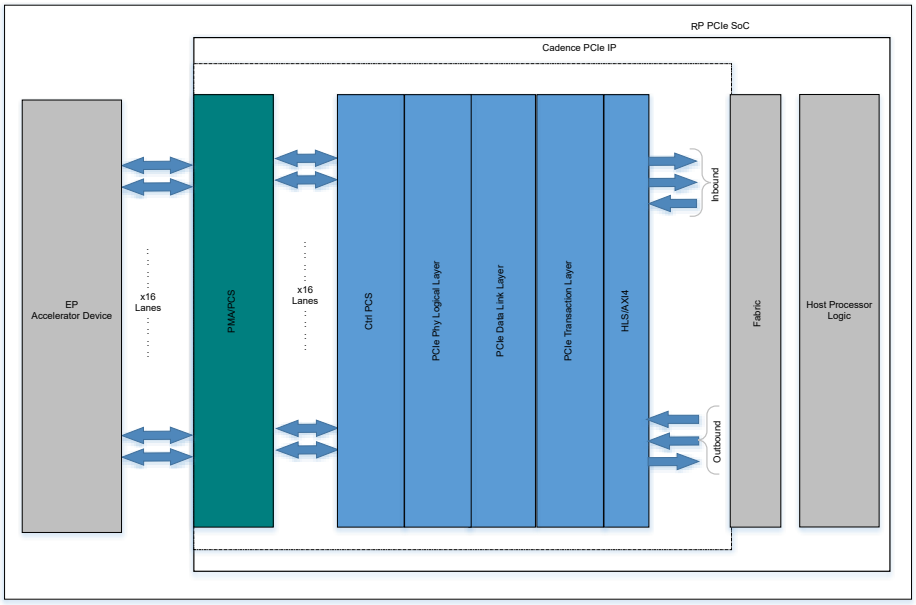


Figure 1.2: RP PCIe System-Level Block Diagram Example

1.3 Core Configuration

The Cadence PCIe IP consists of a range of features designed to meet the needs of different ASIC architectures, ranging from the low-power applications such as mobile and storage to high-performance server, and accelerator applications requiring large bus bandwidth. *Table 1.1* lists the architectural features that are allowed for core configuration.

Table 1.1: Core configuration details

Section	Feature Name	Description	Default Value	Value Range
Architecture Selection	SOC Target	Selects the features for *ASIC Only* or *ASIC With Support for FPGA*. To select *ASIC With FPGA Support*, the IP should include FPGA Delivery.	ASIC	ASIC, ASIC_WITH_FPGA_SUPPORT
	Core Protocol Support	By Default, selects *PCIe*, .	PCIE	PCIE
	Core Base Spec version	Selects the PCIe Base Spec version supported by this IP.	PCIE 5	PCIe 5
	Core Max Speed	Selects the maximum link width of the IP.	32GT/s	32GT/s, 16GT/s, 8GT/s, 5GT/s,2.5GT/s
	Core PIPE Spec version	Selects the PIPE Spec version supported by this IP.	PIPE_4.4.1, PIPE5.2	PIPE_4.4.1, PIPE5.2 SerDes Architecture with Low Pin Count
	Max Link Width	Max link width of the IP.	16	4, 8, 16
	Number of Independent Links	Selects the number of independent links supported by the IP.	1	1, 2, 4
	Number of Physical Functions	Selects the total number of Physical functions.	1	1, 2, 4, 8, 16
	Number of Virtual Functions	Selects the total number of Virtual functions.	0	2,4,8,16..upto 255
	Client Interface Type for PCIe	Selects the type of Client interface needed. HLS (HAL Streaming Interface), AXI4 (AMBA AXI4)	HLS	HLS, AXI4
Client (HLS) I/F Selection	Clent Interface Width	Selects from a legal choice based on PCIe Spec version and Link Width selected by you.	512	128, 256, 512
Client (AXI4) I/F Selection	AXI PCIe Manager Data Bus Width	Selects the AXI data bus width. For best performance the AXI data bus width should be equal or greater than the internal HLS data width.	512	128, 256, 512
	AXI PCIe Subordinate Data Bus Width	Selects the AXI data bus width. For best performance the AXI data bus width should be equal or greater than the internal HLS data width.	512	128, 256, 512
Delivery Selection	PIPE Width	PIPE Width selected for this configuration.	32	32
	RTL Module Prefix	Adds prefix that you select to ALL rtl modules, so that the modules can be unique.	dbPCIe_	dbPCIe_

Chapter 2: Architecture Overview

2.1 Architectural Block Diagram

This section describes the operation of internal functional blocks and the external interfaces of the Cadence PCIe core. The Controller IP consists of following key layers:

- CPCS (*Controller PCS*)
- PLL (*PHY Logical Layer*)
- DLL (*Data Link Layer*)
- TL (*Transaction Layer*)
- AL (*Application Layer*)

The PCIe Controller IP HPA architecture is optimized to handle multiple Transaction Layer packets and multiple Data Link Layer packets in parallel on both the receive and the transmit paths. The key feature of this architecture is to optimize latency in the design by reducing the amount of data re-alignment that is done between different internal blocks of the IP.

The PCIe Controller IP has two key Interface Boundaries:

- The PIPE interface which connects to the PMA/PCS
- Choice of Client interfaces; HLS (HAL Stream Interface) or the AXI4 I/F which connects to the application logic.

The PCIe Controller IP uses industry standard specification for these interfaces.

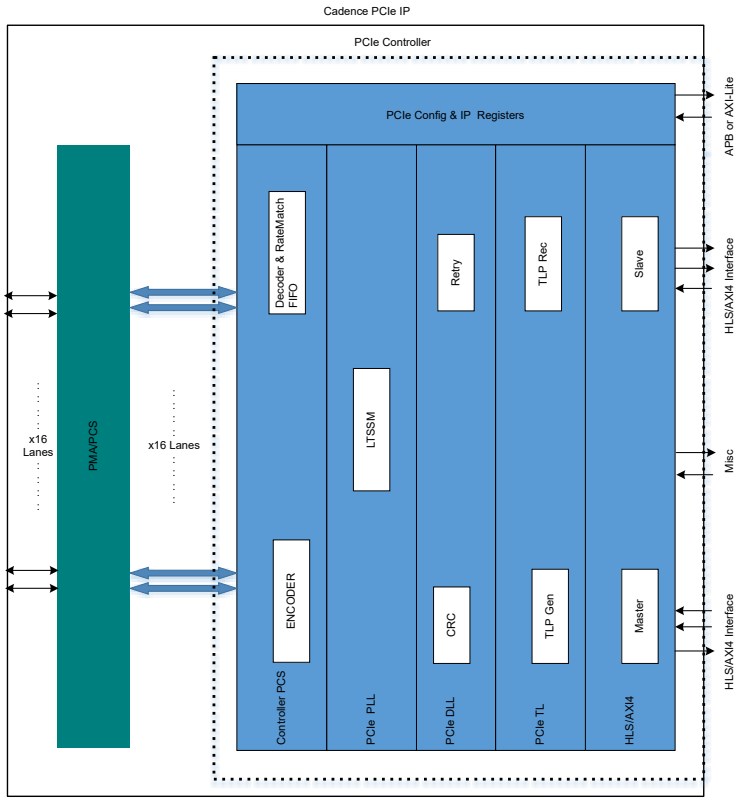


Figure 2.1: PCIe controller IP architecture overview

2.2 Controller PCS

The COntroller PCS IP is implemented to support PCIe PIPE Specification 5.2 (SerDes Architecture with Low Pin Count). The logic is implemented per lane and also supports multi-link feature. This module interacts with a PIPE 5.2 compliant PHY on one side and the data transfer between Controller IP internally using custom PIPE 4.4.1 Interface on the other side.

The CPCS interfaces to the PHY at 20bit for Gen1/2 and 40bit interface for Gen3/4/5. On the PIPE 4.4.1 the interface width is fixed at 32bits. This module supports nominal empty RM (Rate Match) FIFO for reducing latency. It also supports test mode for internal loopback option from receiver to transmitter.

Figure 2.2 shows the block diagram of Controller PCS.

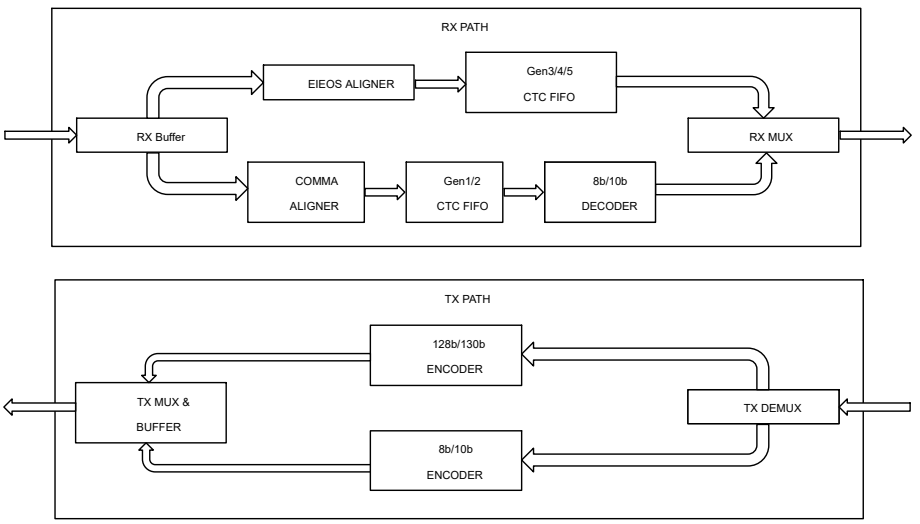


Figure 2.2: Controller PCS block diagram

This block module supports:

- **Fixed Core Clock mode**: In this operation mode, the Controller PCS requires a fixed Core Clock input that is provided to the core blocks irrespective of the current speed of operation on the PIPE interface. This mode adds various clock transfer FIFOs hence, adding latency in the design.

2.3 PHY Logical Layer

The receive operation of the Physical layer is as follows:

1. Data arrives on the PIPE Interface from the CPCS block through one or more lanes, over a 32-bit interface per lane.
2. The data from each lane is de-scrambled independently and goes through the logic to detect link power state transitions.
3. The lanes are then de-skewed by passing the data through FIFOs that are aligned on SKP Ordered Set (Gen1/2) and SDS Ordered Set (Gen3 and higher). The lanes are aligned as a single unit.
4. The de-skewed data is decoded by a frame decoder.
5. The frame decoder removes the SOP/EOP framing delimiters from the packet and aligns them on the internal data path. The frame decoder is designed to deal with varying link widths and all possible alignments of packets on the lanes.
6. The decoded data is passed to the Data Link Layer with indications such as packet-type and errors detected (if any).
7. The received data from each lane is also routed to a Link Training Receive State Machine, which is responsible for detecting and decoding training sequences received from the lane.
8. Each of these state machines pass on information extracted from the training sequences to the LTSSM.

The transmit operation of the Physical layer is as follows:

1. Data arrives from the Data Link Layer.
2. The data passes through a frame encoder, which adds SOP and EOP delimiters to the packets and stripes the data

on the lanes. The frame encoder is capable of dealing with varying link widths and all legal alignments of packets on the lanes.

3 The outgoing packets from the frame encoder are multiplexed with training sequences generated by the LTSSM.

4 The multiplexer disables the data path from the frame encoder during link training. It allows the LTSSM to control the lanes. Each lane has a separate scrambler to transform its data before entering the PIPE interface.

5 The precoding block performs the receive and transmit precoding as defined in the PCIe Specification. The outgoing PIPE interface is configured for 32bits per lane for Gen1/2/3/4/5.

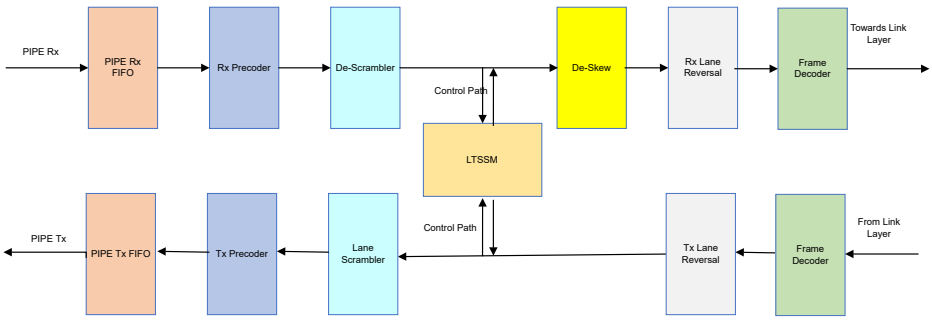


Figure 2.3: Physical Layer block diagram

2.4 Data Link Layer

The receive operation of the DLL is as follows:

1 The data from the link passes through a decoder state machine for each link.

2 The decoders verify the integrity of packets by matching the received CRC with the generated CRC, and comparing their sequence numbers with the expected values (for TLPs).

3 After comparing and removing the link-layer CRC, the DLLP decoders pass on the received packet to its target module based on the packet type.

4 All received TLPs with their sequence number and LCRC fields-removed are passed on to the Transaction Layer.

5 Data Link Layer acknowledgements ACKs and NAKs are processed within the Data link layer itself, and credit DLLPs are sent to the flow control module.

6 After the CRC check, the received Data Link Layer acknowledgements are sent to the transmit side of the Data Link Layer for processing.

7 The logic on the transmit side matches the acknowledgements with the outstanding packets and deal with all the error conditions.

8 The ACKs and NAKs for the received TLPs are sent to the transmit side of the Data Link Layer, where they are multiplexed with the outgoing TLPs.

The transmit operation of the DLL is as follows:

1 The received data from the Transaction Layer is formatted for transmission to the Physical Layer by inserting a sequence number and CRC.

2 The formatted TLPs are multiplexed with outgoing DLLPs such as ACKs/NAKs and credit packets, and sent to the Physical Layer over a data path shared by both links of the core. The transmit side also contains the replay buffer associated with the link, which is responsible for re-transmitting packets when needed. The replay buffer uses an external single-port RAM for storing the packets. The single-port RAM is used as a replay buffer. It can either be written or read. It never a writes and reads simultaneously. There is also an internal pointer RAM for keeping track of packets stored in the replay buffer.

The Data Link Layer also contains logic to generate Power management DLLPs to facilitate transitions of the link to L1 and L2 states as part of the PCIe Power Management.

2.5 Transaction Layer

Transaction Layer (TL) handles TLP buffering, Ordering, Credit gating, PCIe tag check and error checks.

2.5.1 TL Receive Side Operation

The following are high level details of operations done on Transaction layer receive side.

1. TLPs are decoded and checked for malformed errors, length mismatches, ECRC errors.
2. Order of received TLP is stored. It is used to enforce ordering while packets are read from receiver buffer and transferred to application logic.
3. TLPs are stored in store-and-forward RAM. The 128bit data segments are used to efficiently utilize RAM space if small TLPs are received in a cycle. There is a separate receive buffer RAM for Posted, Non-Posted and Completion packets.
4. The TLPs are split into the following separate streams and forwarded for further processing:
   - Posted.
   - Non-Posted.
   - Completion.
5. The receive flow control parameters of the core (payload and header credit for the three types of packets: Posted, Non-Posted and Completion) are set based on the available space in the receive buffers (stored in RAM). The flow control protocol ensures that the receive buffer do not overflow. The receive buffers communicate their state to the flow control module so that when the packet is forwarded out of the receive buffers, the corresponding credit becomes available and can be advertised to the link.
6. Based on the configuration, up to 8 DW aligned TLPs are possible to receive from PCIe link side. Transaction layer converts those packets into 128/64bit aligned 1/2/4 TLPs per cycle based on the selected configuration.

2.5.2 TL Transmit Side Operation

1. Posted, Non-Posted and Completion (SC) packets arrive from the Application logic over separate interfaces.
2. Logic within the Transaction Layer multiplexes the packets, inserts ECRC if enabled, and forwards them to the Data link layer.
3. The Completions and messages generated within the core are also multiplexed on the same data path to the Data link layer.
4. Credit gating logic blocks the TLPs if the sufficient credits are not available from the remote device.
5. PCIe Ordering of TLPs are enforced from HLS (Posted, Non-posted, Completion) and interrupt interface towards PCIe link in this layer.

2.6 Host Adaptation Layer Operation

The following features in Host Adaptation Layer (HAL) eases application integration with PCIe IP:

1. Targeted PF and VF is decoded after comparing all the BAR registers and the TLP address.
2. PCIe tags from outbound request towards PCIe link is stored and compared with completions for error checks. These are also used to generate completion timeouts, if completions are not received from PCIe link within a programmed time.
3. Endpoints are expected to accept all completions from PCIe link. The Throttling logic present in this layer stops outbound Non-Posted TLPs, if space for receive completion buffer is insufficient. This is an useful feature, if application logic back-pressure TLPs from PCIe-link.
4. All read and write requests to configuration registers are routed to the register set of the function addressed by the request, and returns completion packets back to the link.
5. All interrupt related messages are processed by a separate interrupt processing module, which controls the Interrupt interface of the core. The error messages are processed by an error handling module. There are also blocks to generate messages internally based on programmable registers.
6. When FLR received for any function, this layer automatically creates completions with FLR status towards AL for all outstanding requests.
7. Logic to generate UR Completion towards PCIe link for unsupported requests detected in IP.

2.7 Application Layer

The Application Layer of the core provides interface to the application logic implemented outside the IP, insulating you from the complexity of the PCI Express protocols. The Application layer provides the following key interfaces:

AXI4 Interface

The AXI4 interface connects to the existing HLS interface through an AXI4 Bridge logic. This interface is based on the AMBA AXI4 specification. This bridge implements all the PCIe and AXI4 ordering rules, any exceptions to this are explicitly stated. See *Chapter Application Interface AXI4* for additional information.

APB/AXI-Lite

Note:

The content is applicable based on whether APB or AXI-Lite is selected during configuration.

The IP provides access to the register's implemented PCIe core through a 32-bit user selectable APB/AXI-Lite bus. Through this bus, a local processor may access and modify the contents of the IP and access the PCIe Specification Configuration's registers associated with the PCIe Functions. The core adheres to protocol as defined in the *AMBA® APB Protocol Specification or AMBA® AXI and ACE Protocol Specification Issue E.*

Chapter 3: Signal List and Description

3.1 Pin List Description

For information on the Signal List and Description, refer to the *<package_path>/docs/cdnhpa_ips_IO_descriptions.html* file of the delivery package.

To open the html files from linux, type:

```
<web_browser_name> <filename>.html
```

Chapter 4: Clocks and Reset

4.1 IP Clocks

The Cadence IP has following two key clocking architectures.

4.1.1 Fixed Core Clock (FCC)

In this mode, the IP operates on a Fixed frequency which is the highest PIPE clock frequency, irrespective of the speed of operation at the PIPE I/F. These clocks are shown in *Figure 4.2*

Table 4.1: Clock Domains

Label	Domain Name	Source	Synchronicity	Frequency
A	Recovered clock domain	PHY CDR	Async to all Clocks	CDR recovered clock varies with PCIe speed of operation and PIPE I/F width
B	PIPE I/F Clock domain	PHY Ref Clock	Sync to Ref Clock	Varies with PCIe speed of operation and PIPE I/F width. • 62.5MHz for Gen1 • 125MHz for Gen2 • 250MHz for Gen3 • 500MHz for Gen4 • 1GHz for Gen5
C	Core Clock domain	PHY Ref Clock	Sync to Ref Clock	Max Operating PIPE I/F frequency
D	AXI I/F clock domain	User	Async to All Clocks	Higher or Equal to Core Clock Frequency
E	Power Management (PM) clock domain	User	Async to All Clocks	Up to Max Core Clock Frequency. Range: 2MHz to 60Mhz
F	APB I/F Clock domain	User	Async to All Clocks	Up to Max Core Clock Frequency

Note: The following figures are for representative purpose only, and the blocks are applicable based on the configuration selected.

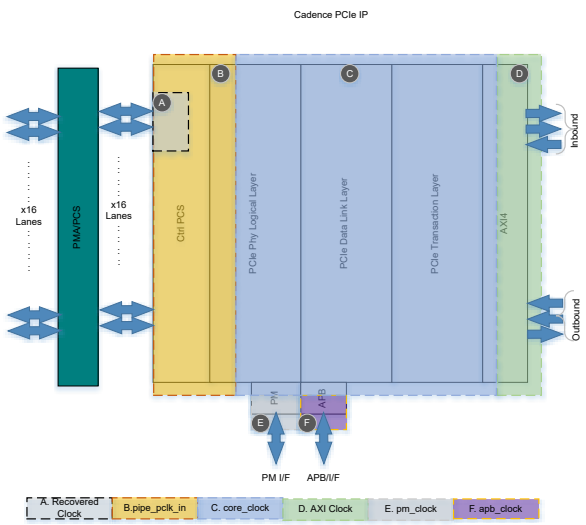


Figure 4.1: IP Clocks

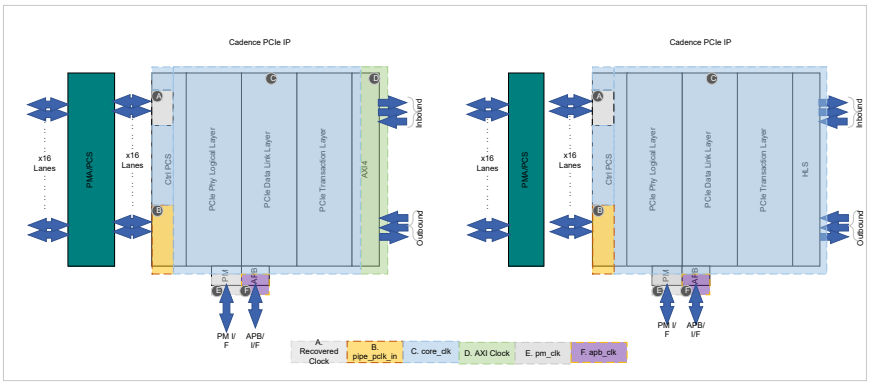


Figure 4.2: FCC PCIe IP with AXI and HLS I/F (PL TX/RX FIFO Bypassed)

4.1.2 PM Clock Frequency

In L1-substate *core_clk* can be turned off. A separate clock (*pm_clk*) is required for the L1-substate machine to operate. Default frequency of this clock is selected in configurator and the Client firmware needs to program *pm_clk Frequency Register/pm_clk Frequency Select* field in local management register for changing *pm_clk* frequency. The change in *pm_clk* frequency is supported only when LTSSM is not in L1. The Controller supports the frequency of up to 2MHz to 60MHz for *pm_clk*. The reset value reflects the *pm_clk* frequency chosen during Controller configuration.

Note: *pm_clk* should be timed at 62.5MHz and the Controller SDC file will be generated accordingly. If the timing is to be closed at a different frequency, then you need to update the SDC accordingly.

4.1.3 Reg Clock Usage

The controller has a separate *reg_clk input* which is used by PCIe configuration space and ip-mgmt registers in the Core. The controller with AXI interface has reg_axi_clk which is used by the registers controlling AXI layer. Separate clock input for registers allows firmware to access registers even when main clock is not available in the controller. When *core_clk* is available, *reg_clk* is expected to be connected to *core_clk*. When *axi_clk* is available, *reg_axi_clk* is expected to be connected to *axi_clk*.

For clock ports details, refer to the *IO_description.html* in the package.

If your application does not require to access registers during low power states or low power states are always disabled, then main clock can be directly connected to respective register clock.

The following figure shows an example of *reg_clk* and *reg_axi_clk* muxing in the customer application logic external to the IP.

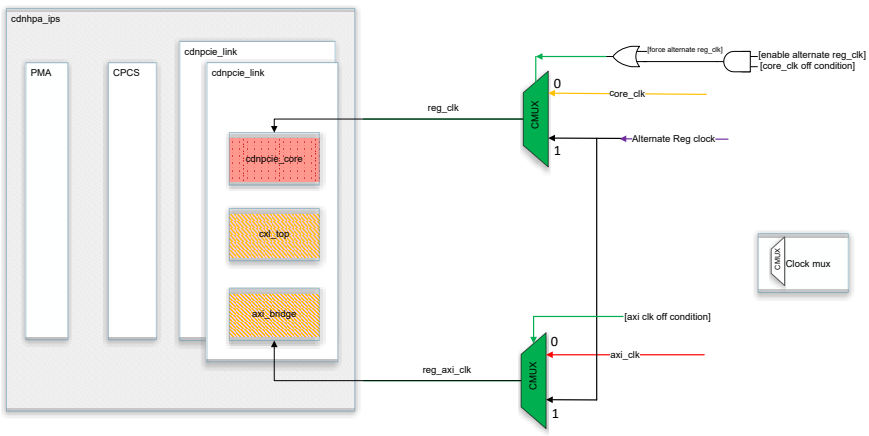


Figure 4.3: Reg clock

When the link is in L1_substate, the *core_clk* is shut off. In this state, the *reg_clk* can be connected to an *Alternate Reg Clock*. If *axi_clk* is gated to the IP under any conditions, *reg_axi_clk* can be connected to an alternate clock.

Note: Any clock to access registers while main clock is off. Frequency must be slower than *core_clk* or *axi_clk*, depending which one is slower.

Description of the logic in diagram

[**core_clk off condition**] = low_pwr_l1_pm_substate_out==3'b101 || low_pwr_l1_pm_substate_out==3'010

[f**orce alternate reg_clk**] :: System register bit implemented in application logic to force alternate clock for *reg_clk*. Useful for debug.

[**enable alternate reg_clk**] :: system register bit implemented in application logic to enable switching to alternate clock. Useful for debug to prevent switching to alternate clock.

[**axi_clk off condition**] :: system register or logic indicating that *axi_clk* is off

Alternate reg_clk :: Any clock to access registers while main clock is off. Frequency must be slower than *core_clk* or *axi_clk*, depending which one is slower.

4.2 Reset Signals

For information on the The PCIe Controller reset signals, refer to the /doc directory of the delivery package.

4.2.1 Reset Outputs

The Reset signals are output to the client logic as Status Information. The client logic may use this information as a part of its reset sequencer. For information on the reset output signal of the Controller IP, refer to the */doc* directory of the delivery package.

4.2.2 *link_down_handling_in_progress*

The *link_down_handling_in_progress* output signal is driven on the Client Interface Clock, when it detects a Link Down event. The controller autonomously performs a synchronous reset internally. Please check the link down handling section for link down handshake from application side. This signal is kept asserted till the link_down handshake is complete.

- All the internal state machines, timers and control registers in the Controller are reset upon the Link Down Reset.
- In the PCIe defined configuration register space, all the registers, except those which are **Sticky**, are reset upon *Link Down Reset*.
- The Local Management Registers are not cleared upon *Link Down Reset*. These registers hold their programmed values and are not required to be re-programmed.
- The *link_down_handling_in_progress* output is sent to the client logic for the link down event. For example, the client logic may reset its own FIFOs, registers, or state machines if link goes down. It might also need to set some default values for some of the PCIe Configuration space registers as a result of this link down event.

Note:

The PHY is not required to be reset upon *link_down_handling_in_progress* , though this can be done for debug purposes.

4.2.3 Hot Reset Out

The intent of the Hot Reset state is to allow a configured Link and associated downstream device to be reset using in-band signaling. Hot reset allows Root port to reset only a selected portion of PCIe hierarchy or Device.

The *hot_reset_out* output signal is driven on the CORE_CLK in Endpoint mode, when it detects a HOT_RESET from the Host in the received Training Sequences. This reset causes the following events in the Controller.

- The *hot_reset_out* output sent to the client logic indicates that the Link Down event has occurred due to a Hot Reset directive from the host.
- The controller autonomously performs synchronous reset for the PCIe Configuration registers upon a hot reset event.
- The outstanding requests on the Client interface are handled similarly as the *link_down_handling_in_progress* .
- Hot Reset causes the LTSSM to go to Detect State (i.e. Link Down.). Hence, the *link_down_handling_in_progress* is always asserted together with the *hot_reset_out*. Host_reset_out is asserted only for 8 clocks.

- Client may choose to ignore the HOT_RESET_OUT output and only use the *link_down_handling_in_progress* output.

Chapter 5: IP Register Interface

5.1 Overview

The Controller IP Register block supports access to the core address spaces specified in *Table 5.1*. The table lists high-level location of all the registers located in the Controller IP. The access to the AXI address space is reserved when AXI Bridge is not present. The Controller IP and IP AXI Bridge/DMA logic registers can be accessed via below two paths.

- PCIe Link Access (A Path)
  - If the IP is operating as an EP, the remote device can access all the Controller IP registers through PCIe CFG transactions via indirect addressing using the VSEC Capability.
  - Access to the IP registers via this path can be disabled using the IP Port.
- AXI-Lite/APB Access (B Path)

Note:

The register access protocol would be either APB or AXI-Lite based on the selected configuration.

- The Application logic can access all the IP registers through AXI-Lite port on the IP via Direct addressing.
- Optionally APB port when enabled can also access these registers. The Port uses an AXI-Lite to APB Bridge which is built in the IP for the access.
- Please note, for an IDE configured controller, the IDE Stream Management and Encryption core registers are not accessed via this interface, but via a separate Secure APB interface. Please refer to the IDE overview section for details.

Table 5.1: Register Spaces

Various Registers Space	Access through PCIe link	Access through AXI-Lite/APB Interface
PCIe Spec defined TYPE1 CFG Space	Yes	Yes
PCIe Spec defined TYPE0 CFG Space for PF	Yes	Yes
PCIe Spec defined TYPE0 CFG Space for VF	Yes	Yes
IP Specific Register for Core logic	Yes (indirectly via VSEC[11])	Yes
IP Specific Register for AXI Bridge/DMA logic	Yes (indirectly via VSEC[1])	Yes
IDE Stream Management and Encryption Core registers	No (accessed via secure APB interface)	No (accessed via secure APB interface

1 See *IP Registers Access through the PCIe Link*

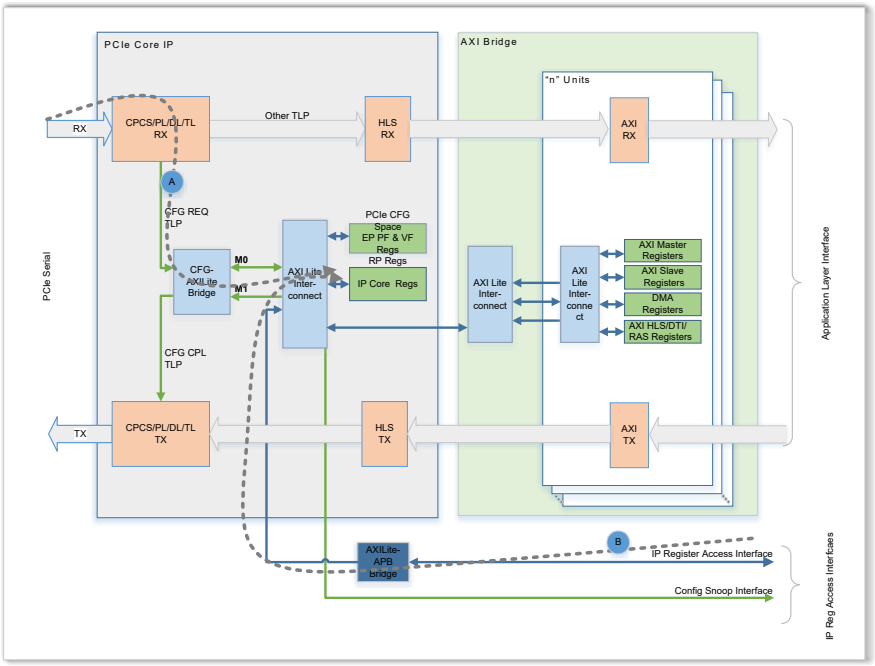


Figure 5.1: IP Register Interface

In the following sections, the addresses that are not defined are reserved for future use and the addresses that are described are DWORD addresses. In the write operation, the individual bytes are addressed by byte-enable bits. A configuration access from the link to a reserved address causes the core to return a packet with the UR (Unsupported Request) completion code. A read from the local management bus to a reserved address returns all zero's, whereas a write to a reserved address does not modify any of the registers. All registers (with the exception of reserved or hardwired fields) can be written from the local management bus.

5.2 Register Address Map

The address range decoded by the Controller is expanded, as shown in the table below, to support the registers defined by the UCIe specification. The full PCIe range is further broken down in additional tables below.

Table 5.2: Register Address Map

Access Path	Bank Decode	Function Select	Register Addresses	IP Mode	Description
PCIe Link Access	addr[25:24]=0	addr[23:12] = 0 - 2047	addr[11:0] = 0 - 4095	EP	PCIe Spec Config Registers Type0 PFn/VFn Registers
	addr[25:24]=0	addr[23:12] = 0	addr[11:0] = 0 - 4095	RP	PCIe Spec Config Registers Type1
	addr[25:24]=1	addr[23:14] = 0 - 255	addr[13:0] = 0 - 16383	EP	IP Core Registers [1]
	addr[25:24]=2 addr[25:24]=3	x	addr[17:0] = 0 - 131071	x	IP AXI/DMA Registers [1]

Table 5.2: Register Address Map

Access Path	Bank Decode	Function Select	Register Addresses	IP Mode	Description
Appl Layer APB/AXI-Lite Access[2]	addr[25:24]=0	addr[23:12] = 0 - 2047	addr[11:0] = 0 - 4095	EP	PCIe Spec Config Registers Type0 PFn/VFn Registers
	addr[25:24]=0	addr[23:12] = 0	addr[11:0] = 0 - 4095	RP	PCIe Spec Config Registers Type1
	addr[25:24]=1	addr[23:14] = 0 - 255	addr[13:0] = 0 - 16383	x	IP Core Registers, [3]
	addr[25:24]=2 addr[25:24]=3	x	addr[17:0] = 0 - 131071	x	IP AXI/DMA Registers[3]

1 Indirect addressing, “addr” taken from value programmed in VSEC Capability Reg0(addr). Enabled only when Access is enabled via IP Port (STRAP_IPREG_LINK_ACCESS_ENABLE).

2 This address uses the BDF format as per PCIE specification in the 'Function Select' Field of addr i.e [23:12] .

3 Direct addressing, “addr” as provided on the APB/AXI-Lite interface

5.2.1 IP Controller Registers

The IP Controller Register space is further categorized based on layers and functionality as given in the following table.

Table 5.3: IP Controller Register Map

REG ADDR[25:24]	REG_ADDR[23:14]	REG_ADDR[13:10]	REG_ADDR[9:0]	IP Core Register Space
1	0	0	0 - 1023	Controller PCS Block Registers
1	0	1	0 - 1023	Physical Layer Block Registers
1	0	2	0 - 1023	Data Link Layer Block Registers
1	0	3	0 - 1023	Transaction Layer Block-Registers
1	0	4	0 - 1023	HLS Block Registers
1	0	5	0 - 1023	Common RAS Registers
1	0	6-14	0 - 1023	Reserved-Unused
1	0 - 256	15	0 - 1023	Per Function Core Registers

5.3 IP Register Access

5.3.1 AXI-Lite Interface (B Path)

The Controller IP Register Interface used by the client software to access the core registers is based on the AXI-Lite specification. The following diagrams detail the Write and Read timings on the interface.

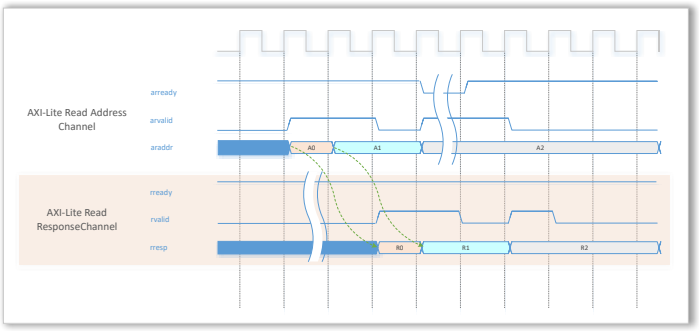


Figure 5.2: AXI-Lite Interface Read

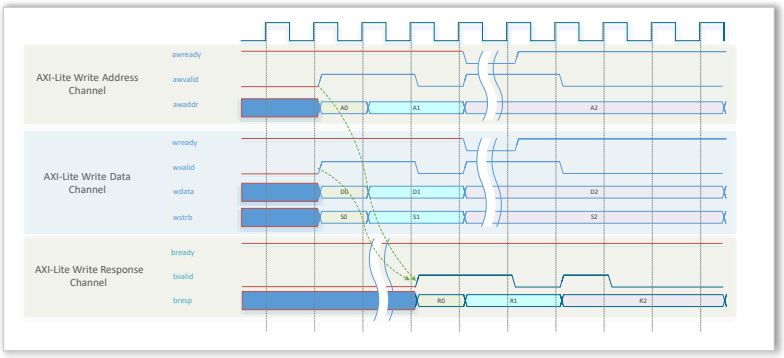


Figure 5.3: AXI-Lite Interface Write

5.3.2 APB Interface (B Path)

The controller IP Register Interface by the client software to access the Controller IP registers also supports APB interface for more flexibility. If you choose APB, an APB to AXI-Lite bridge is instantiated in the Core Registers block and in the CDC blocks. This ensures the APB Writes and Reads are converted from the APB Clock to the AXI-Lite Clock domain.

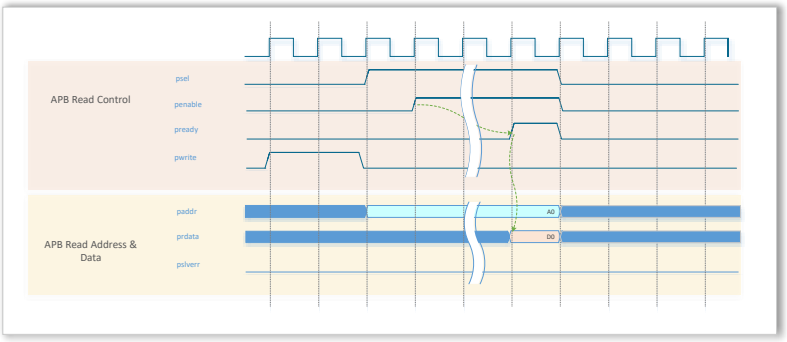


Figure 5.4: APB Interface Read

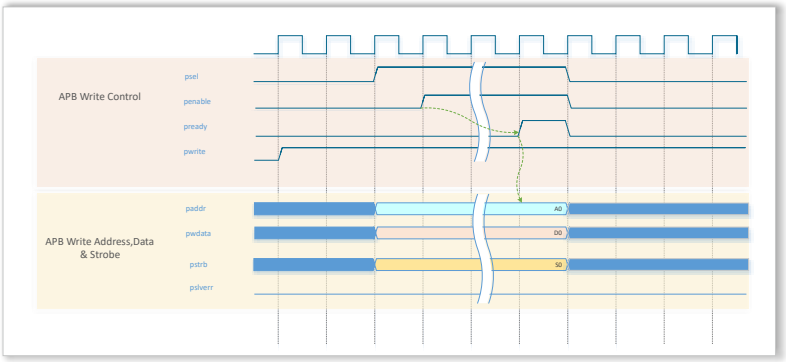


Figure 5.5: APB Interface Write

5.4 IP Registers Access through the PCIe Link

As shown in *Table 5.2*, the HPA controller contains Implementation specific registers in addition to the Specification Mandated Configuration registers to aid in efficient and error free operation of the core. These IP registers are mainly accessed through the APB/AXILite Interface. The HPA Controller also supports access of these register (including AXI/DMA registers) through the PCIe Link (Host). To enable this, the strap input *strap_ipreg_link_access_enable* must be set . This functionality is achieved using the Vendor Specific Capability(VSEC) Registers in the Config space, using an indirect mechanism defined below.

Each such Access would consist of an “Address” phase and a “Data” phase. The *strap_ipreg_link_access_enable* strap would need to be high for this access to be allowed, else an “UR” completion would be returned . The following Vendor Specific Registers would be used for the access:

- Vendor Specific Address Register (*i_vendor_specific_addr_reg0*): Used for the Address to be read/written
- Vendor Specific Data Register (*i_vendor_specific_data_reg1*): Used for the Data to be Written

1 For Writing into an IP Register through the PCIe Link , the steps to follow would be:

   a Address Phase: Host sends a config write targeting the ‘Vendor Specific Address Register 0’, the config write payload data should contain the target IP/AXI Register’s address. The controller would return a ‘SC’ Cpl for the config write as per PCIe rules.
   b Data Phase: Host sends next config write targeting the ‘Vendor Specific Data Register 1’, the config write payload data should contain the actual Data to be written to the IP/AXI Register, the bytevalid’s in the config write TLP would be used for the ByteValid/Strobe for the IP Register Write access too. The controller would return a ‘SC’ Cpl for the config write as per PCIe rules.
   c Only after the receipt of the second config write to the ‘Vendor Specific Data Register 1’ would the controller internally generate a IP Register Write access using the Address, Data and Strobe captured in the above steps, the AxPROT[0] would be set to 0 to signify it’s a config access. The targeted register in IP/AXI reg space would be updated.

2 For Reading an IP Register through the PCIe Link, the steps to follow would be:

   a Address Phase: Host sends a config write targeting the ‘Vendor Specific Address Register 0’, the config write payload data would contain the target IP/AXI Register’s address. The controller would return a ‘SC’ Cpl for the config write as per PCIe rules.
   b Data Phase: The Host next sends a config read targeting the ‘Vendor Specific Data Register 1’.
   c Only after the receipt of the second config read to the ‘Vendor Specific Data Register 1’ would the controller internally generate an IP Register Read Access using the Address captured in the address phase above. The AxPROT[0] would be set to 0 to signify it’s a Config Access .The Read data and status returned for the read would be used to generate a CplD for the Config Read in step 2 as per PCIe Rules to return the IP Register Read data and complete the access.

5.5 Config Snoop Interface

The Cadence PCIe controller implements PCIe configuration space registers per function as defined in the specifications. It supports the PCI compatible configuration space and the PCIe extended space registers. Depending

on the features selected during the configuration process, the controller automatically builds a linked list of capability structures.

During the enumeration process, the Host can traverse through the linked list of structures in the controller to discover supported features. The same capability structures are used by Root Port software for control and status information. The PCIe Configuration Read/Write TLPs mechanism is used for this purpose. The controller maps an incoming CFG RD/WR TLP into a read/write access to the configuration space registers that is implemented internally. The control fields of the capability structures are routed to different layers of the controller IP to correctly provide the information.

The purpose of this Config Snoop Interface is to provide flexibility to implement your own set of registers in the PCIe configuration space. The interface simply snoops on incoming CFG RD/WR TLPs received from the link and places them on a simple configuration snoop read/write interface. The core implements a snoop interface using AXI-Lite.

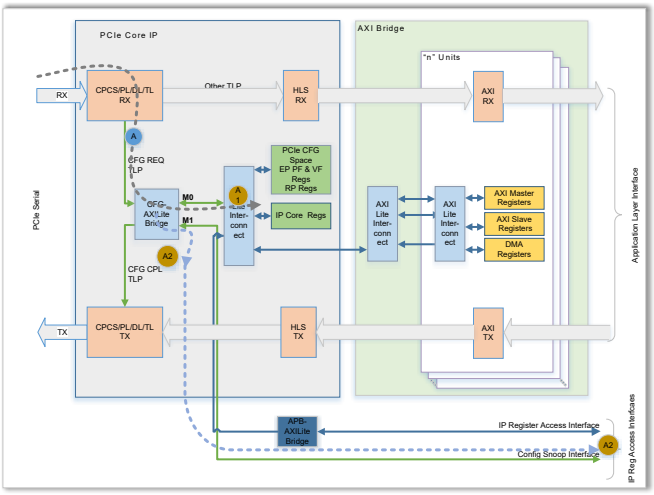


Figure 5.6: Config Snoop Interface

5.5.1 Inbound CFG Write Request Flow (A Path)

The Cadence IP internally has two AXI-Lite Master ports M0 and M1 as shown in top-level diagram, *Figure 5.6*.

- M0 port is used for accessing internal register (A1 Path)
- M1 port is used for accessing external configuration registers (A2 Path)

When a CFG Write is received, the core places the same data on the write address and data channels of M0 and M1 as received on the TLP . The client can use Write Address/Data channel on M1 to track and monitor any IP CFG registers that are updated internally by the IP. The write Response channel on M1 is implemented but not used by the controller such that the returned completion response for a write is based purely on M0 port as follows:

- When M0 port respond with DECERR/SLVERR, Completion is sent out UR
- When M0 port responds with OKAY, Completion is sent out as Successful

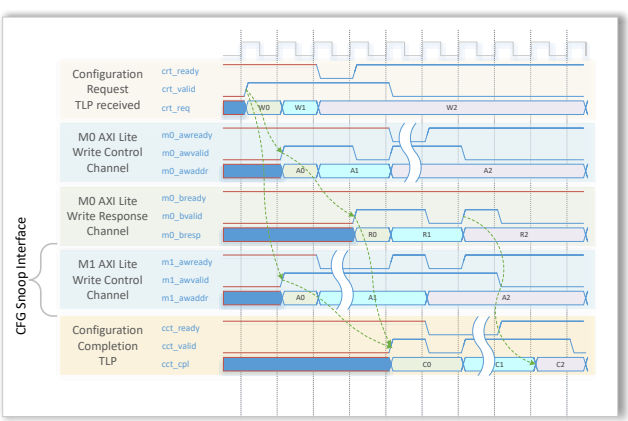


Figure 5.7: Config Snoop Interface Write

5.5.2 Inbound CFG Read Request Flow (A Path)

During a Config Read, the core places the address on the M0 and M1 read address channel. The client may choose to respond with its own Read Data on M1 rather than the one returned from the internal register. The RRESP signal from M1 is used by the core to decide which Read Data and Response to choose to send in the completion.

When a CFG Read Request is received on the inbound the request is passed to M0 and M1 Interface. A TLP completion is sent only after receiving responses from both M0 and M1 ports;

- When M1 port respond with DECERR/SLVERR/Timeout, completion is sent out with data from M0
- When M1 port responds with OKAY, completion is sent out as Successful with the data from M1. If the user does not respond on M1 with any RRESP for a read, the bus may hang, and the Config Path may stall. A user programmable timeout mechanism through IP Core register is implemented to ensure the Config Path is never stuck.

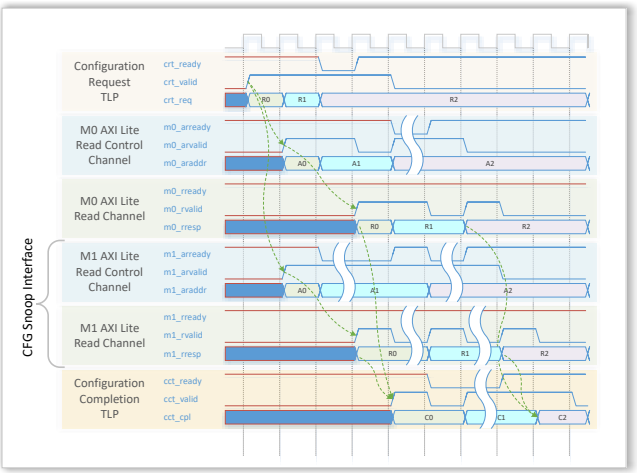


Figure 5.8: Config Snoop Interface Read

Chapter 6: Registers

6.1 Register Description

For information on the Register Description, refer to the `<package_path>/models/ipxact/*` directory of the delivery package.

There are up to three sets of register descriptions, depending upon configuration:

- `* _ep_local.xml/html` – describes End Point register space seen from the Device AXI Lite interface
- `* _ep_remote.xml/html` – describes remote End Point register space seen from the Host
- `* _rp_local.xml/html` – describes Root Port register space seen from Host AXI Lite interface
- `* _rp_remote.xml/html` – describes Root Port register space seen from remote interfaces

To open the html files from linux, type:

```
<web_browser_name> <filename>.html
```

Chapter 7: Controller and PHY Integration Solution

7.1 Controller IP Integrated Protocol Stack (IPS)

Note: This chapter is relevant if this release is a controller and PHY solution.

The Controller IP Integrated Protocol Stack (IPS) integrates the Controller IP and PHY IP for PCIe into a single configurable design.

The IPS consists of a wrapper around the Controller IP and PHY IP. It contains top-level interfaces to the outside world and the interconnects between the two modules. A small amount of logic is present for the control registers at the IPS top-level. The block diagram of this design is shown in the following figure.

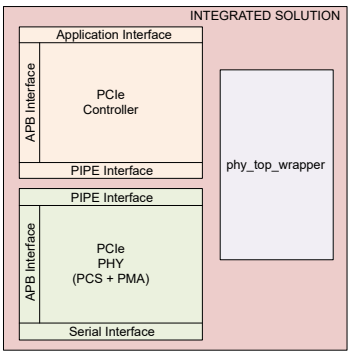


Figure 7.1: IPS block diagram

Note: Refer to the PHY documents available in the release package `dbPcie_ios_top/hdl/hdl_src/phy_ip/<pcs>/doc`

As shown in the figure above,

- The Controller IP has an Application interface which connects directly through the wrapper.
- The Controller IP connects to the PHY IP via the PIPE interface which varies in size depending on the PHY IP that is selected.
- The PHY IP consists of a digital PCS layer and mixed-signal PMA, its serial interface connecting directly through the wrapper.
- The phy_top_wrapper implements all the tie-offs and simple logic to keep the `hdl/hdl_src/ips/cdnhpa_ips.v` clean.

Note: Refer to the Cadence Verilog Testbench User Guide for information on how to run simulations at IPS level.

Chapter 8: Application Interface AXI4

8.1 AXI Overview

The following figure shows the overview of the Controller IP including AXI Bridge for AXI4 interface support. The DMA (blue) and DTI-ATS (green) modules are optional and present only when corresponding features are enabled during IP configuration. AXI4 Stream interface is not available when the DTI-ATS feature is disabled. AXI4 DMA DS interface is available only when both DMA and separate DMA Descriptor Server port are enabled. Otherwise, DMA Descriptor Server uses the main AXI4 Manager interface for reading the DMA descriptors.

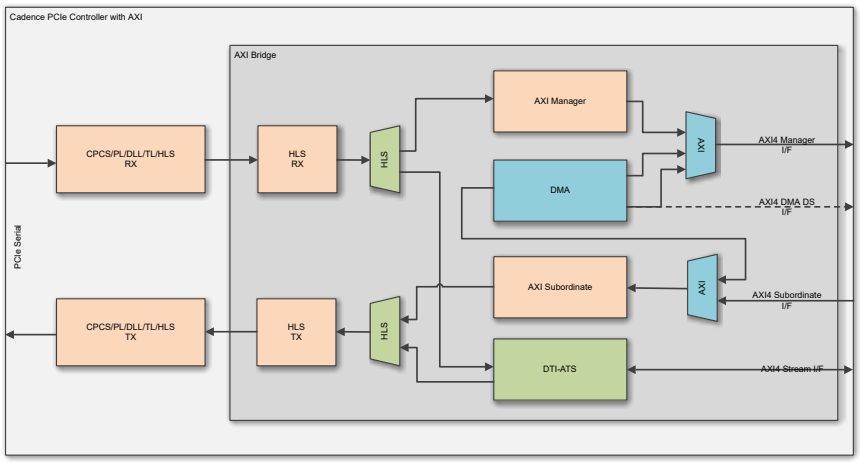


Figure 8.1: PCIe Controller with AXI

8.2 AXI Subordinate

The AXI Subordinate interface enables an Application Layer to initiate PCIe transactions across the PCIe link as Bus Master.

For Endpoints (EP), this interface has to be connected to Application Layer only when the Application Layer has Bus Master capability.

The AXI Subordinate initiates Requests and sends it through the PCIe link to the Completer. AXI Write and Read accesses, via regions or sideband descriptor, are decoded and mapped to the respective Posted and Non-Posted Write and Read TLP generator to initiate the PCIe Requests. Memory Write and Message (with/without data) Requests are generated in the Posted generator and mapped to the Posted HLS TX interface. Configuration Write, IO Write, AtomicOp, and DMWr Requests are generated in the Non-Posted Write generator and mapped to the Non-Posted HLS TX interface. Memory Read, Configuration Read, IO Read Requests are generated in the Non-Posted Read generator and mapped to the Non-Posted HLS TX interface.

8.2.1 AXI Subordinate Operation Types

The following table lists the PCIe TLP types that Application Layer can request as Write/Read operations on AXI Subordinate. Additional information can be found in the upcoming subsections.

Table 8.1: Allowed AXI Subordinate PCIe TLP types

AXI Operation	Write operations	Read operations
Root Port (RP)	**MWr** Memory Write Request **AtomicOp** Atomic Request **DMWr** Deferrable Memory Write Request **CfgWr0/1** Cfg Write Type 0/1 Request **IOWr** IO Write Request **MSG** Message Request **VDM** Vendor Defined Message Request	**MRd** Memory Read Request **CfgRd0/1** Cfg Read Type 0/1 Request **IORd** IO Read Request
Endpoint (EP)	**MWr** Memory Write Request **AtomicOp** Atomic Request **DMWr** Deferrable Memory Write Request **MSG** Message Request **VDM** Vendor Defined Message Request	**MRd** Memory Read Request

The following table lists the allowed transfer types for each type of the Request. Additional information can be found in the upcoming subsections. The limitations are the results of the AXI and PCIe specification rules and the conversion of the Request from AXI transaction to PCIe TLP. Application Layer may use a combination of the allowed transfers in single Request.

Table 8.2: Allowed transfer types for each request type

Transfer type	Split logic	Wrap transfer	NCBE	Narrow transfer
MWr - Memory Write Request	Yes (split at Max Payload Size)	Yes	Yes	Yes
AtomicOp – AtomicOp Request See *Atomic Operations* for more details	No	No / Yes (for AXI-5 Compare and Swap signalization)	No	No
DMWr – Deferrable Memory Write Request	No	No	No	Yes
CfgWr0/1 - Cfg Write Type 0/1 Request	N/A	No	Yes	No
IOWr - IO Write Request	N/A	No	Yes	No
MSG/VDM – (Vendor Defined) Message Request	No	No	No	Yes
MRd - Memory Read Request	Yes (split at Max Read Request Size)	Yes	No	Yes
CfgRd0/1 - Cfg Read Type 0/1 Request	N/A	No	No	Yes
IORd - IO Read Request	N/A	No	No	Yes

8.2.2 AXI Subordinate Write Operations

A Write transaction in the AXI Protocol is a split transaction with independent Address, Data, and Response Phases associated with the corresponding Channels. The Application Layer must follow the Manager protocol for the Write Address, Data, and Response Channel as described in the *AXI Specification*. The AXI Subordinate as part of IP shall follow the Subordinate protocol for the Write Address, Data, and Response Channel as described in the *AXI Specification*.

Application Layer may send Memory Write Request on AXI with payload bigger than current PCIe Maximum Payload Size (MPS). AXI Subordinate handles the split of the Request into multiple PCIe Memory Write TLPs, which length does not exceed MPS. The first split starts from the address aligned to the datapath width.

The following figure shows an example for MPS set to 128 bytes and 512-bit (64-byte) datapath. The first split of AXI packet occurs after the first two beats of AXI - the bytes are counted from the address aligned to the datapath (0x00 in the example) until 128 bytes. The second split occurs after next two full data beats, and so on.

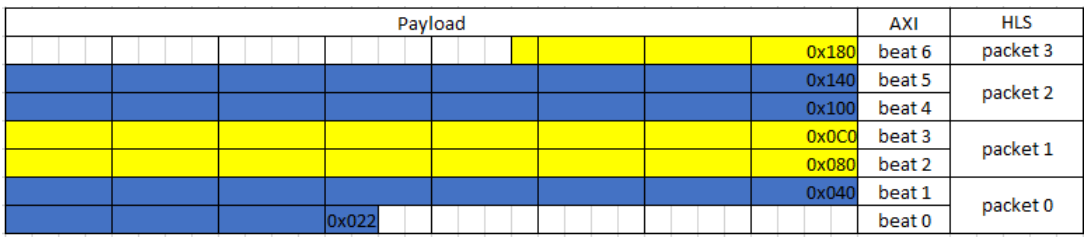


Figure 8.2: AXI Subordinate Memory Write Request Split

The *axi_s_awsize* variation from its maximum value of the datapath size is not allowed when *axi_s_awlen* is not zero in a Request (e.g. for mult-beat accesses) if narrow transaction support is disabled.

When narrow access support is enabled, Memory Requests can be sent with *axi_s_awsize* not equal to maximum value of the datapath width for any valid *axi_s_awlen* (single- or multi-beat transactions). The following figure shows a request sent through AXI using narrow access and payload size bigger than the MPS value.

Figure 8.3: AXI Subordinate Memory Write Narrow Request Split

AXI Subordinate does not support FIXED burst type. If the Application Layer needs to send data to the same address, data should be split into multiple single-beat (*axi_s_awlen* equal 0) transactions with INCR burst type, that have the same address.

If the Application Layer sends a Request with WRAP burst type, then the transaction is split at wrap boundary address. The following figure shows a request sent through AXI using narrow access, WRAP burst type and payload size bigger than MPS value.

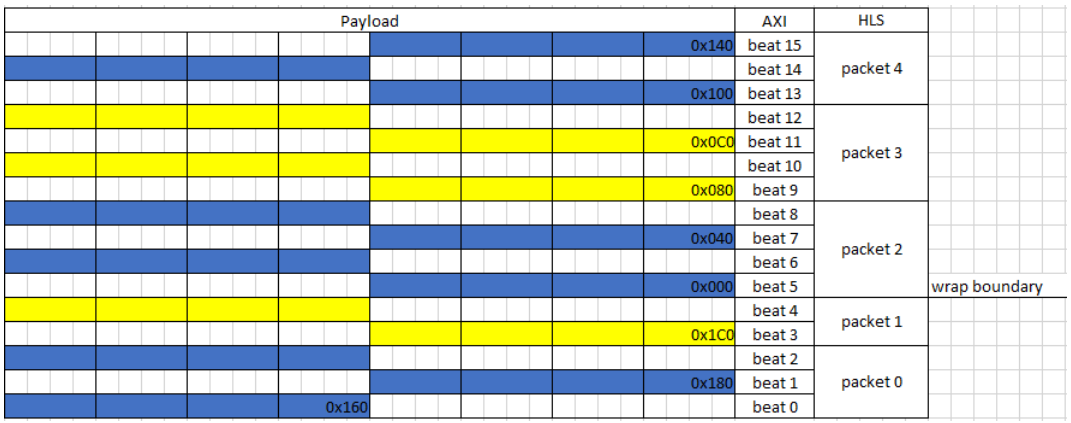


Figure 8.4: AXI Subordinate Memory Write Narrow Wrap Request Split

Configuration and I/O (Non-Posted) Writes are handled in a similar manner, except that the data payload is only one DWord in length. The core issues a response for Configuration and I/O Writes only when it receives the Completion back from the PCIe Link.

8.2.2.1 Deferrable Memory Write (DMWr)

Deferrable Memory Write is a PCIe Non-Posted Write Request. The Request requires to a PCIe Completion (without data) that carries a status, which is Completer's response to the Request. This gives the Completer a mechanism to accept the Request (Successful Completion status) or to temporarily refuse (defer) the Request (Request Retry Status). DMWr data is expected to be transferred (and accepted/refused by a Completer) as a whole, in a single Request, so it cannot be split due to various conditions (smaller than MPS, no WRAP burst type, no NCBE).

The access and generation of the DMWr request is described in section *AXI Subordinate Access*.

8.2.2.2 AXI Subordinate Write ID Management

Write requests on the AXI Subordinate interface are performed as split transactions. That is, the client may make additional read and write requests while the response for a write request is pending. Each outstanding write requests can be issued by the client with the same *axi_s_awid* or unique ones. For non-posted writes, the AXI Subordinate internally maps the *axi_s_awid* to an internally generated PCIe tag. See *Tag Management for Non-Posted Transactions* for more details. This mapping is looked up to translate an incoming completion to a corresponding *axi_s_bid* value sent with the write response channel.

8.2.2.3 Zero-Length Writes

A zero-length memory write transaction can be initiated on the AXI Subordinate interface by proceeding as in the case of a 1- byte memory write transaction, with the exception that the byte valid bits in *axi_s_wstrb* are all set to zero during the data cycle. This results in the core sending a memory write request on the PCIe link with length field set to 1 double-word, and the byte-enable fields set to all zeroes.

8.2.2.4 Non-contiguous Byte Enable Writes

The PCIe Specifications allow memory writes with non-contiguous byte enables (NCBE) for single-DWord writes, and for two-DWord writes when the address is aligned on an 8-byte boundary. For these write transactions, the user logic sets the byte valid bits on the *axi_s_wstrb* signal based on the valid bytes indicated in the header of the request TLP.

For AXI writes of a single DWord, the byte-valids may be non-contiguous, as allowed by the PCIe Specifications. Likewise, for 2-DWord writes, the byte-valids may be non-contiguous if the starting address is aligned on an even DWord boundary.

For AXI writes with payload bigger than 2DWord, the request is split into multiply two-DWord NCBE PCIe write requests. Otherwise, it is forbidden to send transaction with the byte-valids are non-contiguous with payload bigger than two-DWord.

8.2.3 AXI Subordinate Read operations

A Read Transaction in the AXI Protocol is a split transaction with independent Address and Data Phases associated with the corresponding Channels. The Client must follow the Manager protocol for the Read Address and Data Channel as described in the *AXI Specification*. The AXI Subordinate follows the Subordinate protocol for the Read Address and Data Channel as described in the *AXI Specification*.

User logic may send a memory read request on AXI with payload bigger than the allowed maximum read request size in PCIe. AXI Subordinate handles the split of the request into multiple PCIe memory read TLPs. The request splits into requests with *Maximum Read Request Size* (MRRS) where the first split starts counting from the address aligned to the datapath. The following figure shows an example for MRRS set to 256 bytes and 512bits datapath. The first split with Tag 0 reads the first four beats on the AXI, that is, the byte count starts from the address aligned to the datapath (here 0x00) until 256 bytes that is 4 beats read on AXI. The second split happens after the four full of valid data beats, and so on.

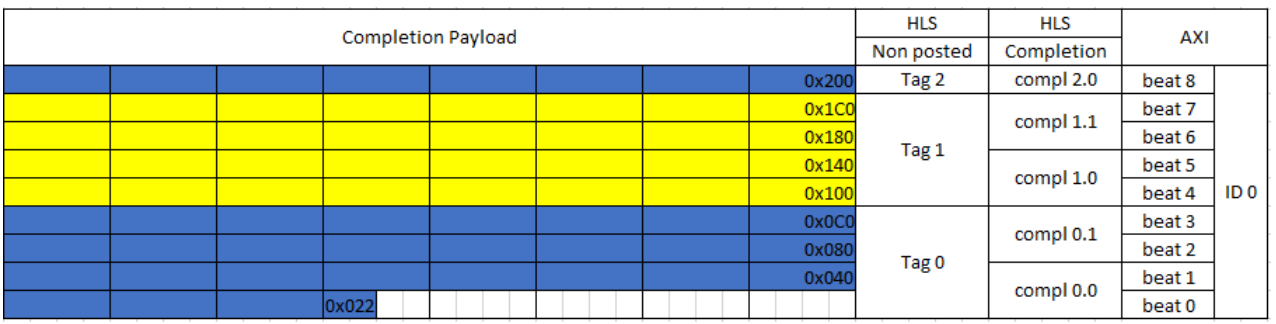


Figure 8.5: AXI Subordinate Memory Read Request Split

Note:

Completions from the split request with Tag 1 and 2 may come before the completions from Tag 0 but their payload is saved and put on AXI only after the payload from Tag 0 completions comes over PCIe link. That is, data is sent over AXI in an increased address order to abide by AXI rules.

Note:

The *axi_s_arsize* variation from its maximum value of the datapath size is not allowed when *axi_s_arlen* is not zero in a request if narrow transaction is disabled. Memory request can be sent with narrow access, that is, *axi_s_arsize* not equal maximum value of the datapath size for any *valid axi_s_arlen*.

The following figure shows a request sent through AXI using narrow access and payload size bigger than the MRRS value.

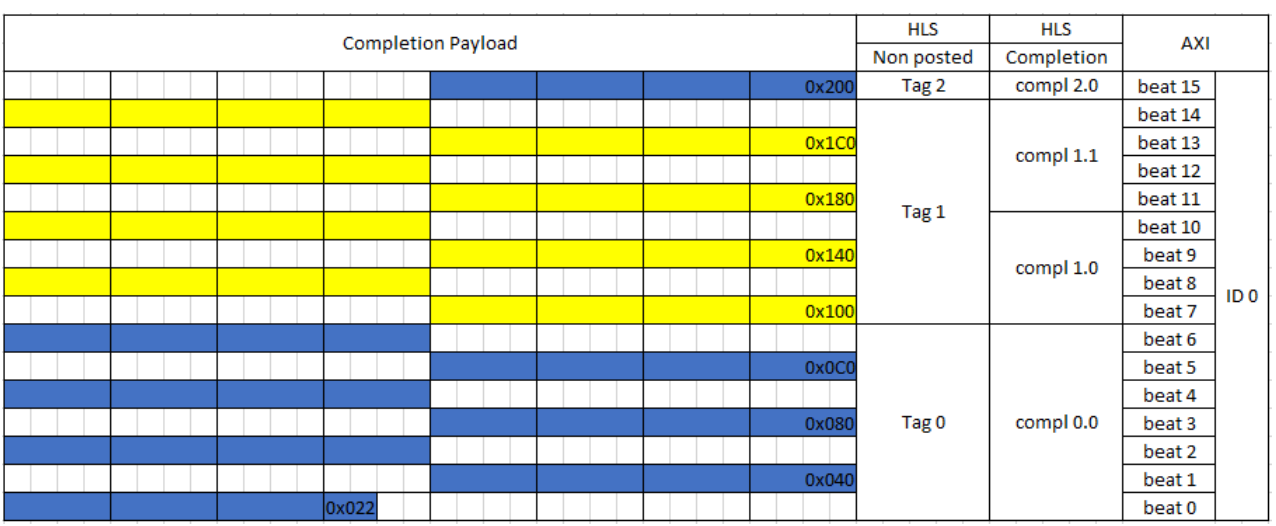


Figure 8.6: AXI Subordinate Memory Read Narrow Request Split

If user logic sends a request with WRAP burst type, then the transaction is split at wrap boundary address. The following figure shows a request sent through AXI using narrow access, WRAP burst type and payload size bigger than the MRRS value.

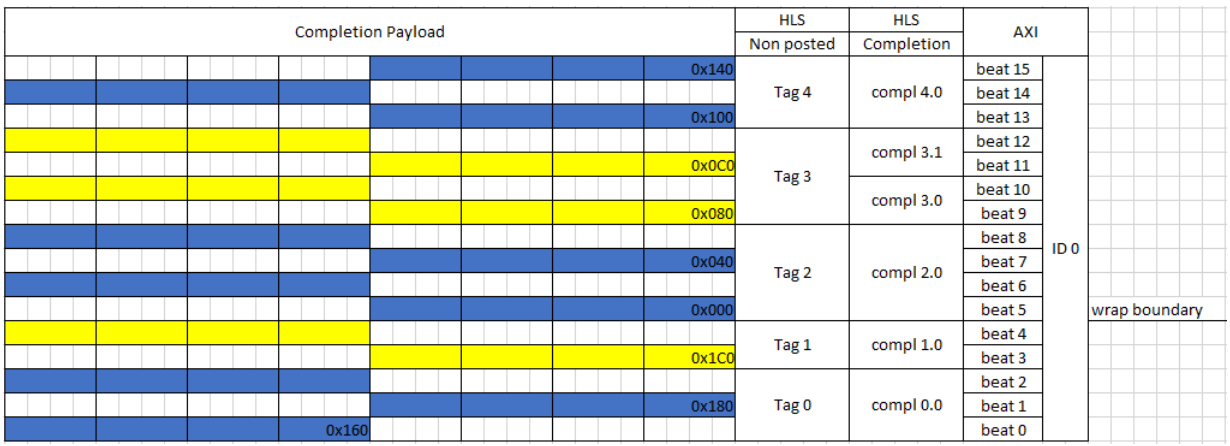


Figure 8.7: AXI Subordinate Memory Read Narrow Wrap Request Split

In Root Port, the AXI Subordinate interface is also used to initiate Configuration and I/O read requests. Configuration and I/O reads work in the same manner as memory reads. These requests are distinguished from memory requests by the transaction-type field. The data returned in response to these requests is always 4 bytes long and is delivered aligned to the request address.

8.2.3.1 AXI Subordinate Read ID Management

Read requests on the AXI Subordinate interface are performed as split transactions. That is, the client may make additional read and write requests while the completion for a read request is pending. Each outstanding read request can be issued by the client with the same *axi_s_arid* or unique ones. The AXI Subordinate internally maps the *axi_s_arid* to an internally generated PCIe tag. See *Tag Management for Non-Posted Transactions* for more details. This mapping is looked up to translate an incoming completion to a corresponding *axi_s_rid* value sent with the read data channel.

8.2.3.2 Completion Data Ordering

AXI specifications mandate that if there are outstanding read requests with same *axi_s_arid*, read data should be returned by the AXI Subordinate interface in-order. However, as each of these outstanding read requests are assigned unique tags on the PCIe side, the PCIe ordering rules permit the read completions come back in any order. The AXI Subordinate re-orders the out-of-order completions and issues them out on the AXI Subordinate Read Data channel in the correct order. If the client logic issues read requests with unique *axi_s_arid*, the AXI ordering rules permit the read data to come back out-of-order and the AXI Subordinate issues the read data in the order they come back from the PCIe Link.

8.2.3.3 R Channel Data Interleaving

A data interleaving may happen on the R channel of AXI. Lets assume that two AXI read requests have AXI ID ‘h11 and ‘h22. The request with AXI ID ‘h11 has ARLEN equal ‘h0A, which means 11 beats. 11 * 64b = 704 bytes, that exceeds MRRS limit of 512 bytes. Therefore, AXI Subordinate splits this AXI Request into two PCIe memory reads to avoid MRRS violation.

- One with TAG ‘h000 has length of 128 DWs (512 bytes).
- The other with TAG ‘h001 has length of 48 DWs (192 bytes).

The request with AXI ID ‘h22 has ARLEN equal ‘h05, which is 6 beats. This does not exceed MRRS limit of 512 bytes. Hence, this AXI request is translated into single PCIe memory read, which is assigned TAG ‘h002. Completions for PCIe requests with different TAGs may come back to the Manager in any order. Assume that the three PCIe request were completed in following order:

1 TAG ‘h000 (AXI ID ‘h11)
2 TAG ‘h002 (AXI ID ‘h22)

3 TAG ‘h001 (AXI ID ‘h11)

It is the order in which payload are sent on AXI R channel. That is, data from AXI ID ‘h22 with Tag ‘h002 is interleaved between the data of AXI ID ‘h11 with Tag ‘h000 and of AXI ID ‘h11 with Tag ‘h001. AXI Subordinate block only tracks the order of transactions with different PCIe TAGs within particular AXI ID. Because TAG ‘h002 belongs to different AXI ID, it is presented on AXI Subordinate R channel even though transaction with AXI ID ‘h11 (TAG h’000 and ‘h001) has not been yet concluded with RLAST.

Following options are available in the situation from example above:

- Handle AXI Subordinate R channel data interleaving in Application Layer,
- Avoid split of the AXI transaction into multiple PCIe Memory Read Requests
  - by using ARLEN <= MRRS, e.g. maximum ARLEN in this case is ‘h07, what indicates 8 beats of 64 bytes – 512 bytes
  - by increasing MRRS, e.g. to 1024 bytes, so ARLEN ‘h0A (up to ‘h0F) will not cause the split
- Use single AXI ID for AXI transactions
  - AXI Subordinate tracks the order of PCIe TAGs within single AXI ID pool. In described case it would present data of TAG ‘h000 (without RLAST), withhold data of TAG ‘h002, present data of TAG ‘h001 (with RLAST, as only know first AXI Read is concluded), when it’s received and only then present data of TAG ‘h002.

8.2.4 Tag Management for Non-Posted Transactions

The AXI Subordinate maintains the state of all pending Non-Posted transactions (Memory Reads, I/O Reads and Writes, Configuration Reads and Writes and Deferrable Memory Writes) initiated by the Application Layer, so that the Completions returned by the Completer can be matched against the corresponding Requests. The state of each outstanding Non-Posted transaction is held in the AXI Subordinate Completion LUT in the AXI Subordinate interface module. Its depth determines the maximum number of outstanding Non-Posted transactions. The offset and range of the Tag pool assigned to the Non-Posted PCIe Request are set by Tag Management for Non-Posted Request Register (see register description for more details). Each initiated Tag is unique for each of the pending Non-Posted Request. The returning Completions are matched with the pending Requests using the their Tag. The AXI Subordinate Completion LUT maintains a list of free Tags and assigns one of them to each Request when the Application Layer initiates a Non-Posted transaction.

Note:

AXI Subordinate Completion LUT have single Tag pool common for all Requests. It is defined by single offset and range, which implies that different PCIe Tag ranges (0-31, 0-255, 256-1023, 1024-16383) cannot be used simultaneously.

To use the 10-bit Tag (256-1023) capability, Application Layer should set the 10-bit Tag Requester Enable bit in Device Control 2 Register for all Functions.

To use the 14-bit Tag (1024 - 16383) capability, Application Layer should set 14-bit Tag Requester Enable bit in Device Control 3 Register for all Functions. Please note, that even though 14-bit Tag capability is available, number of available Tags is still limited by AXI Subordinate Completion LUT depth.

If in any Function the Extended Tag Field Enable bit in Device Control Register is clear and therefore also 10-Bit and 14-bit Tag Requester Enable bit should be clear, then Application Layer should set the 5-bit Tag range (0-31) for AXI Subordinate Completion LUT as result of the common Tag pool.

8.2.5 Application Layer Read Size

By default, AXI Subordinate Completion Reorder Buffer RAM size is equal to the maximum number of available PCIe TAGs (AXI Subordinate Completion LUT depth) multiplied by maximum supported MRRS. For example, 64 KB for 128 TAGs and 512 B MRRS.

Note: ECC protection overhead is not included in this calculation.

For large number of TAGs and large MRRS, such Completion Reorder Buffer RAM size might be too much and does not fit controller usage cases. For instance, when Aplication Layer wants to send many Read Requests with smaller size (utilizing maximum number of TAGs), but sometimes less Read Requests, however with bigger size.

AXI Subordinate Completion Reorder Buffer RAM can be downsized during initial IP configuration process. Application Layer Read Size field of AXI Subordinate *axi_slave_ctrl* register can be used to select desired size of Completion Reorder Buffer RAM per TAG chunk during controller operation.

128	Maximum number of TAGs
512 B	Maximum Read Request Size
64 KB	Maximum size of AXI Slave Completion Reorder Buffer RAM
16 KB	Downsized AXI Slave Completion Reorder Buffer RAM

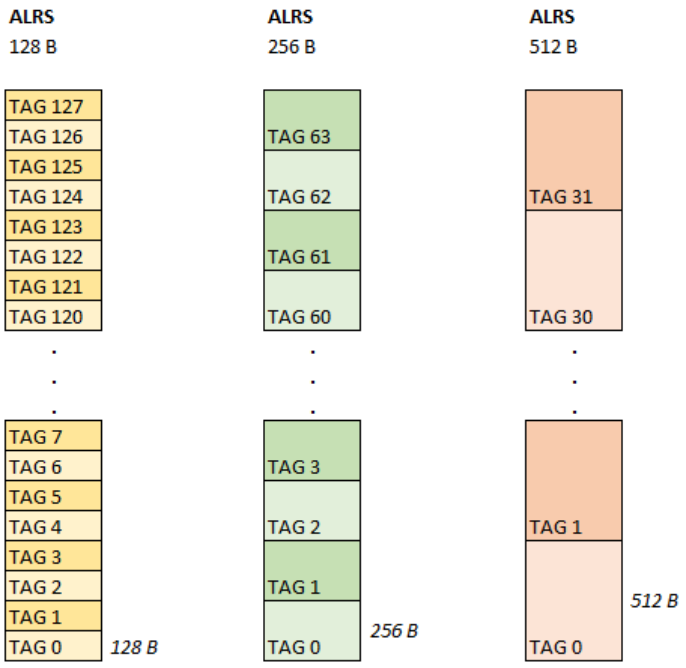


Figure 8.8: AXI Subordinate application layer read size

Figure 8.8 shows layout of AXI Subordinate Completion Reorder Buffer RAM for various ALRS settings. Maximum number of TAGs can be utilized when ALRS equals 128B. When ALRS is set to larger value, number of available TAGs decreases as there is not enough memory for data returned in PCIe Completions – respectively 64 and 32 TAGs are available for 256 B and 512 B ALRS.

Note: Application Layer is responsible for limiting number of available TAGs via *range* field of AXI Subordinate *tag_management* accordingly to prevent Completion Reorder Buffer RAM overflow.

Read Requests larger than ALRS still can be issued by Application Layer as they will be split by AXI Subordinate block into smaller TLPs that do not violate ALRS limit and fit into reserved Completion Reorder Buffer RAM chunk.

8.2.6 PF/VF Encoding

PF/VF (ipfivf) encoding is used at AXI metadata and other controller interfaces.

The PCIe controller allows setting of **SR_IOV** stride to a number, to the power of 2, as a configurable option. The number of VFs attached to a PF can also be modified using the local management register dynamically before the link training. The ipfivf format gives the VF number in the attached PF and the associated PF number. This helps the client logic to decode the BDF requester ID format using *sriov_stride* and allocated VFs to a PF.

PF is also encoded in the same format with a bit to indicate whether the encoding is for a VF or not.

Value of **VF_STRIDE** greater than one is useful only when multiple bus numbers are required to allocate the VFs. **VF_STRIDE** to the power of 2 is useful when uniform number of total VFs is assigned to all PFs and the host might program smaller.

numVFs is to optimize usage of bus numbers. stride of 1 is good if the *total_vf* is distributed based on the application and the host is not likely to optimize bus numbers.

Only stride value of 1 and a number to the power of 2 greater than the number of PFs are supported.

Table 8.3 shows examples of ipfivf encoding with different stride and VF allocations to PF.

Table 8.3: Acronym and Description

Acronym	Description
RID	BDF formatted ID as in PCIe TLP
ipfivf	Format with VF number in the associated PF
total_vf	Total VF allocated to a PF, programmable with LM register. This is an **SRIOV** read-only capability field.
num_vf	enabled number of VF count in **SRIOV** capability programmed by host.
vf_offset	RO field in **SRIOV** capability. (*vf_offset* + RID of the PF) gives the RID of the first VF in that PF. when *vf_stride* is not one, vf_offset is same of vf_stride. When *vf_Stride* is 1, *vf_offset* is (*num_pfs* + (total_vfs-1) of all the previous PFs.
vf_Stride	RO field in the **SRIOV** capability. VF's *RID* + *vf_Stride* gives the next VF's RID in a PF.

Colour code		
Active VF	Invalid VF	Disabled/Unused VF instance

Figure 8.9: Color code

8.2.6.1 Example with Uniform Number of Total VFs

There are 4 VFs in each PFs, the host had enabled only Two Vfs in PF0.

Total PFs: 2, Total VFS:4 per PF, PF0.num_vf=2, PF1.num_vf=4: vf_stride=1			
RID	ipfivf		
BDF format	Associated PF	VF number in PF	Is_VF
0	0	-	0
1	1	-	0
2	0	0	1
3	0	1	1
4	***0***	***2***	***1***
5	***0***	***3***	***1***
6	1	0	1
7	1	1	1
8	1	2	1
9	1	3	1

Total PFs: 2, Total VFS:4 per PF, PF0.num_vf=2, PF1.num_vf=4: vf_stride=2			
RID	ipfivf		
BDF format	Associated PF	VF number in PF	Is_VF
0	0	-	0
1	1	-	0

Total PFs: 2, Total VFS:4 per PF, PF0.num_vf=2, PF1.num_vf=4: vf_stride=2			
2	0	0	1
3	1	0	1
4	0	1	1
5	1	1	1
6	***0***	***2***	***1***
7	1	2	1
8	***0***	***3***	***1***
9	1	3	1

Note: In the above tables, the ***highlighted entries*** have Disabled/Unused VF instance.

8.2.6.2 Example with Non-uniform Number of Total VFs

There are 2 VFs in PF0 and 6 VFs are available PF1. The host had enabled only Four VFs in PF1 and only one VF in PF0.

Total PFs: 2, Pf0.total VF = 2, PF1.total VF=6, PF0.num_vf=1, PF1.num_vf=4, stride=1			
RID	ipfivf		
BDF format	Associated PF	VF number in PF	Is_VF
0	0	-	0
1	1	-	0
2	0	0	1
3	***0***	***1***	***1***
4	1	0	1
5	1	1	1
6	1	2	1
7	1	3	1
8	***1***	***4***	***1***
9	***1***	***5***	***1***

Total PFs: 2, Pf0.total VF = 2, PF1.total VF=6, PF0.num_vf=1, PF1.num_vf=4, stride=2			
RID	ipfivf		
BDF format	Associated PF	VF number in PF	Is_VF
0	0	-	0
1	1	-	0
2	0	0	1
3	1	0	1
4	***0***	***1***	***1***
5	1	1	1
6*	-	-	-
7	1	2	1
8*	-	-	-

Total PFs: 2, Pf0.total VF = 2, PF1.total VF=6, PF0.num_vf=1, PF1.num_vf=4, stride=2			
9	1	3	1
10*	-	-	-
11	***1***	***4***	***1***
12*	-	-	-
13	***1***	***5***	***1***

Note:

- In the above tables, the ***highlighted rows*** have Disabled/Unused VF instance.
- In the above table, the rows ***with** **** have Invalid VF.

8.2.7 AXI Subordinate Access

There are two methods to perform AXI Subordinate access

- Static method: Region Based
  - Region based method is useful when you only have the TLP type (read / write) with you and want to send a pre - programmed TLP information through PCIe in the outbound direction. Any change in the TLP information stored in regions has to be done through APB/AXI Lite reprogramming.
- Dynamic method: Sideband Descriptor Based
  - Sideband descriptor based method is useful when you already have the TLP information stored outside the controller and you want to send it through PCIe in the outbound direction. This will bypass the address translation logic in the PCIe AXI. Customers who already have the TLP information with them can make use of this feature so that they can save the re-programming time if they need to change the TLP information. (Re-programming of PCIe Descriptor registers needs to be done per region if the TLP information change in Region based method.)

8.2.8 AXI Subordinate Access through Regions

AXI Subordinate address space of PCIe core can be divided into a number of regions. There can be a maximum of regions active at a time and each region has a set of registers for its operation. The outbound access is a 2 step process

- Region Setup
  Region registers needs to be programmed before accessing them. The region registers can be programmed using the APB/AXI Lite interface provided.
- Region Access
  Region registers can be accessed through the AXI Subordinate interface (AXI Subordinate or axi_s_*).

Each region has the following set of registers.

Table 8.4: AXI Subordinate Region Access Registers for one region

Register set Name	**Description**
AXI Subordinate to PCIe Address Translation Registers	This register set is used to do the address translation from the AXI address to PCIe Address.
AXI Subordinate Descriptor to TLP Registers	This register set holds the TLP information to be sent outbound.
AXI Subordinate Region Base Address Registers	This register set hold the AXI region base address and the region size of the corresponding region. This register set is used to de-code the region by matching the incoming AXI address.

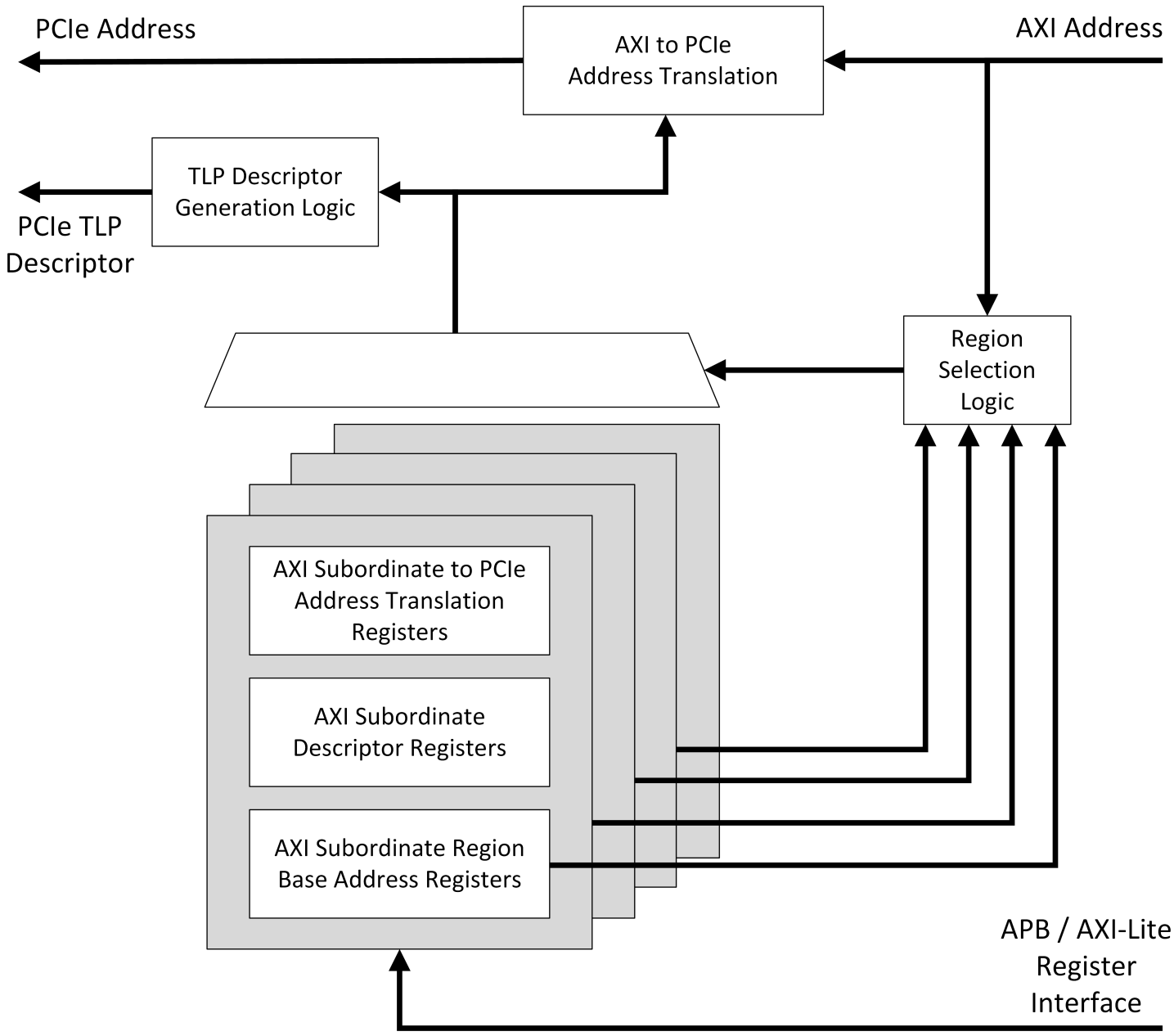


Figure 8.10: PCIe Controller AXI Subordinate access through regions

8.2.8.1 AXI Subordinate to PCIe Address Translation Registers

These registers are used to do the Outbound AXI to PCIe Address Translation for Memory and IO TLPs. The Configuration TLPs Bus Number, Device Number, and the Function Number can be programmed in the Outbound AXI to PCIe Address Translation Registers if the pass bits are programmed to pass less than specific bits. Refer to the PCIe address translation registers in the `<package_path>/models/ipxact/*` directory of the delivery package.

Some of the AXI Subordinate Address (*axi_s_awaddr*) bits are used to convey the MSG_ROUTING and MSG_CODE fields of the Message TLPs.

File *<package_path>/docs/cdnhpa_ips_axi_s_region_access_examples.html* contains examples of various AXI Subordinate accesses via Regions resulting in different PCIe TLPs.

8.2.8.2 AXI Subordinate Descriptor Registers

These registers hold the static information in the TLP (Function Number, Requester ID etc). Refer to the Subordinate descriptor registers in the `<package_path>/models/ipxact/*` directory of the delivery package.

File *<package_path>/docs/cdnhpa_ips_axi_s_region_access_examples.html* contains examples of various AXI Subordinate accesses via Regions resulting in different PCIe TLPs.

8.2.8.3 AXI Region Base Address Registers

The PCIe core uses a region select logic to match outbound AXI address and the preprogrammed AXI address (in the AXI Region Base Address Registers for each region) to figure out the region it belongs to. The matching is done from region 0 to maximum number of regions minus one. The comparator selects the first matching region as the region number which is used to pick the static TLP information (PCIe Descriptor) and PCIe address (for address translation). The AXI region sizes and region base addresses are programmable in this way. Refer to the AXI region base address registers in the `<package_path>/models/ipxact/*` directory of the delivery package.

Note:

Overlapping regions are not supported.

All the AXI Regions have their start address aligned to the region size which is programmed through the AXI Region Base Address Register axi_addr0 [5:0]. If no match is found by the region select logic, an appropriate error response is given backat the AXI.

8.2.9 AXI Subordinate Access through Sideband Descriptor

If user wants to directly send an PCIe address packet without doing any address translation, this method can be used. This is useful in case of address translations that are dynamic.

In this scheme of AXI Subordinate access, PCIe Address is directly driven at the *axi_s_aw/raddr*. There is no address translation performed on the AXI address in this scheme. The Outbound PCIe Descriptor is directly driven at the *axi_s_aw/ruser*. There is a VALID bit to enable sideband access, that is 159th bit in the *axi_s_aw/ruser* which gives sideband scheme of access priority over the region scheme of access.

For sideband descriptor, refer to the axi_s_awuser/axi_s_aruser signal descriptions in `<package_path>/docs/cdnhpa_ips_IO_descriptions.html` file of the delivery package.

File *<package_path>/docs/cdnhpa_ips_axi_s_sideband_access_examples.html* contains examples of various AXI Subordinate accesses via Sideband Descriptor resulting in different PCIe TLPs.

8.2.10 AXI Subordinate Access Note

- The client logic must check the following conditions before making a request on the AXI Subordinate intertace:
  - Only Root Ports can initiate I/O and Configuration.
  - The originating Function is not currently processing a Function-Level Reset (when FLR is implemented in the core).
- AXI Subordinate expects the client logic to program the *AXI Subordinate Descriptor Registers*, from which the TLPs are generated, in compliance with PCIe Specification.
- AXI Subordinate expects the data of the Normal/Vendor Defined Message request must be aligned to the DWORD (4-bytes) on the data channel (*axi_s_wdata*) in AXI transfer.
- Messages without Data should have no byte strobes set (*axi_s_wstrb* = 0).
- AXI Subordinate supports multiple outstanding ATS Translation Requests. However, each of those Requests is allowed to ask only for a single translation (Length of the AT Request TLP must be limited to 2 DWORDs, 8 bytes). AXI address of ATS Translation Requests must be aligned to 8 bytes.
- AXI Subordinate does not allow to use same AXI ID to simultanously make Posted and Non-Posted Write Requests, because it does not assure the order of the AXI B Responses in such case.
  - In case Application Layer wants to use same AXI ID to send both, Posted and Non-Posted Write Requests, it must restrain from making the Request of the other kind until all AXI B Responses to Requests of the first kind return.
- AXI Subordinate blocks all new Requests after access which results in decode error (DECERR), until it handles all outstanding Requests. This prevents violation of AXI response ordering.
- ECRC is generated internally by the Controller IP when *ecrc_capability_support* is enabled. Application Layer

must ensure that ECRC Generation Enable bit (Advanced Error Capabilities and Control Register) is set for given Function.

- **PCIe6, Flit Mode** - in case Application Layer generates Write Request that may have Non-Contiguous Byte Enables, OHC-A must be enabled for such Request

8.2.11 AXI Subordinate Access by Examples

The following figure shows the Outbound AXI Memory Map with different types of regions marked. The map shows just the partitioning of the regions into 3 with corresponding base address and region size. However, the actual region type is assigned once we program the “AXI Subordinate Descriptor to TLP Registers” with correct values for each type of TLP as given in . Program the Base address and region size into the “AXI Region Base Address Registers” for the corresponding region number.

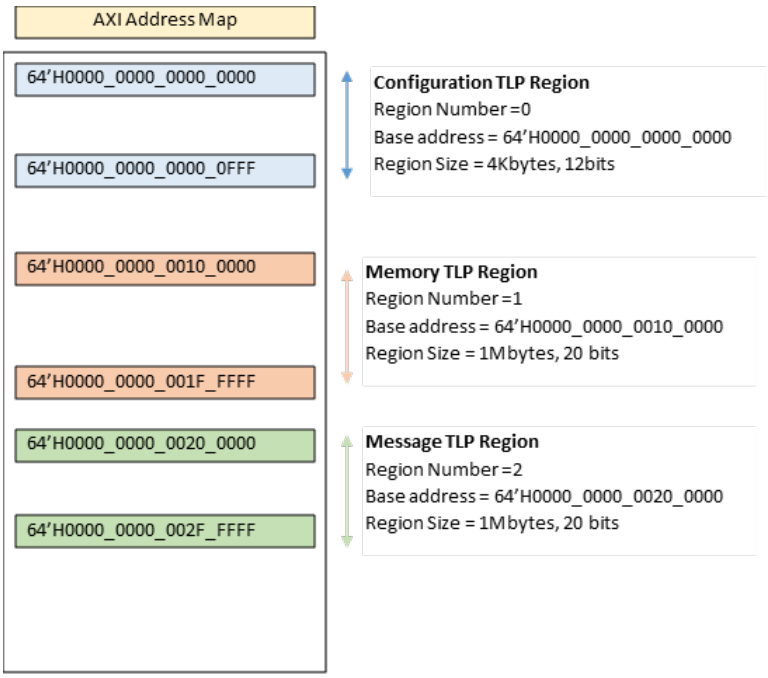


Figure 8.11: AXI Subordinate Address map for TLP accesses

In this example, programming values of the “AXI Region Base Address Registers” is given in the following table.

Table 8.5: Values of the AXI Region Base Address Registers

Region Number	axi_addr0[5:0]	axi_addr0[31:8]	axi_addr1[31:0]
0(configuration TLP)	6’d11	24’h00_0000	32’h0000_0000
1(Memory or IO TLP)	6’d19	24’h00_1000	32’h0000_0000
2(Message TLP)	6’d19	24’h00_2000	32’h0000_0000

8.2.11.1 Configuration TLP Access

There are two methods to convey the BDF (Completer ID) information for a configuration TLP. Configuration TLPs are always routed to the PCIe Config Space of the EP devices.

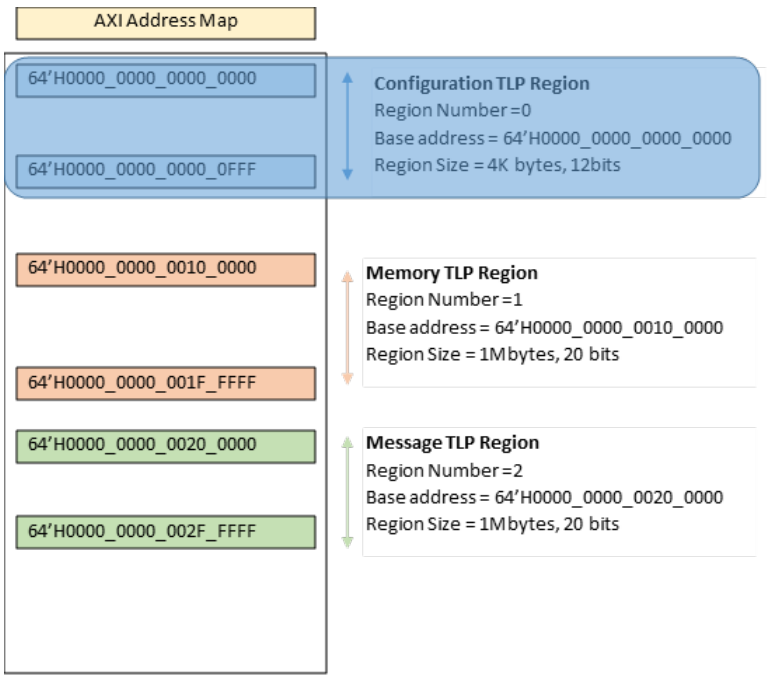


Figure 8.12: AXI Subordinate Address Map for configuration TLP

Method 1: Bus Number, Device Number and Function Number through the "AXI Subordinate to PCIe Address Translation Registers

The Configuration register address can be driven at the axi_s_aw/araddr[11:0] bits. The pass bits or the addr0 [5:0] of the "AXI Subordinate to PCIe Address Translation Registers" should be programmed to 6'd11 so that the lower 12 bits are taken from the axi_s_aw/araddr. In this case the BDF information is programmed in the corresponding region's (region 0 in this examples) "AXI Subordinate to PCIe Address Translation Registers" while the extended register number and the register number are passed from axi_s_aw/araddr[11:0] bits.

Table 8.6: BDF values programming for configuration TLPs through outbound AXI to PCIe Address Translation Register (method 1)

Legacy Mode		ARI Mode	
addr0 bits	**Value**	**addr0 bits**	**Value**
27:20	Bus Number	27:20	Bus Number
19:15	Device Number.	19:12	Function Number
14:12	Function Number		

Table above shows the BDF values to be programmed in the addr0 register of the "AXI Subordinate to PCIe Address Translation Registers" for Legacy and ARI mode.

Note:

AXI region size should be at least 4K bytes if we use this method of sending configuration TLPs.

Method 2: Bus Number, Device Number and Function Number through the axi_s_aw/araddr

In this mode of sending configuration TLPs, we can program the addr0 [5:0] of the "AXI Subordinate to PCIe Address Translation Registers" to 6'd27 so that the lower 28 bits of the AXI address is passed to the PCIe core directly. These bits include the BDF values as indicated below.

Table 8.7: BDF Values Programming for Configuration TLPs through Outbound AXI to PCIe Address Translation Register (method 2)

Legacy Mode		ARI Mode	
axi_s_aw/araddr bits	**Value**	**axi_s_aw/araddr bits**	**Value**
27:20	Bus Number	27:20	Bus Number
19:15	Device Number.	19:12	Function Number
14:12	Function Number		

Note:

AXI region size should be at least 256 Mbytes (28 bits) in this approach since the BDF information is passed through the 27:12 of the axi_s_aw/araddr or AXI address.

In both the methods, addr1 [31:0] should be programmed to zero.

8.2.11.2 AXI Subordinate Descriptor to TLP Registers Programming

AXI Subordinate Descriptor Register programming once done assigns a specific TLP type to the region number. For this example, ensure that you follow the values to be programmed in the region 0 desc0, desc1, etc. registers given in the `<package_path>/models/ipxact/*` directory.

Configuration TLP AXI to PCIe Address Translation

Consider a different example where the AXI configuration TLP address map is not starting from the 64'h0000_0000_0000_0000, then we can use the address translation feature in the PCIe Core to translate the addresses to requester id and configuration register number. This can be thought as a "PCIe Configuration Space visible to RP".

Method 1: In this method (Method 1) AXI region can have a region size of 4K bytes (lower 12 bits of AXI address). The BDF information is captured from the region registers as explained in the Method 1 description above.

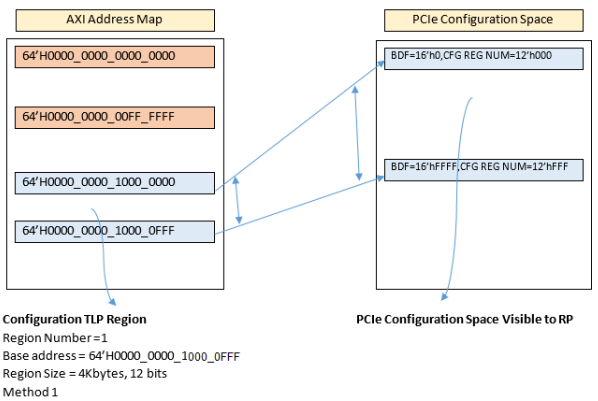


Figure 8.13: AXI Subordinate to PCIe Configuration space address translation for configuration TLPs using method 1

Method 2: In this method (Method 2) lower 32 bits of the AXI address is used to convey configuration register address and BDF information.

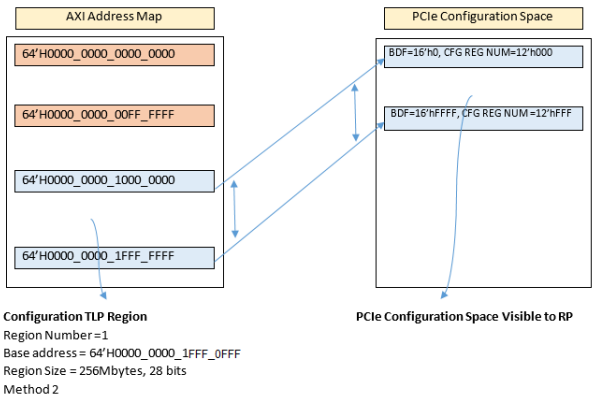


Figure 8.14: AXI Subordinate to PCIe Configuration space address translation for configuration TLPs using method 2

Note: PCIe Configuration write/ IO write requests are having a payload size of 1 DW.

1. The AXI Address should be aligned to the first Data Byte Enable that is provided. The Data/BE starts from the first address location given.
2. One DWord of Data starting from the above address is used for Config/IO write payload.

Table 8.8: PCIe Configuration write/IO write requests examples

	Example1	Example2
axi_s_addr	0x4	0x0
axi_s_wstrb	0xF0	0x0F

8.2.11.3 Memory or IO TLP Access

Let us assume that the one of the PCIe Controller's Memory Region has an AXI Region Base Address as 64'h0000_0000_1000_0000 region 1 as shown in the following figure) with a region size of 1Mbytes. For example, assume that the PCIe Base Address to be written to is 64'h0000_0000_3000_0000.

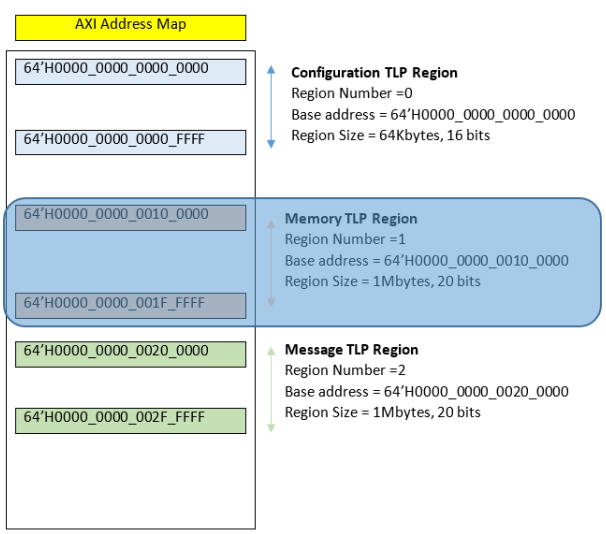


Figure 8.15: AXI Subordinate address map for memory or IO TLP access

AXI Subordinate Descriptor to TLP Registers Programming

AXI Subordinate Descriptor Register programming once done assigns a specific TLP type to the region number. Ensure that you follow the values to be programmed in the desc0, desc1, etc. registers provided in the deliverables. Memory or IO TLPs column.

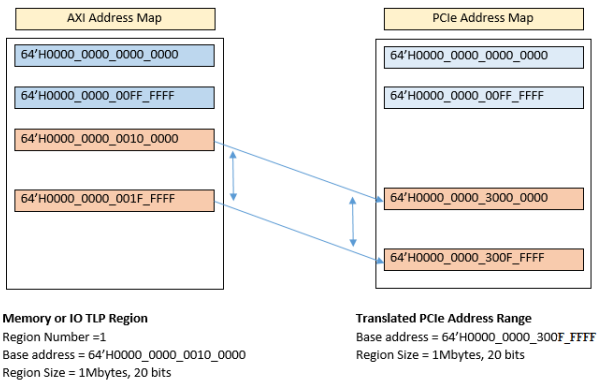


Figure 8.16: AXI Subordinate to PCIe address translation register programming

Table 8.9: AXI Region Base Address Register and AXI Subordinate to PCIe Address Translation Register Values

Register	Value	Register	Value
axi_addr0[5:0]	6'd19	addr0[5:0]	6'd19
axi_addr0[31:8]	24'h00_1000	addr0[31:8]	24'h30_0000
axi_addr1[31:0]	32'h0000_0000	addr1[31:0]	32'h0000_0000

Message TLP Access

When an Outbound Message Access is made to an AXI Region, the PCIe Header fields are driven mainly from the region register values. Only message code and message routing fields are driven through the AXI address *axi_s_awaddr*.

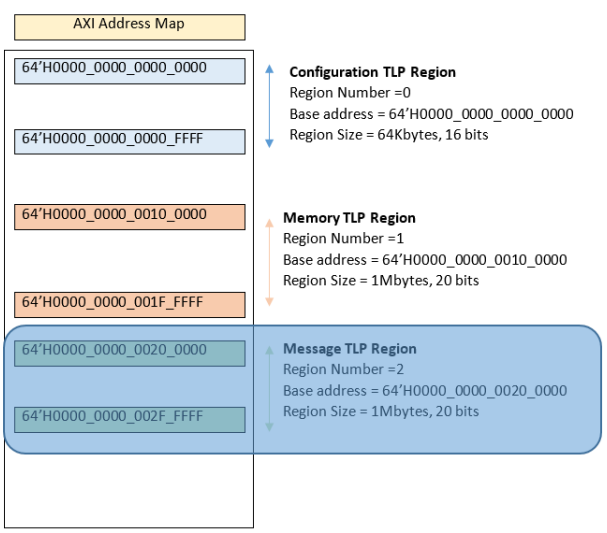


Figure 8.17: AXI Subordinate address map for message TLP access

This example uses region 2 as a Message region. Messages region (region 2) is decoded once an access is made with an AXI address which falls in between 64'h0000_0000_0020_0000 and 64'h0000_0000_002F_FFFF. The **AXI Subordinate Descriptor Registers** for region 2 needs to be programmed with values for Message TLPs.

Note:

Messages should have a minimum region size of 512Kbytes (Message code, message routing and Message with/without data bit is driven through the AXI address). shows how the Message Code Message routing and Message with/without data bit are driven through AXI address.

For vendor defined Messages the vendor defined Message Header [127:64] is driven through the AXI Subordinate to PCIe Address Translation Registers. This is explained in the *AXI Subordinate to PCIe Address Translation Registers* section.

8.2.12 AXI Subordinate (Error) Response

Application Layer has possibility to configure AXI Subordinate response codes (AXI xRESP, *axi_s_bresp, axi_s_rresp*) for several error categories and TLP types, using AXI Subordinate Response Control registers.

The following options are available:

- 2'b00 – OKAY
- 2'b10 – SLVERR
- 2'b11 – DECERR

Value 2'b01 is RESERVED.

Table 8.10 shows different error categories.

Table 8.10: AXI Subordinate Error Response categories

Category	Description
1	**Non-successful Completion Status.** This category includes cases, where Completion with non-successful Completion Status was received, that is, *Completion Abort* (CA), *Unknown Request* (UR), *Retry Request Status (RRS)*. Default error code for this category is SLVERR.
2	**Completion errors, other than non-successful Completion Status.** This category includes all Completion errors, other than non-successful Completion Status. For example, Completion Timeout, Completion byte count mismatch, etc. Default error code for this category is SLVERR.
3	**Link-Down event.** This category includes responses generated by AXI Bridge during Link-Down event flush. Default error code for this category is SLVERR.
4	**(Non-)Prefetchable Base/Limit filtering.** This category includes responses for Memory Transactions which failed (Non-)Prefetchable Base/Limit check and were filtered by AXI Bridge. Default error code for this category is SLVERR.
5	**Errors detected by AXI Bridge.** This category includes error cases detected by AXI Bridge, which are often result of Application Layer mistake and make TLP generation impossible, e.g. usage of unmapped Traffic Class, unknown Request, unmatched region access. Default error code for this category is DECERR. This category is common for all TLP types.

Table 8.11 shows the error categories that are applicable to different TLP types. This is reflected in AXI Subordinate Response Control registers fields.

Note: Category 5 is common for all TLP types. This is because, in case of Category 5 error, it might be impossible to determine TLP type.

Table 8.11: Error categories applicable to different TLP types

TLP Type	Category			
	1	2	3	4
Cfg	YES	YES	YES	NO
IO	YES	YES	YES	NO
Mem	YES	YES	YES	YES

Table 8.11: Error categories applicable to different TLP types

TLP Type	Category			
	1	2	3	4
AtomicOp	YES	YES	YES	YES
DMWr	YES	YES	YES	YES
Msg	NO	NO	YES	NO

Table 8.12 shows detailed error codes of AXI Subordinate response (AXI xUSER, *axi_s_buser, axi_s_ruser*).

Table 8.12: AXI Subordinate response detailed error codes (xUSER, axi_s_buser, axi_s_ruser)

Error origin	Error code ('b)	Error code ('d)	Error description	Category
-	00000	0	No error.	-
PCIe core	00010	2	Completion TLP is poisoned.	2
PCIe core	00100	4	RRS Status received for Vendor ID access. This code is set when CRS software visibility is set and RRS Status is received for a CFG access to Vendor ID address.	1
PCIe core	00110	6	Request terminated by Completion without data or there is a byte count mismatch between Request and Completions from remote side.	2
PCIe core	01000	8	Received Completion has the same PCIe Tag as an outstanding Request, but its Requester ID, TC or Attr fields did not match with parameters of this Request.	2
PCIe core	01010	10	Error in starting address. The Low Address bits in the Completion header did not match with the starting address of the next expected byte for the Request.	2
PCIe core	01100	12	Request terminated as it is not in the Type1 Base/Limit range. Valid only for RP.	4
PCIe core	10000	16	Request terminated by Completion Timeout.	2
PCIe core	10010	18	Request terminated by an FLR targeted at the Function that generated the Request.	2
PCIe core	10100	20	Request terminated with unexpected RRS or RCB violation for the memory packet.	2
PCIe core	11000	24	Request terminated by a Completion with UR status (includes other cases where spec allows to infer UR-Completion).	1
PCIe core	11010	26	Request terminated by a Completion with CA status.	1

Table 8.12: AXI Subordinate response detailed error codes (xUSER, axi_s_buser, axi_s_ruser)

Error origin	Error code ('b)	Error code ('d)	Error description	Category
PCIe core	11100	28	Request terminated by a Completion with RRS status.	1
AXI Bridge	00011	3	Unknown Request – Sideband descriptor or Region is misconfigured.	5
	00101	5	Unmapped Traffic Class.	5
	00111	7	Region based access did not match any region.	5
	01001	9	(Non-)Prefetchable Base/Limit check failed. Valid only for RP.	4
	10101	21	AXI Bridge Link-Down event flush.	3

Note: Data in AXI Subordinate R-response with error will be set to all 1s. This rule has one exception for error response for Completion with RRS Status to Configuration Read Request, when Configuration RRS Software Visibility is enabled. In this case, PCIe Specification requires returned data to be 0xFFFF0001.

8.3 AXI Manager

AXI Manager interface ends inbound path of the PCIe controller. It might be also shared with the DMA module.

It is compliant with *AXI-4 Specification* and incorporates AXI-5 Atomic Transaction feature. AXI Manager module transforms inbound PCIe Transaction Layer Packets (Requests) into AXI transactions (Writes or Reads) and if necessary, generates PCIe TLPs (Completions) based on the AXI response. The Controller might have multiple AXI Manager interfaces (depending on the number of Client Interfaces). Inbound traffic is routed based on the VC to appropriate CIF AXI Manager.

8.3.1 PCIe to AXI Address Translation

SoC address space can be completely different from PCIe address space, therefore AXI Manager provides possibility to map PCIe BARs into AXI address space and perform address translation for incoming traffic.

8.3.1.1 PCIe Messages

PCIe Message TLPs do not have address. AXI Manager provides two registers (*inb_at_msg_addr0, inb_at_msg_addr1*), where target region for those Requests can be specified. Data from these registers is combined to form 64-bit address of target region in AXI space.

This address must be 4KB aligned.

8.3.1.2 RP Mode

In RP mode, two PCIe BARs are available. AXI Manager provides three sets of registers – for BAR0 (*inb_at_rp_bar0_addr0, inb_at_rp_bar0_addr1*), BAR1 (*inb_at_rp_bar1_addr0, inb_at_rp_bar1_addr1*) and for so called "BAR7" (no match) – for Requests that missed BAR0 or BAR1 (*inb_at_rp_bar7_addr0, inb_at_rp_bar7_addr1*).

Data from *addr0* and *addr1* registers is combined to form 64-bit address of region in AXI space.

For Requests targeting BAR0 and BAR1 final AXI address is constructed as following – upper bits are taken from the registers, while lower from TLP address. Number of lower bits depends on the given BAR aperture.

Same mechanism is used in case of no match ("BAR7") - upper bits are taken from the registers, while lower from TLP address, but number of lower bits depends on *PASS_BITS* field from *inb_at_rp_bar7_addr0* register, since BAR7 formally does not exist and so its aperture is unknown.

8.3.1.3 EP Mode

Per PF

In EP mode, every Physical Function has 7 PCIe BARs: BARs from 0 to 5 and Expansion ROM BAR ("BAR6").

AXI Manager provides set of registers per BAR within every PF (*inb_at_ep_pf_X_bar_Y_addr0*, *inb_at_ep_pf_X_bar_Y_addr1*). Data from addr0 and addr1 registers is combined to form 64-bit address of region in AXI space.

For Requests targeting BARs 0 to 6 final AXI address is constructed as follows:

- Upper bits are taken from the registers
- Lower from TLP address.

Number of lower bits depends on the given BAR aperture. Address of TLPs that misses all BARs (so called “BAR7” – no match) is not translated and is passed as it is.

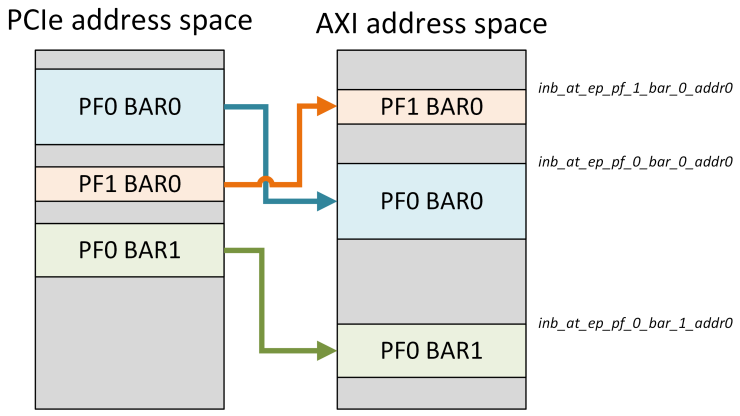


Figure 8.18: Example mapping of PF BARs from PCIe to AXI address space

Per VF

In EP mode, 0 to N Virtual Functions can be assigned to Physical Function. This section is applicable when PF has 1 to N VFs assigned. Every VF has 6 PCIe BARs: BARs from 0 to 5. Memory spaces assigned to BAR0 of VFs belonging to given PF are consecutive. See *Figure 8.19* . AXI Manager provides set of registers per VF BAR within every PF (so 6 sets – *inb_at_ep_pf_X_vf_bar_Y_addr0*, *inb_at_ep_pf_X_vf_bar_Y_addr1*). Data from addr0 and addr1 registers is combined to form 64-bit address of region in AXI space, that is beginning of the VF0 memory space. Upper bits of final AXI address are taken from the registers with adjustment depending on the associated VF number and VF BAR aperture.

$$VF_v\ BAR_b = VF\ BAR_b + v * VF\ BAR_b\ aperture$$

Lower bits are taken from TLP address. Number of lower bits depends on the given VF BAR aperture.

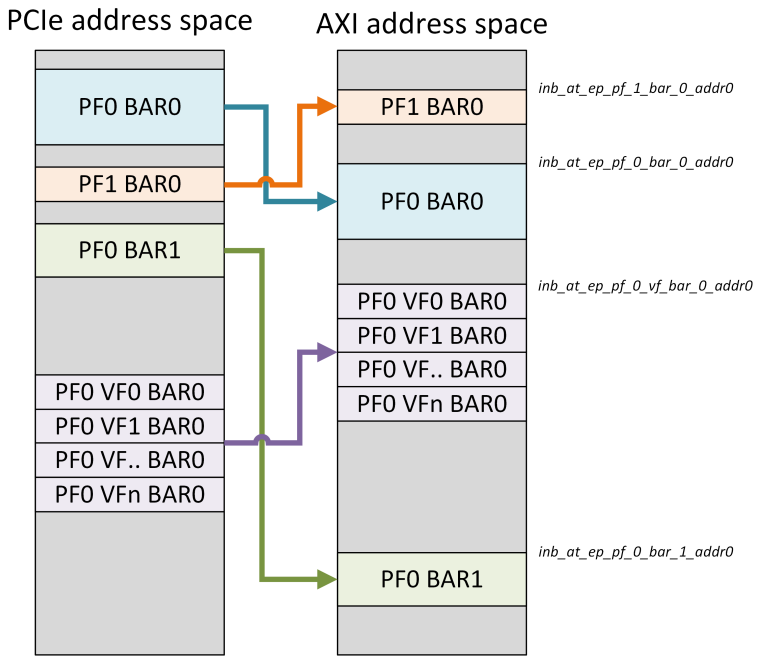


Figure 8.19: Example mapping of PF and VF BARs from PCIe to AXI address space

For example, *inb_at_ep_pf_0_vf_bar_3_addr0* (VF BAR3) = 0x0480, VF BAR3 aperture = 128b (0x80)

TLP targeting VF2 of PF0 with address 0xFF72 in PCIe space is translated to:

- Upper bits:

  $VF_2\ BAR_3 = VF\ BAR_3 + 2 * VF\ BAR_3\ aperture = 0x480 + 2 * 0x80 = 0x0580$
- Lower bits: 0x72 (7 bits)
- Final address = 0x05F2.

CXL

CXL 1.1

EP in CXL 1.1 mode has two additional memory regions – RCRB and MEMBAR0. AXI Manager provides two sets of registers for these regions (*inb_at_ep_cxl_rcrb_addr0, inb_at_ep_cxl_rcrb_addr1* and *inb_at_ep_cxl_membar0_addr0*, *inb_ab_ep_cxl_membar0_addr1*).

Data from *addr0* and *addr1* registers is combined to form 64-bit address of region in AXI space.

For Requests targeting these two regions final AXI address is constructed as following – upper bits are taken from the registers, while lower from TLP address. Number of lower bits depends on the given BAR aperture.

CXL 2.0, 3.0

In CXL 2.0 and higher there is no need to configure these registers as in this mode standard PCIe BAR mechanism is used.

8.3.2 AXI Manager Write Access

This section describes how PCIe Write Requests. Posted Memory Writes and Messages along with Non-Posted IO Writes, Deferrable Memory Writes and Atomic Operations are transformed into AXI Manager Writes.

8.3.2.1 AXI ID Pool

AXI Manager has AWID pool divided into several sections, dependent on the supported TLP-types (AtomicOp, Posted, other Non-Posted). Presence of these sections and their size depends on controller configuration. Detailed information regarding AWID pool can be found in *<package_path>/docs/cdnhpa_ips_axi_m_axid.html.*

8.3.2.2 Request Transformation

Figure 8.20 shows how PCIe Write Request TLPs are transformed into AXI Write transactions.

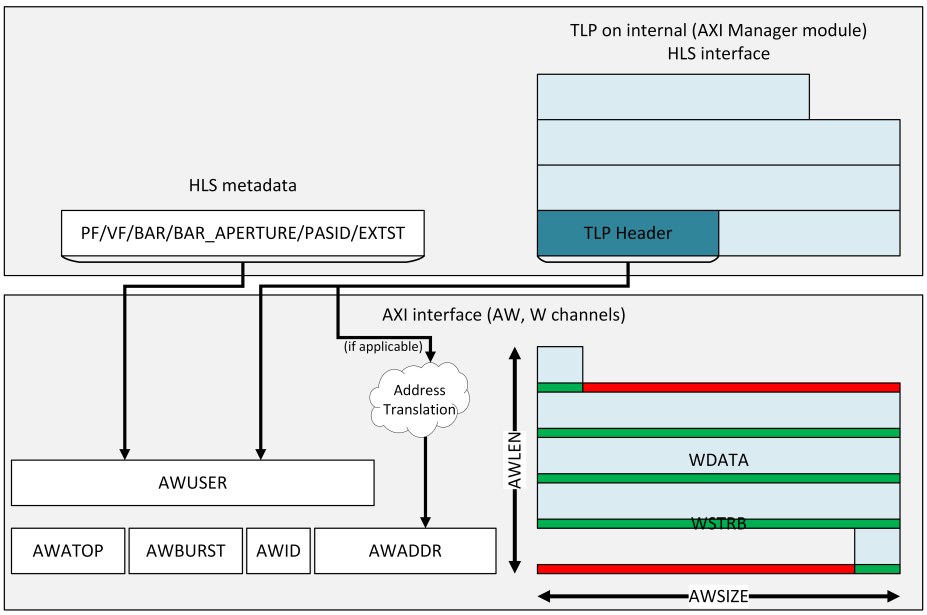


Figure 8.20: Transformation of PCIe Write Request TLP into AXI Write transaction

AXI Manager module receives serialized TLPs. Each of these is accompanied with packet metadata vector, that stores some of PCIe information related to given packet. TLP header is stripped and its contents together with metadata vector are utilized to form AW channel transfer.

AW Channel

- *AWADDR* – TLP address is fetched from the header and PCIe to AXI address translation is performed,
- *AWID*– AXI ID is allocated depending on TLP type and AXI Manager LUTs capacity in case of Non-Posted Requests. For more information see *AXI ID Pool*
- *AWUSER*– TLP header is presented in this signal along with other PCIe information (targeted PF/VF, BAR hit, BAR_APERTURE, PASID, Extended Steering TAG, etc. – details can be found in *<package_path>/docs/cdnhpa_ips_IO_description.html*),
- *AWATOP*– value 6'b000000 indicates non-Atomic access, non-zero value indicates Atomic Transaction. Allowed signalization is described in chapter strictly devoted to AtomicOp Requests (Atomic Transactions might impact *AWLEN* and *AWSIZE* signalization, following AXI-5 Atomic Transaction rules),
  - fully compliant AXI-5 fabric is not required, if AXI-4 fabric can follow AXI-5 Atomic Transaction rules,
- *AWLEN*– length of the data burst is calculated from TLP payload length and address offset (depends on selected data path width),
- *AWSIZE*– Memory TLPs resulting in single-beat AXI bursts will have smallest possible *AWSIZE*; for IO TLPs (always resulting in single-beat AXI bursts) *AWSIZE*will always indicate 4 bytes; Memory TLPs resulting in multi-beat AXI bursts will have *maximal*possible *AWSIZE* – equal data path width,

Note:

AWSIZE is calculated once per TLP and when certain conditions occur for Posted Requests (described in section *Posted Requests – Splits*) and burst must be *split* into multiple AXI transactions, all of them will have same, maximal possible *AWSIZE* – which might be suboptimal for selected single-beat bursts (results of the *split*).

Table 8.13: Example AWLEN and AWSIZE signalization for 512-bit data path width

TLP parameters		AXI AW Channel			Comment
Address	Length in bytes	AWADDR	AWLEN	AWSIZE	-

Table 8.13: Example AWLEN and AWSIZE signalization for 512-bit data path width

TLP parameters		AXI AW Channel			Comment
0x3	1	0x3	0x00	0x0	Single byte.
0x4	4	0x4	0x00	0x2	First byte accessed aligned with natural address boundary.
0x10	8	0x10	0x00	0x3	
0x2	4	0x2	0x00	0x3	First byte accessed unaligned with natural address boundary
0xF	2	0xF	0x00	0x5	
0x3F	2	0x3F	0x01	0x6	Multi-beat bursts. Full *AWSIZE*.
0x20	96	0x20	0x01	0x6	

- *AWBURST* – always INCR for non-Atomic bursts.

W Channel

- *WDATA* –TLP payload is passed on this bus; data are realigned, so the location of the first byte matches target address,
- *WSTRB* – AXI Manager module calculates appropriate strobes for every burst, so that only valid bytes are marked (starting from the first byte indicated by AWADDR, until last byte of data); TLP header fields – First Byte Enable and Last Byte Enable are utilized in the process and influence strobes of first and last fours of bytes. *WSTRB* allows to handle the special cases of PCIe Write TLPs:
  - Zero-length Write. See *Special Case Zero-length Write*.
  - Non-contiguous Byte Enables. See *Special Case Non-contiguous Byte Enables*.

8.3.2.3 Posted Requests

Memory Writes and Messages are Posted Requests supported by AXI Manager module.

B channel responses for Posted Requests are used only to maintain internal order of Posted and Non-Posted TLPs – BRESP code is ignored.

The following section describes conditions of Posted Request splits.

8.3.2.4 Posted Requests – Splits

Size of Posted Requests might reach up to 4kB and it may be desirable from the system perspective to divide large packets into smaller chunks. Posted Request *split* might be result of two conditions, being consequence of following configuration variables settings:

- AXI Manager write burst length limit (*Figure 8.21*),

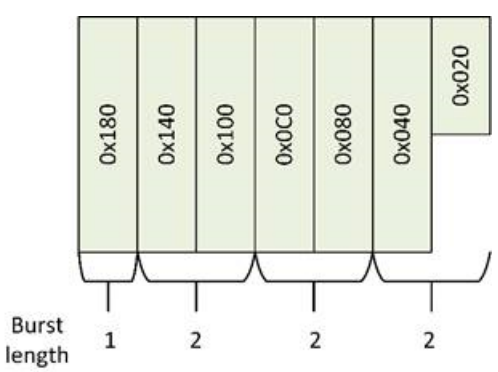


Figure 8.21: Posted Request split – burst length limited to 2

- 1kB address boundary, if enabled (*Figure 8.22*).

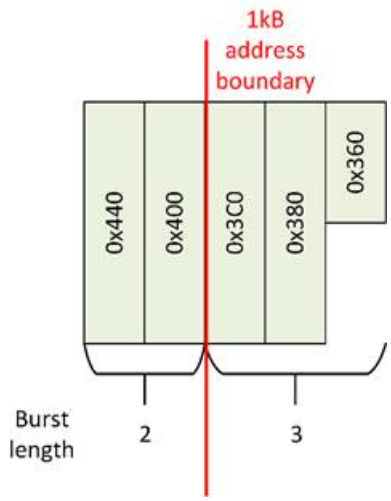


Figure 8.22: Posted Request split – split on 1kB address boundary

It may happen that both conditions are active at the same time – in such situations rule that forces shortest split will be applied. In *Figure 8.23* AXI Write consists of 7 beats. First split was made due to limiting burst length to maximum 2 beats, but for second split both conditions apply – 1kB address boundary rule forces shorter burst length of 1 beat.

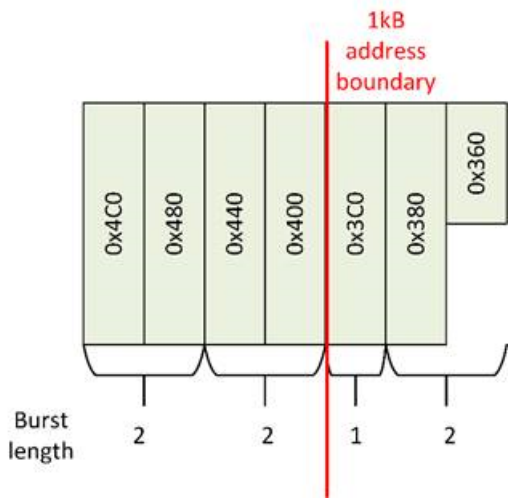


Figure 8.23: Posted Request split - both conditions occurring at the same time

8.3.2.5 Non-Posted Write Requests

IO Writes, Deferrable Memory Writes and Atomic Operations are Non-Posted Write Requests supported by AXI Manager module (last two types are dependent on appropriate configuration variables).

See *Atomic Operations* for additional information.

IOWr and DefMemWr require PCIe Completion. It is generated by AXI Manager module, when B channel response for AXI transaction is received and its Completion Status field value depends on BRESP code and BUSER value (for DefMemWr) as in *Table 8.14*.

Table 8.14: Possible values of Completion Status for IO Writes and Deferrable Memory Writes

	IOWr	DefMemWr	
		BUSER – bit 0	
		0 – accepted	1 - deferred

Table 8.14: Possible values of Completion Status for IO Writes and Deferrable Memory Writes

BRESP	OKAY	SC	SC	RRS
	SLVERR/DECERR	CA	CA	CA

Note: Non-Posted Write Requests are *never* split into multiple AXI transactions.

8.3.2.6 Outstanding Requests

Number of outstanding AXI Manager Write transactions (Posted and Non-Posted Write Requests) generated by AXI Manager module depends on selected configuration (from 4, up to 4096). AXI *READY* and *VALID* signals will be masked, when outstanding transaction counter saturates. Outstanding transaction counter will be decremented, whenever B response (B and R response for Atomic Transactions) is received.

Every split request is counted to outstanding transaction limit.

8.3.2.7 Error Handling

BRESP errors for Posted Requests are ignored.

BRESP errors for Non-Posted Write Requests influence its Completion Status. See *Table 8.14*.

8.3.2.8 Special Case Zero-length Write

A Memory Write Request of 1 DW and no bytes enabled is called zero-length write. Such Request is transformed into single-byte AXI Write (AWLEN = 0x00, AWSIZE = 0x0) with all write strobes disabled.

8.3.2.9 Special Case Non-contiguous Byte Enables

The *PCIe Specification* allows Memory Write Requests with non-contiguous byte enables for 1 DW TLPs and 2 DW TLPs with address aligned to 8-byte boundary (QW – QuadWord). For such Requests appropriate write strobes are generated, so only enabled bytes are marked valid on AXI.

8.3.3 AXI Manager Read Access

This section describes how PCIe Read Requests (IO Reads and Memory Reads) are transformed into AXI Manager Reads.

8.3.3.1 AXI ID pool

AXI Manager has ARID pool divided into several sections, dependent on the supported TLP-types (AtomicOp, other Non-Posted). Presence of these sections and their size depends on controller configuration. Detailed information regarding ARID pool can be found in *<package_path>/docs/cdnhpa_ips_axi_m_axid.html.*

8.3.3.2 Request Transformation

Figure 8.24 shows how PCIe Read Request TLPs are transformed into AXI Read transactions.

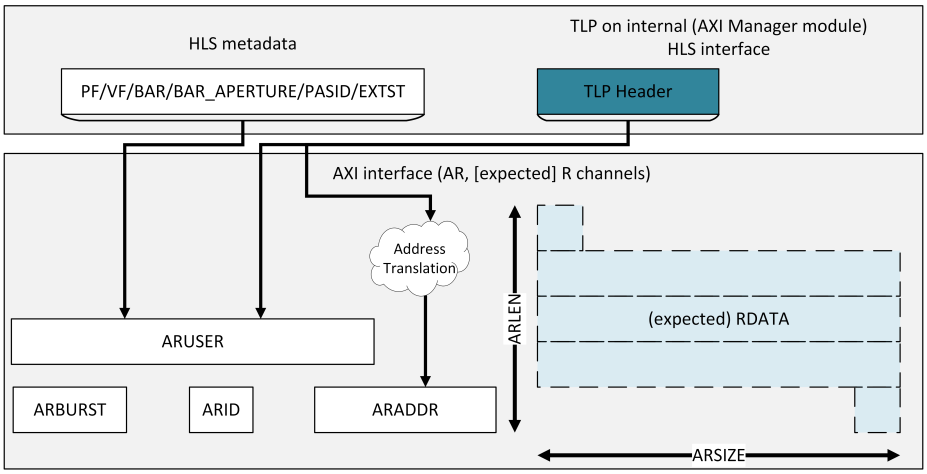


Figure 8.24: Transformation of PCIe Read Request TLP into AXI Manager Read transaction

AXI Manager module receives serialized TLPs. Each of these is accompanied with packet metadata vector, that stores some of PCIe information related to given packet. TLP header together with metadata vector are utilized to form AR channel transfer.

AR Channel

- *ARADDR* – TLP address is fetched from the header and PCIe to AXI address translation is performed,
- *ARID*– AXI ID is allocated depending on TLP type and AXI Manager LUTs capacity. See *AXI ID pool* for more information.
- *ARUSER*– TLP header is presented in this signal along with other PCIe information (targeted PF/VF, BAR hit, BAR_APERTURE, PASID, Extended Steering TAG, etc. – details can be found in *<package_path>/docs/cdnhpa_ips_IO_description.html*),
- *ARLEN*– length of the data burst is calculated from TLP length and address offset (depends on selected data path width),
- *ARSIZE*– Memory TLPs resulting in single-beat AXI bursts will have smallest possible *ARSIZE*; for IO TLPs (always resulting in single-beat AXI bursts) *ARSIZE*will always indicate 4 bytes; Memory TLPs resulting in multi-beat AXI bursts will have *maximal*possible *ARSIZE*– equal data path width,

NOTE: *ARSIZE* is calculated once per TLP and when certain conditions occur for Memory Read Requests as described in *Memory Read Request Splits*, and burst must be *split* into multiple AXI transactions, all of them will have same, maximal possible *ARSIZE* – which might be suboptimal for selected single-beat bursts (results of the *split*).

Table 8.15: Example ARLEN and ARSIZE signalization for 512-bit data path width

Case Number	TLP parameters		AXI AW Channel			Comment
-	**Address**	**Length in bytes**	**ARADDR**	**ARLEN**	**ARSIZE**	-
1	0x3	1	0x3	0x00	0x0	Single byte.
2	0x4	4	0x4	0x00	0x2	First byte accessed aligned with natural address boundary.
3	0x10	8	0x10	0x00	0x3	
4	0x2	4	0x2	0x00	0x3	First byte accessed unaligned with natural address boundary
5	0xF	2	0xF	0x00	0x5	
6	0x3F	2	0x3F	0x01	0x6	Multi-beat bursts. Full *ARSIZE*.
7	0x20	96	0x20	0x01	0x6	

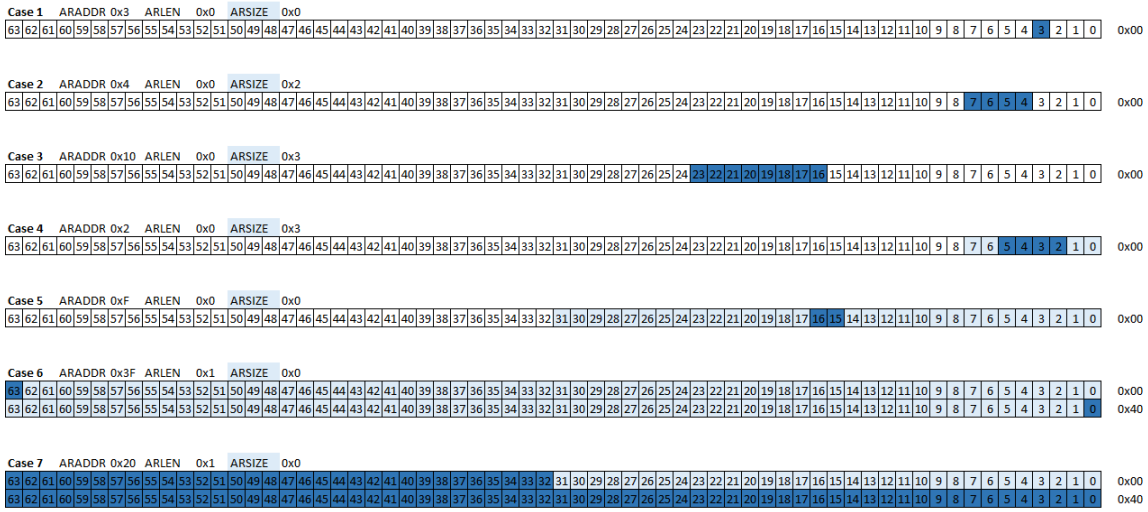


Figure 8.25: Example ARLEN and ARSIZE signalization for 512-bit data path width

- *ARBURST* – always INCR.

8.3.3.3 Memory Read Request Splits

Size of Memory Read Requests might reach up to 4kB and it may be desirable from the system perspective to divide large accesses into smaller chunks. This might occur on following conditions:

- Hitting limit of maximum number of beats within burst. See *Figure 8.26*.
- Maximum Payload Size setting, that limits size of PCIe Completion. For example, 128 bytes, see *Figure 8.26.Figure 8.26*

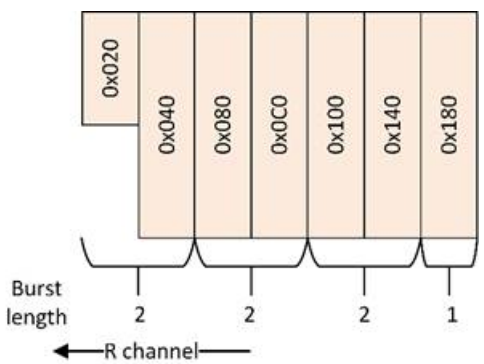


Figure 8.26: Memory Read Request split - burst length limited to 2 beats with selected configuration or by 128-byte MPS

- Crossing 1kB address boundary, if enabled. See *Figure 8.27*.

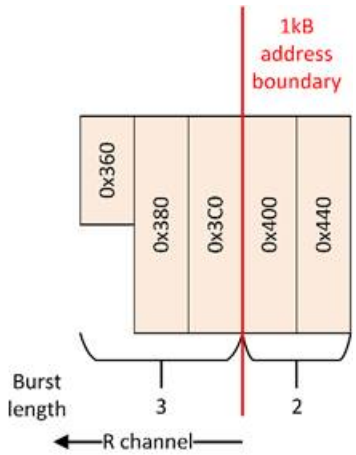


Figure 8.27: Memory Read Request split – split on 1kB address boundary

It may happen that more than one condition is active at the same time – in such situations rule that forces shortest split will be applied. In *Figure 8.28* AXI Read consists of 7 beats. First split was made due to limiting burst length to maximum 2 beats, but for second split also another condition applies – 1kB address boundary rule forces shorter burst length of 1 beat.

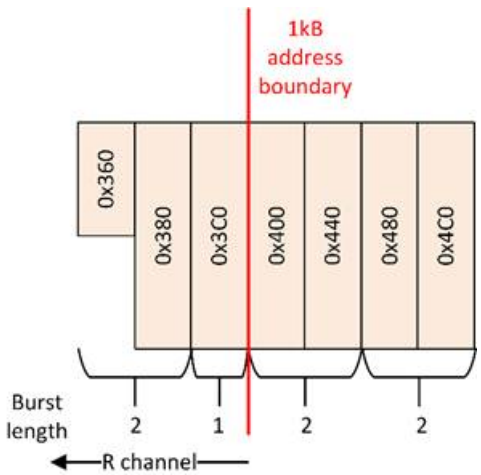


Figure 8.28: Memory Read Request split – two conditions occurring at the same time

When splitting AXI Manager module ensures that Read Completion Boundary rule is satisfied, longest possible burst still might be shortened a bit to enforce RCB compliance.

8.3.3.4 AXI Response Transformation

Every response from R channel is transformed into PCIe Completion for Read Request. AXI Manager module stores Request context, for example, remaining byte count to satisfy the Request.

RRESP code for all R response beats must be OKAY, so Completion with Data and SC Completion Status is generated. When at least one beat is marked with error (SLVERR or DECERR), Completion with CA Completion Status is generated and Request is terminated.

8.3.3.5 Outstanding Requests

Number of outstanding AXI Manager Read transactions (Non-Posted Read Requests) generated by AXI Manager module can be limited by selected configuration (from 4, up to 4096). AXI *READY* and *VALID* signals will be masked, when outstanding transaction counter saturates. Outstanding transaction counter will be decremented, whenever R response is received.

Every split request is counted to outstanding transaction limit.

8.3.3.6 Special Case Zero-length Read

A Memory Read Request of 1 DW with no bytes enabled is called zero-length read. Such Request is transformed into single-byte AXI Manager Read (ARLEN = 0x00, ARSIZE = 0x0). PCIe Completion with Byte Count equal 1 (following the *PCIe Specification*) is generated from R response for this access.

8.3.3.7 Special Case Non-contiguous Byte Enables

The *PCIe Specification* allows Memory Read Requests with non-contiguous byte enables for 1 DW TLPs and 2 DW TLPs with address aligned to 8-byte boundary (QW – QuadWord). Such Requests targeted to both prefetchable and non-prefetchable memory spaces will be *by default* processed the same way – *ARSIZE* of the corresponding AXI Manager Read access will be calculated according to the rules from section and **ALL** bytes from target location are read.

Non-prefetchable Split

Non-prefetchable Read split support can be enabled during IP configuration. When it is supported, Memory Read Requests with non-contiguous byte enables that target non-prefetchable memory space will be split into multiple 1 or 2-byte AXI Manager Reads to ensure that only intended bytes are read.

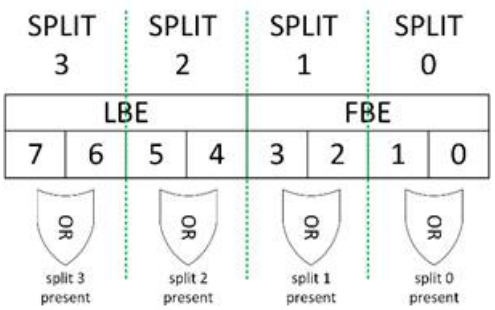


Figure 8.29: NPRF split rule

*Figure 8.29*shows NPRF split rule (maximum) 8 bytes of such Memory Read Request are divided into 4 chunks by 2 bytes. If at least one byte per group is enabled, split AXI Read will be present – *ARSIZE* depends on number of bytes enabled (0x0 or 0x1).

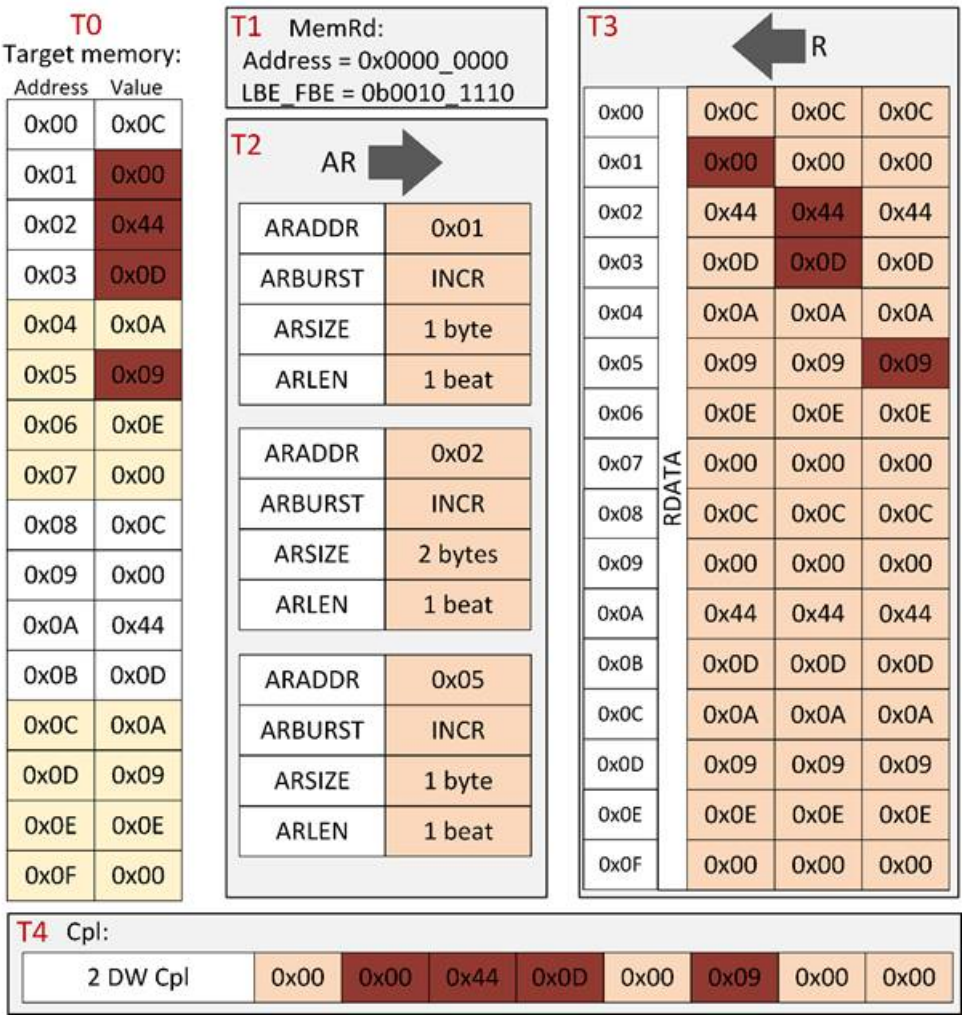


Figure 8.30: Example of NPRF split for 2 DW MemRd TLP with some bytes disabled

Figure 8.30 and *Table 8.16* show examples of non-prefetchable split.

Table 8.16: Examples of NPRF split

No	Address		Split 0		Split 1		Split 2		Split 3	
	LBE	FBE	Address	Size	Address	Size	Address	Size	Address	Size
1	0x4BE4		0x4BE4	1b	0x4BE7	1b	-	-	-	-
	0000	1001								
2	0x7738		-	-	0x773A	2b	0x773C	2b	-	-
	0011	1100								
3	0xABCC		0xABCD	1b	0xABCE	2b	0xABD0	1b	0xABD3	1b
	1001	1110								
4	0x1234		0x1234	1b	-	-	0x1238	2b	0x123A	1b
	0111	0001								
5	0x0000		0x0001	1b	0x0002	1b	-	-	0x0007	1b
	1000	0110								

8.3.3.8 Deinterleave Buffer

If system allows read bursts data to be interleaved (see *Figure 8.31*), they must be sorted by Request (AXI ID) before PCIe Completions are generated. AXI Manager deinterleave buffer is part of the AXI Manager module that provides R response deinterleave capability – all beats of each R response are stored in deinterleave buffer *bins* and burst is only release upon *RLAST* reception.

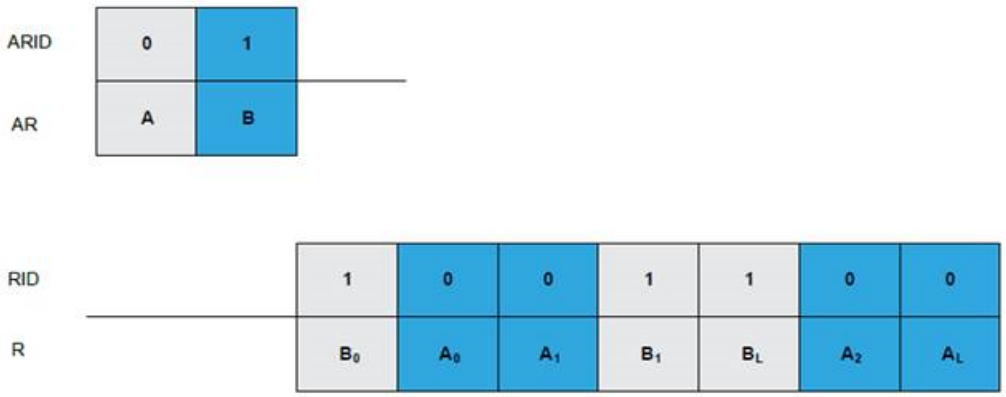


Figure 8.31: Example of interleaved R bursts

When deinterleaving capability is required, number of deinterleave buffer bins should be equal to AXI Manager NP RD LUT depth.

Size of the bin depends on the maximum length of R burst. This results from one of the three conditions:

- AXI Manager Read burst length limit,
- maximum supported Max Payload Size value,
- whether 1kB address boundary split is active.

See *Memory Read Request Splits*.

AXI Manager deinterleave buffer has separate AtomicOp FIFO for Atomic Transaction R responses, if AtomicOp support is enabled.

8.4 AXI Link-Down Handling

A link-down is caused any time after the link is up, that is, when LTSSM is in L0 and later LTSSM transits from L0 State to DETECT state. This is usually a rare event and not expected in normal operation. It is indicated by *link_down_handling_in_progress* port and Link-down Indication bit which is mapped in AXI Subordinate register space.

This event needs to be handled at the Application Layer by performing a corrective action such as flushing all transactions and restarting a DMA.

AXI Subordinate

When link-down event occurs, AXI Subordinate drops all requests and respond adequately until Application Layer resets Link-down Indication bit mapped in AXI Subordinate register space. Link-down handling will end only when all outstanding AXI transactions are completed. This is indicated by *axi_s_link_down_reset_flush_done*.

Note:

- If Application Layer doesn't reset Link-down Indication bit and issues a new request while *axi_s_link_down_reset_flush_done* is high then *axi_s_link_down_reset_flush_done* will be deasserted to indicate that link-down handling is restarted.
- If Application Layer resets Link-down Indication bit before link-down handling is completed, then AXI Subordinate deasserts AW, W, AR ready outputs to stop any new AXI requests.

AXI Manager

When link-down event occurs, AXI Manager waits until all AXI transactions are completed with B or R responses. All PCIe completions that were supposed to be generated, are discarded. Link-down handling will end only when all outstanding AXI transactions are completed. This is indicated by *axi_m_link_down_reset_flush_done*.

DMA

DMA is link-down agnostic, so when the link-down event occurs, it will continue sending scheduled requests. In this case, all the requests issued by DMA will fail on AXI Ext Subordinate error. If DMA is used in the Controller IP, then the application logic should wait untill all DMA channels are complete before resetting link-down indication bit mapped in AXI Subordinate register space.

8.5 Atomic Operations

This section briefly describes concepts of PCIe Atomic Operations and AXI-5 Atomic Transactions. There are several examples for AXI-5 Atomic Transactions to allow reader to better understand functioning of the interface.

8.5.1 PCIe Atomic Operations

Atomic Operation is single PCIe transaction, that reads target location's value, potentially writes a new value back and returns original value. *Read-modify-write* sequence is performed atomically.

Table 8.17: Types of PCIe Atomic Operations

Operation	Description	Operands	Possible operand size
Fetch and Add (FetchAdd)	Request contains single operand - the *add* value. Target location is read, *add* value is added and modified value is written back to target location. Original value of the target location is returned.	1	32 bits 64 bits
Unconditional Swap (Swap)	Request contains single operand - the *swap* value. Target location is read, *swap* value is written back and original value is returned.	1	32 bits 64 bits
Compare and Swap (CAS)	Request contains two operands - the *compare* value and *swap* value. Target location is read and its value is compared with *compare* value. If they are equal, *swap* value is written back to target location. If they are not equal, nothing happens. Original value of the target location is returned.	2	32 bits 64 bits 128 bits

CAS requests contain two operands:

- One in data area is always compare value.
- One is swap value.

8.5.2 AXI-5 Atomic Transactions

Atomic Transaction is issued using AW AXI channel with additional *AWATOP* signal to distinguish between non-atomic and atomic transactions (data is transmitted on W). It must use AXI ID reserved for AtomicOps, common for W and R channels. Both B (write response) and R (read response with original data from target location) responses are expected for Atomic Transaction (*Figure 8.32*).

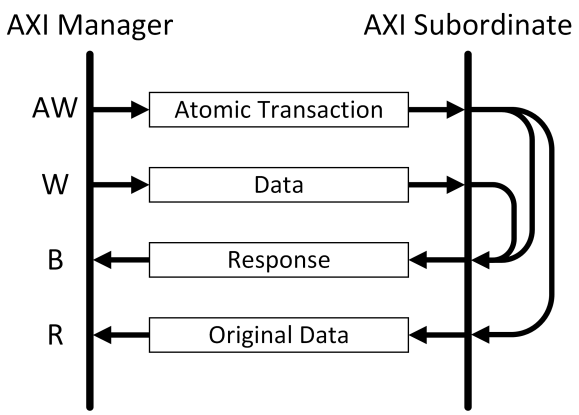


Figure 8.32: Atomic Transaction structure

AXI-5 specification defines many Atomic transaction types with various data operations possible, but PCIe supports only 3 Atomic Operations (*Table 8.18*).

Table 8.18: PCIe AtomicOp's mapped to AXI-5 Atomic Transactions

PCIe AtomicOp type		AXI-5 Atomic transaction type	AWATOP[5:0] (1), section E2.1.6
Non-atomic operation	Normal request	Normal request	0b000000
Atomic operation	FetchAdd	AtomicLoad (ADD operation)	0b10e000 (bit *e* to indicate endiannes of data)
	Swap	AtomicSwap	0b110000
	CAS	AtomicCompare	0b110001

Above table presents mapping between PCIe Atomic Operations and AXI-5 Atomic transactions and contains valid AWATOP values. All other values (other AXI-5 Atomic transaction types) are NOT supported.

Note: Value of bit e for AXI AtomicLoads generated by AXI Requester module can be set with *axi_m_atomicop_endianness* configuration variable.

8.5.2.1 Examples

Following list contains examples of AXI-5 Atomic Transactions. You must remember that:

- Relative timing between AW (request) and W (request data) is not specified,
- Order of W and R transactions is not specified as:
  - Subordinate is permitted to wait for all write data before sending read data,
  - Subordinate is permitted to send all read data before accepting any write data,
- As a result, order of B and R responses is also not specified (B must be given only after it has received all W data and result of Atomic Transaction is observable).

Following examples are only intended to clarify concept of AXI-5 Atomic Transactions and are only subset of possible situations.

AXI-5 AtomicLoad with ADD operation

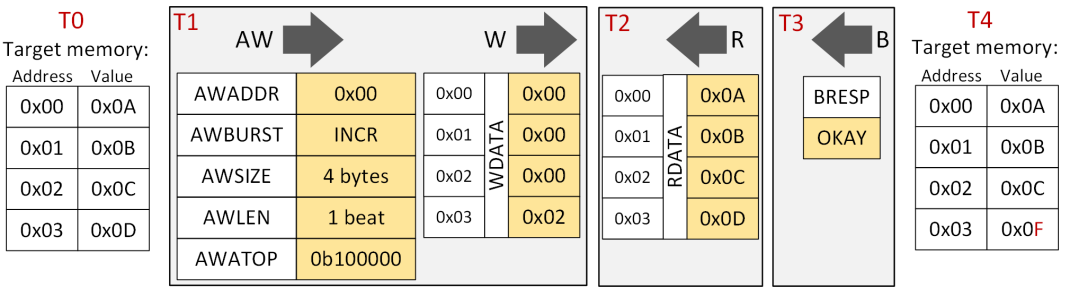


Figure 8.33: AXI-5 Atomic Load with ADD operation

At T0 memory location at address 0x00 has value 0x0D0C0B0A. Atomic Transaction is issued on AW channel at time T1 (Atomic Load with ADD operation - AWATOP = 6'b100000). Atomic Transaction operand is 0x02000000 (W channel). At time T2 original data is returned on R channel - this value is read from target memory location. At T3 write response is received on B channel - result of Atomic Transaction is observable. New value of target memory location is 0x0F0C0B0A - operand was added to original value.

AXI-5 AtomicSwap

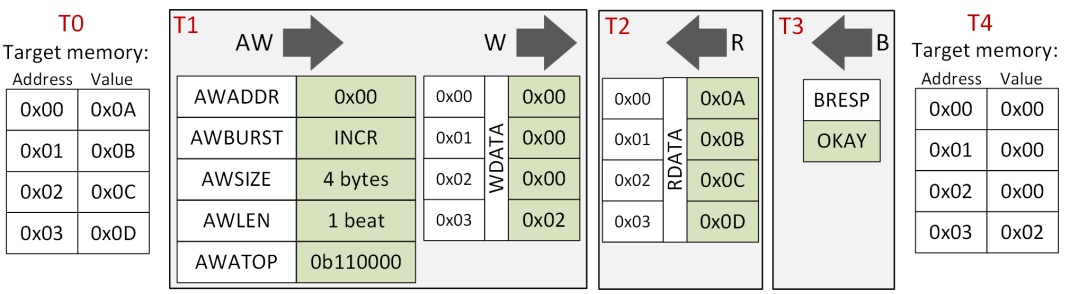


Figure 8.34: AXI-5 Atomic Swap operation

Atomic Swap (AWATOP = 6'b110000) is similar transaction to Atomic Load, but instead performing any operation on target memory location, it writes its operand to it. Original data is returned on R channel and at the end (T4) target memory location value is equal to Atomic Swap operand (W channel).

AXI-5 AtomicCompare with aligned address

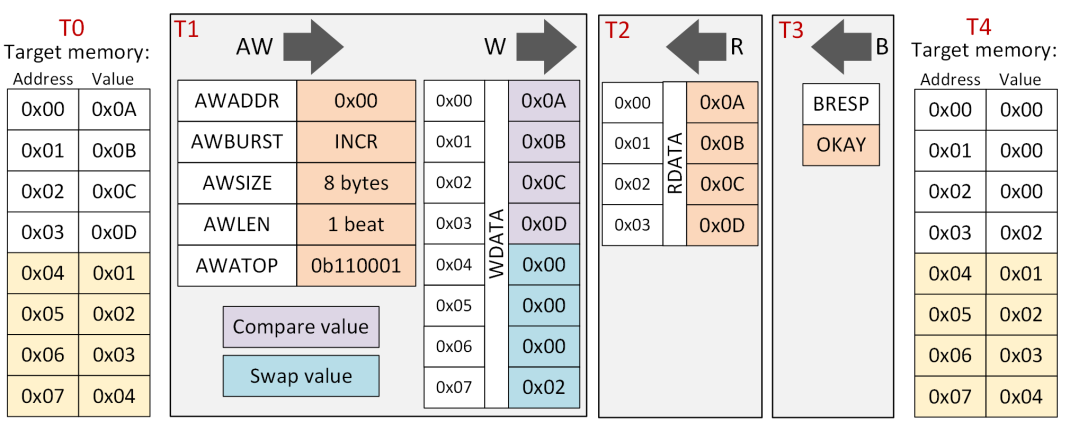


Figure 8.35: AXI-5 Atomic Compare operation with address aligned to total burst size

At T1 AtomicCompare transaction is issued on AW channel (AWATOP = 6'b110001). Operand size is half of total transaction size. Compare value is sent first, because transaction address is aligned to total size of burst. At T2 original

data is returned on R channel – inbound data is only half of the outbound data size (earlier two operands must have been sent).

If target memory location is equal to compare value, it is overwritten with swap value (as shown in the above figure). If these two values are different, nothing happens.

AXI-5 AtomicCompare with unaligned address, AWLEN = 0

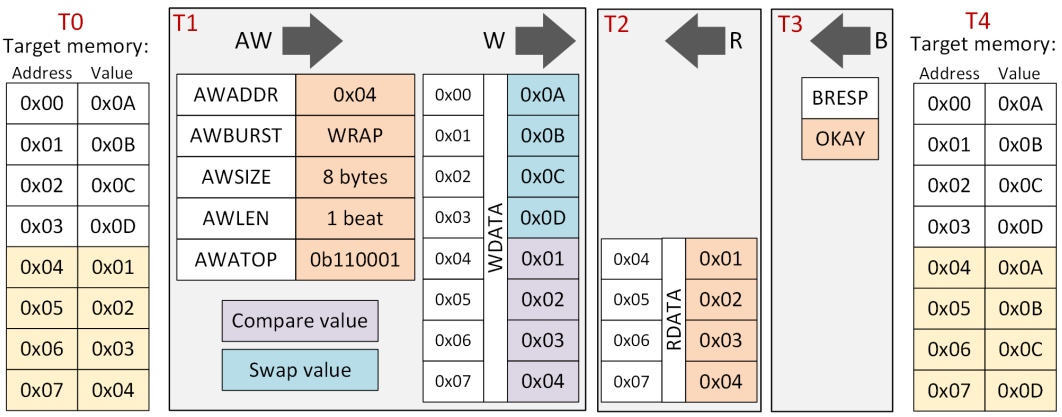


Figure 8.36: Atomic Compare operation with address unaligned to total burst size

In this case swap value is sent first, because transaction address is UNALIGNED to total burst size. (0x04 vs. 8 bytes). This is marked in AW request by AWBURST signal equal WRAP.

AXI-5 AtomicCompare with unaligned address, AWLEN > 0

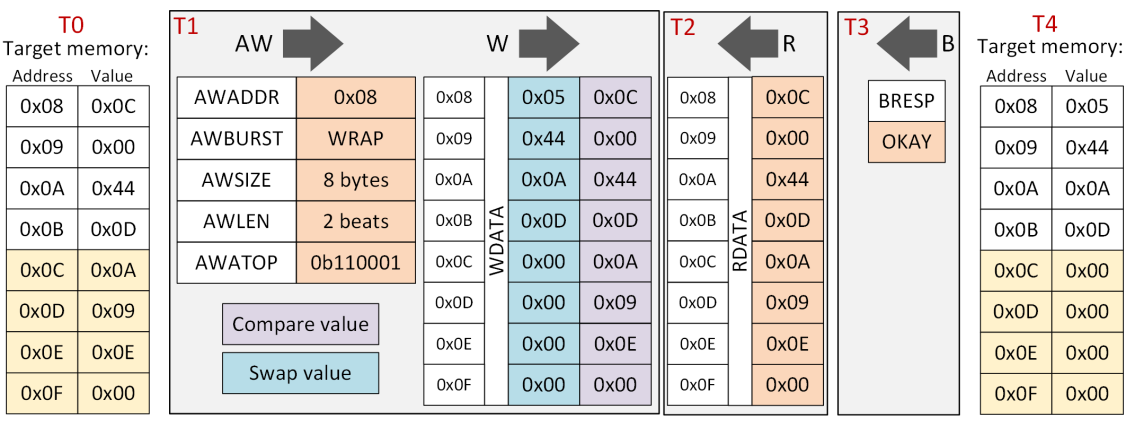


Figure 8.37: Atomic Compare operation with address unaligned to total burst size (AWLEN > 0)

In this case total size of burst is 16 bytes (2 beats * 8 bytes). Operand size is 8 bytes. Transaction address is unaligned to total burst size, so swap value is sent before compare value. Compare value is equal to target memory location, so swap value is written into the memory.

8.5.3 AtomicOp on AXI Subordinate

There are two possible ways to issue AtomicOp using AXI Subordinate:

- AtomicOp access through AXI Regions

  This process is described in section 'PCIe AXI Subordinate Access through Regions'. In this method PCIe-related information is stored in AXI region registers. Region should be programmed as MEMORY region. User must additionally set *axi_s_awatop* signal to appropriate value.
- AtomicOp access using Sideband Descriptor

  This process is described in section *AXI Subordinate Access through Sideband Descriptor*. In this method PCIe-related information is given using AWUSER signal. User must additionally set *axi_s_awatop* signal to appropriate value. See *Table 8.18*.

User may accidentally miss all of the programmed regions with AtomicOp access. If no match is found by the region select logic, appropriate error responses are given back on B and R channels.

Note:

- *axi_s_awsize* and *axi_s_awlen* must match correct size of AtomicOp TLP payload.
- As First BE and Last BE fields are reserved in AtomicOp TLPs, write strobes of W channel does not have specific meaning for AtomicOp accesses. It is required that write strobes are set for valid bytes of AtomicOp access and are contiguous.

To generate correct Compare and Swap AtomicOp request, user must follow all rules of AXI-5 specification regarding AXI AtomicCompare Atomic Transaction. For accesses with address aligned to total size of the burst *axi_s_awburst* must be set to INCR, 'compare' operand must be placed first on *axi_s_wdata* and 'swap' operand must be second. For accesses with address unaligned to total size of the burst *axi_s_awburst* must be set to WRAP, 'swap' operand must be placed first on *axi_s_wdata* and 'compare' operand must be second.

8.5.4 AtomicOp on AXI Manager

8.5.4.1 AtomicOp Request

Atomic Operation TLPs are processed in similar manner as other Non-Posted Write Requests by AXI Manager module (that is header and PCIe context is presented on AWUSER bus or TLPs address undergoes PCIe to AXI address translation. See *AXI Manager* for additonal iformation, but are presented on the interface AXI-5 compliant way, described with examples in *AXI-5 Atomic Transactions*, where AXI Manager module acts as bus Manager.

Number of AXI IDs (common for all channels) is reserved only for Atomic Transactions and is dependent on AXI Manager AtomicOp LUT depth (`axi_m_atomicop_lut_depth` configuration variable) – AXI Manager module might issue up to so many Atomic Transactions.

Endianness of AXI 5 AtomicLoad with ADD operation (result of PCIe Fetch and Add Request) issued by AXI Manager module might be set with `axi_m_atomicop_endianness` configuration variable.

In case of PCIe Compare and Swap Requests with address unaligned to total Request size, AXI Manager module is responsible for changing order of 'compare' and 'swap' operands to be compliant with AXI-5 Atomic Transactions specification.

8.5.4.2 AtomicOp Completion

Since Atomic Operations are Non-Posted, AXI Manager module must generate Completion for every Request.

AXI Manager module sends AtomicOp Completion only when both, B and R AXI responses are received. xRESP code for both responses must be OKAY, so Completion Status is SC. If any xRESP code is either SLVERR or DECERR, Completion with CA Completion Status will be generated.

8.6 Credit Availability Interface

8.6.1 Overview

This section describes the Credit Availability Interface at Client logic. It consists of the following posted and non-posted port groups. For more information on IOs, refer to Signal List and Description available in the IP package.

- Credit limit for /non/posted header (scaled) – 12 bits per VC
- Credit limit for /non/posted payload (scaled) – 16 bits per VC
- Credit available for /non/posted header – 1 bit per VC
- Credit available for /non/posted payload – 1 bit per VC
- Credit infinite for /non/posted header and payload – 2 bits per VC:
    - [0] - header credits are infinite
    - [1] - payload credits are infinite

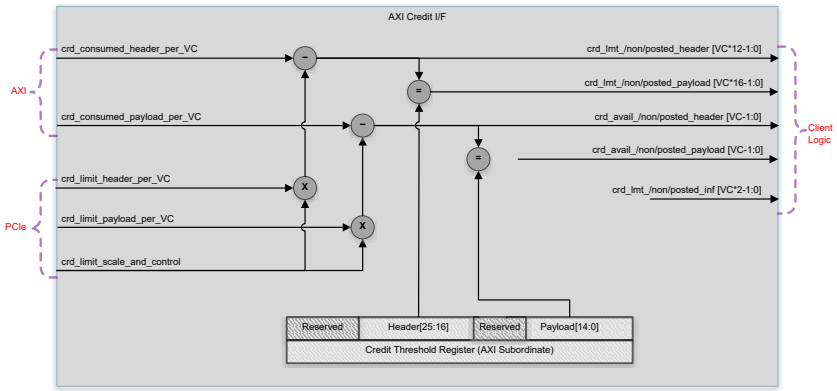


Figure 8.38: Credit availability

All ports include credit values per VC which are packed a in vector that VC0 is located at LSB and VCmax at MSB, as shown in the example below:

```
crd_lmt_posted_header[VC*12-1:0] =
{crd_lmt_posted_header_VCmax[11:0], … , crd_lmt_posted_header_VC1[11:0], crd_lmt_posted_header_VC0[11:0]}
```

The credit limit header and payload values at client logic are equal to the PCIe credit limit reduced by credits consumed by AXI logic. Occasionally, the credit limit value can turn to a negative number coded as 2's complement when AXI logic consumes more credits than the current PCIe credit limit. In this case, the packets transmission will be stopped until new credits are received from PCIe remote device.

8.6.2 Credits Scaling

All credit values at Client logic are scaled based on credit scaling rules defined in the PCIe specification. 12-bit header credit limit ports width and 16-bit payload credit limit ports width can hold values with scale multiplier 16 which corresponds to scale factor 4.

8.6.3 Credit Threshold Register

The Credit Threshold Register mapped in AXI Subordinate registers space holds credit value which is compared with the current credit limit and used to assert the adequate credit availability output flags for corresponding VCs. The threshold is given in 4DW units.

8.6.4 Link Down and Low Power Mode

During the PCIe link down, all credits are set to infinitive and all available outputs are active. On PCIe link down exit event, the credit values are reset, and new credits must be granted from PCIe remote device. The credit I/F outputs are updated during the PCIe low power mode if the credit limit changes, or credits are consumed.

8.6.5 DMA Credit-Based Flow

If the DMA is available, then it transmits the requests through AXI Subordinate interface which is part of the Controller IP. The DMA requests are gated based on credit availability, to prevent blocking AXI Subordinate request pipeline when the credits are not present.

Chapter 9: Miscellaneous Interfaces

9.1 Interrupt Interface

When the IP is selected as EP mode, the application logic can use this interface for generating interrupts that consists of INTx, MSI and MSI-X.

9.1.1 INTx or Legacy Interrupt Interface

Figure 9.1 shows Interrupt Assert and De-assert Messages.

Note:

The below timing diagrams are for reference and example.

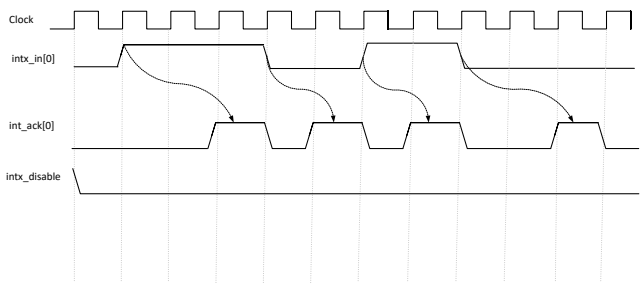


Figure 9.1: INTx interface

The timing diagram above is shown for INTA, which uses intx_in[0] and int_ack[0] . If supported, INTB, C and D are represented by intx_in and int_ack bits 1, 2, 3 respectively.

9.1.2 MSI-X Interrupts

MSI-X interface is used by the client to send MSIX interrupt as shown in the *Figure 9.2* timing diagram.

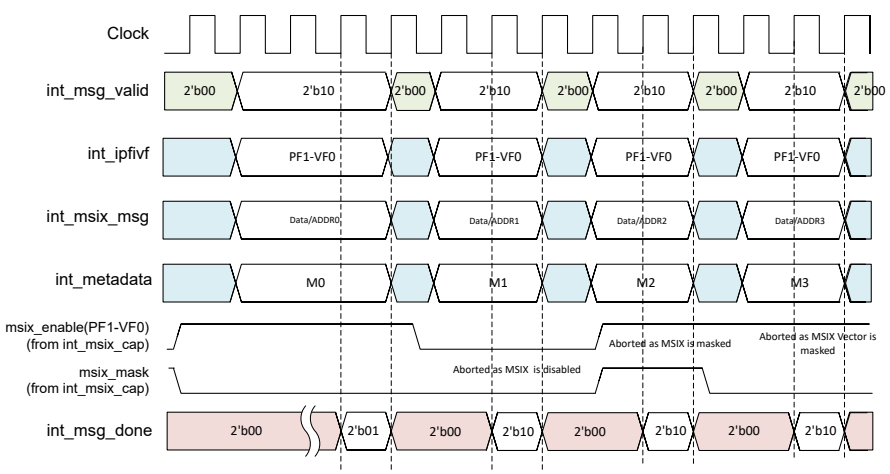


Figure 9.2: MSI-X interface

9.1.3 MSI Interface

MSI interface is used by the client to send MSI interrupt as shown in the *Figure 9.3* timing diagram.

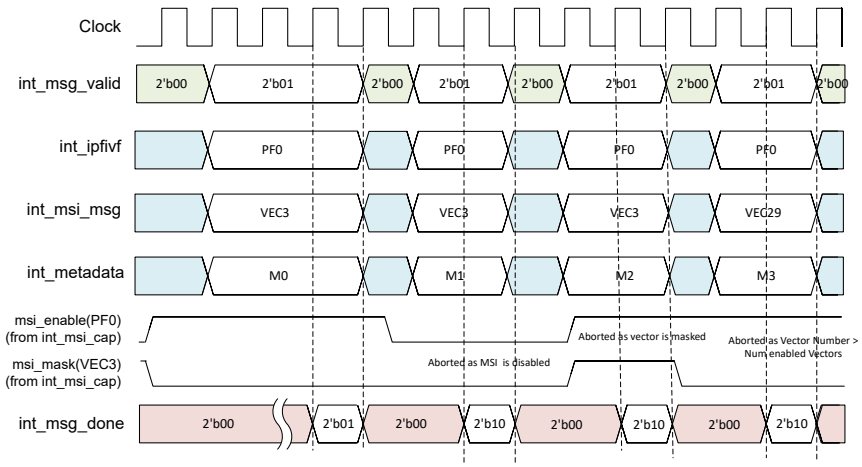


Figure 9.3: MSI interface

9.2 FLR Handling

9.2.1 HLS FLR Handling

The host can use CFG-WR to put a function or VF to undergo FLR. All transactions associated with that functions needs to get flushed. The controller sends dummy completions to all the pending outbound requests from a function undergoing FLR. The host can send multiple FLR to different functions. The controller can also handle multiple FLR for different functions in parallel.

The following are the expectations from application logic upon receiving FLR to a particular PF/VF. If FLR is received on the PF which has associated VFs, the following actions are required for all the associated VFs:

1. Abort all the ongoing transactions for that PF/VF.
2. Stop sending any outbound transactions. Any pending requests or completions in the OB direction for the function under FLR needs to be discarded.
3. Wait for *good-completion* or *dummy-completion-with-flr-status* for all the pending requests in the outbound direction. This is to let all the *np-request-transactions* in the pipeline for this PF/VF to move forward before clearing the FLR status.
4. Assert client_flr_done for the PF/VF.

After *client_flr_done* is received for a PF/VF, the controller asserts *controller_flr_done* after clearing the *flr_in_progress* status internally. Now, the application logic can resume operations on this PF/VF.

Any requests that are received for a function are still under rest, that is, the client logic is yet to acknowledge in the form of client-side FLR done status:

- All MEM/IO requests get silently discarded.
- Completions targeting a function under FLR get silently discarded.
- All configuration requests targeting a function under FLR get silently discarded. In this scenario, there is a LM register setting available to respond with CRS status.

If the host sends FLR to a PF which has active VFs, the controller sends completions with FLR status to all the attached VF's which have NP requests that are pending.

Note: In this case, FLR signal handshake will be performed only for the PF.

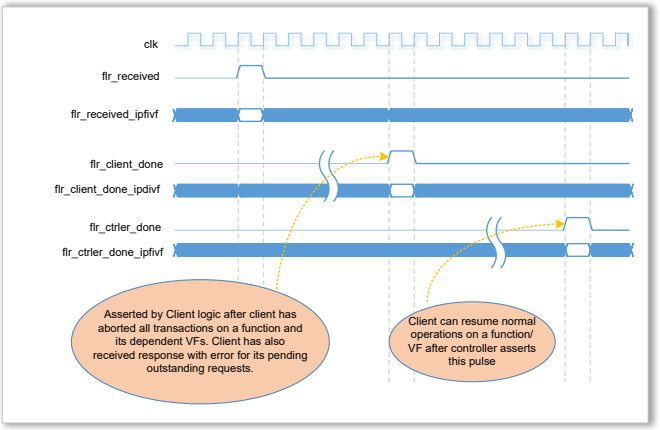


Figure 9.4: HLS FLR handling

Chapter 10: Power Management

10.1 Overview

This chapter describes different power management techniques implemented in Cadence PCIe HPA controller. For detailed description on how to use optional features, for the Specification mandated features and its usage in a PCI system refer to the *PCI-SIG Specification*. User side signals and registers related to power management features are explained in this section.

10.2 Function Power States

The current version of the Cadence PCIe HPA Controller supports the following PCI Function power states:

- D0 (uninitialized and active).
- D1 - has a kwire to enable [SS1] (optional).
- D3-Hot.

By default, the Controller sets the *No_Soft_Reset* bit of the **Power Management Control Register** of all enabled functions to 1. This implies that the state of a Function is not lost when it is in the D3-HOT power state, and its registers need not be re-configured when the Function is set back to D0. The *PCI-SIG Specification* recommends setting this bit for all functions.

10.2.1 Function Power State Change Handshake Interface

This interface is used by the EndPoint-IP to change its Device Power State when a Config Write is received to the Power Management Control and Status Register. The timing diagram shows the handshake required for a successful power state change.

Changing the Endpoint function power state from RP:

1. RP initiates CFGWr to changes device power state in Endpoint
2. EndPoint IP asserts *power_State_change_interrupt* upon receiving this CFgWr
3. Controller does not respond to any other CFG request while *power_State_change_interrupt* is asserted
4. *ACK* from client logic changes power state of that function
5. Client logic must give *ACK* only after halting all traffic from that function if the host is programming to move to a lower power state

Following LM register control available to control which power state change creates interrupt:

- **power_state_change_interrupt_ctrl[0]** : Creates interrupt when power state is changed from D1 or D3 to D0
- **power_state_change_interrupt_ctrl[1]** : Creates interrupt when power state is changed from D0 or D3 to D1
- **power_state_change_interrupt_ctrl[2]** : Creates interrupt when power state is changed from D0 or D1 to D3.

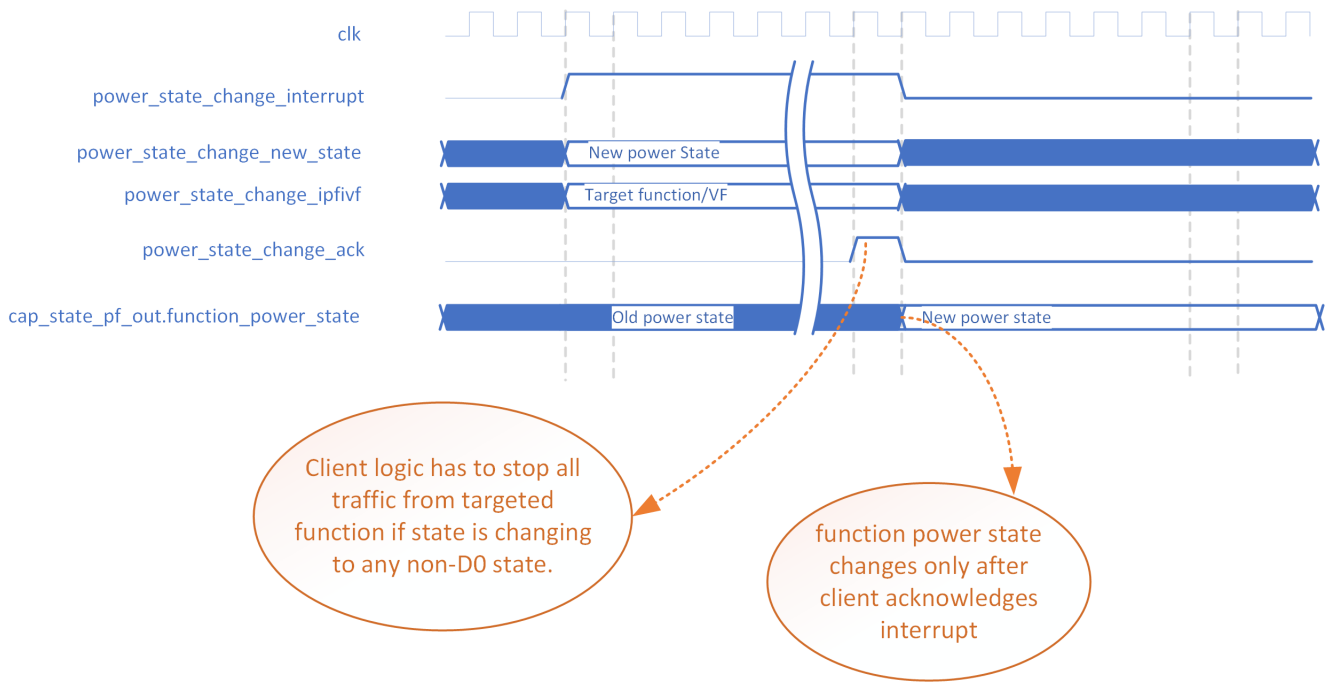


Figure 10.1: Function power state change handshake

10.3 PCIe Link Power States

10.3.1 L0s Link Power State

L0s entry and exit is an autonomous process. L0s state has very low exit latency among the link power states. ASPM L0s can be enabled or disabled in the *i_PCIE_cap_struct/i_link_ctrl_status* configuration register bit **ASPMC**. The Controller automatically initiates entry into ASPM L0s if the TX side is idle, that is, no TLPs and no DLLPs to be transmitted, for a programmable time period (*i_regf_ip_pl/i_pm_cfg_0*).

10.3.1.1 Register Programming details for L0s

L0s entry timeout is programmable through the IP Register *i_regf_ip_pl/i_pm_cfg_0*. The transition from L0 to L0s happens after a time period programmed in the *i_regf_ip_pl/i_pm_cfg_0* field **ASL0ST** elapses with no TLP or DLLP being transmitted. Setting the *i_regf_ip_pl/i_pm_cfg_0* register field **ASL0ST** to 0 disables the transition to L0s state.

10.3.2 L1 Link Power State

L1 link power state gives more power saving at the expense of more latency than L0s state.

10.3.2.1 L1 Entry through ASPM Mechanism

ASPM L1 is an autonomous mechanism. Entry and exit happens without any user handshake. ASPM L1 can be enabled or disabled in the *i_PCIE_cap_struct/i_link_ctrl_status* configuration register bit **ASPMC**. The Controller automatically initiates entry into ASPM L1 if the TX side is idle, that is, no TLPs and no DLLPs to transmit, for a programmable time period in the *i_regf_ip_pl/i_pm_cfg_1* register field **ASL1T**.

The following sequence shows the operation of ASPM L1 Entry in a PCIe system.

1. When the Link TX is idle (that is, no TLPs and no DLLPs to transmit), the EP Controller starts incrementing the ASPM L1 entry timer internally. If the Client requests to transmit a TLP, the timer is immediately cleared.
2. When the ASPM L1 entry timer reaches the programmed value of the *i_regf_ip_pl/i_pm_cfg_1* register field **ASL1T**, the EP Controller checks if sufficient credits are accumulated.
3. The EP Controller blocks new TLPs and initiates ASPM L1 entry by transmitting *PM_Active_State_Request_L1* DLLPs onto its Transmit Lanes.

4 The EP Controller continuously transmits *PM_Active_State_Request_L1 DLLP* until it receives a response from the Upstream device.
5 The Upstream component must immediately respond to the request with either an acceptance or a rejection of the request.
6 If the Upstream component rejects by sending a *PM_Active_State_Nak* message, the Controller aborts the ASPM L1 entry and continues to send TLPs normally.
7 If the Upstream component accepts by sending *PM_Request_Ack* DLLPs, then the Controller (EP) puts its TX into electrical idle and enters ASPM L1.
8 The Upstream component also detects electrical idle and puts its TX also into electrical idle.

Register Integration Details

ASPM L1 entry timeout

ASPM L1 entry timeout is programmable through the ip register *i_regf_ip_pl/i_pm_cfg_1* field **ASL1T**. The field 'ASL1T' contains the timeout value(in units of 16 ns) for transitioning to the ASPM L1 power state. Setting it to 0 disables the transition to the ASPM L1 power state.

10.3.2.2 L1 Entry through PCI-PM Software Compatible Mechanism

The following sequence shows the operation of PCI power management when the Controller is configured as an Endpoint.

1 Assume the Controller is operating normally, with all Functions in D0 active state.
2 The remote Root Port writes into the Power Management Control Register of all Functions to transition the Function to the Non-D0 power state. Root Port should start with VFs if SRIOV is enabled and supported and then program the PFs.
3 For changing the power state of any function, we need an acknowledgement from the Application logic described in the section *Function Power State Change Handshake Interface*.
4 When all enabled Functions in the Controller are in non-D0 state, the Controller initiates a transition of the link power state to L1 by transmitting *PM_Enter_L1* DLLPs.
5 After the Data Link Layer handshake, the link transitions to L1 state.

Register Integration Details

PM-L1 re-entry timeout

While the link is in L1 state and all the Controller's functions are in D1 or D3-HOT state, the link partner may transition the link any time from L1 to L0. The Controller may optionally initiate a re-entry back to L1 if the link has been idle for a set interval and the Controller's functions still remain in D3-HOT state . The re-entry to L1 is controlled by the delay programmed in the *i_regf_ip_pl/i_pm_cfg_2* in ip register space. Setting this register to a non-zero value causes the Controller to initiate entry back to L1 when a delay equal to the number of clock cycles programmed in this register has elapsed with no link activity. Setting this register to 0 prevents re-entry to L1. The initial transition to L1, step *When all enabled Functions in the Controller are in non-D0 state, the Controller initiates a transition of the link power state to L1 by transmitting PM_Enter_L1 DLLPs.* is not affected by the setting of this register.

10.3.2.3 L1 Exit Trigger Events

The following events trigger an L1 exit process.

1 Electrical Idle exit detection.
2 New requests at AXI or HLS interface.
3 When a new TLP is scheduled for transmission internally, for exmaple, *ERR_COR* Message.
4 Assertion of side band signals *low_pwr_client_req_exit_l1*.
5 Retrain request using *link_control* register for Root Ports.
6 Assertion of *hls_ob_activereq** signal for any of the posted/ Nonposted/ completion interfaces.

10.3.2.4 L1 Exit Process for ASPM Entry

Any L1 exit trigger events can cause an L1 exit process and there is no handshake required from the client side.

10.3.2.5 L1 Exit Process for PCI-PM Software Compatible Entry

Any L1 exit trigger events will bring link power state to L0. However, an additional Root Port to Endpoint handshake is required for normal operation after reaching L0.

The steps involved in Root Port initiated PM_L1 exit are:

1. Root Port initiates a CFG write to change Endpoint function's power state to D0.
2. This config write triggers an L1 exit on the link.
3. Endpoint DUT also exits from L1 and responds with Cpl.
4. If the transition was from D3Hot depending on No_Soft_Reset controller can start normal data transfer or needs re-initialization.

The steps involved in Endpoint initiated PM-L1 exit are:

1. When the client sets *PME_STATUS* bit in the **PMSCR**, the Endpoint initiates *PM_PME* message to Root Port to request a power state change.
2. This message triggers an L1 exit on the link.
3. After receiving this message, Root Port initiates a CFG write to change Endpoint function's power state to D0.
4. Endpoint DUT respond with Cpl.
5. Normal data transfer can happen on PCIe link.

10.3.3 Debug Registers for L1

10.3.3.1 L1 Entry Mode Information

The register *i_regf_ip_pl/i_pm_dbg_sts_0* field **L1EM** in ip register space gives information on last L1 entry mode. You will get to know whether PM or ASPM mode was used to enter L1. This is reset before each L1 entry operation.

10.3.3.2 L1 or L1-Substate Exit Trigger Event Information

The register *i_regf_ip_pl/i_pm_dbg_sts_0* field **L1EXCON** in ip register space gives information on last L1 or L1-substate exit triggers. This field gets reset before each L1 entry operation.

10.3.3.3 Explicit L1 Exit from Client Side or Blocking L1 entry for EP Controller

Client logic can trigger an explicit L1(PCI-PM/ASPM) exit by asserting *low_pwr_client_req_exit_l1*. This signal triggers an exit to L0 from L1 or from L1-substates. A new request at HLS or AXI interface also triggers a L1 exit event internally.

This signal can also be used to block L1 entry in Endpoint controllers. Blocking L1 entry is useful when client wants to disable L1 entry for reducing latency for its outstanding operations.

The following figures show the timing diagrams that explain the use of this signals for two purposes.

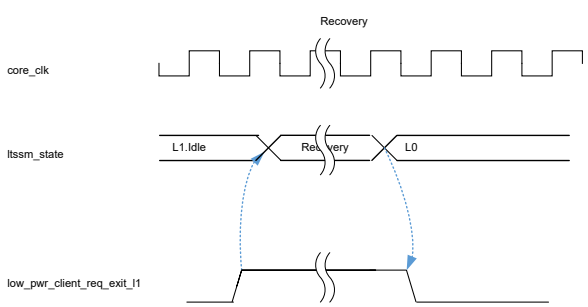


Figure 10.2: Using low_pwr_client_req_exit_l1 to exit from L1

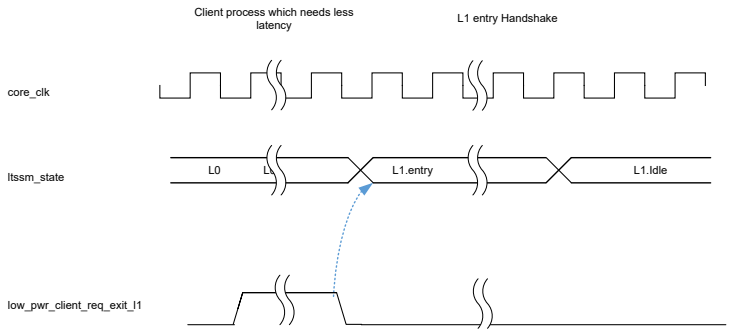


Figure 10.3: Using low_pwr_client_req_exit_l1 for blocking L1 entry

10.3.4 L1 Substate

The L1 Power Management Substates ECN defines an optional mechanism to reduce the idle power in the L1 link state. This is achieved by defining new substates within the L1 state to facilitate the removal of power to the phy, and clocks to the controller. The L1 PM substates are enabled when the link enters L1, due to either PCI Power Management or ASPM.

The ECN provides two options when the link is in the L1 state: The L1.1 substate allows clocks and most of the phy power to be turned off, but requires the phy to maintain common-mode voltages on the transmit side. The L1.2 substate enables further reduction in idle power by not requiring common-mode voltages to be maintained. Both L1.1 and L1.2 states allow the Electrical Idle detection circuitry in the phy to be turned off.

The L1 PM Substates use the *CLKREQ#* sideband signal to control the clocks. The *CLKREQ#* signal is an open-drain active-low signal shared by the Upstream and Downstream Ports, and can be asserted by either side driving it low. This enables the clock generator. The core clock is turned off when both sides de-assert their *CLKREQ#* outputs.

The Cadence PCIe HPA Controller provides a *low_pwr_clkreq_in_n* input and a *low_pwr_clkreq_out_n* output to implement the tri-state *CLKREQ#* pin. The *low_pwr_clkreq_out_n* output, when low, enables the tri-state driver driving the *CLKREQ#* pin, causing assertion of the shared signal. The *CLKREQ#* signal can also be asserted by the Port on the other side driving it low. The state of this shared signal is monitored by the *Cadence PCIe Controller* through the *low_pwr_clkreq_in_n* input, as shown in the following figure.

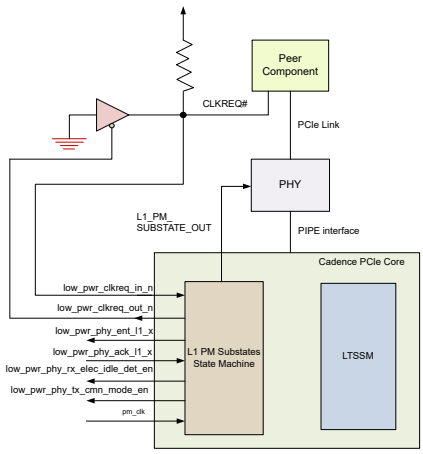


Figure 10.4: Block Diagram of the L1 PM Substates Implementation

As the core clock is turned off in the L1.1 and L1.2 substates, a separate power management clock (*pm_clk*) is used to drive the L1 PM Substates state machine. This clock must always be active, regardless of the power state of the link. There is no requirement on the relative phase of this clock with respect to the other clocks used by the Controller.

The L1 PM Substates state machine also provides handshake signals *low_pwr_phy_ent_l1_x* and *low_pwr_phy_ack_l1_x* to prepare the local phy module for the removal of the reference clock. The state machine asserts the *low_pwr_phy_ent_l1_x* output in the L1.0 substate when it has determined that the conditions for transition to the L1.1 or L1.2 substates are met. It then waits for the phy to assert *low_pwr_phy_ack_l1_x* before de-asserting *low_pwr_clkreq_out_n* and entering L1.1 or L1.2 substates. During L1.x exit, the controller de-asserts

low_pwr_phy_ent_l1_x to the PHY and waits for corresponding de-assertion of *low_pwr_phy_ack_l1_x* before transitioning to the L1.0 state. This is required to ensure that the PHY is fully operational and clocks are stable before entering L1.0. For the case of L1.2, the PHY handshake is performed while in the L1.2 Exit substate.

Note:

De-assertion of *low_pwr_phy_ent_l1_x* changes the PHY state from L1-substate to L1.0, ensure that only a stable *refclk* is given to PHY during the exit process.

The L1 PM Substates state machine provides an output signal *low_pwr_phy_rx_elec_idle_det_en* to inform the local PHY when to enable its electrical idle detection circuits on the receive side. The Controller asserts this output in all states except when the L1 PM Substates state machine is in the L1.1, L1.2.Entry and L1.2.Idle substates.

The L1 PM Substates state machine also provides an output signal *low_pwr_phy_tx_cmn_mode_en* to enable common mode on the transmit side of the PHY. The Controller de-asserts this output when the L1 PM Substates state machine is in the L2.Idle substate, and asserts it at all other times.

10.3.4.1 L1-Substate Entry Conditions

L1-substate entry process is initiated when the link is in L1 and *CLKREQ#* from upstream and downstream components are de-asserted. The Controller enters L1.1 or L1.2 depending on what is enabled in L1 PM Substate Control registers. Controller also internally checks for the pipelines to be empty before entering L1.x substates.

The *low_pwr_clkreq_out_n* is de-asserted in both L1-substates. When the remote device also de-assert CLKREQ#, CORE clock is turned off by the clock controller in user domain.

10.3.4.2 L1-Substate Exit Conditions

Either side may initiate a transition out of the L1-substate. The remote side initiates L1 PM Substate exit by asserting its *CLKREQ#* output, which turns on the core clock. This also asserts the *low_pwr_clkreq_in_n* input to the Controller, causing its L1 PM substate to change to L1.0, and enabling the transition of the LTSSM from L1 into Recovery. Client can also initiate the exit process for L1.

The following events trigger an L1 substate exit process and there are two categories of triggers:

- Remote device initiated exit triggers:
  - CLKREQ# assertion
  - Electrical Idle exit detection(Only during entry into L1.x substates, before *low_pwr_phy_rx_elec_idle_det_en* is deasserted)
- Locally initiated exit triggers:
  - New requests at AXI or HLS interface.
  - Assertion of side band signals *low_pwr_client_req_exit_l1*
  - Assertion of *hls_ob_activereq** signal for any of the posted/ Nonposted/ completion interfaces.

L1-substate exit triggers moves link from L1.x state to L1.0 and then to L0.

The figure below shows the power domain association of different modules in PCIe controller. This is diagram is indicative only.

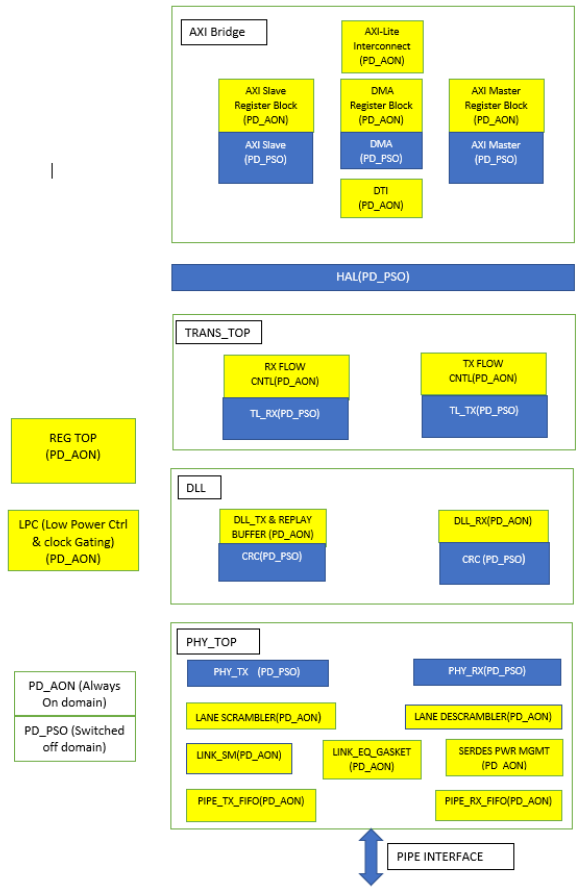


Figure 10.5: Power domain association of different modules in the PCIe controller

10.3.5 L1.1 Operation

L1.1 is entered when L1.1 entry conditions are true and L2.2 entry conditions are false.

10.3.5.1 L1.1 Entry and Locally Initiated Exit

Figure 10.6 shows L1.1 entry and locally initiated exit process. Signal *low_pwr_client_req_exit_l1* represents all local exit triggers in this diagram.

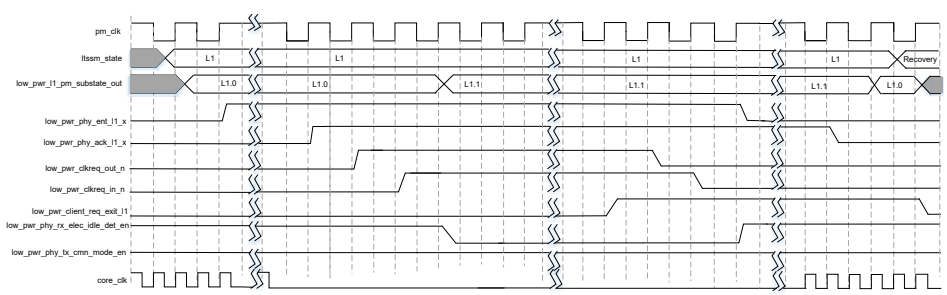


Figure 10.6: L1.1 entry and locally initiated exit

On finding that the conditions for entering the L1.1 Substate are met, the L1 PM Substates State Machine first performs the handshake with the phy using the *low_pwr_phy_ent_l1_x* and *low_pwr_phy_ack_l1_x* signals to prepare the phy for the removal of the reference clock. Once the phy has asserted *low_pwr_phy_ack_l1_x*, the Controller de-asserts *low_pwr_clkreq_out_n*. If the link partner also de-asserts its *CLKREQ#* output, the core clock becomes inactive and the *low_pwr_clkreq_in_n* input to the Controller will be de-asserted. The L1 PM Substates State Machine transitions to L1.1 on sensing the *low_pwr_clkreq_in_n* input high.

Any local L1-substate exit triggers bring back the controller to L1.0 state. During exit, the controller asserts *low_pwr_clkreq_out_n* to turn on the core clock. This results in *low_pwr_clkreq_in_n* becoming asserted. On sensing *low_pwr_clkreq_in_n* low, the L1 PM Substates State Machine performs another handshake with the phy by de-asserting *low_pwr_phy_ent_l1_x* and waiting for the phy to respond with the de-assertion of *low_pwr_phy_ack_l1_x*. This handshake is necessary to prepare the phy for the re-activation of the reference clock. Once this handshake has been completed, the phy transitions back to the L1.0 substate.

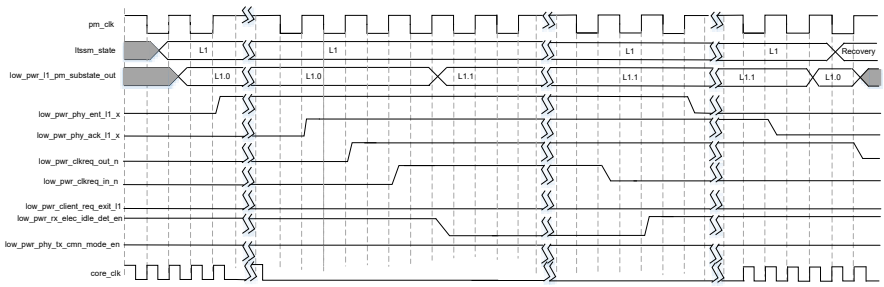


Figure 10.7: L1.1 entry and exit initiated by link partner

Figure 10.7 shows the operation when the exit from L1 is initiated by the link partner that the Controller is connected to. The Controller enters L1.1 from L1.0 when the entry conditions for L1.2 are not satisfied and the entry conditions for L1.1 are satisfied. After completing the *low_pwr_phy_ent_l1_x/ phy_ack_l1_x* handshake with the phy, the Controller de-asserts *low_pwr_clkreq_out_n*. If the link partner also de-asserts its CLKREQ# output, the core clock will become inactive and the *low_pwr_clkreq_in_n* input to the Controller will be de-asserted, causing the L1 PM Substates State Machine to enter the L1.1 state.

The Link Partner initiates the transition of the link from L1 state by asserting its *CLKREQ#* output. This results in assertion of the *low_pwr_clkreq_in_n* input to the Controller. On sensing *low_pwr_clkreq_in_n* low, the L1 PM Substates State Machine prepares the phy for exit from L1.1 by de-asserting *low_pwr_phy_ent_l1_x* and waiting for the phy to de-assert *low_pwr_phy_ack_l1_x*. Once this handshake has been completed, the L1 PM Substates State Machine transitions to the L1.0 substate. Meanwhile, the assertion of CLKREQ# results in the core clock becoming active. This enables the LTSSM to move out of L1 into Recovery.

10.3.6 L1.2 Operation

10.3.6.1 L1.2 Entry Sequence

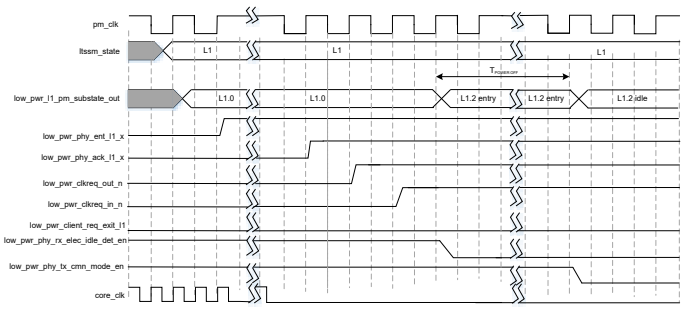


Figure 10.8: L1.2 Substate entry operation

Figure 10.8 shows the sequence for the L1 PM Substates State Machine to enter the L1.2 substate, when the Controller is configured as EP or RP. If the entry conditions for L1.2 are satisfied, it first performs the handshake with the phy using the *low_pwr_phy_ent_l1_x* and *low_pwr_phy_ack_l1_x* signals to prepare the phy for the removal of the reference clock. Once the phy has asserted *low_pwr_phy_ack_l1_x*, the Controller de-asserts *low_pwr_clkreq_out_n*. If the link partner also de-asserts its CLKREQ# output, the core clock will become inactive and the *low_pwr_clkreq_in_n* input to the Controller will be de-asserted. The L1 PM Substates State Machine transitions to L1.2.Entry when it senses the

low_pwr_clkreq_in_n input high. While the L1 PM Substates State Machine is in the L1.2.Entry substate, it monitors the *low_pwr*_clkreq_in_n input and transitions back to the L1.0 substate if it is found asserted. If *low_pwr*_clkreq_in_n remains de-asserted, the state machine will stay in the L1.2.Entry substate for an interval TPOWER_OFF (set to 2 microseconds) and then transition to L1.2.Idle substate.

10.3.6.2 L1.2 Exit Sequence

When the L1 PM Substates State Machine is in the L1.2.Idle substate, either the local client or the link partner may initiate a transition of the link out of the L1-substate. The figure below illustrates the operation of the L1.2 substates when there is an exit trigger from local client. Signal *low_pwr_client_req_exit_l1* represents all local exit triggers in the *Figure 10.9*

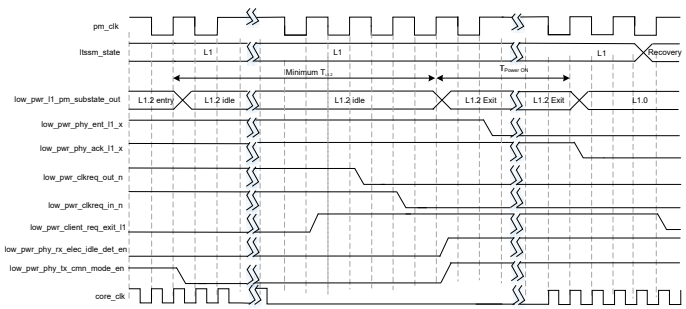


Figure 10.9: Locally initiated exit

Any one of the local exit triggers initiates the L1.2 exit process. Controller first asserts *low_pwr_clkreq_out_n* to turn on the core clock. This results in *low_pwr_clkreq_in_n* becoming asserted. On sensing *low_pwr_clkreq_in_n* low, the L1 PM Substates State Machine transitions to the L1.2.Exit substate.

While in the L1.2.Exit substate, the L1 PM Substates State Machine performs the *low_pwr_phy_ent_l1_x/ phy_ack_l1_x* handshake with the phy to prepare the phy for the re-introduction of the clocks, and subsequently transitions back to the L1.0 substate. The L1 PM Substates State Machine must stay in the L1.2.Exit substate for a minimum interval of TPOWER_ON. The duration of this interval is determined by the setting of the TPOWER_ON value and scale parameters in the ‘i_regf_L1_PM_cap_struct/i_L1_PM_ctrl_2’ register. The interval can vary of 0 to 3100 microseconds based on the setting of these register fields.

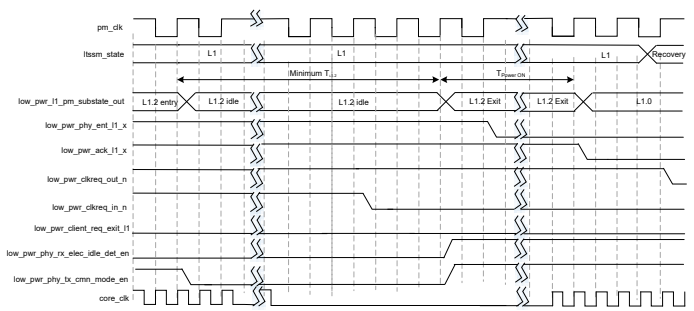


Figure 10.10: Exit initiated by link_partner

Figure 10.10 shows the operation when the exit from L1 is initiated by the link partner that the Controller is connected to. When in the L1.2.Idle substate, the Link Partner initiates the transition of the link from L1 state by asserting its *CLKREQ#* output. This results in assertion of the *low_pwr_clkreq_in_n* input to the Controller. On sensing *low_pwr_clkreq_in_n* low, the L1 PM Substates State Machine transitions to the L1.2.Exit substate (after satisfying the requirement of a minimum 4 us stay in L1.2.Idle). The L1 PM Substates State Machine then transitions to the L1.0 substate, after completing the handshake with the phy for re-enabling its clocks, and staying in the L1.2.Exit substate for a minimum interval of *TPOWER_ON*.

10.3.7 Register Details for L1-Substate

10.3.7.1 Programming of spec defined L1-substate registers

Table 10.1 provides reference to program specification defined registers. Refer to the *Register* chapter in user guide for the full list of registers. Registers with prefix port_* and capability registers are read-only while accessing from PCIe-link. These registers are writable through ip register access interface. These registers need to be initialized to match PHY and system electrical characteristics. These registers are used by the standard system initialization software to program RW control registers in L1-substate capability space of Root Port and Endpoint.

Table 10.1: L1 substate registers

Register Name	Guide to select value	Who updates	Issues with wrong values
L1PrtPvrOnVal and L1PrtPvrOnScale	Port's PHY T_POWER_ON value. For example this is TP1.2_to_P1 value in Cadence PHY	Local Client's controller and PHY initialization firmware.	PCIe system initialization software uses this value to program in **T_power_on** register.
L1PwrOnVal and L1PwrOnSc	Max (EP Port_T_power_on, RC Port_T_power_on)	PCIe System initialization software on Host.	Lower value than the required value makes one device to drive into an unpowered remote device. This can results in link_down event and LTSSM moves to detect state.
L1PrtCmMdReStrTime	Time required for the Port's PHY to establish common_mode actively during the transmission of TS1s. For example in Cadence's PHY this value is t_common_mode.	Local Client's controller and PHY initialization firmware.	Initialization software uses this value to program common_mode_restore_time register.
L1CmMdReStrTime	Max(RC port_common_mode_ restore_time, EP port_common_mode_ restore_time)	PCIe System initialization software	Lower value than the required value makes one device to drive into an unpowered remote device. This can results in link_down event and LTSSM moves to detect state.

Table 10.1: L1 substate registers

Register Name	Guide to select value	Who updates	Issues with wrong values
L1ThrshldVal and L1ThrshldSc *	This is used only if LTR is enabled and ASPM L1.2 is enabled. This is the worst case latency a request would face to get completed in the presence of L1.2. Please refer LTR section of PCIe specification for the usage of LTR mechanism. An example calculation is: LTR_L1_2_threshold = Service_request_latency + 2 * (TL1.2 + TL1O_REFCLK_ON + TP1_to_P0 + TCOMMONMODE + 2us); • Service_request_latency:: Worst case delay for the Root Port side client to respond to Read request or to accept WR request • TL1.2 - Minimum time to stay in L1.2 as per Spec. This is 4us • TL1O_REFCLK_ON - CLKREQ# assertion to Refclk valid when exiting L1.2. This is TPOWER_ON + Additional time taken by the system to enable reference clock; - TP1_to_P0 – Time taken by the PHY to change power state from P1 to P0. TCOMMONMODE - Same as the common_mode_ restore_ time register; - 2us - Margin to account for latency in controller and handshakes; - Multiplication with two in the expression is required to account for the L1.2 exit that a request and/or Completion might require.	PCIe System initialization software	ASPM L1.2 entry happens only when Endpoint's LTR requirement is larger than this threshold. Incorrect programming of threshold can cause unexpected latency for the Endpoint requests.

Note:

- As defined in the *PCIe Specification*, ASPM L1.1 entry does not automatically check for latency threshold requirements. Hence, Port PHY's (TL1.1_TO_REF_CLK_ON + TP1_to_P0) has to be less than possible minimum latency requirement of client application. TL1.1_TO_REF_CLK_ON is the time to turn-on reference clock after assertion of *CLKREQ#* in L1.1. For example, this is tP1.1_to_P1+ additional time taken by system to turn on reference clock for a Cadence PHY. TP1_to_P0 is the time taken by the PHY to change power state from P1 to P0. If this delay is more, it is required to turn off L1.1 to meet LTR requirement of client application.
- If you have selected the internal power shutoff feature, 12 PM_CLK periods must be added to LTR_L1_2_threshold value.

10.3.7.2 L1 Substates Entry Delay

Entry to L1 substates from basic L1 can be delayed with field **L1SED** in the ip register *i_regf_ip_pl/i_pm_cfg_5*. Controller responds to L1 exit event while waiting for expiration of this delay.

10.3.7.3 Explicit L1-Substate Exit from Client Side or Blocking L1-Substate Entry

Client logic can trigger an explicit L1-substate exit by asserting *low_pwr_client_req_exit_l1*. This signal triggers an exit from L1-substates to L0 if controller is already in L1-substates. Ensure that you use this signal if L1-substate shuts off

clocks which drives new request to HLS or AXI interface. It is highly recommended to implement hardware logic by client to assert this signal before initiating new TLP or register access from client side. Hardware logic reduce overhead on firmware or software driver. This signal is also useful if you want to block l1-substate entry to reduce latency for any pending outstanding operation. *Figure 10.11* shows the use of this signal.

Note: L1.x in the figure can be either L1.1 or L1.2.

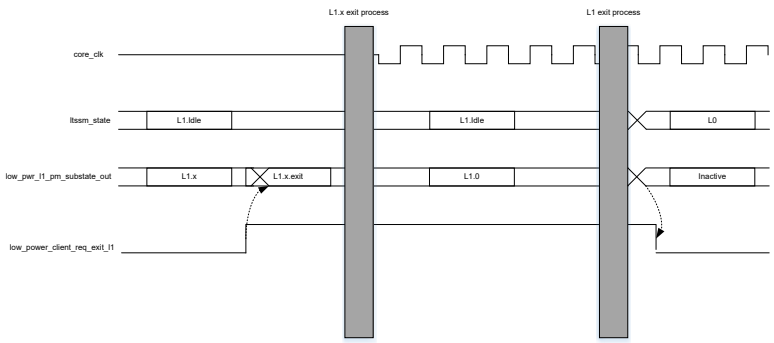


Figure 10.11: Using low_pwr_client_req_exit_l1 to exit from L1.x

10.3.8 General Integration Details for L1 Substates

10.3.8.1 PM_CLK Frequency Selection

In L1-substate *pd_pso_core_clk* can be turned off. A separate clock, *PM_CLK*, is required for the L1-substate machine to operate. Default frequency of this clock is selected in configurator. Client firmware needs to program *i_regf_ip_pl*/*i_pm_cfg_7* field **PMCFS** in ip registers for changing *PM_CLK* frequency. The change in *PM_CLK* frequency is supported only when LTSSM is not in L1.

Pm_clk is required only when L1-substate capability is enabled in PCIe configuration space.

10.3.8.2 Internal Power Shutoff

Low power link state is defined in the *PCI-SIG Specification*. This is a low power state of the link with clocks and PCIe link related circuitry turned off. The PCIe controller has internal power shutoff feature to further improve this power saving. Internal power shutoff reduces leakage current and improves power saving in L1.2 link state. This is accomplished with the help of CPF based power flow. Entering and exiting internal power shutoff do not require any special handshake from client side. There are tuning registers and sideband signals to change default behavior of the PCIe controller.

Conditions to Enter L1.2 with Power Shutoff

After entering basic L1 state, the controller enters L1.2 substate with power shutoff if the following conditions are met:

1. Conditions to enter L1.2 substate are true. Refer to the *PCI-SIG Specification* for detailed description of these conditions.
2. PCIe-link to client side path buffers are empty.
3. No outstanding packets pending from PCIe-link and from client side.

The Controller waits in basic L1 state to meet these conditions. Once these conditions are met, the controller enters L1.2 substate and it shuts off power while in L1.2.idle state. During the wait, the controller normally responds to L1 exit conditions. If power-shutoff is disabled in IP register, controller enter L1.2 from L1.0 without waiting for these powershutoff conditions.

Conditions to Exit Power Shutoff Mode

Conditions to exit L1.2 substate also trigger an exit from power-shutoff mode. Refer to *L1 Substate* section for more information.

Power Domains in PCIE Controller

The PCIe Controller Power Management Logic is divided into two power domains:

- Power Shut Off power domain (PD_PSO)
  - Includes sub-modules of the Controller which can be powered OFF during L1 substates.
  - Contains the power management logic, LTSSM and PIPE interface FIFOs.
  - These are always ON, even during L1 substates.
  - The power management logic runs on an independent clock called “PM_CLK”.
  - The clock to LTSSM and PIPE interface FIFO modules will be shut OFF while in L1 substates but the power will be ON to preserve the state of these modules.
- Always ON power domain (PD_AON)
  - Contains the power management logic, LTSSM and PIPE interface FIFOs.
  - These are always ON, even during L1 substates.
  - The power management logic runs on an independent clock called “PM_CLK”.
  - The clock to LTSSM and PIPE interface FIFO modules will be shut OFF while in L1 substates but the power will be ON to preserve the state of these modules.

Note: The accurate domain information is available in CPF file.

Power Shut-off Schemes Available

CPF based power flow and internal power and isolation controller

Sequence to turn-off and turn-on power and isolation is generated by internal control logic. No handshake is expected from user in this mode. Just following CPF power flow achieves power-shutoff.

CPF based power flow with internal power and isolation controller in bypass mode

A sideband signal LP_CTRL_BYPASS_ENABLE enables bypass mode for power and isolation controller. In this mode sequence to turn-off and turn-on power and isolation is generated by user logic. User can use LP_CTRL_* signals to pass power sequence events to controller. User can still use delivery CPF power flow. In bypass mode, signals which controls CPF flow are MUXed to use LP_CTRL_* signals from user.

Figure 10.12 illustrate power-and-isolation controller bypass mode.

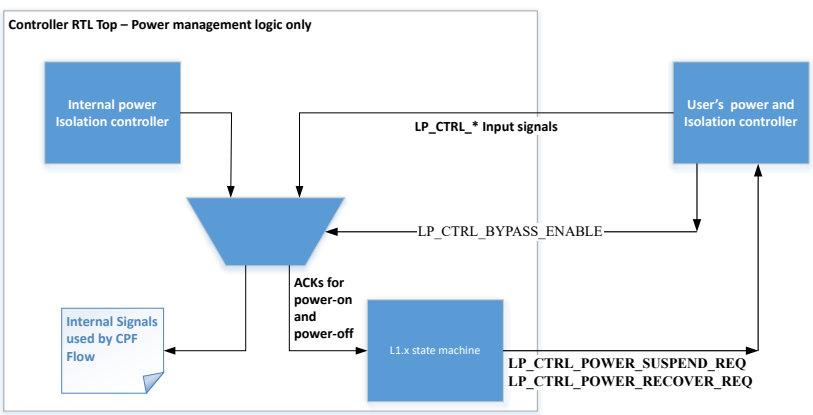


Figure 10.12: Bypass Mode for Internal Power and Isolation Controller

Clock Gating Cell and Reset Synchronizers

There are clocking gating latches present in the Controller IP, that should be considered in post-rtl tool flows.

These latches are used to turn off clocks internally before shutting down power domains. The Clock gating latch instances present in the Controller IP are named as: **.i_lp_ctrl.u_clk_gating_cell_**

There are reset synchronizers present in the Controller IP, that should be considered in post-rtl tool flows. These are used to synchronize fundamental resets and the resets generated internally during power-on sequence.

The reset synchronizer instances present in the Controller IP are named as: **.i_lp_ctrl.u_reset_sync_**

Shutting off External SRAMs

External SRAMs can also be turned off by using PD_PSO power domain turn off conditions. Refer to the `CPF file` to get condition to turn off PD_PSO. An alternative method is to use *low_pwr_ctrl_power_suspend_req* and *low_pwr_ctrl_power_recover_req* sideband signals to trigger SRAM power-off and power-on sequences. These signals are asserted in L1.2 only if internal power shutoff conditions are met. Zero to one transition on this signal indicate trigger for respective sequences. In normal mode, User's SRAM power-on and power-off sequences are expected to finish in 4 *PM_CLK* cycles.

Registers Programming to Control Power Shutoff

The following tuning registers control the default behavior of controller.

Turning off Power-shut-off Feature

Power-shut-off feature can be turned off by using field “Low Power debug and control Register 1/Enable Power shutoff in L1.2” in local management space. If power-shut-off is turned off, controller enters L1.2 substate without waiting for power-shutoff conditions. Internal power is not turned–off during L1.2 substate if this field is set.

Timeout Mechanism while Waiting for Power-shutoff Condition

A timeout mechanism can be enabled with field “Low Power debug and control Register 1/Timeout while waiting for RX IDLE and OUTSTANDING Idle” in local management space. A timeout will force controller to enter L1.2 substate without power-shutoff mechanism. After L1.2 exit controller normally responds to pending packets from PCIe link and client side. Customer can use this feature if there exists some condition in the system which takes long time for responses. A timeout will make controller enter L1.2 and saves power while waiting for responses. A status field “Low Power debug and control Register 1/ Timeout occurred during RX and outstanding wait” in local management space gives an indication on whether L1.2 entry was triggered after this timeout.

Sideband Signals to Control Power Shutoff

Bypass mode for internal isolation and power-shutoff controller

The following table list describes the signal required for bypass mode.

Pin Name	Direction	Clock	Description
low_pwr_ctrl_bypass_enable	IN	static	Puts isolation and power-shutoff controller to bypass mode. Client logic has full control on the isolation and power-shutoff sequence in bypass mode. If bypass-mode is not required, tie this signals to ‘0’.
low_pwr_ctrl_power_suspend_req	OUT	PM_CLK	This is an indication for user to initiate power_suspend-sequence. User has to assert LP_CTRL_POWER_SUSPEND_ACK after completing power-off-sequence if LP_CTRL_BYPASS_ENABLE is true. This signal is available for user for SRAM shut-off or other user side logic shutoff. A ‘0’ to ‘1’ transition is power_suspend sequence request. User is expected to finish power-off—sequence in 4 PM_CLKs if LP_CTRL_BYPASS_ENABLE is false.
low_pwr_ctrl_power_suspend_ack	IN	PM_CLK	User has to assert this signal after completing power-off-sequence in bypass mode. This is used by controller only if LP_CTRL_BYPASS_ENABLE is true.

Pin Name	Direction	Clock	Description
low_pwr_ctrl_power_recover_req	OUT	PM_CLK	This is an indication for user to initiate power_recover-sequence. You need to assert LP_CTRL_POWER_RECOVER_ACK after completing power-on-sequence if LP_CTRL_BYPASS_ENABLE is true. This signal is available for user for SRAM power-on or other user side logic power-on. A ‘0’ to ‘1’ transition is power_recover sequence request. You are expected to finish power-on-sequence in 4 PM_CLKs if LP_CTRL_BYPASS_ENABLE is false.
low_pwr_ctrl_power_recover_ack	IN	PM_CLK	You need to assert this signal after completing power-on-sequence in bypass mode. This is used by controller only if LP_CTRL_BYPASS_ENABLE is true.
low_pwr_ctrl_state_ret_save	IN	PM_CLK	A pulse in this signal save internal states before power shutoff. This is used by controller only if LP_CTRL_BYPASS_ENABLE is true. Note: *This is not used in this configuration User can tie this to ‘0’.*
low_pwr_ctrl_state_ret_restore	IN	PM_CLK	A pulse in this signal restores internal states. This is used by controller only if LP_CTRL_BYPASS_ENABLE is true. Note: *This is not used in this configuration User can tie this to ‘0’.*
low_pwr_ctrl_power_en_n	IN	PM_CLK	Power enable phase 1: This signal controls phase 1 of the power island power up. A two phased enable process is supported by this state machine, so power up of components on the power island can be staggered, in order to prevent excessive current draw. Note: *This is an **active low** signal.* • 1’b0 : Power enabled • 1’b1 : Power disabled This is used by controller only if LP_CTRL_BYPASS_ENABLE is true.
low_pwr_ctrl_power_isolation_en	IN	PM_CLK	Power isolation enable: Activating this signal enables the isolation function for power islands described in CPF flow. This is used by controller only if LP_CTRL_BYPASS_ENABLE is true.
low_pwr_ctrl_power_reset_n	IN	PM_CLK	Power reset: This signal is used to reset any logic that requires a reset when the power island being controlled is switched on after being disabled. This is used by controller only if LP_CTRL_BYPASS_ENABLE is true.

Retention and Power off Sequence>

Figure 10.13 shows the expected timing relationship for low_pwr_ctrl_* signals in power-controller-bypass mode. Power controller bypass is enabled by asserting LP_CTRL_BYPASS_ENABLE. Power_suspend_ack completes the power_down sequence as shown in the below diagram. Application logic can insert delays between each step. LP_CTRL_ prefix is omitted in timing diagram signal names. L1.2.Idle exit triggers are processed only after the power down sequence is completed.

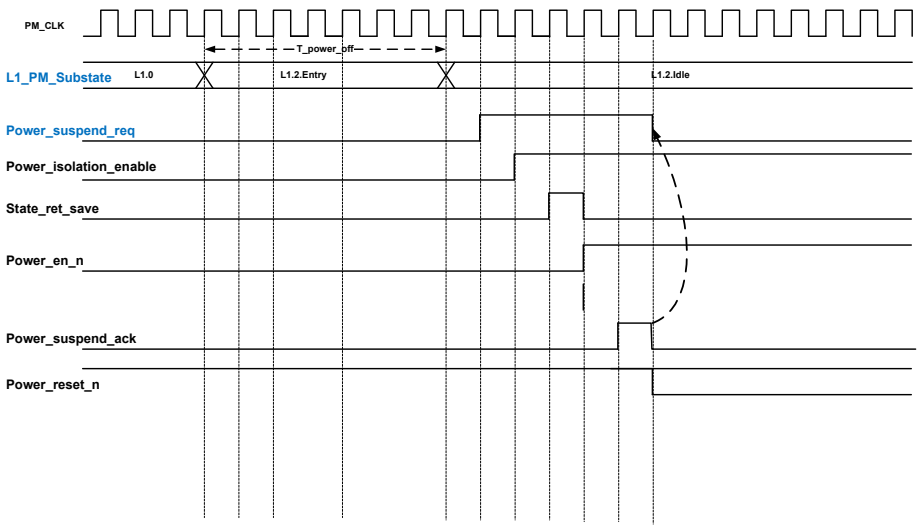


Figure 10.13: Power Off Sequence

Power ON and Restore Sequence

Figure 10.14 shows the expected timing relationship for *low_pwr_ctrl_** signals in power-controller-bypass mode. Power controller bypass is enabled by asserting *low_pwr_ctrl_bypass_enable*. The timing diagram shows a power-on sequence. *low_pwr_ctrl prefix* is omitted in timing diagram signal names.

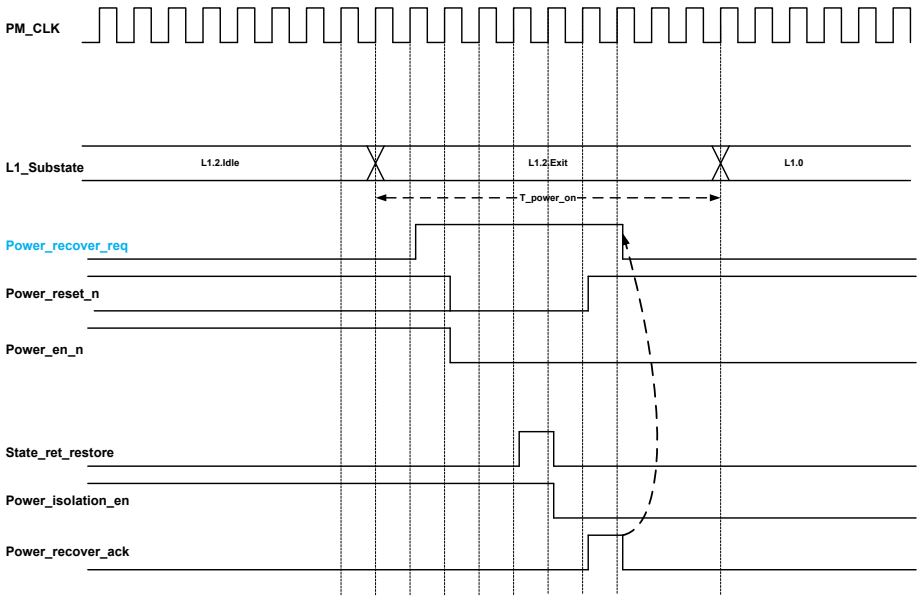


Figure 10.14: Power Restore Sequence

Power Domains

The PCIe Controller Power Management Logic is divided into two power domains.

- Power Shut Off power domain (PD_PSO)
  - Includes sub-modules of the Controller which can be powered OFF during L1 substates.
- Always ON power domain (PD_AON)
  - Contains the power management logic, LTSSM and PIPE interface FIFOs. These are always ON, even during L1 substates.

The power management logic runs on an independent clock called 'PM_CLK'.

The figure below shows the power domain association of different modules in PCIe controller. The below diagram is indicative and the accurate domain information available in CPF file.

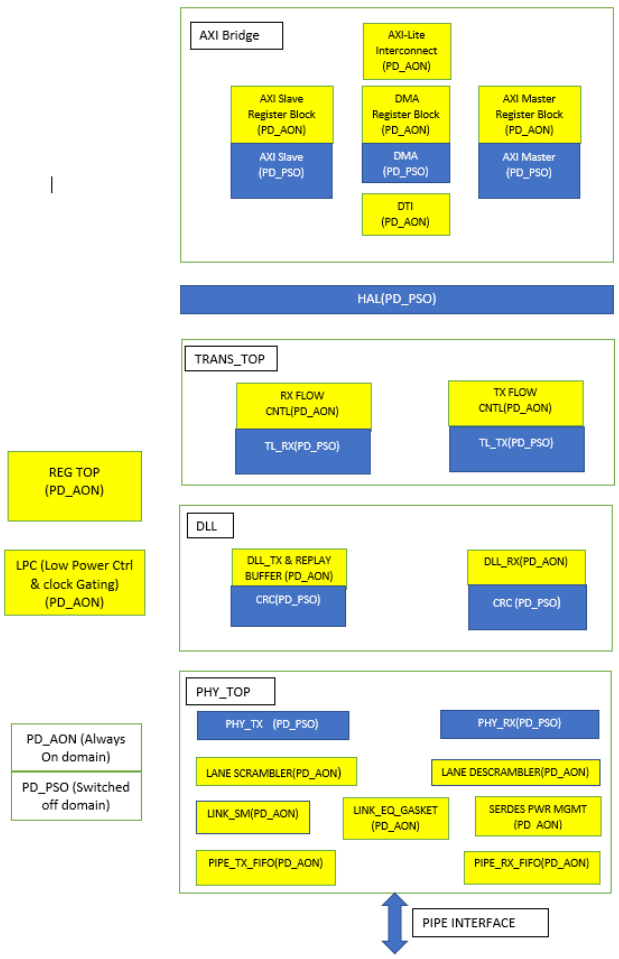


Figure 10.15: Power Domain Association

10.3.9 L2 and L3

10.3.9.1 L2 and L3 Entry Process

Both L2 and L3 are power saving states with no power supply to the whole controller. Management of power in this state has to be managed by the Application logic. Handshake to enter L2/L3 starts by RP sending PME_Turn_Off message.

Steps required for Endpoint

1 When the EP received a PME_Turn_OFF Message from the RP , the controller sets the following signals. Application Logic(AL) can choose either one of the option to get notified of this event. PCIe spec allows RP to send PME_Turn_off message in both D0 and non-D0 states.

   a low_pwr_l2_pme_turn_off_received
   b local_interrupt and interrupt_signals_sideband[32] , which mainly reflect bit[0] of ‘Local Error Status Register 1” . Client can mask this using ‘Local Error Mask Register 1’ in which case none of the signals listed here would get set.

2 AL needs to acknowledge power off request from RP within 10ms as required by PCIe specification. AL may have outstanding packets ongoing which it may need to flush prior to acknowledging the PME Turn OFF Request. AL can set this acknowledgement once it is ready in any of the two ways as shared below:

   a Set low_pwr_l2_pme_turn_off_ack for one cycle .
   b Write 1 to bit[0] of ‘Local Error Status Register 1” to clear it if using the Local Interrupt method . This will create an ACK pulse similar to ‘a’ above internally.

3 Once the AL sets the ACK, the controller will wait for Timeout in ‘i_pm_cfg_7[31:24]: PM L2 Entry OB Packet Flush Timeout’ for pending OB packets to be flushed before it schedules PME Turn oFF ACK Msg and post that start the DLLP handshake for L2 entry.
4 To Determine successful entry to L2 , AL can look at ‘cap_state_common[53:50]’ to equal 4’h8 , the ‘cap_state_common_change_pulse’ can be used as trigger to check for the same.
5 AL can then turn off the Clocks and Power.

Steps required for RP

1 After transmitting PME Turn Off Message, Controller waits for PM Enter L2 DLLPs from the Endpoint.
2 Once these DLLPs are received, controller initiates steps to enter L2/L3 after the outbound pipelines are empty or after a programmed timeout.

For debug purpose there are ip-mgmt registers available to delay or block the entry to L2 upon receiving DLLPs. Please refer “i_pm_cfg_4/Block PM_Request_ACK for Enter L23 DLLPs” and “i_pm_cfg_7/HLS OB Deactivate Timeout’.

10.3.9.2 Wakeup or Exit from L2,L3 State

The PCIe Controller supports systems that use a wakeup mechanism.

The PME context of the PCIe controller shall be captured by the Application logic before sending the PME_TurnOff Acknowledge message to the Root Complex. Client logic must store the controller’s bus number and device number form local management before acknowledging pme_turnoff. The client logic shall specify the Requester ID of the PME message sent when power is re-applied to the controller and the link reaches L0. The following sequence describes the process of supporting *WAKE#*.

1 Assume the Controller is in L2. Client logic is maintaining PME context and Requester ID using client application logic that is powered, either by Vaux or full power.
2 Now the client logic may decide to wake up the controller. The client logic shall drive the *WAKE#* out-of-band signal to signal to the Power Management Controller that the controller requires power to be reapplied.

Note: The *WAKE#* signal is external to the PCIe Controller and is not used by the PCIe Controller.

3 Power is reapplied to the controller.
4 A hard reset is required upon restoring power to bring the link back up to L0
    a The client logic then restores any PME Context to the controller registers via the ip register access interface.
    b Prevent the Controller from automatically transmitting PM_PME message by programming the bit-20 *disable_pme_message_on_pm_status* to '1' in ip register PME Service Timeout Delay Register..
    c The *disable_pme_message_on_pm_status* bit needs to be programmed to '1' before setting the *PME Status* bit to '1'.
5 The client logic then sends a *PM_PME* message over the client interface using the Requester ID that was captured before entering L2/L3.
6 The Root Complex may then perform a configuration write to the controller in order to move the device from the D3-hot state.

Chapter 11: Synthesis and Physical Design Guidelines

11.1 Synthesis Setup

The IC synthesis strategy is to synthesize and optimize each IP in the design individually in isolation from all other IP. Reliance is then made on the following characteristics of the IP level synthesis constraints to ensure that no further optimization is required at the top level. Hence, the top level netlist integration task is simplified to IP uniquification and concatenation of the constituted netlists.

Table 11.1: Deliverables

Category	Description
Synthesis	Synthesis has been run with Genus Version `/genus/162/16.21.000`
Library	Point to your specific library.
Constraints	Under the `dbPcie_ios_top/` directory there is a `synth/constraints` directory which has the constraints file: `cdnhpa_ips_default.func.sdc`.
RTL/Libraries	The `project.tcl` and the `setup.tcl` point to our sample synthesis run. Ensure that you change the pointers to pick up the right rtl and libraries for our environment.
Script	The following script is used for synthesis: `synth/scripts/genus/genus_synth.tcl`
Command	Run the synthesis using, `./run_phys.csh -synth`

Note:

- If you are using Genus as your synthesis tool, ensure that the optimization switch is on. Also, ensure that there is no module preserve attributes on.
- If you are using Synopsys DC as your synthesis tool, ensure that the Remove redundanct logic" switch is on.

11.2 Technology Specific Model

You can replace the below component provided `/hdl/hdl_src/reuse_components` directory with your technology-specific components retaining the same filenames.

- cdnsdru_clk_gating_latch_synth
- cdnsdru_scan_mux_synth
- cdnsdru_reset_sync_synth
- cdnsdru_datasync_synth_sr
- cdnsdru_datasync_synth_ar
- cdnsdru_clk_mux_synth

11.3 Clock MUX for CPCS Module

The clock mux *cpcs_synth_clk_mux.v* for CPCS is a reuse module that is present inside the `rtl/cpcs/` folder. Ensure that you replace *cpcs_synth_clk_mux.v* with technology specific library module. Location of the *cpcs_synth_clk_mux.v* module:

`<trunk>/rtl/cpcs/cpcs_synth_clk_mux.v`

11.4 Handling Asynchronous FIFOS in Physical design

Asynchronous FIFOs use gray coded vectors across clock domains to pass read and write pointers. The skew between the bits of this vector must be minimum in comparison to the clock. PNR (Place & Route) tool needs to carefully handle this vector synchronizer blocks to not add skew between vector bits.

The gray coded vector synchronizers need special treatment.

- Every synchronizer group is treated as a structured data-path that requires special handling.
- Metastability hardened flops of the lowest Vt available recommended to be used. Some foundries provide special synchronizer cells themselves.
- If single flops are used to create the synchronizer group, use aligned placement is recommended. The flops will also be grouped in columns, which significantly eases the clock routing since the clock pins will align if the flops are the same type.
- Drive the flops in the group with the same clock-buffer or the same clock-gating cell and place them close to optimize routes
- No combo-logic between the Q and D of the destination flop. PD (Physical Design) must not insert buffers between the flops in the synchronizer group.

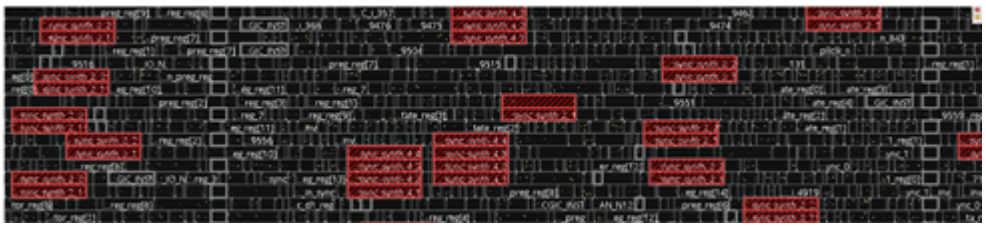


Figure 11.1: Example guided placement of Gray coded pointer synchronizers

PCIe Controller IP uses common reuse block for asynchronous FIFOs. Clock domain synchronizers are also common across the design. Customer is expected to replace synchronizers with metastability hardened library specific cells.

Asynchronous FIFO gray coded vectors can be obtained by searching following instance patterns:

- *ptr_gray**[*]**.sync_cell.cdns_syncflop_inst*
- *sync**.***.cdns_syncflop_inst.*.
- *ptr**.***.cdns_syncflop_inst.*

The following async FIFO modules are used in IP. Source wire to input of the synchronizer wire is listed for each module. This is listed here only for information purposes.

- <prefix>cpcs_rctc
- wr_ptr_gray -> i_data_wr_ptr_gray[*].sync_cell.cdns_syncflop_inst.*. u_cdnsdru_datasync_synth.d_in
- <prefix>cpcs_128b_rctc_32b
- wr_ptr_gray -> i_phy_data_sync_wr_ptr_gray[*].sync_cell.cdns_syncflop_inst.*. u_cdnsdru_datasync_synth.d_in
- <prefix>pipe_rx_fifo
- write_addr_gray -> write_addr_sync.*.cdns_syncflop_inst.*. u_cdnsdru_datasync_synth.d_in
- read_addr_gray -> read_addr_sync .*.cdns_syncflop_inst.*. u_cdnsdru_datasync_synth.d_in
- <prefix>pipe_tx_fifo
- write_addr_gray -> write_addr_sync.*.cdns_syncflop_inst.*. u_cdnsdru_datasync_synth.d_in
- read_addr_gray -> read_addr_sync.*.cdns_syncflop_inst.*. u_cdnsdru_datasync_synth.d_in
- <prefix>afifo
- afifo_wclklog.wrPtr -> afifo_rclklog.sync_wr_ptr.*.cdns_syncflop_inst.*. u_cdnsdru_datasync_synth.d_in
- afifo_rclklog.rdPtr -> afifo_wclklog.sync_rd_ptr.*.cdns_syncflop_inst.*. u_cdnsdru_datasync_synth.d_in
- <prefix>async_fifo
- wptr_gray -> i_cdnsdru_datasync_wptr_gray.sync_wr_ptr. *.cdns_syncflop_inst.*. u_cdnsdru_datasync_synth.d_in
- rptr_gray -> i_cdnsdru_datasync_rptr_gray.sync_wr_ptr. *.cdns_syncflop_inst.*. u_cdnsdru_datasync_synth.d_in

11.5 Synthesis App notes

- If you are using Genus as your synthesis tool, please make sure the optimization switch is on.
- Also ensure that there are no “module preserve” attributes on.

- If you are using Synopsys DC as your synthesis tool, please make sure the “Remove redundant logic” switch is on.
- Please use hierarchical synthesis flow and not flattened synthesis flow. This will help reduce LEC run time and adapt to the Smart LEC flow.
- Ensure that you always use the latest version of Genus.

Chapter 12: Integration Guide

12.1 Start-Up Sequence

The Cadence PCIe IP subsystem has different reset inputs that are configured as synchronous to the respective clocks. The reset inputs can be asserted asynchronously, but should be synchronously de-asserted to the respective clocks.

The Controller internally synchronizes all the reset inputs to respective clock domains. Hence, the client logic is not required to add any reset synchronizers outside the IP.

The following sections provide information on the reset inputs and the recommended sequences in which they should be activated.

12.2 Reset Sequence

The start-up sequence for the Controller and PHY subsystem is as follows.

Note: The relative order of reset de-assertion in the below sequence is important with recommendation of at least 1 clock cycle or more of delay between the steps.

1 Power On the PCIe subsystem and assert all resets to the PHY and Controller, this:

- Enables reference clocks to the PHY.
- Enables all free running clocks to the IP *phy_apb_clk*, *core_apb_clk*, *pm_clk* and *axi_clk*.

2 De-assert *POR_N* (*Power On Reset*) and *PERST_N*.

Note: These signals are system Level reset and not present in the PCIe IP.

3 De-assert APB reset *link*_apb_rst_n* to ensure the Controller's APB interface is active.

4 De-assert APB reset *phy_apb_rst_n* to ensure the PHY's APB interface is active.

5 Program PHY registers and set PHY and Controller static configuration pins.

6 De-assert *phy_rst_n* reset inputs to activate the PHY.

Note: These are asynchronous reset inputs.

Wait for *pipe_l_phy_status* to transition from 1 to 0 before going to next step.
This indicates the PHY output clocks *(core_clk* and *pipe_clk*) that the controller uses are stable.

Note: For *phy_p*_reset_n* inputs:

- For single link configuration, all *phy_p*_reset_n* inputs should be driven high. It is recommended to connect SoC programmable register with default value 1)
- For multi-link configuration, upon de-assertion of *phy_reset_n*:
  - at least 1 of the *phy_p*_reset_n* inputs must be enabled, that is, by having its *phy_p*_reset_n high* upon de-assertion of *phy_reset_n*.
  - for remaining disabled links (if any), the respective *phy_p*_reset_n* should be driven to low (1'b0) from time 0 or before de-assertion of *phy_reset_n*.
  - After initial PHY startup, any link disabled at the de-assertion of *phy_reset_n* can be enabled by de-asserting its *phy_p*_reset_n* (i.e. driving it 1'b1).

7 De-assert the Controller Reset Inputs in the following sequence order:

a *core_rst_n*
b *axi_rst_n*
c *reg_sticky_rst_n*
d **_reg_sticky_rst_n* is asserted only first time entering after power up
e De-assert *reg_axi_rst_n* and *reg_rst_n*.
f De-assert *pm_rst_n*

8 Wait for *ip_reg_init_in_progress* to get de-asserted if already asserted after the ip management registers are programmed.
9 Set up Controller IP registers as per *Programming IP Registers* section.
10 Assert *link_training_enable* input to allow link training to start.

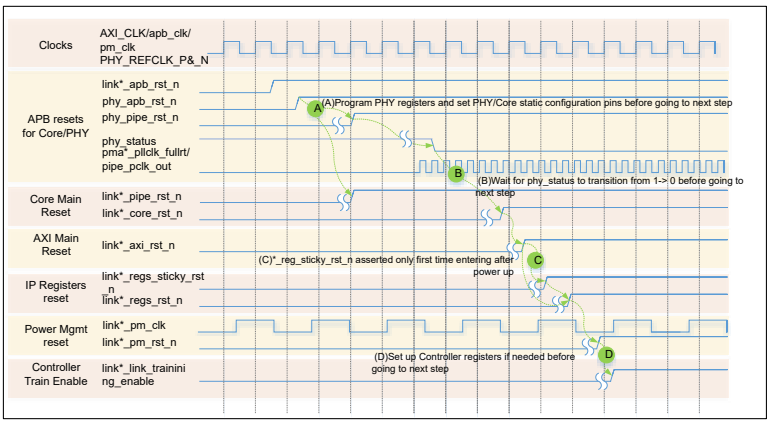


Figure 12.1: Reset sequence

12.2.1 Reset Reference Block Diagram

The below diagram is used as reference for integrating the resets to the PCIe IP (PCIe Controller with integrated PHY) within a SoC, and connecting to application logic. The Reset Sequencer implementing the reset sequence is described in the *Reset Sequence*. Reset Synchronizers are implemented inside the Controller IP to synchronize the resets to respective clocks.

Note: Reset Synchronizers are not required to be added outside the PCIe IP.

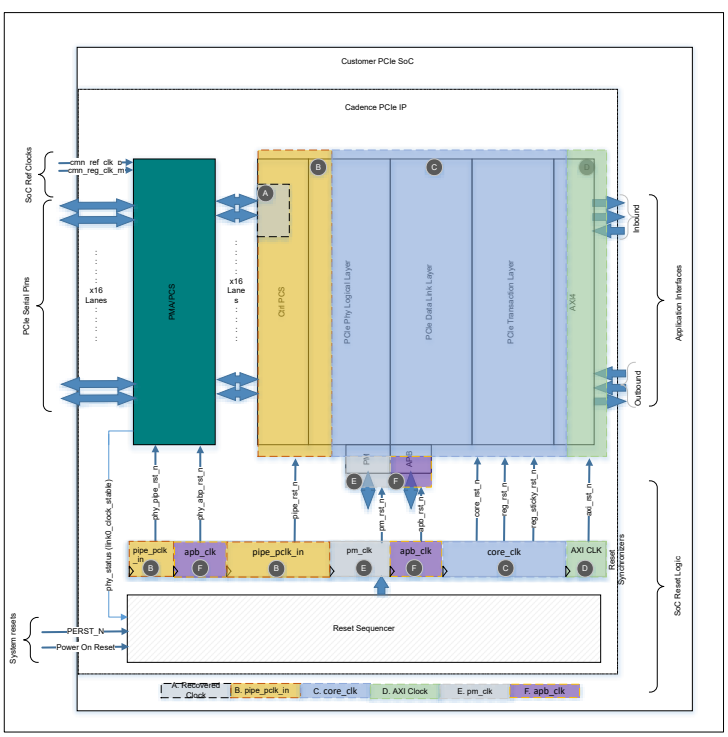


Figure 12.2: Example Reset reference block diagram for PCIe

12.3 Precision Time Measurement

Precision Time Measurement (PTM) enables precise coordination of events across multiple components with independent local time clocks. Ordinarily, such precise coordination would be difficult given that individual time clocks

have differing notions of the value and rate of change of time. To work around this limitation, PTM enables components to calculate the relationship between their local times and a shared PTM Master Time, an independent time domain associated with a PTM Root.

12.3.1 Features

The key features in the Controller's implementation of PTM are listed below:

- PTM Capability Register set is implemented in Physical Function 0.
- Supports Enhanced Precision Time Management.
- Supports interpretation A and interpretation B using Propagation Delay Adaptation Capability.
- PTM Requester Mode features:
  - Functions as PTM Requester when configured for Endpoint Mode.
  - Implements a local time clock based on core_clk.
  - Provides a simple register programming mechanism to local Firmware to trigger PTM dialogs when PTM Enable is set.
  - Computes current PTM Master time at the end of PTM dialogs and outputs to user via ptm_master_timer_out[63:0] and ptm_master_timer_out_valid.
  - Local PTM Context Status readable through APB interface.
- PTM Responder Mode features:
  - Functions as PTM Responder when configured for Root Port Mode.
  - Implements local time clock based on core_clk.
  - Has provision for user to preset the local time clock through the ptm_master_timer_in[63:0] and ptm_master_timer_in_valid inputs.
  - Automatically responds to PTM Requests when PTM Enable is set.
  - Local PTM Context Status readable through APB interface.

12.3.2 Functional Description

12.3.2.1 PTM Link Protocol

The following figure shows the PTM mechanism implemented in the Controller.

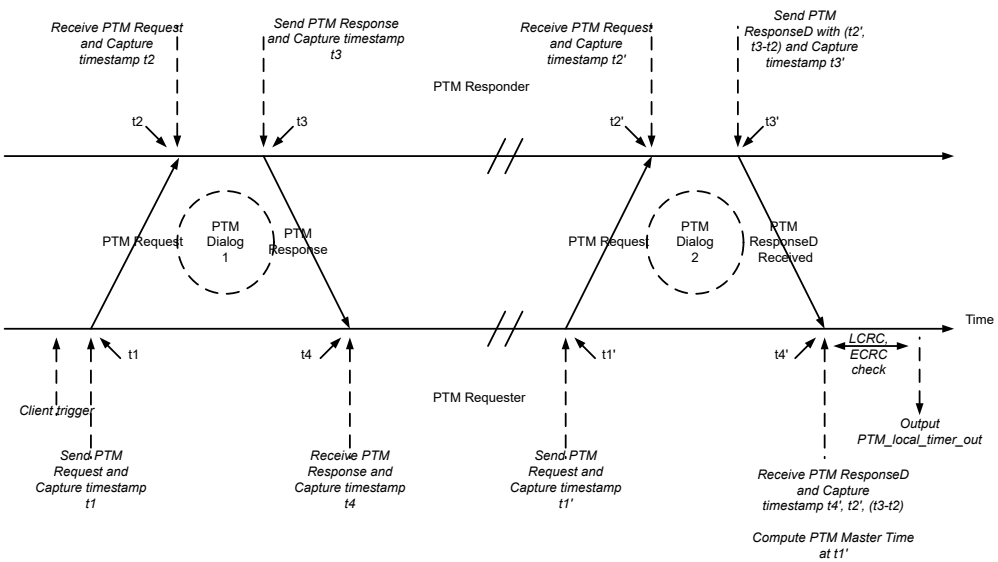


Figure 12.3: PTM Link Protocol

PTM Requester

The Controller functions as a PTM Requester when operating in Endpoint mode. In this mode, the Controller provides a mechanism to the Client to trigger the PTM Dialog process. The Client can trigger PTM dialog by programming the local "PTM Control Register" through the APB Interface.

Upon this trigger event, the Controller transmits a PTM Request message over the PCIe Link. Controller captures the timestamp (t1/t1') when the PTM Request message header (STP token) is transmitted.

The Controller waits to receive the PTM Response/PTM ResponseD message. The Controller captures the timestamp (t4/t4') when a PTM Response/ResponseD message is received. The Controller performs the LCRC, ECRC, TLP Header checks before registering the captured timestamp.

When the PTM ResponseD message is received, the Controller extracts the PTM Responder timestamps (t2' and t3-t2) from the message.

Using this information ,the Controller computes the PTM Master time at t1' using the expression:

- * (PTM Master Time at t1') = t2' - ((t4 - t1) - (t3 - t2)) / 2
  - The timestamps t1, t4, t1', t4', propagation delay t3-t2 received in ResponseD message, are stored in the Controller's local "PTM Context Registers" and are readable by Client through APB interface.

    Further, the current PTM Master Time is computed using the expression:
- * (Current PTM Master Time) = (PTM Master Time at t1') + (Current PTM Local Timer) - (t1')
  - The "Current PTM Master Time" is output on PTM_MASTER_TIMER_OUT[63:0] and PTM_MASTER_TIMER_OUT_VALID is asserted for 1 core_clk period.

PTM Responder

The Controller functions as a PTM Responder when operating in Root Port mode. In this mode, the Controller can be programmed to automatically respond with PTM Response/ResponseD message when a PTM Request is received.

When a PTM Request is received, the Controller captures timestamp (t2/t2'). When the PTM Response/ResponseD is sent, the Controller captures timestamp (t3/t3'). The (t3-t2) value stored from the previous PTM dialog and the captured t2' are used in the PTM ResponseD message.

The timestamps t2, t3, t2', t3'are stored in the local "PTM Context Registers" and are readable by Client through APB interface.

12.3.2.2 PTM Timestamp Capture

All the timestamps captured by the Controller must be based on the STP Symbol or Token that frames the PTM Message TLP, as if observing the first bit of that Symbol or Token at the Port's pins.

However, it is practically not feasible to capture timestamp on the Port's pins that transmit and receive serial data. The Controller captures timestamps at an internal point in Tx and Rx datapaths and then adjusts the timestamp with the delay factors till the Port's pins. The Controller's internal datapath for PTM measurements is shown in figure below:

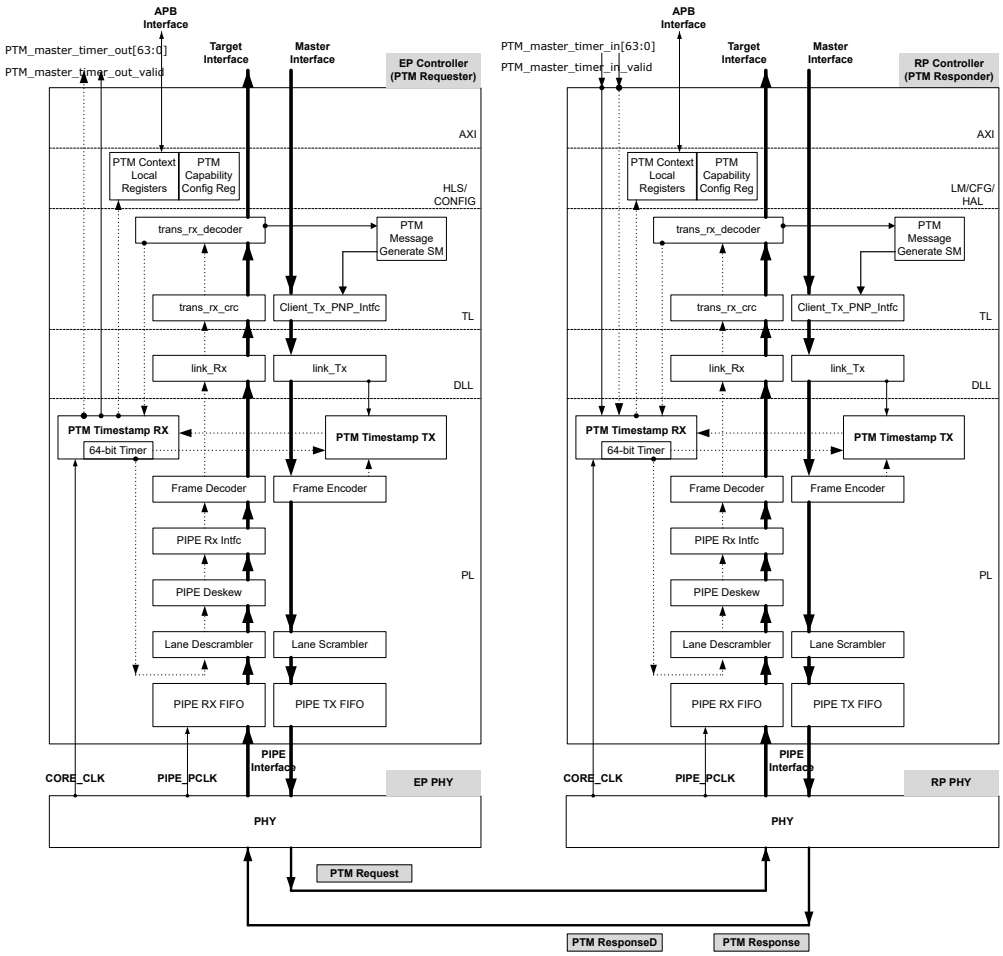


Figure 12.4: Controller Block Diagram of PTM Implementation

12.3.2.3 Accuracy of PTM Timestamps

As per the PCIe specification, the timestamps must be based on the STP Symbol or Token that frames the TLP, as if observing the first bit of that Symbol or Token at the Port's pins. However, this is not practically feasible and the Controller captures timestamp's at an internal point in the datapath as seen in previous sections. A static programmable latency value is used for the PHY. However, there may be dynamic variations in the receive latency in the Elastic Buffer. This is currently not accounted in the Controller and PHY design. As a result, the PTM measurements may be inaccurate to the extent shown in the following table.

Table 12.1: Accuracy of PTM timestamps

Clock Architecture	Maximum Inaccuracy
Common Clock (0 ppm)	0 ns (No Elastic Buffer Variations.)
SRNS (600 ppm)	• At 64 GT/s = +/- 0.5ns • At 32 GT/s = +/- 1ns • At 16 GT/s = +/- 2 ns • At 8 GT/s = +/- 4 ns • At 5.0 GT/s = +/- 8 ns • At 2.5 GT/s = +/- 16 ns
SRIS (5600 ppm)	• At 64 GT/s = +/- 1ns • At 32 GT/s = +/- 2ns • At 16 GT/s = +/- 4 ns • At 8 GT/s = +/- 8 ns • At 5.0 GT/s = +/- 16 ns • At 2.5 GT/s = +/- 32 ns

System Settings to Ensure Best Accuracy

The following system settings are recommended to minimize delay variations and to ensure maximum accuracy:

- Disable ASPM L0s, L1
  - Entry into L0s, L1 in between PTM dialogs can cause changes to the Rx Skew and Receive FIFO levels in the Controller.
- Link Operating at 16 GT/s
  - This ensures the devices are running at highest frequency and smallest possible variation in PTM measurements.
- Use Common Clock
  - Using common clock architecture, if feasible, ensures 0 ppm and 0 variations in the Receive Elastic Buffer.

12.3.2.4 Exception Handling

- Unsupported Requests
  - When Controller is configured as a Downstream Port, If a PTM Message is received when the PTM Enable bit is clear, the Message is treated as an Unsupported Request.
    - This error is reported in AER and Device Status registers as an Unsupported Request.
    - The PTM message is not forwarded to the Client Interface.
  - When Controller is configured as an Upstream Port, If a Properly formed PTM Response message is received when the PTM Enable bit is clear, then this message is silently discarded.
    - The PTM message is not forwarded to the Client Interface.

12.3.2.5 PTM Context Invalidation

PCIe specification, strongly recommends that an Upstream Port invalidate its internal PTM context when the relationship between PTM Master Time and the Upstream Port's local time changes, as determined by implementation specific criteria. For example, this may occur as a result of a transition to a non-D0 state or due to accumulated PPM drift.

Controller implements two modes for PTM Context invalidation when in Endpoint mode:

- Client Invalidate PTM Context:
  - Client Firmware can invalidate PTM Context by writing a '1' to the "Client Invalidate PTM Context" bit-27 of the " PTM Local Control Register".
  - Client can enable PTM dialogs by setting PTM Requester Enable bit in the "PTM Local Control Register" to get current PTM Context and PTM Master Time.
- Hardware auto-invalidate PTM Context:
  - Controller can automatically invalidate PTM Context when the LTSSM transitions out of L0 State.
  - This mode is enabled by default. This mode can be disabled by programming the "Disable Auto Invalidate PTM Context", bit-28 of the "PTM Local Control Register".
  - When PTM Context is automatically invalidated, the Controller notifies the Client by asserting the local_interrupt output. The "PTM Context Auto Invalidated" status bit-27 is also set in the "Local Error Status Register 0".

12.4 SRAM Integration

The interface signals associated with the RAMs used by the core are available at the top level of the design for easy interfacing to customer-generated RAM modules. This also enables users to add BIST capability to the RAMs without modifying any of the Cadence PCIe core design files.

Details of SRAM required including the clock relationship and read latency are avaialble at docs/cdnpcie_sram_requirements.html document

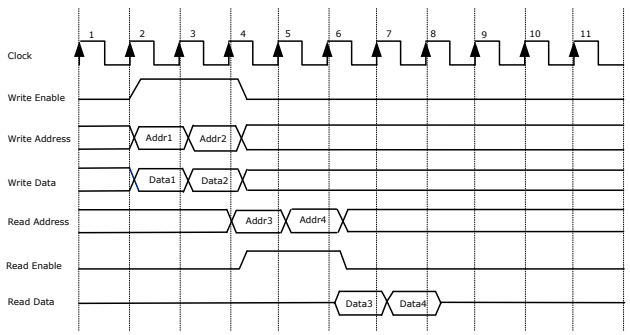
The Read and Write accesses issued by the Controller are described below:

RAM Read

- Controller drives Read Address and Read Enable for 1 clock and samples the Read Data after 2 cycles as shown

in the RAM Timing Diagram.At configuration time, 3 Cycle SRAM access latency can also be selected.

- Controller never issues Read access to an uninitialized RAM location.
- Controller could issue contiguous back to back RAM Read accesses.
- After writing a RAM location, the Controller could issue Read to the same location 1 cycle after the Write.
- Controller has internal mechanisms to prevent Read and Write access collision.
- Controller never issues Read and Write to the same RAM location simultaneously. Controller ensures this even when Read and Write clocks are Asynchronous.



RAM Write

- Controller drives Write Address, Write Data and Write Enable for 1 clock period.
- Controller could issue contiguous back to back RAM Write accesses.
- After reading a RAM location, the Controller could issue Write to the same location 1 cycle after the Read.
- Controller has internal mechanisms to prevent Read and Write access collision.
- Controller never issues Read and Write to the same RAM location simultaneously. Controller ensures this even when Read and Write clocks are Asynchronous.

Integration Note: The Controller could issue a Read immediately after a Write or a Write immediately after a Read to the same location. Hence it is important to ensure the Read and Write latencies are identical during the integration. As an example implementation, the Integrator can use a standard 1-cycle RAM Macro and add a register slice on all of the RAM inputs namely (address/write data/read enable/write enable). Write followed by read ordering has to be maintained in the RAM Implementation.

12.5 Vital Product Data (VPD) Integration

Vital Product Data (VPD) is information that uniquely identifies hardware and, potentially, software elements of a system. The VPD can provide the system with information on various Field Replaceable Units such as, part number, serial number, and other detailed information. The objective from a system point of view is to make this information available to the system owner and service personnel.

VPD resides in a storage device, for example, a serial EEPROM, in a PCI device. Access to the VPD is provided using the Capabilities List in Configuration Space

This VPD capability register is implemented per Physical Function and is present in the End Point.

Refer to the VPD Capability registers of the Register Description present in the `<package_path>/models/ipxact/*` directory of the delivery package. This is a configurable feature. This feature and registers are available only if selected in the IP configurator.

The following example shows how this is used.

12.5.1 VPD Read Example

1. The Host that is trying to read the VPD will send a CFG Write to the VPD capability structure with the following information:
    - VPD Flag = ZERO, VPD Address = VPD Address that needs to be read by the Host
2. The Controller IP on seeing a write to the VPD Address of the VPD capability structure will generate a VPD_INTERRUPT (part of *cap_state_pf_out* signal) single cycle pulse that indicates to the user application that the

VPD capability has new data.

- A pulse on PF0 group indicates a VPD event for PF 0, a pulse on PF1 group indicates a VPD event for PF 1, and so on.

3 On seeing the “VPD_INTERRUPT”, the local software running on the End Point must read the VPD Capability Register of the corresponding Function to determine Read/Write is being requested by the host.

- “VPD Flag” ZERO indicates a read is requested by the HOST.

4 The local software should read the VPD Flag, VPD Address using ip-mgmt interface

5 Once the local software has the data for the VPD read request, it should write the data to the VPD Data register in the VPD capability structure, using the ip-mgmt interface, and set the VPD Flag to ONE.

6 The Host Software will be monitoring the VPD Flag and, and once this is set to ONE, will use the VPD information from the VPD Data register.

12.5.2 VPD Write Example

1 The Host that is trying to write the VPD will send a CFG Write to the VPD capability structure with the following information:

a VPD Data = VPD Data to be written

b VPD Flag = ONE, VPD Address = VPD Address that needs to be written

2 The Controller IP on seeing a write to the VPD Address of the VPD capability structure will generate a VPD_INTERRUPT (part of c*ap_state_pf_out* signal) single cycle pulse that indicates to the user application that the VPD capability has new data.

- A pulse on PF0 group indicates such a VPD event for PF 0, a pulse on PF1 group indicates that a VPD event for PF 1, and so on.

3 On seeing the VPD_INTERRUPT, the local software running on the End Point must read the VPD Capability Register of the corresponding Function to determine Read/Write is being requested by the host.

- VPD Flag ONE indicates a write is requested by the HOST.

4 The local software should read the VPD Flag, VPD Address and VPD Data using the ip-mgmt interface.

5 Once the local software written the data for the VPD Write request, it should set the VPD Flag to ZERO Using the ip-mgmt interface.

6 The Host Software will be monitoring the VPD Flag and, and once this is set to ZERO, will now know that the VPD Write request is now complete.

12.6 Hot-Plug Integration

12.6.1 Overview

The PCI Express defines the mechanism for hot-plug support for downstream ports and provides a “toolbox” of mechanisms that allow different user/operator models to be supported using a self-consistent infrastructure. This feature is applicable to Root Complex and switch downstream ports.

The elements of Hot-Plug defined in PCIe Specification are described in the *Table 12.2*.

Table 12.2: Elements of Hot-Plug

Element	Purpose
Indicators	Show the power and attention state of the slot
Manually-operated Retention Latch (MRL)	Holds adapter in place
MRL Sensor	Allows the Port and system software to detect the MRL being opened
Electromechanical Interlock	Prevents removal of adapter from slot
Attention Button	Allows user to request hot-plug operations
Software User Interface	Allows user to request hot-plug operations

Table 12.2: Elements of Hot-Plug

Element	Purpose
Slot Numbering	Provides visual identification of slots
Power Controller	Software-controlled electronic component or components that control power to a slot or adapter and monitor that power for fault conditions

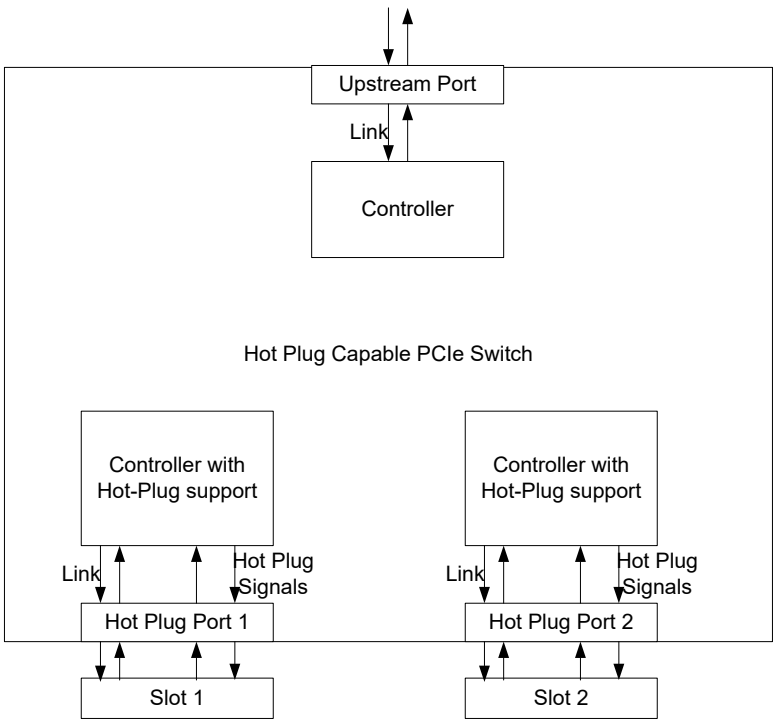


Figure 12.5: Conceptual view of a hot plug capable PCIe switch

Several interface signals are required to support the Hot-Plug elements as described by the PCIe Specification. The Interface signals required to connect the Controller to the Hot-Plug Slot are shown in *Figure 12.6*.

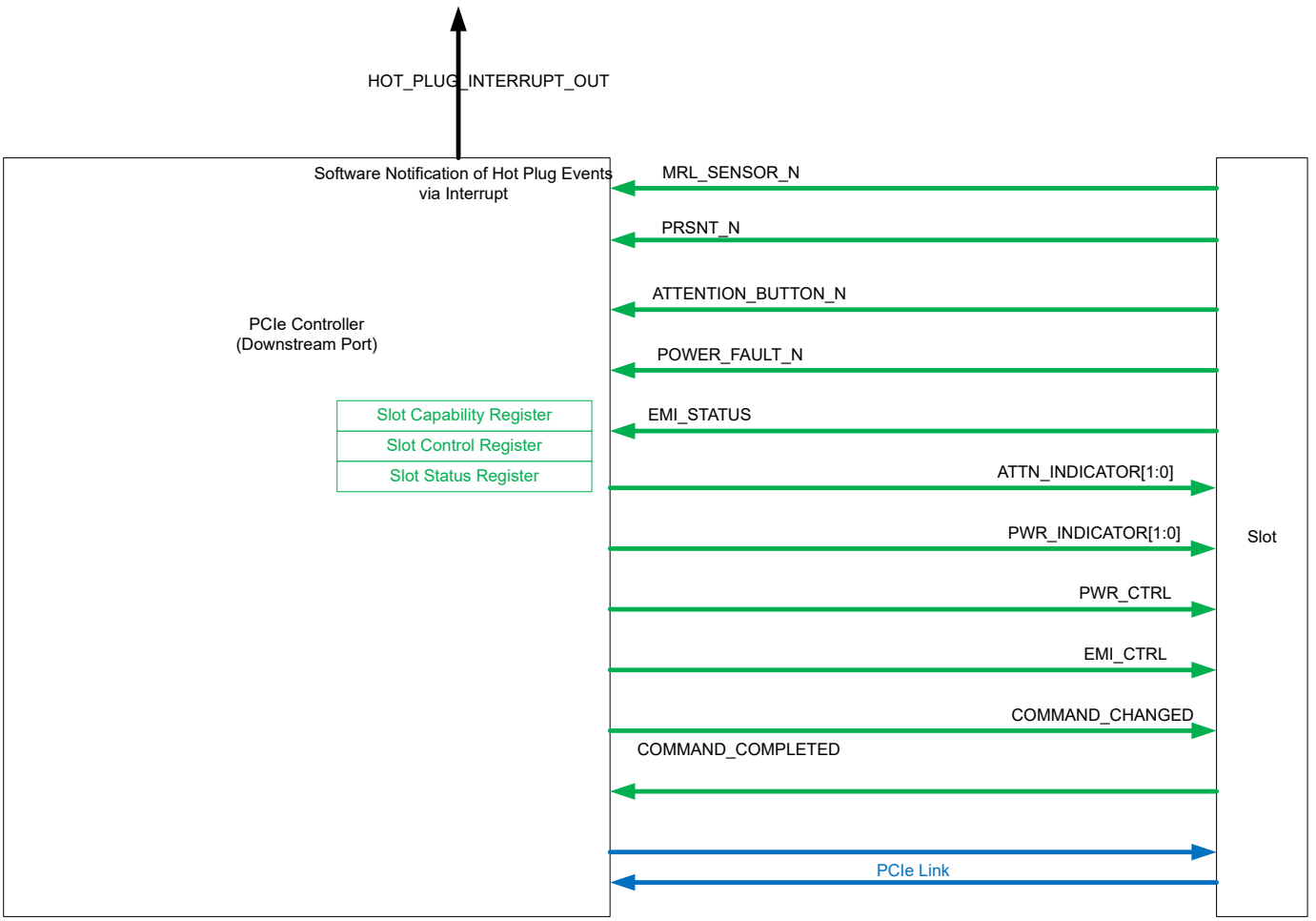


Figure 12.6: Hot plug interface block diagram

12.6.2 Signal Description

Table 12.3 shows the signals associated with the Hot Plug implementation.

Table 12.3: Hot-Plug interface inputs and outputs

Signal	Direction	Width	Description
hot_plug_mrl_sensor_n	Input	1	Indicates the status of the Manually-operated Retention Latch (MRL). The MRL Sensor is a switch, optical device, or other type of sensor that reports the position of a slot's MRL to the Downstream Port. The MRL Sensor reports closed when the MRL is fully closed and open at all other times (that is, if the MRL fully open or in an intermediate position). If MRL fully closed => 0 At all other times => 1
hot_plug_prsnt_n	Input	1	Active Low Input, indicating the presence of an adapter in the slot.

Table 12.3: Hot-Plug interface inputs and outputs

Signal	Direction	Width	Description
hot_plug_attention_button_n	Input	1	This bit is set when an Attention Button electrically controlled by the chassis is pressed. 0 => Attention Button is pressed 1 => Default state
hot_plug_power_fault_n	Input	1	Power fault indicator generated by external power fault detection logic. Active Low.
hot_plug_emi_status	Input	1	Electromechanical Interlock Status input.
hot_plug_command_changed	Output	1	1-bit output to indicate that a write to the Slot Control Register occurred and this resulted in one of the Hot Plug command output signal to change. This will be set high until COMMAND_COMPLETED input is set to ‘1’. If the “No Command Completed Support”=1 in Slot Capabilities Register, this Output can be ignored and will be tied low.
hot_plug_command_completed	Input	1	Command completed status input in response to COMMAND_CHANGED. Active high input, asserted for 1 clock period. If the “No Command Completed Support”=1 in Slot Capabilities Register, this Input can be tied to ‘0’.
hot_plug_hot_plug_interrupt_out	Output	1	Hot Plug Interrupt Output for Software Notification of Hot Plug events.
hot_plug_attn_indicator	Output	2	Attention Indicator Control Output. Defined encodings are: 00b: Reserved 01b: - On - Attention - Operational problem at this slot 10b: - Blink - Locate - Slot is being identified at the user’s request 11b: - Off - Normal - Normal operation

Table 12.3: Hot-Plug interface inputs and outputs

Signal	Direction	Width	Description
hot_plug_pwr_indicator	Output	2	Power Indicator Control Output: Defined encodings are: 00b: Reserved 01b: On - Power On - Insertion or removal of the adapter is not permitted. 10b: Blink - Power Transition - Hot-plug operation is in progress and insertion or removal of the adapter is not permitted. 11b: Off - Power Off - Insertion or removal of the adapter is permitted.
hot_plug_pwr_ctrl	Output	1	Sets the power state of the slot per the defined encodings. The defined encodings are: 0b: Power On 1b: Power Off
hot_plug_emi_ctrl	Output	1	Electromechanical Interlock Control Output. Defined encodings are: 0b: Electromechanical Interlock Disengage 1b: Electromechanical Interlock Engage

12.6.3 Cadence PCIe Controller Hot Plug Implementation

Figure 12.7 shows the Implementation of the Hot Plug feature in the Controller.

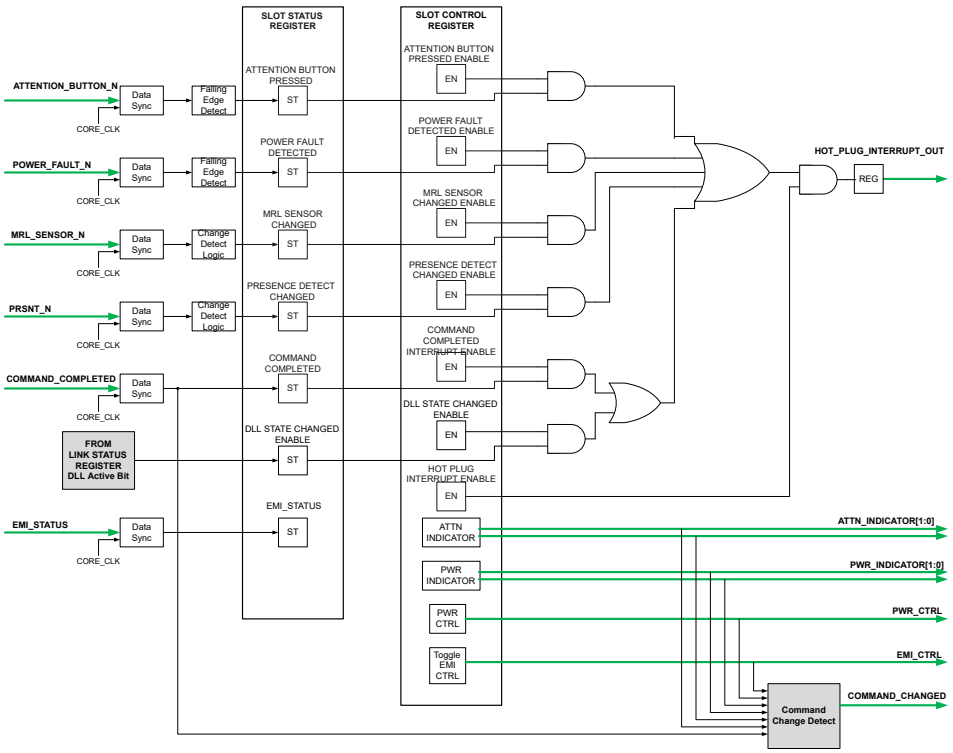


Figure 12.7: Controller Hot Plug implementation

12.6.3.1 Slot Status Register Implementation

Downstream Port with hot-plug capabilities supports the following hot-plug events:

- Slot Events:
  - **Attention Button Pressed** – Indicated by the hot_plug_attention_button_n input.
  - **Power Fault Detected** – Indicated by the hot_plug_power_fault_n input.
  - **MRL Sensor Changed** – Indicated by the hot_plug_mrl_sensor_n Input. A change in this input is detected by the Controller and captured in status register.
  - **Presence Detect Changed** – Indicated by the hot_plug_prsnt_n input. A change in this input is detected by the Controller and captured in status register.
- Command Completed Events
  - Indicated by hot_plug_command_completed input.
- Data Link Layer State Changed Events
  - Indicated by “Link Status Register” DLL Active Bit.

Each of these events has a status field, which indicates that an event has occurred but has not yet been processed by software, and an enable field, which indicates whether the event is enabled for software notification. Some events also have a capability field, which indicates whether the event type is supported on the Port.

12.6.3.2 Slot Control Register Implementation

The enable bits are used to enable interrupt generation for the various hot-plug events. The Control outputs are used to drive the outputs directly to the Slot as shown in *Figure 12.7*.

12.6.3.3 Software Notification of Hot-Plug Events

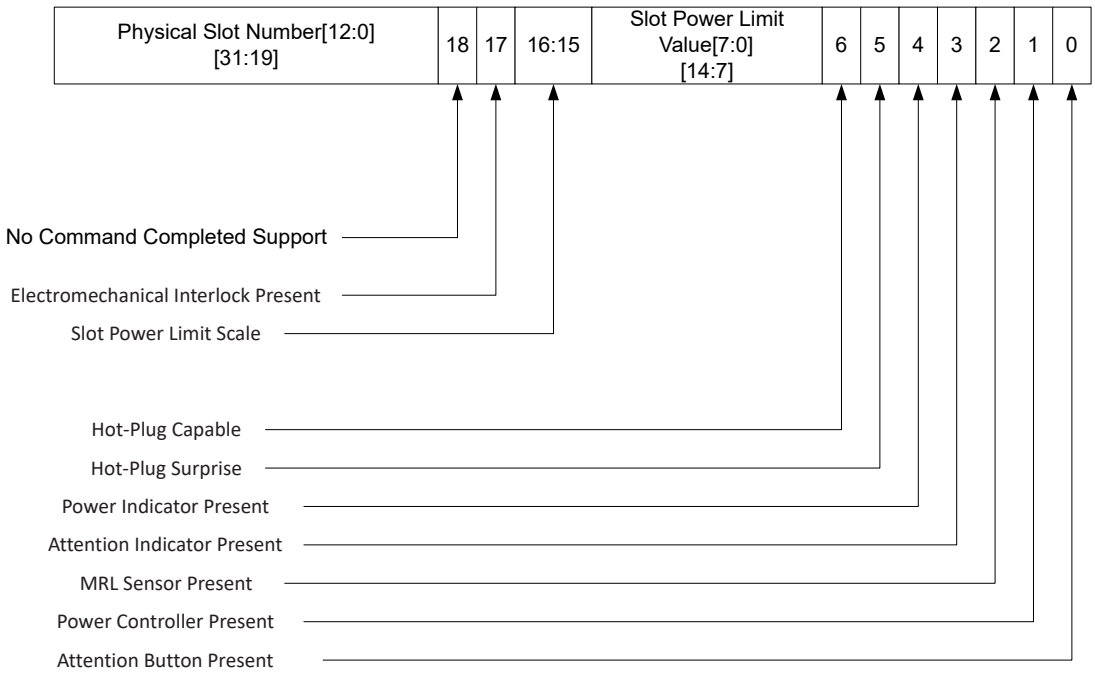
The Hot Plug Events are notified via the hot_plug_interrupt_out interrupt output. This output will remain asserted until the Status bits are cleared by writing a ‘1’ to the bits in the Slot Status Register.

12.6.4 Register Description

The Hot Plug Capability Structure is required ONLY for Downstream Ports.

12.6.4.1 Slot Capabilities Register

Address: 0xD4



Bit	Register Description	Attributes	Implementation
0	**Attention Button Present** – When Set, this bit indicates that an Attention Button for this slot is electrically controlled by the chassis	HwInit	No Logic. RW from APB Interface
1	**Power Controller Present** – When Set, this bit indicates that a software programmable Power Controller is implemented for this slot/adapter (depending on form factor).	HwInit	No Logic. RW from APB Interface
2	**MRL Sensor Present** – When Set, this bit indicates that an MRL Sensor is implemented on the chassis for this slot.	HwInit	No Logic. RW from APB Interface
3	**Attention Indicator Present** – When Set, this bit indicates that an Attention Indicator is electrically controlled by the chassis.	HwInit	No Logic. RW from APB Interface
4	**Power Indicator Present** – When Set, this bit indicates that a Power Indicator is electrically controlled by the chassis for this slot.	HwInit	No Logic. RW from APB Interface

Bit	Register Description	Attributes	Implementation
5	**Hot-Plug Surprise** – When Set, this bit indicates that an adapter present in this slot might be removed from the system without any prior notification. This is a form factor specific capability. This bit is an indication to the operating system to allow for such removal without impacting continued software operation.	HwInit	No Logic. RW from APB Interface
6	**Hot-Plug Capable** – When Set, this bit indicates that this slot is capable of supporting hot-plug operations.	HwInit	No Logic. RW from APB Interface
14:7	**Slot Power Limit Value** – In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by the slot (see Section 6.9) or by other means to the adapter. Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field except when the Slot Power Limit Scale field equals 00b (1.0x) and Slot Power Limit Value exceeds EFh, the following alternative encodings are used: F0h = 250 W Slot Power Limit F1h = 275 W Slot Power Limit F2h = 300 W Slot Power Limit F3h to FFh = Reserved for Slot Power Limit values above 300 W This register must be implemented if the Slot Implemented bit is Set. Writes to this register also cause the Port to send the Set_Slot_Power_Limit Message. The default value prior to hardware/firmware initialization is 0000 0000b.	HwInit	In Root Port mode, write into this register causes Controller to transmit a Set_Slot_Power_Limit Message.

Bit	Register Description	Attributes	Implementation
16:15	**Slot Power Limit Scale** – Specifies the scale used for the Slot Power Limit Value (see Section 6.9). Range of Values: 00b = 1.0x 01b = 0.1x 10b = 0.01x 11b = 0.001x This register must be implemented if the Slot Implemented bit is Set. Writes to this register also cause the Port to send the Set_Slot_Power_Limit Message. The default value prior to hardware/firmware initialization is 00b.	HwInit	In Root Port mode, write into this register causes Controller to transmit a Set_Slot_Power_Limit Message.
17	**Electromechanical Interlock Present** – When Set, this bit indicates that an Electromechanical Interlock is implemented on the chassis for this slot.	HwInit	No Logic. RW from APB Interface
18	**No Command Completed Support** – When Set, this bit indicates that this slot does not generate software notification when an issued command is completed by the Hot-Plug Controller. This bit is only permitted to be Set if the hot-plug capable Port is able to accept writes to all fields of the Slot Control register without delay between successive writes.	HwInit	No Logic. RW from APB Interface
31:19	**Physical Slot Number** – This field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is unique within the chassis, regardless of the form factor associated with the slot. This field must be initialized to zero for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.	HwInit	No Logic. RW from APB Interface

12.6.4.2 Slot Control Register

Address: 0xD8

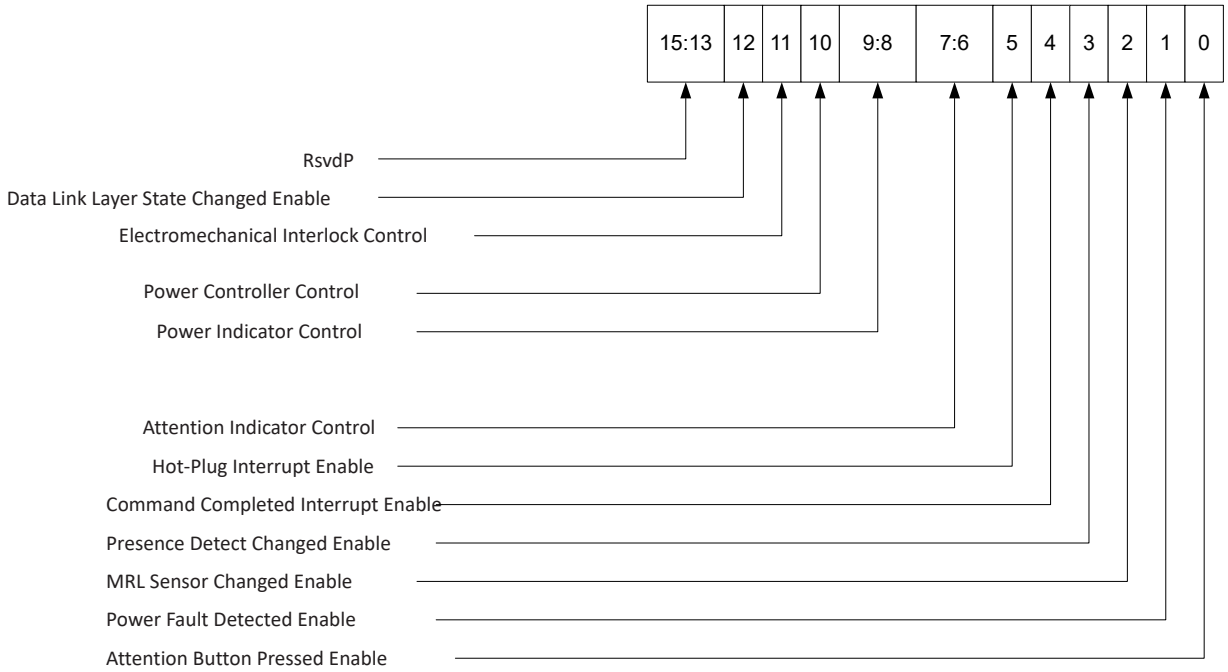


Table 12.4: Slot Control Register Description

Bit	Register Description	Attributes	Implementation
0	**Attention Button Pressed Enable** – When Set to 1b, this bit enables software notification on an attention button pressed event (see Section 6.7.3). If the Attention Button Present bit in the Slot Capabilities register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW	ANDed with **Attention Button Pressed Status** to generate Interrupt
1	**Power Fault Detected Enable** – When Set, this bit enables software notification on a power fault event (see Section 6.7.3). If a Power Controller that supports power fault detection is not implemented, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW	ANDed with **Power Fault Detected Status** to generate Interrupt

Table 12.4: Slot Control Register Description

Bit	Register Description	Attributes	Implementation
2	**MRL Sensor Changed Enable** – When Set, this bit enables software notification on a MRL sensor changed event (see Section 6.7.3). If the MRL Sensor Present bit in the Slot Capabilities register is Clear, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW	ANDed with **MRL Sensor Changed Status** to generate Interrupt
3	**Presence Detect Changed Enable** – When Set, this bit enables software notification on a presence detect changed event (see Section 6.7.3). If the Hot-Plug Capable bit in the Slot Capabilities register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW	ANDed with **Presence Detect Changed Status** to generate Interrupt
4	**Command Completed Interrupt Enable** – If Command Completed notification is supported (if the No Command Completed Support bit in the Slot Capabilities register is 0b), when Set, this bit enables software notification when a hot-plug command is completed by the Hot-Plug Controller. If Command Completed notification is not supported, this bit must be hardwired to 0b. Default value of this bit is 0b.	RW	ANDed with **Command Completed Status** to generate Interrupt
5	**Hot-Plug Interrupt Enable** – When Set, this bit enables generation of an interrupt on enabled hot-plug events. If the Hot Plug Capable bit in the Slot Capabilities register is Clear, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW	Master enable for all hot plug events interrupt generation

Table 12.4: Slot Control Register Description

Bit	Register Description	Attributes	Implementation
7:6	**Attention Indicator Control** – If an Attention Indicator is implemented, writes to this field set the Attention Indicator to the written state. Reads of this field must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write without waiting, if required to, for the previous command to complete in which case the read value is undefined. Defined encodings are: 00b Reserved 01b On 10b Blink 11b Off Note: The default value of this field must be one of the non-Reserved values. If the Attention Indicator Present bit in the Slot Capabilities register is 0b, this bit is permitted to be read-only with a value of 00b.	RW	Drives ATTN_INDICATOR[1:0] output

Table 12.4: Slot Control Register Description

Bit	Register Description	Attributes	Implementation
9:8	**Power Indicator Control** – If a Power Indicator is implemented, writes to this field set the Power Indicator to the written state. Reads of this field must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write without waiting, if required to, for the previous command to complete in which case the read value is undefined. Defined encodings are: 00b Reserved 01b On 10b Blink 11b Off Note: The default value of this field must be one of the non- Reserved values. If the Power Indicator Present bit in the Slot Capabilities register is 0b, this bit is permitted to be read-only with a value of 00b.	RW	Drives PWR_INDICATOR[1:0] output

Table 12.4: Slot Control Register Description

Bit	Register Description	Attributes	Implementation
10	**Power Controller Control** – If a Power Controller is implemented, this bit when written sets the power state of the slot per the defined encodings. Reads of this bit must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write, if required to, without waiting for the previous command to complete in which case the read value is undefined. Note that in some cases the power controller may autonomously remove slot power or not respond to a power-up request based on a detected fault condition, independent of the Power Controller Control setting. The defined encodings are: 0b Power On 1b Power Off If the Power Controller Present bit in the Slot Capabilities register is Clear, then writes to this bit have no effect and the read value of this bit is undefined.	RW	Drives PWR_CTRL output
11	**Electromechanical Interlock Control** – If an Electromechanical Interlock is implemented, a write of 1b to this bit causes the state of the interlock to toggle. A write of 0b to this bit has no effect. A read of this bit always returns a 0b.	RW	Drives EMI_CTRL output

Table 12.4: Slot Control Register Description

Bit	Register Description	Attributes	Implementation
12	**Data Link Layer State Changed Enable** – If the Data Link Layer Link Active Reporting capability is 1b, this bit enables software notification when Data Link Layer Link Active bit is changed. If the Data Link Layer Link Active Reporting Capable bit is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW	ANDed with **DLL State Changed Status** to generate Interrupt

12.6.4.3 Slot Status Register

Address: 0xDA

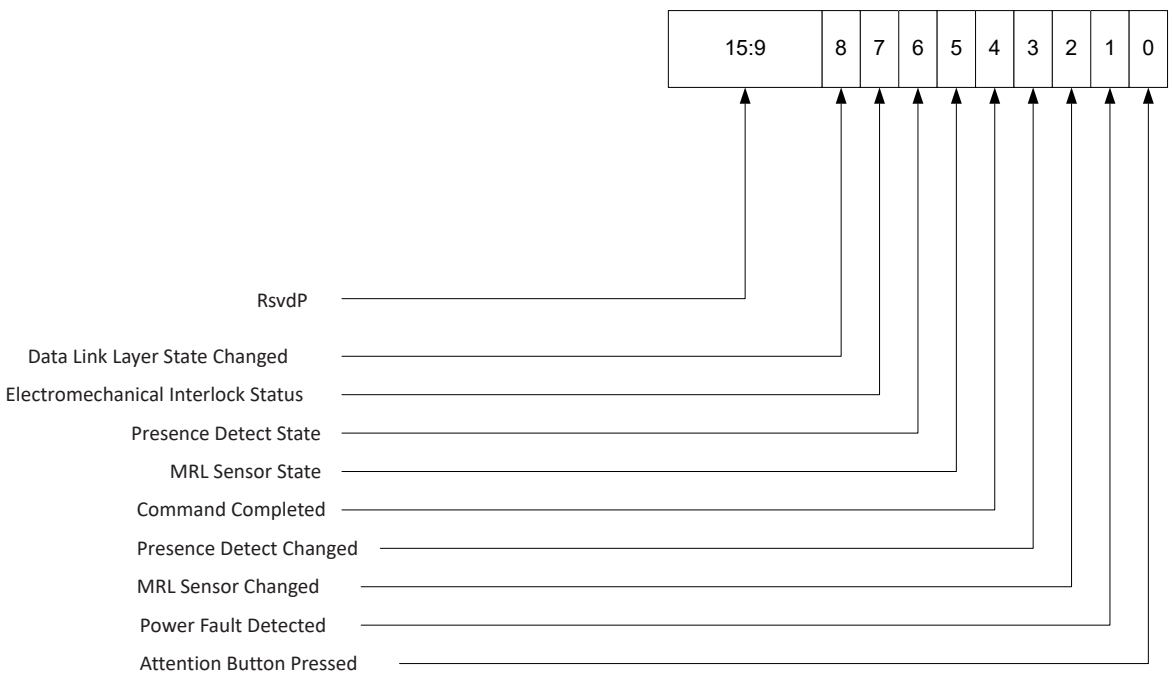


Table 12.5: Slot Status Register Description

Bit	Register Description	Attributes	Implementation
0	**Attention Button Pressed** – If an Attention Button is implemented, this bit is Set when the attention button is pressed. If an Attention Button is not supported, this bit must not be Set.	RW1C	Driven by the ATTENTION_BUTTON_N input. This bit is set to ‘1’ when ATTENTION_BUTTON_N changes from 1->0. Cleared when a ‘1’ is written into this bit.

Table 12.5: Slot Status Register Description

Bit	Register Description	Attributes	Implementation
1	**Power Fault Detected** – If a Power Controller that supports power fault detection is implemented, this bit is Set when the Power Controller detects a power fault at this slot. Note that, depending on hardware capability, it is possible that a power fault can be detected at any time, independent of the Power Controller Control setting or the occupancy of the slot. If power fault detection is not supported, this bit must not be Set.	RW1C	Driven by the POWER_FAULT_N input. This bit is set to ‘1’ when POWER_FAULT_N changes from 1->0. Cleared when a ‘1’ is written into this bit.
2	**MRL Sensor Changed** – If an MRL sensor is implemented, this bit is Set when a MRL Sensor state change is detected. If an MRL sensor is not implemented, this bit must not be Set.	RW1C	Driven by the MRL_SENSOR_N input. This bit is set to ‘1’ when MRL_SENSOR _N changes from 1->0 OR 0->1. Cleared when a ‘1’ is written into this bit.
3	**Presence Detect Changed** – This bit is set when the value reported in the Presence Detect State bit is changed.	RW1C	Driven by the PRSNT_N input. This bit is set to ‘1’ when PRSNT _N changes from 1->0 OR 0->1. Cleared when a ‘1’ is written into this bit.
4	**Command Completed** – If Command Completed notification is supported (if the No Command Completed Support bit in the Slot Capabilities register is 0b), this bit is Set when a hot-plug command has completed and the Hot-Plug Controller is ready to accept a subsequent command. The Command Completed status bit is Set as an indication to host software that the Hot-Plug Controller has processed the previous command and is ready to receive the next command; it provides no guarantee that the action corresponding to the command is complete. If Command Completed notification is not supported, this bit must be hardwired to 0b.	RW1C	Driven by the COMMAND_COMPLETED input. This bit is set to ‘1’ on the COMMAND_COMPLETED pulse. Cleared when a ‘1’ is written into this bit.

Table 12.5: Slot Status Register Description

Bit	Register Description	Attributes	Implementation
5	**MRL Sensor State** – This bit reports the status of the MRL sensor if implemented. Defined encodings are: 0b MRL Closed 1b MRL Open	RO	Connected to MRL_SENSOR_N input.
6	**Presence Detect State** – This bit indicates the presence of an adapter in the slot, reflected by the logical “OR” of the Physical Layer in-band presence detect mechanism and, if present, any out-of-band presence detect mechanism defined for the slot’s corresponding form factor. Note that the in-band presence detect mechanism requires that power be applied to an adapter for its presence to be detected. Consequently, form factors that require a power controller for hot-plug must implement a physical pin presence detect mechanism. Defined encodings are: 0b Slot Empty 1b Card Present in slot This bit must be implemented on all Downstream Ports that implement slots. For Downstream Ports not connected to slots (where the Slot Implemented bit of the PCI Express Capabilities register is 0b), this bit must be hardwired to 1b.	RO	Connected to ~PRSNT_N input.
7	**Electromechanical Interlock Status** – If an Electromechanical Interlock is implemented, this bit indicates the status of the Electromechanical Interlock. Defined encodings are: 0b Electromechanical Interlock Disengaged 1b Electromechanical Interlock Engaged	RO	Connected to EMI_STATUS input

Table 12.5: Slot Status Register Description

Bit	Register Description	Attributes	Implementation
8	**Data Link Layer State Changed** – This bit is Set when the value reported in the Data Link Layer Link Active bit of the Link Status register is changed. In response to a Data Link Layer State Changed event, software must read the Data Link Layer Link Active bit of the Link Status register to determine if the Link is active before initiating configuration cycles to the hot plugged device.	RO	Connected to Link Status Register DLL Active Bit

12.7 PF/VF Encoding at IP Interfaces

12.7.1 HLS PF/VF Encoding

PF/VF (ipfivf) encoding is used at HLS metadata and other controller interfaces.

SR_IOV stride is set to 1 internally in the IP. The number of VFs attached to a PF can be modified using the local management register dynamically before the link training. The ipfivf format gives the VF number in the attached PF and the associated PF number.

PF is also encoded in the same format with a bit to indicate whether the encoding is for a VF or for PF .

The table shows examples of ipfivf encoding with different VF allocations to PF.

Table 12.6: Acronym and Description

Acronym	Description
RID	BDF formatted ID as in PCIe TLP
ipfivf	Format with VF number in the associated PF
total_vf	Total VF allocated to a PF, programmable with LM register. This is an **SRIOV** read-only capability field.
num_vf	enabled number of VF count in **SRIOV** capability programmed by host.
vf_offset	RO field in **SRIOV** capability. (*vf_offset* + RID of the PF) gives the RID of the first VF in that PF. when *vf_stride* is not one, vf_offset is same of vf_stride. When *vf_Stride* is 1, *vf_offset*is (*num_pfs* + (total_vfs-1) of all the previous PFs.
vf_Stride	RO field in the **SRIOV** capability. VF's *RID* + *vf_Stride* gives the next VF's RID in a PF.

Colour code		
Active VF	Invalid VF	Disabled/Unused VF instance

Figure 12.8: Color code

12.7.1.1 Example with Uniform Number of Total VFs

There are 4 VFs in each PFs, the host had enabled only Two Vfs in PF0.

Total PFs: 2, Total VFS:4 per PF, PF0.num_vf=2, PF1.num_vf=4: vf_stride=1			
RID	ipfivf		
BDF format	Associated PF	VF number in PF	Is_VF
0	0	-	0
1	1	-	0
2	0	0	1
3	0	1	1
4	***0***	***2***	***1***
5	***0***	***3***	***1***
6	1	0	1
7	1	1	1
8	1	2	1
9	1	3	1

Note:

In the above tables, the ***highlighted entries*** have Disabled/Unused VF instance.

12.7.1.2 Example with Non-uniform Number of Total VFs

There are 2 VFs in PF0 and 6 VFs are available PF1. The host had enabled only Four VFs in PF1 and only one VF in PF0.

Total PFs: 2, Pf0.total VF = 2, PF1.total VF=6, PF0.num_vf=1, PF1.num_vf=4, stride=1			
RID	ipfivf		
BDF format	Associated PF	VF number in PF	Is_VF
0	0	-	0
1	1	-	0
2	0	0	1
3	***0***	***1***	***1***
4	1	0	1
5	1	1	1
6	1	2	1
7	1	3	1
8	***1***	***4***	***1***
9	***1***	***5***	***1***

Note:

- In the above tables, the ***highlighted rows*** have Disabled/Unused VF instance.
- In the above table, the rows ***with **** have Invalid VF.

12.7.1.3 HLS Multi-VC Handling

VC Activation Deactivation and HLS Interface

A VC becomes active when the flow control handshake for that VC is complete. The VC can be disabled by clearing the VC enable in the VC Capability Register. It is not expected for the host to clear the VC when pending transactions are present in that VC.

The controller activates inbound HLS interface only when any one of the associated VCs becomes active. At the outbound, the controller responds to the activation request only when one of the attached VCs becomes active. After HLS becomes active, the controller deactivates the inbound HLS bus when:

- All the VCs in that client interface is inactive
- *rx_pipeline* is empty
- *deact_hint* is asserted by the client logic.

At the outbound HLS, the controller asserts deact_hint when all the VCs associated with the client interface becomes inactive. There is an IP register available to change this behavior. The hal_control_hls field select whether to STOP HLS bus upon VC deactivation.

Multi VC Integration Details

No special integration step is required if only one VC is required for Application.

Inbound Datapath Integration

If configuration has only one client interface, all VC's TLPs are transferred in order to application logic. Application logic is not expected to block a particular VC's traffic. If Application use case expects unblocked paths for every VCs, then multiple client interface configuration needs to be selected.

Outbound Integration

If multiple VCs are mapped to one client interface, Application logic must check Available VC credits before transmitting TLP. This is to prevent head of the line blocking in that client interface. crd_lmt_to_client* signals helps to implement credit check logic at Application side. Not implementing this logic at application side can cause temporary blockage til credits are available.

Chapter 13: Programming Guide

13.1 Programming IP Registers

The Controller IP registers can be programmed through AXI-Lite/APB Interface after the following Controller resets are de-asserted:

- *core_rst_n*
- *reg_rst_n*
- *reg_sticky_rst_n*
- *apb_rst_n*.

Note:

- The LTSSM will be in *Detect.Quiet* state and will start Link Training process, after the controller resets are released.
- All the register programming must be complete before the link training.
- It is recommended to keep the *link_training_enable* input of the Controller to 0 and complete the **IP Register** programming.
- After the **IP Register** programming is complete, wait for *ip_REg_init_in_progress* to get de-asserted if asserted already.
- LTSSM will remain in *Detect.Quiet* as long as the *link_training_enable* input is 0.
- *link_training_enable* should be set to 1 after the programming to allow link training to begin.

13.2 Software Driver Setup

The software driver delivery package includes the following documentation which can be refered for driver usage.

Table 13.1: Documentation Deliverables

Category	Item Detail
Driver Porting Guide	Explanation on how to use the drivers on different platforms
Driver Quick Start Guide	Steps on how to start using the driver, with references to the driver documents
IP Driver Guide	The User Guide for the driver (with API reference generated from Doxygen)

13.3 Link Equalization Setup

13.3.1 Gen3 Link Equalization Setup in RP Mode

Register Name: Lane Equalization Control Register.

(i_PCIE_sec_ext_cap_struct/i_lane_equalization_control_reg_[0,1,2,3,4,5,6,7])

Register Offset: 0x000030C, 0x0000310, 0x0000314, 0x0000318, 0x000031C, 0x0000320, 0x0000324, 0x0000328

Register Write Data: 0x00000000* (*Check Description for Write Data)

Description: In RP Mode, the Lane Equalization Control Register contains the following fields:

- Downstream Port 8.0 GT/s Transmitter Preset - Default Value: 1111b
- Downstream Port 8.0 GT/s Receiver Preset Hint - Default Value: 111b
- Upstream Port 8.0 GT/s Transmitter Preset - Default Value: 1111b
- Upstream Port 8.0 GT/s Receiver Preset Hint - Default Value: 111b

The default values of these presets are Invalid as per PCIe Specification. The Client Application FW must program Valid values into these fields prior to Link Training.

- Valid values for 8.0 GT/s TX Preset are: 0000b through 1010b.

- Valid values for 8.0 GT/s RX Preset Hint are: 000b through 110b.

The Values to be programmed are application specific and need to be determined by the Client.

Note: Link Training to Gen3 will be unsuccessful if this register is not initialized in RP.

13.3.2 Gen4 Link Equalization Setup in RP Mode

Register Name: PL 16GTs Lane Equalization Control Register. (i_regf_pl_16gts_cap/ i_pl_16gts_lane_equalization_control_reg_[0,1,2,3])

Register Offset: 0x00009E0, 0x00009E4, 0x00009E8, 0x00009EC

Register Write Data: 0x00000000* (*Check Description for Write Data)

Description: In RP Mode, the PL 16GTs Lane Equalization Control Register contains the following fields:

- Downstream Port 16.0 GT/s Transmitter Preset - Default Value: 1111b
- Upstream Port 16.0 GT/s Transmitter Preset - Default Value: 1111b

The default values of these presets are Invalid as per PCIe Specification. The Client Application FW must program Valid values into these fields prior to Link Training.

- Valid values for 16.0 GT/s TX Preset are: 0000b through 1010b.

The Values to be programmed are application specific and need to be determined by the Client.

Note: Link Training to Gen4 will be unsuccessful if this register is not initialized in RP.

13.3.3 Gen5 Link Equalization Setup in RP Mode

Register Name: PL 32GTs Lane Equalization Control Register. (i_regf_pl_32gts_cap/ i_pl_32gts_lane_equalization_control_reg_[0,1,2,3])

Register Offset: 0x0000A60, 0x0000A64, 0x0000A68, 0x0000A6C

Register Write Data: 0x00000000* (*Check Description for Write Data)

Description: In RP Mode, the PL 32GTs Lane Equalization Control Register contains the following fields:

- Downstream Port 32.0 GT/s Transmitter Preset - Default Value: 1111b
- Upstream Port 32.0 GT/s Transmitter Preset - Default Value: 1111b

The default values of these presets are Invalid as per PCIe Specification. The Client Application FW must program Valid values into these fields prior to Link Training.

- Valid values for 32.0 GT/s TX Preset are: 0000b through 1010b.

The Values to be programmed are application specific and need to be determined by the Client.

Note: Link Training to Gen5 will be unsuccessful if this register is not initialized in RP.

13.3.4 Gen3 Link Equalization Setup in EP Mode

Register Name: Gen3 Link Equalization Control Register 1. (i_regf_ip_pl/i_g3_eq_ctrl_1)

Register Offset: 0x1000478

Register Write Data: 0x00000410

Note: This setup is required to operate with CDNS PHY.

Description: Program “P4” as the preset request during Gen3 Equalization Phase2.

13.3.5 Gen4 Link Equalization Setup in EP Mode

Register Name: Gen4 Link Equalization Control Register 1. (i_regf_ip_pl/i_g4_eq_ctrl_1)

Register Offset: 0x100048C

Register Write Data: 0x00000430

Note: This setup is required to operate with CDNS PHY.

Description: Program “P4” as the preset request during Speed change to Gen4 and during Gen4 Equalization Phase2

13.3.6 Gen5 Link Equalization Setup in EP Mode

Register Name: Gen5 Link Equalization Control Register 1. (i_regf_ip_pl/i_g5_eq_ctrl_1)

Register Offset: 0x10004A0

Register Write Data: 0x00000630

Note: This setup is required to operate with CDNS PHY.

Description: Program “P6” as the preset request during Speed change to Gen5 and during Gen5 Equalization Phase2

13.4 Error Handling

13.4.1 Receive Side Errors (Inbound)

Table 13.2: Detected errors and related action

Error description	AER	Packet/Data path
NPRT Table ram read error	Local interrupt with parity error indication	Requires fundamental reset on controller
MSG Tlp with invalid msg_code	UR error is flagged in AER	TLP is discarded Possible to disable this behavior with LM *register hal_control[0]*
Receiving PTM when capability is not supported.	UR error is flagged in AER	TLP is discarded
PTM MSG in Rootport, and PTM capability is supported but disabled.	UR error is flagged in AER	TLP is discarded
PTM MSG in endpoint, PTM capability is supported but disabled.	Silent discard	TLP is discarded
Root port receiving LTR when LTR is disabled	UR error is flagged in AER	TLP Discarded
Rootport receiving OBFF	UR error is flagged in AER	TLP Discarded
EP receiving OBFF when OBFF enabled is not 01 or 10	UR error is flagged in AER	TLP Discarded
TLP targets an overlapped BAR	UR and LM registers to indicate BAR overlap is flagged	TLP Discarded
TLP not hitting any BARs	UR error is flagged in AER	TLP Discarded
WR TLP targeting EROM	UR error is flagged in AER	TLP Discarded
MEM request targets a function which is in non-D0 state	UR error is flagged in AER	TLP Discarded
TLp with non PASID or EXT_STEERING tag prefix	UR error is flagged in AER	TLP Discarded
TLP having PASID received while pasid not enabled	UR error is flagged in AER	TLP Discarded
TLP having extended steering tag while extended steering tag is not enabled	UR error is flagged in AER	TLP Discarded
Poisoned Atomic	Uncorrectable error is flagged in AER	TLP Discarded. Generates UR CPl.
Poisoned VDM message with data	Advisory error is flagged in AER	TLP Discarded

Table 13.2: Detected errors and related action

Error description	AER	Packet/Data path
Poisoned Non-Atomic/VDM	Advisory error/uncorrectable error based on the LM register **rx_poisoned_tlp_as_advisory** setting	TLP Discarded.
Poisoned TLP without data	Poison bit is ignored for TLPs without data	
Request targeting function undergoing FLR	No errors	TLP silent discard
MSG with route by ID and EP, received with incorrect Bus number and device number	No errors Bus and device number are not checked in this case	TLP not discarded.
MSG with route by ID and EP, received with un implemented function number	UR error is flagged in AER	TLP Discarded
Request Targeting MEM or IO for PF/VF while it is not enabled in endpoint	UR error is flagged in AER	TLP Discarded
Completions targeting a function in non-D0	Unexpected completion error	TLP Discarded. Detection this error as unexpected completion can be changed by LM register *hal_control[2]*
Completions targeting a function under FLR	Unexpected completion error/silent discard based on LM register *hal_conrol[3]*	TLP discarded. Detection of this error as unexpected can be changed by LM register *hal_control[3]*
CPL with invalid tag	Unexpected completion error is flagged in AER	TLP discarded
CPL with requester id mismatch	Unexpected completion error is flagged in AER	TLP Transferred to client interface with complete bit and error-code set. tag information is discarded.
Poisoned Completions	Poisoned completion is flagged in AER	TLP packet is discarded. tag information is also discarded
Completion Timeout	Completion timeout is reported in AER Request header's lower address, tag, ATR, TC and type fields are logged	Completion TLP is generated at HLS_IB with completion timeout flag set in Metadata
Memory request Completions with Byte count, Low Address Issues.	Unexpected completion error is flagged in AER	TLP is discarded by default. Possible to use **ip_management** register to transfer this TP to HLS interface with appropriate *error_code* in metadata set.

On the inbound path, the following errors default discard behavior can be changed using the IP Register **hal_ctrl_decoded_tlp_filter**:

- Poisoned advisory error
- Poisoned uncorrectable error
- Unsupported posted
- Packet which are marked for silent discard due to certain conditions.
- Completions with invalid tag
- Unexpected completions with valid tag
- Poisoned completions.

The following errors are marked as function non-specific errors and are logged in all PFs:

- TLP not matching any BARs
- Completions with RID field mismatch.

13.4.2 Malformed Errors

Table 13.3: Malformed errors

Error description	AER	Credit update
ATS is selected and AT=1 MEM Rd has length=1 or greater than RCB settings	Malformed error	Packet is discarded without writing to Receive buffers. Credits are not returned.
RRS (CRS/MRS) Completion received for non CFG and DMWR requests	Malformed error	Credits are returned for this error. TLP is discarded by default.
Non-Memory Completion with matching ID but unexpected TC, Attr or TC or Byte count or low address	Malformed error	Credits are returned for this error. TLP is discarded by default.
Memory Completion with matching ID but unmatched TC, Attr from request	Malformed error	Credits are returned for this error. TLP is discarded by default.

13.4.3 HLS Outbound Errors

Table 13.4: HLS outbound errors

Error description	Action by HLS	Packet dropped (Yes/No)
OB completion with CA status	NP request header from metadata is logged in AER	No
TLP with poisoned bit set	Poisoned TLP send to PCIe link	No
TLP with incorrect credit type enters a client interface	reported in LM register **aue_status**	No. Client logic is not expected to create this error. This might cause hang in the system.
TLP with wrong TC setting which belongs to different Client interface	reported in LM register **aue_status**	No. Client logic is not expected to create this error. This might cause hang in the system.
TLP size more than Max payload size / Max read request size	reported in LM register **aue_status**	No. Client logic is not expected to create this error. This might cause hang in the system.
PCIe tag value more than supported	reported in LM register **aue_status**	No. Client logic is not expected to create this error. This might cause hang in the system.
PCIe tag value is already in use	reported in LM register **aue_status**	No. Client logic is not expected to create this error. This might cause hang in the system.

Chapter 14: Debugging Guide

14.1 Link Training Debug

14.1.1 Link Training Status Registers

To check the status of the link training parameters, the following registers can be read:

14.1.1.1 Physical Layer Debug Status Register 0

i_regf_ip_pl/i_dbg_sts_0, Register Offset: 0x1000420

Includes Physical Layer Functional Status.

Table 14.1: Physical Layer Debug Status Register 0

Bit Location	Register Field	Attr	Description	Reset
0	Link Status	RO	Current state of link 1 = link training complete, 0= link training not complete	0
3:1	Reserved			
6:4	Negotiated Lane Count	RO	Lane count negotiated with other side during link training 000 = x1, 001 = x2, 010 = x4, 011 = x8 100 = x16	0
9:7	Negotiated Speed	RO	Current operating speed of link 000 = 2.5G, 001 = 5G, 010 = 8G, 011 = 16G 100=32G	0
11:10	Reserved			
19:12	Received Link ID	RO	Link ID received from other side during link training.	0
27:20	LTSSM State	RO	Current state of the LTSSM. The encoding of the states is given in Appendix C.	0

Table 14.1: Physical Layer Debug Status Register 0

Bit Location	Register Field	Attr	Description	Reset
28	Remote Linkwidth Upconfigure Capability Status	RO	A 1 in this field indicates that the remote node advertised Linkwidth Upconfigure Capability in the training sequences in the Configuration.Complete state when the link came up. A 0 indicates that the remote node did not set the Link Upconfigure bit.	0
31:29	Reserved			

14.1.1.2 Physical Layer Debug Status Register 1

i_regf_ip_pl/i_dbg_sts_1, Register Offset: 0x1000424

Includes Physical Layer Functional Status.

Table 14.2: Physical Layer Debug Status Register 1

Bit Location	Register Field	Attr	Description	Reset
15:0	Negotiated Lane Map	RO	Bit i of this field is set to 1 at the end of link training if Lane i is part of the PCIe link. The value of this field is valid only when the link is in L0 or L0s states.	0
16	Lane Reversal Status	RO	This bit set by the Controller at the end of link training if the LTSSM had to reverse the lane numbers to form the link.	0
31:17	Reserved			

14.1.1.3 Physical Layer Debug Status Register 2

i_regf_ip_pl/i_dbg_sts_2, Register Offset: 0x1000428

Received FTS Count Status Register.

Table 14.3: Physical Layer Debug Status Register 2

Bit Location	Register Field	Attr	Description	Reset

Table 14.3: Physical Layer Debug Status Register 2

7:0	Received FTS Count for Gen1	RO	FTS count received from the other side during link training for use at the 2.5 GT/s link speed. The Controller transmits this many FTS sequences while exiting the L0S state, when operating at the 2.5 GT/s speed.	0x0
15:8	Received FTS Count for Gen2	RO	FTS count received from the other side during link training for use at the 5.0 GT/s link speed. The Controller transmits this many FTS sequences while exiting the L0S state, when operating at the 5.0 GT/s speed.	0x0
23:16	Received FTS Count for Gen3	RO	FTS count received from the other side during link training for use at the 8.0 GT/s link speed. The Controller transmits this many FTS sequences while exiting the L0S state, when operating at the 8.0 GT/s speed.	0x0
31:24	Received FTS Count for Gen4	RO	FTS count received from the other side during link training for use at the 16.0 GT/s link speed. The Controller transmits this many FTS sequences while exiting the L0S state, when operating at the 16.0 GT/s speed.	0x0

14.1.1.4 Physical Layer Debug Status Register 3

i_regf_ip_pl/i_dbg_sts_3, Register Offset: 0x100042C

Received FTS Count Status Register.

Table 14.4: Physical Layer Debug Status Register 3

Bit Location	Register Field	Attr	Description	Reset

Table 14.4: Physical Layer Debug Status Register 3

7:0	Received FTS Count for Gen5	RO	FTS count received from the other side during link training for use at the 32.0 GT/s link speed. The Controller transmits this many FTS sequences while exiting the L0S state, when operating at the 32.0 GT/s speed.	0x0

14.1.2 Link Error Status Registers

If errors are seen on the link, the following registers can be read to check the nature of errors detected by the Controller:

14.1.2.1 Physical Layer Debug Status Register 4

i_regf_ip_pl/i_dbg_sts_4, Register Offset: 0x1000430

Includes Physical Layer Error Status.

Table 14.5: Physical Layer Debug Status Register 4

Bit Location	Register Field	Attr	Description	Reset
0	TLP PHY Error Status	RO	This bit indicates that a PHY Error was detected on the PIPE_RX_STATUS within a TLP. Write a 1 to clear this field.	
1	OS Block After Skip OS	RO	This bit indicates that an Ordered Set BLock was received immediately after a SKIP OS. This is a framing error. Write a 1 to clear this field.	
2	Illegal OS Block After EDS	RO	The Valid OS blocks after an EDS are EIOS, EIEOS and SKP. If any other OS blocks are received after EDS, then it is a framing error and this bit is asserted.	
3	Data Block After EDS	RO	This bit is set if a Data Block is received after an EDS. Write a 1 to clear this error.	
4	OS Block Received without EDS	RO	This bit is set if an Ordered Set Block is received without an EDS. This is a framing error. Write a 1 to clear this error.	

Table 14.5: Physical Layer Debug Status Register 4

Bit Location	Register Field	Attr	Description	Reset
5	Gen3 Framing Error Detected	RO	This bit is set if a framing error is detected in Gen3. Example, if an invalid token is received in a data stream, this error is flagged. Write a 1 to clear this error.	
6	OS Block Received After SDS	RO	This bit is set if an SDS is received after an SDS. This is a framing error. Write a 1 to clear this error.	
7	Invalid Sync Header Error	RO	This bit is set if an invalid Sync Header is detected. 00 and 11 are Invalid Sync Headers. Write a 1 to clear this error.	
8	Loss of Block Alignment Error	RO	This bit is set if the PHY Loses Block Alignment during data stream. This is detected based upon an unexpected PIPE_RX_VALID input deassertion during data stream. Write a 1 to clear this error.	
9	Deskew FIFO Overflow Error	RO	The Controller implements a 80 symbol FIFO for deskew across lanes. After Deskew is initially done, there may be ppm variations due to jitter across lanes. If the skew + ppm variation exceeds 80 symbol limit, then the Deskew FIFO overflow occurs. The overflow status is captured in this register. Write a 1 to clear this error.	

14.1.2.2 AER i_uncorr_err_status

i_AER_cap_struct/i_uncorr_err_status, Register Offset: 0x0000104

Description: Uncorrectable Error Status Register.

This register provides the status of the various uncorrectable errors detected by the PCI Express Controller. Software may clear any error bit by writing a 1 into the corresponding bit position.

Table 14.6: Uncorrectable Error Status Register

BitLocation	Register Field	Attr	Description	Reset
22	Uncorrectable Internal Error Status.	WOCLR	Uncorrectable Internal Error Status. This bit is set when the Controller has detected an internal uncorrectable error (HAL parity error or an uncorrectable ECC error while reading from any of the RAMs). This bit is also set in response to the client signaling an internal error through the input UNCORRECTABLE_ERROR_IN. This error is not Function-specific. This error is considered fatal by default, and is reported by sending an ERR_FATAL message. STICKY.	
20	Unsupported Request Error Status.	WOCLR	Unsupported Request Error Status. This bit is set when the Controller has received a request from the link that it does not support. This error is not Function-specific. This error is considered non-fatal by default. In the special case described in Sections 6.2.3.2.4.1 of the PCI Express Specifications, the error is reported by sending an ERR_COR message. In all other cases, the error is reported by sending an ERR_NONFATAL message. The header of the received request that caused the error is logged in the Header Log Registers. STICKY.	
19	ECRC Error Status.	WOCLR	ECRC Error Status. This bit is set when the Controller has detected an ECRC error in a received TLP. This error is not Function-specific. The header of the received TLP with error is logged in the Header Log Registers. STICKY.	

Table 14.6: Uncorrectable Error Status Register

BitLocation	Register Field	Attr	Description	Reset
18	Malformed TLP Status.	WOCLR	Malformed TLP Status. This bit is set when the Controller receives a malformed TLP from the link. This error is not Function-specific. This error is considered fatal by default, and is reported by sending an ERR_FATAL message. The header of the received TLP with error is logged in the Header Log Registers. STICKY.	
17	Receiver Overflow Status.	WOCLR	Receiver Overflow Status. This bit is set when the Controller receives a TLP in violation of the receive credit currently available. This error is not Function-specific. STICKY.	
16	Unexpected Completion Status	WOCLR	Unexpected Completion Status. This bit is set when the Controller has received an unexpected Completion packet from the link. This error is not Function-specific. STICKY.	
15	Completer Abort Status.	WOCLR	Completer Abort Status. This bit is set when the Controller has returned the Completer Abort (CA) status to a request received from the link. This error is Function-specific. The header of the received request that caused the error is logged in the Header Log Registers. STICKY.	
14	Completion Timeout Status.	WOCLR	Completion Timeout Status. This bit is set when the completion timer associated with an outstanding request times out. This error is Function-specific. This error is considered non-fatal by default. STICKY.	

Table 14.6: Uncorrectable Error Status Register

BitLocation	Register Field	Attr	Description	Reset
13	Flow Control Protocol Error Status.	WOCLR	Flow Control Protocol Error Status. This bit is set when certain violations of the flow control protocol are detected by the Controller. Controller reports FCPE upon the following conditions: (i) InitFC/UpdateFC DLLP received which issues more than 2047 cumulative outstanding unused credits to the Transmitter for data payload or 127 for header. (ii) InitFC_P is received with Payload Credits less than 128B, Or (iii)InitFC_CPL is received with Payload Credits less than 128B. If a FC error is seen on a VC on a credit type, subsequent errors on that VC/credit type will not be reported until first error is cleared by either a good DLLP, VC de-activate, link down or reset. This error is not Function-specific STICKY.	
12	Poisoned TLP Status	WOCLR	Poisoned TLP Status. This bit is set when the Controller receives a poisoned TLP from the link. This error is Function-specific. This error is considered non-fatal by default. The error is reported by sending an ERR_NONFATAL message. The header of the received TLP with error is logged in the Header Log Registers. STICKY.	

Table 14.6: Uncorrectable Error Status Register

BitLocation	Register Field	Attr	Description	Reset
4	Data Link Protocol Error Status.	WOCLR	Data Link Protocol Error Status. This bit is set when the Controller receives an Ack or Nak DLLP whose sequence number does not correspond to that of an unacknowledged TLP or that of the last acknowledged TLP (for details, refer to PCI Express Base Specification 1.1, Section 3.5.2). This error is not Function-specific, and is reported by Function 0. STICKY.	

14.1.2.3 AER i_corr_err_status

i_AER_cap_struct/i_corr_err_status, Register Offset: 0x0000110

Description: This register provides the status of the various correctable errors detected by the PCI Express Controller. Software may clear any error bit by writing a 1 into the corresponding bit position.

Table 14.7: Correctable Error Status Register

Bit Location	Register Field	Attr	Description	Reset
15	Header Log Register overflow	R/WOCLR sticky	This bit is set on a Header Log Register overflow, that is, when the header could not be logged in the Header Log Register because it is occupied by a previous header. STICKY.	
14	Correctable Internal Error Status	R/WOCLR sticky	This bit is set when the Controller has detected an internal correctable error condition (a correctable ECC error while reading from any of the RAMs). This bit is also set in response to the client signaling an internal error through the input CORRECTABLE_ERROR_IN. This error is not Function-specific. STICKY.	

Table 14.7: Correctable Error Status Register

Bit Location	Register Field	Attr	Description	Reset
13	Advisory Non-Fatal Error Status	R/WOCLR sticky	This bit is set when an uncorrectable error occurs, which is determined to belong to one of the special cases described in Section 6.2.3.2.4 of the PCI Express 2.0 Specifications. This causes the Controller to generate an ERR_COR message in place of an ERR_NONFATAL message. STICKY.	
12	Replay Timer Timeout Status	R/WOCLR sticky	This bit is set when the replay timer in the Data Link Layer of the Controller times out, causing the Controller to retransmit a TLP. This error is not Function-specific. STICKY.	
8	Replay Num Rollover Status	R/WOCLR sticky	This bit is set when the replay count rolls over after three re-transmissions of a TLP at the Data Link Layer of the Controller. This error is not Function-specific STICKY.	
7	Bad DLLP Status	R/WOCLR sticky	This bit is set when an LCRC error is detected in a received DLLP, and no errors were detected by the Physical Layer. This error is not Function-specific. STICKY.	
6	Bad TLP Status	R/WOCLR sticky	This bit is set when an error is detected in a received TLP by the Data Link Layer of the Controller. The conditions causing this error are: (i) An LCRC error (ii) The packet terminates with EDB symbol, but its LCRC field does not equal the inverted value of the calculated CRC. This error is not Function-specific. STICKY.	

Table 14.7: Correctable Error Status Register

Bit Location	Register Field	Attr	Description	Reset
0	Receiver Error Status	R/WOCLR sticky	This bit is set when an error is detected in the receive side of the Physical Layer of the Controller (e.g. a bit error or coding violation). This bit is set upon any of the following errors: (1) PHY reported 8B10B error, Disparity Error, Elastic Buffer Overflow Error, Underflow Error (2) GEN3 TLP, DLLP Framing Errors (3) OS Block Received Without EDS (4) Data Block Received After EDS (5) Illegal OS Block After EDS (6) OS Block Received After SKIP OS (7) OS Block Received After SDS (8) Sync Header Error (9) Loss of Gen3 Block Alignment This error is not Function-specific. STICKY.	

14.1.2.4 i_ltssm_recovery_count_reg_[n]

i_regf_ip_pl/i_ltssm_recovery_count_reg_0, Register Offset: 0x10006F4

Description: This register Contains the count of the number of times the LTSSM transitioned from L0 to Recovery in 2.5GT/s speed.

Table 14.8: LTSSM Recovery Count Register

Name	Bits	Access	Reset value	Description
[RMTCNT] Remote Initiated L0 Recovery Transition Count.	31:16	WOCLR	0	Remote Initiated L0 Recovery Transition Count. Count of number of times the LTSSM transitioned to Recovery due to the Remote End. Writing all 1's clears this.
[LCLCNT] Local Initiated L0 Recovery Transition Count.	15:0	WOCLR	0	Local Initiated L0 Recovery Transition Count. Count of number of times the LTSSM transitioned from L0 to Recovery locally due to local RX Errors. Writing all 1's clears this.

Similar L0->Recovery Transition Count Registers are implemented or Gen2, Gen3, Gen4, Gen5 and Gen6:

Gen2: i_regf_ip_pl/i_ltssm_recovery_count_reg_1, Register Offset: 0x10006F8

Gen3: i_regf_ip_pl/i_ltssm_recovery_count_reg_2, Register Offset: 0x10006FC

Gen4: i_regf_ip_pl/i_ltssm_recovery_count_reg_3, Register Offset: 0x1000700

Gen5: i_regf_ip_pl/i_ltssm_recovery_count_reg_4, Register Offset: 0x1000704

Gen6: i_regf_ip_pl/i_ltssm_recovery_count_reg_5, Register Offset: 0x1000708

14.1.2.5 i_ltssm_recovery_count_local_ctrl_reg

i_regf_ip_pl/i_ltssm_recovery_count_local_ctrl_reg, Register Offset: 0x10006F0

Description: This register Controls the events that will be counted under Locally intiated LTSSM Transition from L0 to Recovery in i_ltssm_recovery_count_reg.

Table 14.9: LTSSM Recovery Count Local Control Register

Name	Bits	Access	Reset value	Description
[LCLTSRCVBM] Local Ltssm Recovery Bitmap	31:0	RW	00	Local Ltssm Recovery Bitmap – Controls the Events that will be counted in the i_ltssm_recovery_count_reg for the locally initiated LTSSM transition from L0 to Recovery. Encoding is : 1: Event is enabled for counting in i_ltssm_recovery_count_reg. 0: Event is not enabled for counting in i_ltssm_recovery_count_reg. bit[0] : RP initiated retrain from CFG Retrain Link bit[1] : EP Initiated Speed Change Retrain Link bit[2] : EP 8GT Request EQ Retrain Link bit[3] : EP 16GT Request EQ Retrain Link bit[4] : EP 32GT Request EQ Retrain Link bit[5] : RP Initiated Disable Link Request bit[6] : RP Initiated Hot Reset Request bit[7] : RP Initiated Loopback Request bit[8] : NFTS Timeout bit[9] : RP Auto Equalization Speed Change to Gen3 and higher bit[10] : RP Auto Speed Change to Gen1 bit[11] : Upper Layer initiated L1 entry bit[12] : Upper Layer Initiated L2 entry bit[13] : Error case: RX ELEC IDLE detected without EIOS bit[14] : Error case: PIPE_RX_VALID de-asserted in L0 bit[15] : Reserved. bit[16] : DL Replay Rollover Retrain Link Request bit[17] : Framing Error Retrain Link bit[18] : FC Timeout Retrain Link bit[19] : EP/RP Link Upconfigure Retrain Link bit[20] : Reserved bit[21] : LM initiated IORECAL Request Retrain Link

14.1.3 Deskew Status Registers

14.1.3.1 Physical Layer Debug Status Register 5

The following registers record the skew detected by the Controller at various speeds:

i_regf_ip_pl/i_dbg_sts_5, Register Offset: 0x1000434

Includes Physical Layer De-Skew Status.

Table 14.10: Physical Layer Debug Status Register 5

Bit Location	Register Field	Attr	Description	Reset
6:0	Gen1 Skew	RO	This field indicates the Skew observed on the lanes at Gen1. Reflects the Skew during the most recent entry into Config.Idle/Recovery.Idle at Gen1.	0
7	Gen1 Skew Valid	RO	This bit validates the Gen1 Skew. Set to 1 when Deskew is Completed at Gen1.	0
14:8	Gen2 Skew	RO	This field indicates the Skew observed on the lanes at Gen2. Reflects the Skew during the most recent entry into Config.Idle/Recovery.Idle at Gen2.	0
15	Gen2 Skew Valid	RO	This bit validates the Gen2 Skew. Set to 1 when Deskew is Completed at Gen2.	0
22:16	Gen3 Skew	RO	This field indicates the Skew observed on the lanes at Gen3. Reflects the Skew during the most recent entry into Config.Idle/Recovery.Idle at Gen3.	0
23	Gen3 Skew Valid	RO	This bit validates the Gen3 Skew. Set to 1 when Deskew is Completed at Gen3.	0
30:24	Gen4 Skew	RO	This field indicates the Skew observed on the lanes at Gen4. Reflects the Skew during the most recent entry into Config.Idle/Recovery.Idle at Gen4.	0
31	Gen4 Skew Valid	RO	This bit validates the Gen4 Skew. Set to 1 when Deskew is Completed at Gen4.	0

14.1.3.2 Physical Layer Debug Status Register 6

i_regf_ip_pl/i_dbg_sts_6, Register Offset: 0x1000438

Includes Physical Layer De-Skew Status.

Table 14.11: Physical Layer Debug Status Register 6

Bit Location	Register Field	Attr	Description	Reset
6:0	Gen5 Skew	RO	This field indicates the Skew observed on the lanes at Gen5. Reflects the Skew during the most recent entry into Config.Idle/Recovery.Idle at Gen5.	0
7	Gen5 Skew Valid	RO	This bit validates the Gen5 Skew. Set to 1 when Deskew is Completed at Gen5.	0
14:8	Gen6 Skew	RO	This field indicates the Skew observed on the lanes at Gen6. Reflects the Skew during the most recent entry into Config.Idle/Recovery.Idle at Gen6.	0
15	Gen6 Skew Valid	RO	This bit validates the Gen6 Skew. Set to 1 when Deskew is Completed at Gen6.	0

14.1.4 Link Equalization Control Setup Register

The algorithm used by the Controller to perform equalization can be programmed using the following registers:

14.1.4.1 Gen3 Equalization Control Register 0

i_regf_ip_pl/i_g3_eq_ctrl_0, Register Offset: 0x1000474

Table 14.12: Gen3 Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
10:0	Supported Gen3 Preset	RW	This register can be used to program the Presets that are supported by local Transmitter at 8Gbps. Default value of this register is determined by the SUPPORTED_PRESET strap input. Note: At 8.0 GT/s and higher, all preset values must be supported for Full swing signaling. Reduced swing signaling must implement presets #4, #1, #9, #5, #6, and #3.	
14:11	Default Gen3 Transmitter Preset	RW	Default Gen3 transmitter preset value used for a lane that did not receive EQ TS2 in Recovery.RcvrCfg LTSSM state	
17:15	Default Gen3 Receiver Preset Hint	RW	Default Gen3 receiver preset hint value used for a lane that did not receive EQ TS2 in Recovery.RcvrCfg LTSSM state	
20:18	Gen3 Max Eval Convergence Count	RW	Controls the number of consecutive RxEqEval iterations with direction change feedback of 00s before Equalization Convergence is inferred. 0 : Infer Convergence after 1 feedback of 000000 1 : Infer Convergence after 2 feedback of 000000 2 : Infer Convergence after 3 consecutive feedback of 000000 .. 7 : Infer Convergence after 8 consecutive feedback of 000000. Note: Each lane independently counts consecutive feedback of 000000. Note: Count is reset after a non-000000 feedback on each lane.	

Table 14.12: Gen3 Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
21	EP 8GTs Request Equalization Retrain Link	RW	This bit can be used by Endpoint Device FW to request for 8GT/s Equalization redo. This bit can be set at any time after the Link is Up. Writing a 1 into this field results in the Controller to transition to Recovery. The Request Equalization bit in TS2 Ordered Sets will be set to 1 in Recovery.Rcvr.Cfg to request equalization at 8GTs. This bit is auto-cleared by the internal logic of the Controller after the re-training has been completed and link has reached the L0 state. This bit is also auto-cleared when not in Gen3 or Gen4. Device Firmware must wait for the bit to be clear before any subsequent retrain requests.	
22	Quiesce Guarantee 8GTs	RW	This bit can be used to program the Quiesce Guarantee bit of the TS2 in Recovery.Rcvr.Cfg state during 8GT/s Request Equalization.	

Table 14.12: Gen3 Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
26:23	Max 8GTs Equalization Request Limit	RW	The number of 8GT/s Equalization Requests must be finite as per PCIe specification. This register can be used to program the maximum number of 8GT/s equalization requests automatically initiated by the Endpoint. 0000: Automatic 8GT/s Equalization Request Disabled. 0001: Automatic 8GT/s Equalization request limit is 1. 0010: Automatic 8GT/s Equalization request limit is 2. .... 1111: Automatic 8GT/s Equalization request limit is 15,	0
27	Disable MAX EVAL ITERATION	RW	The MAX_EVAL_ITERATION input controls the maximum number of iterations in Equalization Phase2 (EP) or Phase3 (RP). This bit can be used to mask the MAX_EVAL_ITERATION and perform more iterations till Convergence or a 24ms timeout.	0

14.1.4.2 Gen3 Equalization Control Register 1

i_regf_ip_pl/i_g3_eq_ctrl_1, Register Offset: 0x1000478

Table 14.13: Gen3 Equalization Control Register 1

Bit Location	Register Field	Attr	Description	Reset
3:0	Gen3 Equalization Debug Status Lane Select	RW	Select the Lane for checking the status in the Gen3 Lane Equalization Status Register 0, Gen3 Lane Equalization Status Register 1.	0

Table 14.13: Gen3 Equalization Control Register 1

Bit Location	Register Field	Attr	Description	Reset
4	Gen3 EQ Phase2 Remote Tx Preset Enable	RW	Used only in EP Mode. This bit enables the Controller, to feedback a Tx Preset for the Remote end Transmitter in the first iteration of Link Equalization Phase2 at Gen3 speed. The Gen3 Tx Preset that is used for feedback is programmable in bits [4:1] of this register. Reserved for RP Mode.	0
5	Gen3 Local Override Tx Preset Enable	RW	This is a debug bit. Can be used in both EP and RP Mode. If enabled, the Controller locally applies the Gen3 Local Override Tx Preset to the Local Transmitter throughout Gen3. The Controller performs the Override Preset to Coefficient mapping and then drives on PIPE_TX_DEEMPHASIS signal on the PIPE Interface.	0
11:8	Gen3 EQ Phase2 Remote Tx Preset	RW	Used only in EP Mode. When enabled using bit-0, this Tx Preset will be transmitted in TS1s for the Remote end Transmitter in the first iteration of Gen3 Equalization Phase2. Reserved for RP Mode.	0
15:12	Gen3 Local Override Tx Preset	RW	This is a debug register field. Can be used in both EP and RP Mode. When enabled using bit-5, this Tx Preset will be applied to the local Transmitter throughout Gen3 regardless of Gen3 Equalization.	0

14.1.4.3 Gen4 Equalization Control Register 0

i_regf_ip_pl/i_g4_eq_ctrl_0, Register Offset: 0x1000488

Table 14.14: Gen4 Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
10:0	Supported Gen4 Preset	RW	This register can be used to program the Presets that are supported by local Transmitter at 16GT/s. Default value of this register is determined by the SUPPORTED_PRESET strap input. Note: At 8.0 GT/s and higher, all preset values must be supported for Full swing signaling. Reduced swing signaling must implement presets #4, #1, #9, #5, #6, and #3.	
14:11	Default Gen4 Transmitter Preset	RW	Default Gen4 transmitter preset value used for a lane that did not receive 8G EQ TS2 in Recovery.RcvrCfg LTSSM state	
20:18	Gen4 Max Eval Convergence Count	RW	Controls the number of consecutive RxEqEval iterations with direction change feedback of 00s before Equalization Convergence is inferred. 0 : Infer Convergence after 1 feedback of 000000 1 : Infer Convergence after 2 feedback of 000000 2 : Infer Convergence after 3 consecutive feedback of 000000 .. 7 : Infer Convergence after 8 consecutive feedback of 000000. Note: Each lane independently counts consecutive feedback of 000000. Note: Count is reset after a non-000000 feedback on each lane.	

Table 14.14: Gen4 Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
21	EP 16GTs Request Equalization Retrain Link	RW	This bit can be used by Endpoint Device FW to request for 16GT/s Equalization redo. This bit can be set at any time after the Link is Up. Writing a 1 into this field results in the Controller to transition to Recovery. The Request Equalization bit in TS2 Ordered Sets will be set to 1 in Recovery.Rcvr.Cfg to request equalization at 16GTs. This bit is auto-cleared by the internal logic of the Controller after the re-training has been completed and link has reached the L0 state. This bit is also auto-cleared when not in Gen3 or Gen4. Device Firmware must wait for the bit to be clear before any subsequent retrain requests.	
22	Quiesce Guarantee 16GTs	RW	This bit can be used to program the Quiesce Guarantee bit of the TS2 in Recovery.Rcvr.Cfg state during 16GT/s Request Equalization.	

Table 14.14: Gen4 Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
26:23	Max 16GTs Equalization Request Limit	RW	The number of 16GT/s Equalization Requests must be finite as per PCIe specification. This register can be used to program the maximum number of 16GT/s equalization requests automatically initiated by the Endpoint. 0000: Automatic 16GT/s Equalization Request Disabled. 0001: Automatic 16GT/s Equalization request limit is 1. 0010: Automatic 16GT/s Equalization request limit is 2. .... 1111: Automatic 16GT/s Equalization request limit is 15,	

14.1.4.4 Gen4 Equalization Control Register 1

i_regf_ip_pl/i_g4_eq_ctrl_1, Register Offset: 0x100048C

Table 14.15: Gen4 Equalization Control Register 1

Bit Location	Register Field	Attr	Description	Reset
3:0	Gen4 Equalization Debug Status Lane Select	RW	Select the Lane for checking the status in the Gen4 Lane Equalization Status Register 0, Gen4 Lane Equalization Status Register 1.	0
4	Gen4 EQ Phase2 Remote Tx Preset Enable	RW	Used only in EP Mode. This bit enables the Controller, to feedback a Tx Preset for the Remote end Transmitter in the first iteration of Link Equalization Phase2 at Gen4 speed. The Gen4 Tx Preset that is used for feedback is programmable in bits [15:12] of this register.	0

Table 14.15: Gen4 Equalization Control Register 1

Bit Location	Register Field	Attr	Description	Reset
5	Gen4 EP 8GT EQ TS2 Enable	RW	Used only in EP Mode. During Gen3 to Gen4 Speed Change negotiation, this bit enables the Controller in EP Mode to transmit 8G EQ TS2 in Recovery.Rcvr.Cfg state instead of standard TS2 as defined in PCIe specification. The Tx Preset that will be used in the 8GT EQ TS2 is programmable in bits [15:12] of this register.	0
6	Gen4 Local Tx Preset Override Enable	RW	This is a debug bit. Can be used in both EP and RP Mode. If enabled, the Controller locally applies the Gen4 Local Override Tx Preset to the Local Transmitter throughout Gen4. The Controller performs the Override Preset to Coefficient mapping and then drives on PIPE_TX_DEEMPHASIS signal on the PIPE Interface.	0
11:8	Gen4 EQ Phase2 Remote Tx Preset	RW	Used only in EP Mode. If enabled, this Tx Preset will be transmitted in 8G EQ TS2s during Gen3 to Gen4 Speed Change negotiation. Also, If enabled, this Tx Preset will be transmitted in TS1s in the first iteration of Gen4 Equalization Phase2. Note: If the Remote end device advertised the same preset at start of Equalization Phase2, then this Preset will be skipped.	0
15:12	Gen4 Local Override Tx Preset	RW	This is a debug register field. Can be used in both EP and RP Mode. When enabled using bit-16, this Tx Preset will be applied to the local Transmitter throughout Gen4 regardless of Gen4 Equalization.	0

14.1.4.5 Gen5 Equalization Control Register 0

i_regf_ip_pl/i_g5_eq_ctrl_0, Register Offset: 0x100049C

Table 14.16: Gen5 Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
10:0	Supported Gen5 Preset	RW	This register can be used to program the Presets that are supported by local Transmitter at 32GT/s. Default value of this register is determined by the SUPPORTED_PRESET strap input. Note: At 8.0 GT/s and higher, all preset values must be supported for Full swing signaling. Reduced swing signaling must implement presets #4, #1, #9, #5, #6, and #3.	
14:11	Default Gen5 Transmitter Preset	RW	Default Gen5 transmitter preset value used for a lane that did not receive 8G EQ TS2 in Recovery.RcvrCfg LTSSM state	
16:15	Gen5 Max Eval Convergence Count	RW	Controls the number of consecutive RxEqEval iterations with direction change feedback of 00s before Equalization Convergence is inferred. 0 : Infer Convergence after 1 feedback of 000000 1 : Infer Convergence after 2 feedback of 000000 2 : Infer Convergence after 3 consecutive feedback of 000000 .. 7 : Infer Convergence after 8 consecutive feedback of 000000. Note: Each lane independently counts consecutive feedback of 000000. Note: Count is reset after a non-000000 feedback on each lane.	

Table 14.16: Gen5 Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
17	EP 32GTs Request Equalization Retrain Link	RW	This bit can be used by Endpoint Device FW to request for 32GT/s Equalization redo. This bit can be set at any time after the Link is Up. Writing a 1 into this field results in the Controller to transition to Recovery. The Request Equalization bit in TS2 Ordered Sets will be set to 1 in Recovery.Rcvr.Cfg to request equalization at 32GTs. This bit is auto-cleared by the internal logic of the Controller after the re-training has been completed and link has reached the L0 state. This bit is also auto-cleared when not in Gen3 or Gen4 or Gen5. Device Firmware must wait for the bit to be clear before any subsequent retrain requests.	
18	Quiesce Guarantee 32GTs	RW	This bit can be used to program the Quiesce Guarantee bit of the TS2 in Recovery.Rcvr.Cfg state during 32GT/s Request Equalization.	

Table 14.16: Gen5 Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
22:19	Max 32GTs Equalization Request Limit	RW	The number of 32GT/s Equalization Requests must be finite as per PCIe specification. This register can be used to program the maximum number of 32GT/s equalization requests automatically initiated by the Endpoint. 0000: Automatic 32GT/s Equalization Request Disabled. 0001: Automatic 32GT/s Equalization request limit is 1. 0010: Automatic 32GT/s Equalization request limit is 2. .... 1111: Automatic 32GT/s Equalization request limit is 15,	

14.1.4.6 Gen5 Equalization Control Register 1

i_regf_ip_pl/i_g5_eq_ctrl_1, Register Offset: 0x10004A0

Table 14.17: Gen5 Equalization Control Register 1

Bit Location	Register Field	Attr	Description	Reset
3:0	Gen5 Equalization Debug Status Lane Select	RW	Select the Lane for checking the status in the Gen5 Lane Equalization Status Register 0, Gen5 Lane Equalization Status Register 1.	0
4	Gen5 EQ Phase2 Remote Tx Preset Enable	RW	Used only in EP Mode. This bit enables the Controller, to feedback a Tx Preset for the Remote end Transmitter in the first iteration of Link Equalization Phase2 at Gen5 speed. The Gen5 Tx Preset that is used for feedback is programmable in bits [26:23] of this register.	0

Table 14.17: Gen5 Equalization Control Register 1

Bit Location	Register Field	Attr	Description	Reset
5	Gen5 EP EQ TS2 Enable	RW	Used only in EP Mode. During Gen4 to Gen5 Speed Change negotiation, this bit enables the Controller in EP Mode to transmit EQ TS2 (Or 128b/130b EQ TS2) in Recovery.Rcvr.Cfg state instead of standard TS2 as defined in PCIe specification. The Tx Preset that will be used in the EQ TS2 is programmable in bits [26:23] of this register.	0
6	Gen5 Local Tx Preset Override Enable	RW	This is a debug bit. Can be used in both EP and RP Mode. If enabled, the Controller locally applies the Gen5 Local Override Tx Preset to the Local Transmitter throughout Gen5. The Controller performs the Override Preset to Coefficient mapping and then drives on PIPE_TX_DEEMPHASIS signal on the PIPE Interface.	0
11:8	Gen5 EQ Phase2 Remote Tx Preset	RW	Used only in EP Mode. If enabled, this Tx Preset will be transmitted in EQ TS2s (Or 128b/130b EQ TS2) during speed change negotiation to Gen5. Also, If enabled, this Tx Preset will be transmitted in TS1s in the first iteration of Gen5 Equalization Phase2. Note: If the Remote end device advertised the same preset at start of Equalization Phase2, then this Preset will be skipped.	0

Table 14.17: Gen5 Equalization Control Register 1

Bit Location	Register Field	Attr	Description	Reset
15:12	Gen5 Local Override Tx Preset	RW	This is a debug register field. Can be used in both EP and RP Mode. When enabled using bit-16, this Tx Preset will be applied to the local Transmitter throughout Gen5 regardless of Gen5 Equalization.	0

14.1.4.7 Equalization Control Register 0

i_regf_ip_pl/i_eq_ctrl_0, Register Offset: 0x1000470

Table 14.18: Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
0	RXEQINPROGRESS Abort Disable	RW	In a unexpected case where the PIPE_PCLK stops due to error in equalization, this bit can be set to de-couple RxEqIn-Progress from the rest of the equalization state machine. This bit should not be set for normal usage.	0

Table 14.18: Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
2:1	RXEQINPROGRESS Abort Timer Mode Control	RW	When a 24ms timeout occurs in the LTSSM Equalization Phase 2, the Controller aborts Equalization Phase 2 and transitions to Recovery.Rcvr.Lock. In this case, the RxEqEval output on the PIPE Interface will be deasserted immediately (if it was asserted). The RxEqInProgress output will stay high and waits for PhyStatus pulse. Controller implements a timer to select an upper limit to wait for this PhyStatus pulse during an abort to de-assert RxEqInProgress. 00: Wait for a maximum of 4 PIPE_PCLK period. 01: Wait for a maximum of 8 PIPE_PCLK period. 10: Wait for a maximum of 16 PIPE_PCLK period. 11: Disabled. Wait till PhyStatus Pulse is received. Note: This register is used only if RxEqEval was asserted when LTSSM 24ms timeout occured in Equalization.	0
3	Disable LOCALLF, LOCALFS sampling after speed change	RW	As per PIPE 4.2 specification, the LOCALLF, LOCALFS outputs from PHY can be sampled uponf PHYSTATUS pulse after Reset# OR upon the first PHYSTATUS pulse after speed change to GEN3. This bit can be set to 1 to disable sampling after speed change to GEN3 or higher	0

Table 14.18: Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
4	Enable Retry RXEQEVAL After Feedback Error	RW	When the Controller receives a direction change feedback from the PHY, the feedback is applied to the current coefficients and checked for errors. During Equalization Phase2(EP) or Phase3(RP), 1 iteration is defined as 1 set of feedback issued by the device to the remote end in TS1s over PCIe link. This bit controls the device feedback in an iteration in case of direction change errors: 0: If feedback is invalid, then the Controller asserts PIPE_INVALIDREQUEST to PHY, discards the feedback and does not retry PIPE_RXEQEVAL in same iteration. The coefficient settings is kept unchanged in the iteration. Retry PIPE_RXEQEVAL in next iteration. 1: If feedback is invalid, then the Controller asserts PIPE_INVALIDREQUEST to PHY, discards the feedback and retries PIPE_RXEQEVAL in same iteration till a valid feedback is received.	0

Table 14.18: Equalization Control Register 0

Bit Location	Register Field	Attr	Description	Reset
5	Disable LOCALLF, LOCALFS sampling after rate change for Loopback.		As per PIPE specification, the LOCALLF, LOCALFS outputs from PHY can be sampled upon PHYSTATUS pulse after Reset# OR upon the first PHYSTATUS pulse after rate change to GEN3. Rate change is initiated during Loopback Entry when current link rate is not same as the Loopback Rate. This bit can be set to 1 to disable sampling after rate change to GEN3 or higher during Loopback.	
6	Disable Spurious PIPE Local TX Coefficients Valid Check.		As per PIPE specification, the pipe_localtxcoefficientsvalid input is expected to be asserted only after MAC queries preset to coefficient mapping on pipe_getlocalpresetcoefficients. The Controller checks and discards pipe_localtxcoefficientsvalid input when no request issued on pipe_getlocalpresetcoefficients. 0: Discard spurious responses on pipe_localtxcoefficientsvalid input. 1: Store all responses on pipe_localtxcoefficientsvalid input and use for any future preset query.	
31:7	Reserved			

14.1.4.8 IORECAL Control Status Register

i_regf_ip_pl/i_iorecal_ctrl_stat, Register Offset: 0x10006B0

The PIPE IO Recalibration functionality of the Controller can be controlled using this register. The status of PIPE IORECAL signals is also reflected in this register.

Table 14.19: IORECAL Control Status Register

Bit Location	Register Field	Attr	Description	Reset
5:0	Controller Initiated IORECAL Disable	RW	This field is used to disable Controller Initiated IORECAL upon Receiver/Framing errors at each speed. (1: Disable, 0: Enable). Bit[0] - Reserved Bit[1] - Reserved Bit[2] - Disable IORECAL at GEN3. Bit[3] - Disable IORECAL at GEN4. Bit[4] - Disable IORECAL at GEN5. Bit[5] - Disable IORECAL at GEN6.	0
7:6	Reserved			
13:8	PHY Initiated IORECAL Mask	RW	This field is used to mask PHY Initiated IORECAL request at each speed. (1: Mask PHY IORECAL Request, 0: Process PHY IORECAL Request). Bit[0] - Mask pipe_iorecal_request at GEN1. Bit[1] - Mask pipe_iorecal_request at GEN2. Bit[2] - Mask pipe_iorecal_request at GEN3. Bit[3] - Mask pipe_iorecal_request at GEN4. Bit[4] - Mask pipe_iorecal_request at GEN5. Bit[5] - Mask pipe_iorecal_request at GEN6.	0
15:13	Reserved			

Table 14.19: IORECAL Control Status Register

Bit Location	Register Field	Attr	Description	Reset
16	IORECAL Timeout Enable	RW	This field is used to enable an independent 24ms timeout while waiting in Recovery.Rcvr.Lock for pipe_iorecal_done response from PHY. If no response on pipe_iorecal_done, then: 1 : LTSSM transitions Recovery.Rcvr.Lock -> Recovery.Rcvr.Cfg after 24 ms timeout. 0 : LTSSM transitions Recovery.Rcvr.Lock -> Recovery.Rcvr.Cfg/ Recovery.Speed/Configuration/Detect after 24ms timeout as per PCIe specification for 24 ms timeout. Setting this bit to 1 prevents Link Down in case pipe_iorecal_done takes longer than 24ms.	0
17	IORECAL Timeout Status	WOCLR	This status bit indicates if 24ms timeout occurred while waiting for pipe_iorecal_done response. Note: This bit will be set only when IORECAL Timeout Enable (bit-16) is programmed to to 1.	0
18	IORecal Retrain Link	RW	Writing a 1 into this field results in the Controller initiating L0-> Recovery transition with IORecal asserted. This bit is cleared by the internal logic of the Controller after the IORecal completes and the LTSSM transitions out of Recovery.Rcvr.Lock state. Software must wait for the bit to be clear before setting it again.	0
28:19	Reserved			
29	PIPE IORECAL DONE Status	RO	This status bit indicates the current status of pipe_iorecal_done input.	0

Table 14.19: IORECAL Control Status Register

Bit Location	Register Field	Attr	Description	Reset
30	PIPE IORECAL REQUEST Status	RO	This status bit indicates the current status of pipe_iorecal_request_status input.	0
31	PIPE IORECAL Status	RO	This status bit indicates the current status of pipe_iorecal output.	0

14.1.5 Link Equalization Debug Status Registers

The result of the phases of link equalization are reflected in the status registers. These registers can be used to debug link equalization process.

14.1.5.1 Gen3 Lane Equalization Status Register 0

i_regf_ip_pl/i_g3_ln_eq_sts_0, Register Offset: 0x100047C

Table 14.20: Gen3 Lane Equalization Status Register 0

Bit Location	Register Field	Attr	Description	Reset
3:0	Gen3 Local TX Preset	RO	TX Preset agreed upon for this lane.	
4	Gen3 Local TX Preset Valid	RO	TX Preset Valid Indicator. This bit is set when a TX Preset is received in TS1s with the use_preset bit set in Endpoint Mode Phase 3 or RC Mode Phase 2.	
7:5	Reserved			
25:8	Gen3 Local TX Coefficient	RO	TX Coefficients agreed upon for this lane. [25:20] : Post Cursor Coefficient [19:14] : Cursor Coefficient [13:8] : Pre-Cusror Coefficient	

14.1.5.2 Gen3 Lane Equalization Status Register 1

i_regf_ip_pl/i_g3_ln_eq_sts_1, Register Offset: 0x1000480

Table 14.21: Gen3 Lane Equalization Status Register 1

Bit Location	Register Field	Attr	Description	Reset
3:0	Gen3 Remote TX Preset	RO	TX Preset feedback assigned to Remote TX for this lane.	

Table 14.21: Gen3 Lane Equalization Status Register 1

Bit Location	Register Field	Attr	Description	Reset
4	Gen3 Remote TX Preset Valid	RO	TX Preset Valid Indicator. This bit is set when a TX Preset feedback is provided in TS1s with the use_preset bit set in End-point Mode Phase 2 or RC Mode Phase 3.	
22:8	Gen3 Remote TX Coeffi-cient	RO	TX Coefficients feedback assigned to Remote TX for this lane. [25:20] : Post Cursor Co-efficient [19:14] : Cursor Coefficient [13:8] : Pre-Cusror Coefficient	

14.1.5.3 Gen4 Lane Equalization Status Register 0

i_regf_ip_pl/i_g4_ln_eq_sts_0, Register Offset: 0x1000490

Table 14.22: Gen4 Lane Equalization Status Register 0

Bit Location	Register Field	Attr	Description	Reset
3:0	Gen4 Local TX Preset	RO	TX Preset agreed upon for this lane.	
4	Gen4 Local TX Preset Valid	RO	TX Preset Valid Indicator. This bit is set when a TX Preset is received in TS1s with the use_preset bit set in Endpoint Mode Phase 3 or RC Mode Phase 2.	
7:5	Reserved			
25:8	Gen4 Local TX Coeffi-cient	RO	TX Coefficients agreed upon for this lane. [25:20] : Post Cursor Co-efficient [19:14] : Cursor Coefficient [13:8] : Pre-Cusror Coefficient	

14.1.5.4 Gen4 Lane Equalization Status Register 1

i_regf_ip_pl/i_g4_ln_eq_sts_1, Register Offset: 0x1000494

Table 14.23: Gen4 Lane Equalization Status Register 1

Bit Location	Register Field	Attr	Description	Reset
3:0	Gen4 Remote TX Preset	RO	TX Preset feedback as-signed to Remote TX for this lane.	

Table 14.23: Gen4 Lane Equalization Status Register 1

Bit Location	Register Field	Attr	Description	Reset
4	Gen4 Remote TX Preset Valid	RO	TX Preset Valid Indicator. This bit is set when a TX Preset feedback is provided in TS1s with the use_preset bit set in Endpoint Mode Phase 2 or RC Mode Phase 3.	
7:5	Reserved			
25:8	Gen4 Remote TX Coefficient	RO	TX Coefficients feedback assigned to Remote TX for this lane. [25:20] : Post Cursor Coefficient [19:14] : Cursor Coefficient [13:8] : Pre-Cusror Coefficient	

14.1.5.5 Gen5 Lane Equalization Status Register 0

i_regf_ip_pl/i_g5_ln_eq_sts_0, Register Offset: 0x10004A4

Table 14.24: Gen5 Lane Equalization Status Register 0

Bit Location	Register Field	Attr	Description	Reset
3:0	Gen5 Local TX Preset	RO	TX Preset agreed upon for this lane.	
4	Gen5 Local TX Preset Valid	RO	TX Preset Valid Indicator. This bit is set when a TX Preset is received in TS1s with the use_preset bit set in Endpoint Mode Phase 3 or RC Mode Phase 2.	
7:5	Reserved			
22:5	Gen5 Local TX Coefficient	RO	TX Coefficients agreed upon for this lane. [25:20] : Post Cursor Coefficient [19:14] : Cursor Coefficient [13:8] : Pre-Cusror Coefficient	

14.1.5.6 Gen5 Lane Equalization Status Register 1

i_regf_ip_pl/i_g5_ln_eq_sts_1, Register Offset: 0x10004A8

Table 14.25: Gen5 Lane Equalization Status Register 1

Bit Location	Register Field	Attr	Description	Reset
3:0	Gen5 Remote TX Preset	RO	TX Preset feedback assigned to Remote TX for this lane.	

Table 14.25: Gen5 Lane Equalization Status Register 1

Bit Location	Register Field	Attr	Description	Reset
4	Gen5 Remote TX Preset Valid	RO	TX Preset Valid Indicator. This bit is set when a TX Preset feedback is provided in TS1s with the use_preset bit set in End-point Mode Phase 2 or RC Mode Phase 3.	
7:5	Reserved			
25:8	Gen5 Remote TX Coefficient	RO	TX Coefficients feedback assigned to Remote TX for this lane. [25:20] : Post Cursor Coefficient [19:14] : Cursor Coefficient [13:8] : Pre-Cusror Coefficient	

14.1.5.7 Equalization Debug FIFO Control Register

i_regf_ip_pl/i_eq_dbg_fifo_ctrl, Register Offset: 0x10004F0

Table 14.26: Equalization Debug FIFO Control Register

Bit Location	Register Field	Attr	Description	Reset
0	Clear All capture	RW	Setting this bit clears all captured information in the EQ Debug Status Registers. If it is unset then capture is allowed in status registers.	
4:1	Capture Lane Select	RW	Selects the Lane whose Equalization Debug information is to be captured. Please note, this signifies the physical lane number.	
6:5	Capture EQ Phase Select	RW	Selects the Equalization Phase when capture is to be done 01 : Phase 2, 10 : Phase 3, 11 : Phase 2 and 3	
9:7	Capture EQ Speed Select	RW	Capture EQ Speed Select. Selects the Link Speed at which capture is to be done: 000 : Any speed, 001 : Gen3, 010 : Gen4, 100 : Gen5, 101 : Gen6	

Table 14.26: Equalization Debug FIFO Control Register

Bit Location	Register Field	Attr	Description	Reset
10	Eq Debug Capture Behavior	RW	Eq Debug Capture Behavior. If this is set , the first 64 equalization info events are captured else the last 64 events are captured	

14.1.5.8 Equalization Debug FIFO Status 0 Register

i_regf_ip_pl/i_eq_dbg_fifo_sts_0, Register Offset: 0x10004F4

Description:

All the Dynamic Equalization information is captured in this Register.

This is implemented using a synchronous FIFO which stores all the captured events as separate 32-bit entries. Every read increments the read pointer and the client must store the data read. The FIFO can be cleared using the Clear all Capture bit in the EQ Debug Monitor Control Register.

Table 14.27: Equalization Debug FIFO Status 0 Register

Bit Location	Register Field	Attr	Description	Reset
5:0	Local FS	RO	Local PHY's FS Value Of the Lane and Speed Selected.	
11:6	Local LF	RO	Local PHY's LF Value Of the Lane and Speed Selected.	
17:12	Remote FS	RO	Remote PHY's FS Value Of the Lane and Speed Selected.	
23:18	Remote LF	RO	Remote PHY's LF Value Of the Lane and Speed Selected.	

14.1.5.9 Equalization Debug FIFO Status 1 Register

i_regf_ip_pl/i_eq_dbg_fifo_sts_1, Register Offset: 0x10004F8

Table 14.28: Equalization Debug FIFO Status 1 Register

Bit Location	Register Field	Attr	Description	Reset
17:0	Coefficients	RO	Phase0: Stores Initial Local TX Coefficients mapped from Initial Preset. Phase1: Stores Initial Remote Coefficients advertised in Phase1. (Cp, LF, FS) , EP Ph2/RC Phase3: Stores Current Coefficients of the Remote Device. EP Ph3/RC Phase2: Stores Coefficients Received from Remote Device.	
21:18	Preset	RO	Phase0: Stores Initial Local TX Preset received in Phase0. Phase1: Stores Initial Remote Preset advertised in Phase1. EP Ph2/RC Phase3: Stores Current Preset of the Remote Device. EP Ph3/RC Phase2: Stores Preset Received from Remote Device.	
22	Preset Valid	RO	1: Preset Valid, Indicates [21:18] is valid. Phase0: Set to '1' to indicate that the intial Local Preset is Valid. Phase1: Set to '1' to indicate that the advertised Remote Preset is Valid. EP Ph2/RC Ph3: Set to 1 if controller provide preset feedback and to 0 for coefficient feedback. EP Ph3/RC Ph2: Reflects the use preset bit received from the remote end.	

Table 14.28: Equalization Debug FIFO Status 1 Register

Bit Location	Register Field	Attr	Description	Reset
23	Coefficient Reject	RO	Phase0: Set to '1' if an unsupported preset is received in Phase0. Phase1: Set to '0' since no reject in phase1. EP Ph2/RC Ph3: Indicates Reject by the Remote end device. This bit indicates that the current Coefficient or Preset was rejected by the remote end device. EP Ph3/RC Ph2: Indicates that Controller Rejected the received settings to Remote Device in the TX TS1/TS2. This Reject indicates the current Coefficients or Preset received from Remote Device are rejected	
29:24	Direction Feedback	RO	EP Ph2/RC Ph3: Stores Direction Change Feedback or Preset feedback Transmitted to Remote Device. Bit-22, EQPREVD, indicates if this is a Preset feedback or Direction Change Feedback. EP Ph3/RC Ph2: Reserved	
31:30	Equalization Phase	RO	Equalization Phase during Capture 00 : Phase 0, 01 : Phase 1, 10 : Phase 2, 11 : Phase 3	

14.1.5.10 Equalization Debug FIFO Status 2 Register

i_regf_ip_pl/i_eq_dbg_fifo_sts_2, Register Offset: 0x10004FC

Description: This register contains the upper 32-bits of the Equalization Debug FIFO status. This register must be read after reading i_eq_dbg_fifo_sts_1 register

Table 14.29: Equalization Debug FIFO Status 2 Register

Bit Location	Register Field	Attr	Description	Reset
5:0	Coefficients	RO	This is valid only at Gen6. Phase0: Stores Initial Local TX Coefficients mapped from Initial Preset. (Cm2) Phase1: Reserved. EP Phase2/RC Phase3: Stores Current Coefficients of the Remote Device. (Cm2) EP Phase3/RC Phase2: Stores Coefficients Received from Remote Device.	
11:6	Direction Change Feedback	RO	Direction Change Feedback. This is valid only at Gen6. EP Phase2/RC Phase3: Stores Direction Change Feedback (upper 6-bits for Gen6). EP Phase3/RC Phase2: Reserved	

14.1.6 LTSSM Timeout Control Registers

The LTSSM Timeout values are implemented as per PCIe specification by default. These timeout values can be programmed for debug purposes using following registers:

14.1.6.1 LTSSM Timer Control 0

i_regf_ip_pl/i_ltssm_tmr_ctrl_0, Register Offset: 0x1000560

Table 14.30: LTSSM Timer Control 0

Bit Location	Register Field	Attr	Description	Reset
11:0	LTSSM 1ms Time Interval	RW	This register holds the 1ms timer interval in units of (1024 ns). This register can be tuned to vary the 1ms timeout in the LTSSM.	12’d977
15:12	Reserved			
27:16	LTSSM 2ms Time Interval	RW	This register holds the 2ms timer interval in units of (1024 ns). This register can be tuned to vary the 2ms timeout in the LTSSM.	12’d1954
28	Reserved			

Table 14.30: LTSSM Timer Control 0

Bit Location	Register Field	Attr	Description	Reset
29	LTSSM Time Interval Scaling Enable	RW	This is a debug register field, used to Enable LTSSM Time Interval Scaling. 0: LTSSM Time Interval Register values are in units of 1024ns. 1: LTSSM Time Interval values are in units of 1024us. Note: Setting this bit to 1 enables extended time interval range, beyond PCIe spec defined ranges. Hence, this can be used only for debug purposes and not in functional mode.	0
30	Disable PHY LTSSM Timers	RW	Disable all LTSSM Timers	0
31	Enable Fast Link Training	RW	Used to speed up link training by scaling down the following LTSSM timers: - In Detect.Quiet, 12ms timer is scaled down to 120us. - In Detect.Active, 12ms timer is scaled down to 120us.	0

14.1.6.2 LTSSM Timer Control 1

i_regf_ip_pl/i_ltssm_tmr_ctrl_1, Register Offset: 0x1000564

Table 14.31: LTSSM Timer Control 1

Bit Location	Register Field	Attr	Description	Reset
15:0	LTSSM 4ms Time Interval	RW	This register holds the 4ms timer interval in units of (1024 ns). This register can be tuned to vary the 4ms timeout in the LTSSM.	16’d3907
31:16	LTSSM 8ms Time Interval	RW	This register holds the 8ms timer interval in units of (1024 ns). This register can be tuned to vary the 8ms timeout in the LTSSM.	16’d7813

14.1.6.3 LTSSM Timer Control 2

i_regf_ip_pl/i_ltssm_tmr_ctrl_2, Register Offset: 0x1000568

Table 14.32: LTSSM Timer Control 2

Bit Location	Register Field	Attr	Description	Reset
15:0	LTSSM 12ms Time Interval	RW	This register holds the 12ms timer interval in units of (1024 ns). This register can be tuned to vary the 12ms timeout in the LTSSM.	16'd11719
31:16	LTSSM 24ms Time Interval	RW	This register holds the 24ms timer interval in units of (1024 ns). This register can be tuned to vary the 24ms timeout in the LTSSM.	16'd23438

14.1.6.4 LTSSM Timer Control 3

i_regf_ip_pl/i_ltssm_tmr_ctrl_3, Register Offset: 0x100056C

Table 14.33: LTSSM Timer Control 3

Bit Location	Register Field	Attr	Description	Reset
15:0	LTSSM 32ms Time Interval	RW	This register holds the 32ms timer interval in units of (1024 ns). This register can be tuned to vary the 32ms timeout in the LTSSM.	16'd31250
31:16	LTSSM 48ms Time Interval	RW	This register holds the 48ms timer interval in units of (1024 ns). This register can be tuned to vary the 48ms timeout in the LTSSM.	16'd46875

14.1.6.5 LTSSM Timer Control 4

i_regf_ip_pl/i_ltssm_tmr_ctrl_4, Register Offset: 0x1000740

Table 14.34: LTSSM Timer Control 4

Bit Location	Register Field	Attr	Description	Reset
15:0	[LT64MSINT] LTSSM 64ms Time Interval	RW	This register holds the 64ms timer interval in units of (1024 ns). This register can be tuned to vary the 64ms timeout in the LTSSM.	0xF424

14.1.7 LTSSM Transition Freeze, Interrupt Control Status Registers

The LTSSM can be programmed to freeze after specific LTSSM transition events. The LTSSM can also be programmed to assert an interrupt on specific LTSSM transition events. This can be programmed in the following registers:

14.1.7.1 i_ltssm_trans_debug_ctrl_reg_0

i_regf_ip_pl/i_ltssm_trans_debug_ctrl_reg_0, Register Offset: 0x10006CC

This register enables firmware to program 2 specific LTSSM state transitions to be detected and optionally paused for firmware control.

Table 14.35: i_ltssm_trans_debug_ctrl_reg_0

Bit Location	Register Field	Attr	Description	Reset
6:0	Current LTSSM State 0	RW	Current LTSSM State 0. This is the 7-bit Current LTSSM State of LTSSM transition that is required to be checked and optionally paused. Please refer to the 7-bit LTSSM Encoding table for details.	
14:8	Previous LTSSM State 0	RW	Previous LTSSM State 0. This is the 7-bit Previous LTSSM State of LTSSM transition that is required to be checked. Please refer to the 7-bit LTSSM Encoding table for details.	
22:16	Current LTSSM State 1	RW	Current LTSSM State 1. This is the 7-bit Current LTSSM State of LTSSM transition that is required to be checked and optionally paused. Please refer to the 7-bit LTSSM Encoding table for details.	
30:24	Previous LTSSM State 1	RW	Previous LTSSM State 1. This is the 7-bit Previous LTSSM State of LTSSM transition that is required to be checked. Please refer to the 7-bit LTSSM Encoding table for details.	

14.1.7.2 i_ltssm_trans_debug_ctrl_reg_1

i_regf_ip_pl/i_ltssm_trans_debug_ctrl_reg_1, Register Offset: 0x10006D0

This register enables firmware to program 2 specific LTSSM state transitions to be detected and optionally paused for firmware control.

Table 14.36: i_ltssm_trans_debug_ctrl_reg_1

Bit Location	Register Field	Attr	Description	Reset
6:0	Current LTSSM State 2	RW	Current LTSSM State 2. This is the 7-bit Current LTSSM State of LTSSM transition that is required to be checked and optionally paused. Please refer to the 7-bit LTSSM Encoding table for details.	
14:8	Previous LTSSM State 2	RW	Previous LTSSM State 2. This is the 7-bit Previous LTSSM State of LTSSM transition that is required to be checked. Please refer to the 7-bit LTSSM Encoding table for details.	
22:16	Current LTSSM State 3	RW	Current LTSSM State 3. This is the 7-bit Current LTSSM State of LTSSM transition that is required to be checked and optionally paused. Please refer to the 7-bit LTSSM Encoding table for details.	
30:24	Previous LTSSM State 3	RW	Previous LTSSM State 3. This is the 7-bit Previous LTSSM State of LTSSM transition that is required to be checked. Please refer to the 7-bit LTSSM Encoding table for details.	

14.1.7.3 i_ltssm_trans_debug_ctrl_reg_2

i_regf_ip_pl/i_ltssm_trans_debug_ctrl_reg_2, Register Offset: 0x10006D4

This register enables firmware to program 2 specific LTSSM state transitions to be detected and optionally paused for firmware control.

Table 14.37: i_ltssm_trans_debug_ctrl_reg_2

Bit Location	Register Field	Attr	Description	Reset
6:0	Current LTSSM State 4	RW	Current LTSSM State 4. This is the 7-bit Current LTSSM State of LTSSM transition that is required to be checked and optionally paused. Please refer to the 7-bit LTSSM Encoding table for details.	

Table 14.37: i_ltssm_trans_debug_ctrl_reg_2

Bit Location	Register Field	Attr	Description	Reset
14:8	Previous LTSSM State 4	RW	Previous LTSSM State 4. This is the 7-bit Previous LTSSM State of LTSSM transition that is required to be checked. Please refer to the 7-bit LTSSM Encoding table for details.	
22:16	Current LTSSM State 5	RW	Current LTSSM State 5. This is the 7-bit Current LTSSM State of LTSSM transition that is required to be checked and optionally paused. Please refer to the 7-bit LTSSM Encoding table for details.	
30:24	Previous LTSSM State 5	RW	Previous LTSSM State 5. This is the 7-bit Previous LTSSM State of LTSSM transition that is required to be checked. Please refer to the 7-bit LTSSM Encoding table for details.	

14.1.7.4 i_ltssm_trans_debug_ctrl_reg_3

i_regf_ip_pl/i_ltssm_trans_debug_ctrl_reg_3, Register Offset: 0x10006D8

This register enables firmware to program 2 specific LTSSM state transitions to be detected and optionally paused for firmware control.

Table 14.38: i_ltssm_trans_debug_ctrl_reg_3

Bit Location	Register Field	Attr	Description	Reset
6:0	Current LTSSM State 6	RW	Current LTSSM State 6. This is the 7-bit Current LTSSM State of LTSSM transition that is required to be checked and optionally paused. Please refer to the 7-bit LTSSM Encoding table for details.	
14:8	Previous LTSSM State 6	RW	Previous LTSSM State 6. This is the 7-bit Previous LTSSM State of LTSSM transition that is required to be checked. Please refer to the 7-bit LTSSM Encoding table for details.	

Table 14.38: i_ltssm_trans_debug_ctrl_reg_3

Bit Location	Register Field	Attr	Description	Reset
22:16	Current LTSSM State 7	RW	Current LTSSM State 7. This is the 7-bit Current LTSSM State of LTSSM transition that is required to be checked and optionally paused. Please refer to the 7-bit LTSSM Encoding table for details.	
30:24	Previous LTSSM State 7	RW	Previous LTSSM State 7. This is the 7-bit Previous LTSSM State of LTSSM transition that is required to be checked. Please refer to the 7-bit LTSSM Encoding table for details.	

14.1.7.5 i_ltssm_trans_debug_control_status_reg

i_regf_ip_pl/i_ltssm_trans_debug_control_status_reg, Register Offset: 0x10006DC

This register includes ltssm transition Control and Status of the 8 LTSSM State Transitions that can be programmed for firmware monitor and control.

Table 14.39: i_ltssm_trans_debug_control_status_reg

Bit Location	Register Field	Attr	Description	Reset
7:0	LTSSM State Transition Without Freeze Status.	RW1Clr	LTSSM State Transition Without Freeze Status. This field contains 1-bit status for each of the 8 programmable LTSSM State transitions. 1: Indicates the programmed ltssm transition occured and was not programmed to freeze on that transition. 0: Indicates that either the programmed ltssm transition has not occured or was programmed to freeze. Write 1 to Clear	

Table 14.39: i_ltssm_trans_debug_control_status_reg

Bit Location	Register Field	Attr	Description	Reset
15:8	LTSSM State Transition With Freeze Status.	RW1Clr	LTSSM State Transition With Freeze Status. This field contains status 1-bit for each of the 8 programmable LTSSM State transitions. 1: Indicates the programmed ltssm transition occured and was programmed to freeze on that transition. LTSSM is frozen at the Current LTSSM State, waiting for firmware intervention to un-freeze. 0: Indicates that either the programmed ltssm transition has not occured or was not programmed to freeze. Write 1 to Clear.	
23:16	LTSSM State Transition Freeze Enable.	RW	LTSSM State Transition Freeze Enable. This field can be used by firmware to freeze the LTSSM after an LTSSM transition programmed in i_ltssm_trans_debug_status_reg occurs. Each bit is used to enable the corresponding LTSSM State Transition in i_ltssm_trans_debug_status_reg. 1: LTSSM State will be frozen after the programmed transition occurs. LTSSM will stay in current_state, by dropping all OS being received. 0: LTSSM State will not be frozen after the programmed transition. LTSSM will continue to next legal state.	

Table 14.39: i_ltssm_trans_debug_control_status_reg

Bit Location	Register Field	Attr	Description	Reset
31:24	LTSSM State Transition Check Enable.	RW	LTSSM State Transition Check Enable. This bit enables the LTSSM transition check. This field enables the LTSSM transition check programmed in i_ltssm_trans_debug_status_reg . Each bit is used to enable the corresponding LTSSM State Transition in i_ltssm_trans_debug_status_reg. 1: LTSSM State transition check is enabled. 0: LTSSM State transition check is disabled Note: When LTSSM State Transition 1 Check Enable==1, then the Local Interrupt is asserted if the programmed transition occurs and the LTSSM State Transition Interrupt is not masked in the LTSSTTRANM bit.	

14.1.8 LTSSM Transition Cause Status Registers

Controller reflects the details of the each LTSSM transition in status registers. The following registers can be used to check the description of LTSSM transition events.

14.1.8.1 i_regf_ip_pl/i_ltssm_trans_cause_status_reg

Register Offset: 0x10006E0

This register reflects the most recent LTSSM State Transition and the Cause for the Transition.

Table 14.40: i_regf_ip_pl/i_ltssm_trans_cause_status_reg

Name	Bits	Access	Reset value	Description
LTSSM State Transition Cause [LTSTTRC]	23:16	R	hard: 0x00	LTSSM State Transition Cause. This field contains the 8-bit encoded value of the cause of the most recent LTSSM Transition.
Current LTSSM State [CRLTST]	14:8	R	hard: 0x00	Current LTSSM State. This field contains the 7-bit Current LTSSM State of the most recent LTSSM Transition.
Previous LTSSM State [PRLTST]	6:0	R	hard: 0x00	Previous LTSSM State. This field contains the 7-bit Previous LTSSM State of the most recent LTSSM Transition.

14.1.8.2 i_ltssm_transition_cause_enable_[n]

i_regf_ip_pl/i_ltssm_transition_cause_enable_0, Register Offset: 0x1000750

Description: LTSSM Transition Cause Enable Register0

Some LTSSM transitions may not be useful for debug purpose. Such transitions can be masked from being logged. This register controls logging LTSSM Transition cause in i_ltssm_trans_cause_status_reg register. There are about 160 internal LTSSM states in the Controller. This register provides 1-bit per LTSSM state to enable/disable from being reported in i_ltssm_trans_cause_status_reg.

Table 14.41: i_ltssm_transition_cause_enable_[n]

Name	Bits	Access	Reset value	Description
[LTTRCSEN0] LTSSM Transition Cause Reporting Enable	31:0	RW	0xFFFFFFFF	[LTTRCSEN0] LTSSM Transition Cause Reporting Enable for LTSSM States 0 to 31. This register contains one bit per LTSSM state to enable logging the LTSSM Transition cause on entry to that LTSSM state in the i_ltssm_trans_cause_status_reg register. 1: The LTSSM Transition Cause logging is enabled on entry to this state. 0: The LTSSM Transition Cause logging is not enabled on entry to this state.

Bits for Other LTSSM States are implemented in:

LTSSM States[63:32]: i_regf_ip_pl/i_ltssm_transition_cause_enable_1, Register Offset: 0x1000754

LTSSM States[95:64]: i_regf_ip_pl/i_ltssm_transition_cause_enable_2, Register Offset: 0x1000758

LTSSM States[127:96]: i_regf_ip_pl/i_ltssm_transition_cause_enable_3, Register Offset: 0x100075C

LTSSM States[159:128]: i_regf_ip_pl/i_ltssm_transition_cause_enable_4, Register Offset: 0x1000760

14.1.8.3 i_ltssm_trans_cause_fifo_data_out

i_regf_ip_pl/i_ltssm_trans_cause_fifo_data_out, Register Offset: 0x1000764

Description: Controller logs LTSSM Transition events in an internal FIFO. This FIFO can be accessed by reading this register. Up to 64 LTSSM State Transitions and the Cause for the Transitions are logged in this FIFO. This register must be read multiple times to fetch the recorded entries.

Table 14.42: i_ltssm_trans_cause_fifo_data_out

Name	Bits	Access	Reset value	Description
LTSSM Trans Cause FIFO Data Out	31:0	RO	0	Bits[23:16] - LTSSM State Transition Cause Bits[14:8] - Current LTSSM State Bits[6:0] - Previous LTSSM State

14.1.8.4 i_ltssm_trans_cause_fifo_ctrl

i_regf_ip_pl/i_ltssm_trans_cause_fifo_ctrl, Register Offset: 0x100076C

Description: This register is used to control the LTSSM transition cause debug FIFO behaviour.

Table 14.43: i_ltssm_trans_cause_fifo_ctrl

Name	Bits	Access	Reset value	Description
LTSSM Transition Cause FIFO Control [LTCFC]	0	RW	0	1: Store first 64 LTSSM transitions, 0: Store Most recent 64 LTSSM transitions.

14.1.9 7-bit LTSSM State Encoding

The 7-bit LTSSM State encoding used in LTSSM registers is defined below:

Table 14.44: 7-bit LTSSM State Encoding

DETECT_QUIET	7'd0
DETECT_QUIET_ENTRY	7'd1
DETECT_ACTIVE	7'd2
DETECT_ACTIVE_1	7'd3
DETECT_ACTIVE_2	7'd4
DETECT_ACTIVE_3	7'd5
RCVR_DETECTED_ST	7'd6
RCVR_DETECTED_1	7'd7
POLLING_ACTIVE	7'd8
POLLING_ACTIVE_1	7'd9
POLLING_ACTIVE_2	7'd10
POLLING_ACTIVE_3	7'd11
POLLING_COMPLIANCE	7'd12
POLLING_COMPLIANCE_1	7'd13
POLLING_CONFIGConfig_LW	7'd14
POLLING_CONFIG_1	7'd15
POLLING_CONFIG_2	7'd16
CONFIG_LW_START_RC	7'd17
CONFIG_LW_START_RC_1	7'd18
CONFIG_LW_START_RC_2	7'd19
CONFIG_LW_ACC_RC	7'd20
CONFIG_LANENUM_WAIT_RC	7'd21
CONFIG_LANENUM_WAIT_RC_1	7'd22
CONFIG_LANENUM_ACC_RC	7'd23
CONFIG_LW_START_EP	7'd24
CONFIG_LW_START_EP_1	7'd25
CONFIG_LW_START_EP_2	7'd26
CONFIG_LW_ACC_EP	7'd27
CONFIG_LANENUM_WAIT_EP	7'd28
CONFIG_LANENUM_WAIT_EP_1	7'd29
CONFIG_LANENUM_ACC_EP	7'd30
CONFIG_LANENUM_ACC_EP_1	7'd31
DUMMY_STATE_1	7'd32
CONFIG_COMPLETE	7'd33
CONFIG_COMPLETE_1	7'd34
CONFIG_COMPLETE_2	7'd35
CONFIG_IDLE	7'd36

Table 14.44: 7-bit LTSSM State Encoding

CONFIG_IDLE_1	7'd37
DUMMY_STATE_2	7'd38
DUMMY_STATE_3	7'd39
DUMMY_STATE_4	7'd40
L0_STATE	7'd41
RECOVERY_RCVR_LOCK	7'd42
RECOVERY_RCVR_LOCK_1	7'd43
RECOVERY_RCVR_CFG	7'd44
RECOVERY_RCVR_CFG_1	7'd45
RECOVERY_IDLE	7'd46
RECOVERY_IDLE_1	7'd47
DISABLE_LINK	7'd48
DISABLE_LINK_1	7'd49
DISABLE_LINK_2	7'd50
DISABLE_LINK_3	7'd51
DISABLE_LINK_4	7'd52
DISABLE_LINK_5	7'd53
DISABLE_LINK_6	7'd54
DISABLE_LINK_7	7'd55
HOT_RESET	7'd56
HOT_RESET_1	7'd57
HOT_RESET_2	7'd58
HOT_RESET_3	7'd59
L0S_ENTRY	7'd60
L0S_1	7'd61
L0S_2	7'd62
L0S_3	7'd63
L0S_4	7'd64
L0S_5	7'd65
WAIT_FOR_LINK_TX	7'd66
TX_FTS_ENTRY	7'd67
TX_FTS_1	7'd68
TX_FTS_2	7'd69
TX_ELEC_IDLE_ST	7'd70
TX_ELEC_IDLE_1	7'd71
TX_ELEC_IDLE_2	7'd72
TX_ELEC_IDLE_3	7'd73
RECOVERY_SPEED	7'd74
RECOVERY_SPEED_1	7'd75

Table 14.44: 7-bit LTSSM State Encoding

RECOVERY_SPEED_2	7'd76
RECOVERY_SPEED_3	7'd77
POLLING_COMPLIANCE_GEN23	7'd78
POLLING_COMPLIANCE_GEN23_1	7'd79
POLLING_COMPLIANCE_GEN23_2	7'd80
POLLING_COMPLIANCE_GEN23_3	7'd81
POLLING_COMPLIANCE_GEN23_4	7'd82
POLLING_COMPLIANCE_GEN23_5	7'd83
POLLING_COMPLIANCE_GEN23_6	7'd84
POLLING_COMPLIANCE_GEN23_7	7'd85
POLLING_COMPLIANCE_GEN23_8	7'd86
LOOPBACK_SLAVE_ENTRY	7'd87
LOOPBACK_SLAVE_ENTRY_FROM_RECOVERY	7'd88
LOOPBACK_SLAVE_EXIT_1	7'd89
LOOPBACK_SLAVE_EXIT	7'd90
LOOPBACK_SLAVE_GEN2_1	7'd91
LOOPBACK_SLAVE_GEN2_2	7'd92
LOOPBACK_SLAVE_GEN2_3	7'd93
LOOPBACK_SLAVE_GEN2_4	7'd94
LOOPBACK_SLAVE_GEN2_5	7'd95
LOOPBACK_SLAVE_ACTIVE	7'd96
L1_ENTRY	7'd97
L1_1	7'd98
L1_2	7'd99
L1_3	7'd100
L1_4	7'd101
L1_IDLE	7'd102
L1_EXIT	7'd103
L2_ENTRY	7'd104
L2_1	7'd105
L2_2	7'd106
L2_3	7'd107
L2_4	7'd108
L2_5	7'd109
L2_IDLE	7'd110
LOOPBACK_MASTER_ENTRY	7'd111
LOOPBACK_MASTER_ENTRY_1	7'd112
LOOPBACK_MASTER_ENTRY_2	7'd113
LOOPBACK_MASTER_ENTRY_3	7'd114
LOOPBACK_MASTER_ENTRY_4	7'd115

Table 14.44: 7-bit LTSSM State Encoding

LOOPBACK_MASTER_ENTRY_5	7'd116
LOOPBACK_MASTER_ENTRY_FROM_RECOVERY	7'd117
LOOPBACK_MASTER_ACTIVE	7'd118
LOOPBACK_MASTER_EXIT	7'd119
LOOPBACK_MASTER_EXIT_1	7'd120
LOOPBACK_MASTER_EXIT_2	7'd121
RECOVERY_EQUALIZATION_PHASE0	7'd122
RECOVERY_EQUALIZATION_PHASE1	7'd123
RECOVERY_EQUALIZATION_PHASE2_1	7'd124
RECOVERY_EQUALIZATION_PHASE2_2	7'd125
RECOVERY_EQUALIZATION_PHASE3_1	7'd126
RECOVERY_EQUALIZATION_PHASE3_2	7'd127

14.1.10 L0->Recovery LTSSM State Transition Cause Encoding

The LTSSM State Transition Cause for L0->Recovery transition is listed below:

Table 14.45: L0->Recovery LTSSM State Transition Cause Encoding

LTSSM State Transition Cause[7:0]	Description
8’d0	RP Retrain Link. Valid only in RP Mode.
8’d1	EP Link Speed Change Retrain Link. Valid only in EP Mode.
8’d2	EP 8GT/s Request EQ Retrain Link. Valid only in EP Mode.
8’d3	EP 16GT/s Request EQ Retrain Link. Valid only in EP Mode.
8’d4	EP 32GT/s Request EQ Retrain Link. Valid only in EP Mode.
8’d255	EP 64GT/s Request EQ Retrain Link. Valid only in EP Mode.
8’d5	RP Link Disable Retrain. Valid only in RP Mode.
8’d6	RP Hot Reset Retrain. Valid only in RP Mode.
8’d7	RP Loopback Master Retrain. Valid only in RP Mode.
8’d8	NFTS Timeout during L0s.
8’d9	RP Autonomous Equalization Speed Change Retrain Link. Valid only in RP Mode.
8’d10	RP Autonomous Gen1 to Gen2 Speed Change Retrain Link. Valid only in RP Mode.
8’d11	L0->L1 Entry.
8’d12	L0->L2 Entry.
8’d13	Pipe_rxelecidle asserted without EIOS received.
8’d14	Pipe_rxvalid deasserted during data stream.
8’d15	TS1/TS2 received during datastream.
8’d16	DL Link Retrain Request asserted.
8’d17	Framing Error Detected locally.
8’d18	FC Timeout Retrain Link.
8’d19	Link Upconfigure Retrain Link initiated by local FW.
8’d20	L0->Tx.L0s entry

Table 14.45: L0->Recovery LTSSM State Transition Cause Encoding

LTSSM State Transition Cause[7:0]	Description
8’d21	Controller FW Initiated IORecal Retrain Request.
8’d22	PHY Initiated IORecal Request.

14.1.11 LTSSM Transition Encoding

Table 14.46: Detect

S.No.	Previous State	Current State	Trans Cause	Transition Reason
1	DETECT_QUIET	DETECT_QUIET	8'd0	Enable link training not programmed
2		DETECT_ACTIVE	8'd1	Electrical Idle is broken on any lane
3			8'd2	Electrical Idle is broken on any lane with 100 us wait timer
4			8'd3	Electrical Idle is broken on any lane with 1 ms wait timer
5			8'd4	Electrical Idle is broken on any lane with 2 ms wait timer
6			8'd5	After 12 ms time-out. (Timeout value will be 120 us if enable fast link training is programmed)
7	DETECT_QUIET_ENTRY	DETECT_QUIET	8'd0	PHY status has been de-asserted.
8			8'd1	PHY has completed the power state transition to P1
9			8'd2	After 12 ms time-out.
10	DETECT_ACTIVE	DETECT_ACTIVE_1	8'd0	PHY Status received.
11	DETECT_ACTIVE_1	RCVR_DETECTED_ST	8'd0	Receiver has been detected on all the lanes.
12		DETECT_ACTIVE_2	8'd1	Receiver has been detected on some lanes.
13		DETECT_QUIET_ENTRY	8'd2	No receiver has been detected.
14			8'd3	After 12 ms time-out.
15	DETECT_ACTIVE_2	DETECT_ACTIVE_3	8'd0	Restarted receiver detection sequence after 12 ms time-out. (Time out will be 120 us if enable fast link training is programmed)
16	DETECT_ACTIVE_3	RCVR_DETECTED_ST	8'd0	Receiver detected on same lanes as previous state.
17		DETECT_QUIET_ENTRY	8'd1	Receiver detected on different lanes as previous state.
18		DETECT_QUIET_ENTRY	8'd2	No receiver detected and 12 ms time-out expired.
19	RCVR_DETECTED_ST	RCVR_DETECTED_1	8'd0	Always done after setting the PHY status as P0.
20	RCVR_DETECTED_1	POLLING_ACTIVE	8'd0	Always done once the PHY status is received.

Table 14.47: Polling

S.No.	Previous State	Current State	Trans Cause	Transition Reason
1	POLLING_ACTIVE	POLLING_COMPLIANCE	8'd0	Reg enter compliance set and not gen23 capable
2		POLLING_COMPLIANCE _GEN23	8'd1	Reg enter compliance set and gen23 capable
3		POLLING_ACTIVE_1	8'd2	Reg enter compliance not set

Table 14.47: Polling

S.No.	Previous State	Current State	Trans Cause	Transition Reason
4	POLLING_ACTIVE_1	POLLING_CONFIG	8'd0	Any Lane, which detected a Receiver during Detect, received eight consecutive training sequences (or their complement) satisfying any of the following conditions: 1 TS1 with Lane and Link numbers set to PAD and the Compliance Receive bit (bit 4 of Symbol 5) is 0b. 2 TS1 with Lane and Link numbers set to PAD and the Loopback bit (bit 2 of Symbol 5) is 1b. 3 TS2 with Lane and Link numbers set to PAD and a minimum of 1024 TS1 Ordered Sets are transmitted after receiving one TS1 or TS2 Ordered Set. At least a predetermined set of Lanes that detected a Receiver during Detect have detected an exit from Electrical Idle at least once since entering Polling.Active. Note: This may prevent one or more bad Receivers or Transmitters from holding up a valid Link from being configured, and allow for additional training in Polling.Configuration. The exact set of predetermined Lanes is implementation specific. Note: • Up to [PCIe-1.1] this predetermined set was equal to the total set of Lanes that detected a Receiver. • Any Lane that receives eight consecutive TS1 or TS2 Ordered Sets should have detected an exit from Electrical Idle at least once since entering Polling.Active.

Table 14.47: Polling

S.No.	Previous State	Current State	Trans Cause	Transition Reason
5	POLLING_ACTIVE_1	POLLING_COMPLIANCE	8'd1	If either (a) or (b) is true: a. not all Lanes from the predetermined set of Lanes from (ii) above have detected an exit from Electrical Idle since entering Polling.Active. b. any Lane that detected a Receiver during Detect received eight consecutive TS1 Ordered Sets (or their complement) with the Lane and Link numbers set to PAD, the Compliance+F7 Receive bit (bit 4 of Symbol 5) is 1b, and the Loopback bit (bit 2 of Symbol 5) is 0b. Note: If a passive test load is applied on all Lanes then the device will go to Polling.Compliance. Implementation Check if one or more lanes that detected Rx are still in electrical idle Go into Polling.Compliance State
6	POLLING_ACTIVE_1	POLLING_ COMPLIANCE_GEN23	8'd2	If either (a) or (b) is true: a. not all Lanes from the predetermined set of Lanes from (ii) above have detected an exit from Electrical Idle since entering Polling.Active. b. any Lane that detected a Receiver during Detect received eight consecutive TS1 Ordered Sets (or their complement) with the Lane and Link numbers set to PAD, the Compliance+F7 Receive bit (bit 4 of Symbol 5) is 1b, and the Loopback bit (bit 2 of Symbol 5) is 0b. Note: If a passive test load is applied on all Lanes then the device will go to Polling.Compliance. Implementation Check if one or more lanes that detected Rx are still in electrical idle Go into Polling.Compliance State Also if gen23 capable
7	POLLING_ACTIVE_1		8'd3	Any 8 TS1 received with the compliance receive
8	POLLING_ACTIVE_1	TX_ELEC_IDLE_ST	8'd4	If TS1 not received with the compliance set
9	POLLING_ACTIVE_1	POLLING_CONFIG	8'd5	1024 TS1 ordered sets transmitted and 8 TS1 or TS2 ordered sets received on all active channels
10	POLLING_COMPLIANCE	POLLING_ACTIVE	8'd0	Return to POLLING ACTIVE if lane transitions out of elec idle
11	POLLING_COMPLIANCE_GEN23	POLLING_COMPLIANCE _GEN23_1	8'd0	EIOS to be transmitted before Compliance Pattern so sending IDL ordered sets first
12		POLLING_COMPLIANCE _GEN23_4	8'd1	Compliance rate at Gen1 and tx_state at TX_COMPLIANCE_1
13		POLLING_COMPLIANCE _GEN23_2	8'd2	No need to send IDL ordered sets before compliance transmit

Table 14.47: Polling

S.No.	Previous State	Current State	Trans Cause	Transition Reason
14	POLLING_COMPLIANCE_GEN23_1	POLLING_COMPLIANCE _GEN23_2	8'd0	Second IDL sequence sent
15	POLLING_COMPLIANCE_GEN23_2	POLLING_COMPLIANCE _GEN23_3	8'd0	Drive SERDES to electrical idle
16	POLLING_COMPLIANCE_GEN23_3	POLLING_COMPLIANCE _GEN23_4	8'd0	Sending compliance pattern or modified compliance pattern after waiting for 2ms timeout (tx_state = TX_COMPLIANCE_1)
17	POLLING_COMPLIANCE_GEN23_4	POLLING_COMPLIANCE _GEN23_5	8'd0	One or more lane exited electrical idle
18			8'd1	Link control 2 register bit changed from 1 to 0 (enter compliance bit)
19			8'd2	
20	POLLING_COMPLIANCE_GEN23_5	POLLING_ACTIVE	8'd0	If currently operating at 5G, or if we entered Polling.Compliance because of setting the Enter Compliance bit of Link Control 2 Register, we must transmit 8 EIOS sequences before going back to Polling.Active.
21		POLLING_COMPLIANCE _GEN23_6	8'd1	If compliance bit of the link control 2 register is still set
22	POLLING_COMPLIANCE_GEN23_6	POLLING_COMPLIANCE _GEN23_7	8'd0	6 clocks of idle seq transmitted
23	POLLING_COMPLIANCE_GEN23_7	POLLING_COMPLIANCE _GEN23_8	8'd0	Drive SERDES to electrical idle
24	POLLING_COMPLIANCE_GEN23_8	POLLING_ACTIVE	8'd0	Set speed wait for 8 clks after tx_elec_idle_reg before changing current_speed required with variable_core_clk_en, to ensure pipe_rate does not change before pipe_tx_elec_idle
25	POLLING_CONFIG	POLLING_CONFIG_1	8'd0	Always done
26	POLLING_CONFIG_1	POLLING_CONFIG_2	8'd0	Valid TS2 is received
27		TX_ELEC_IDLE_ST	8'd1	Error free TS2 not received and 48ms timeout counter expired
28	POLLING_CONFIG_2	CONFIG_LW_START_RC	8'd0	Wait until 8 consecutive error-free TS2 ordered sets are received on any lane that detected a receiver, and 16 TS2 ordered sets are transmitted. When port is RP
29		CONFIG_LW_START_EP	8'd1	Wait until 8 consecutive error-free TS2 ordered sets are received on any lane that detected a receiver, and 16 TS2 ordered sets are transmitted. When port is EP
30		TX_ELEC_IDLE_ST	8'd2	8 Error free TS2 not received on detected lanes and 48ms timeout counter expired

Table 14.48: Configuration

S.No.	Previous State	Current State	Trans Cause	Transition Reason
1	CONFIG_LW_ START_RC	DISABLE_LINK	8'd0	Link disable request received or rx has asserted disable all links
2		LOOPBACK_MASTER_ENTRY	8'd1	If downstream port and loopback requested
3		LOOPBACK_SLAVE_ENTRY	8'd2	If upstream port and loopback is requested
4		LOOPBACK_SLAVE_ENTRY	8'd3	Next state is Loopback if any Lane that is transmitting TS1 Ordered Sets receives two consecutive TS1 Ordered Sets with the Loopback bit asserted and with the Enhanced Link Behavior Control bits set to 01b.
5		CONFIG_LW_START_RC_1	8'd4	If loopback bit is not set or received
6	CONFIG_LW_ START_RC_1	DISABLE_LINK	8'd0	Link disable request received or rx has asserted disable all links
7		LOOPBACK_MASTER_ENTRY	8'd1	If downstream port and loopback requested
8		LOOPBACK_SLAVE_ENTRY	8'd2	If upstream port and loopback is requested
9			8'd3	Next state is Loopback if any Lane that is transmitting TS1 Ordered Sets receives two consecutive TS1 Ordered Sets with the Loopback bit asserted and with the Enhanced Link Behavior Control bits set to 01b.
10		CONFIG_LW_START_RC_2	8'd4	2nd TS1 received with PAD
11		TX_ELEC_IDLE_ST	8'd5	2nd TS1 with PAD not received and 24 ms timeout counter expired
12		CONFIG_LW_START_RC_2	8'd6	Look for a PAD/PAD link/lane number to be received on any lane
13		CONFIG_LW_START_EP	8'd7	Cross link enabled and cross link timer timed out
14		TX_ELEC_IDLE_ST	8'd8	24 ms timeout counter expired
15	CONFIG_LW_ START_RC_2	LOOPBACK_SLAVE_ENTRY	8'd0	If the loopback bit is found set in two consecutive TS1 frames received on all active lanes, go to LOOPBACK_SLAVE state.
16			8'd1	Next state is Loopback if any Lane that is transmitting TS1 Ordered Sets receives two consecutive TS1 Ordered Sets with the Loopback bit asserted and with the Enhanced Link Behavior Control bits set to 01b.
17		CONFIG_LW_ACC_RC	8'd2	If any 2 link ID matched in the TS ordered sets
18		TX_ELEC_IDLE_ST	8'd3	24 ms timeout counter expired
19	CONFIG_LW_ACC_RC	CONFIG_LANENUM_WAIT_RC	8'd0	Wait for 4 TS cycles to allow link ids to be received on all good lanes. Assign lane numbers And if tx_state = TX_TS_ID_LAST
20		TX_ELEC_IDLE_ST	8'd1	2 TS1 received with PAD or 24 ms timeout counter expired
21	CONFIG_LANENUM _WAIT_RC	CONFIG_LANENUM_WAIT_RC_1	8'd0	Always transitions

Table 14.48: Configuration

S.No.	Previous State	Current State	Trans Cause	Transition Reason
22	CONFIG_LANENUM _WAIT_RC_1	CONFIG_LANENUM_ACC_RC	8'd0	Received the TS ordered sets with no change in the transmitted lane numbers (When modified TS are used)
23		TX_ELEC_IDLE_ST	8'd1	2 TS1 received with PAD or 24 ms timeout counter expired (When modified TS are used)
24		CONFIG_LANENUM_ACC_RC	8'd2	Received the TS ordered sets with no change in the transmitted lane numbers
25		TX_ELEC_IDLE_ST	8'd3	2 TS1 received with PAD or 24 ms timeout counter expired
26	CONFIG_LANENUM _ACC_RC	CONFIG_COMPLETE	8'd0	Wait for APN ports to be set with correct TS2 values before entering CONFIG COMPLETE (Modified TS are used)
27		CONFIG_LANENUM_WAIT_RC	8'd1	Link ID and lane number not matched on all the active lanes (Modified TS are used)
28		TX_ELEC_IDLE_ST	8'd2	2 TS1 received with PAD or 24 ms timeout counter expired (Modified TS are used)
29		CONFIG_COMPLETE	8'd3	Link ID and Lane number all matched.
30		CONFIG_LANENUM_WAIT_RC	8'd4	Link ID and lane number not matched on all the active lanes
31		TX_ELEC_IDLE_ST	8'd5	2 TS1 received with PAD or 24 ms timeout counter expired
27	CONFIG_LW_ START_EP	DISABLE_LINK	8'd0	Rx assert disable link
28		LOOPBACK_SLAVE_ENTRY	8'd1	Next state is Loopback if all Lanes that are transmitting TS1 Ordered Sets, that are also receiving TS1 Ordered Sets , receive the Loopback bit Loopback bit asserted in two consecutive TS1 Ordered Sets. TS1 Ordered Sets
29			8'd2	Next state is Loopback if Any Lane that is transmitting TS1 Ordered Sets receives two consecutive TS1 Ordered Sets with the Loopback bit asserted and with the Enhanced Link Behavior Control bits set to 01b.
30		CONFIG_LW_START_EP_1	8'd3	If loopback bit is not set or received
31	CONFIG_LW _START_EP_1	LOOPBACK_SLAVE_ENTRY	8'd0	If the loopback bit is found set in two consecutive TS1 frames received on all active lanes, go to LOOPBACK_SLAVE state.
32			8'd1	Next state is Loopback if Any Lane that is transmitting TS1 Ordered Sets receives two consecutive TS1 Ordered Sets with the Loopback bit asserted and with the Enhanced Link Behavior Control bits set to 01b.
33		DISABLE_LINK	8'd2	Rx assert disable link
34		CONFIG_LW_START_EP_2	8'd3	Look for a valid non-PAD link number to be received on any lane
35		CONFIG_LW_START_RC	8'd4	If cross link enable and cross link timer timed out
36		TX_ELEC_IDLE_ST	8'd5	24 ms timeout counter expired

Table 14.48: Configuration

S.No.	Previous State	Current State	Trans Cause	Transition Reason
37	CONFIG_LW_ START_EP_2	LOOPBACK_SLAVE_ENTRY	8'd0	If the loopback bit is found set in two consecutive TS1 frames received on all active lanes, go to LOOPBACK_SLAVE state.
38			8'd1	Next state is Loopback if Any Lane that is transmitting TS1 Ordered Sets receives two consecutive TS1 Ordered Sets with the Loopback bit asserted and with the Enhanced Link Behavior Control bits set to 01b.
39		DISABLE_LINK	8'd2	Rx assert disable link
40		CONFIG_LW_ACC_EP	8'd3	Link ID is configured
41		TX_ELEC_IDLE_ST	8'd4	24 ms timeout counter expired
42	CONFIG_LW_ACC_EP	CONFIG_LANENUM_WAIT_EP	8'd0	Wait for Last TX TS state to start sending MTS1
43		CONFIG_LANENUM_WAIT_EP	8'd1	If modified TS1 are not to be used
44		TX_ELEC_IDLE_ST	8'd2	2 TS1 received with PAD or 24 ms timeout counter expired
45	CONFIG_LANENUM _WAIT_EP	CONFIG_LANENUM_WAIT_EP_1	8'd0	Always transitions
46	CONFIG_LANENUM _WAIT_EP_1	CONFIG_LANENUM_ACC_EP	8'd0	If any Lane receives two consecutive TS2 Ordered Sets (Modified TS are used)
47		TX_ELEC_IDLE_ST	8'd1	2 TS1 received with PAD or 2 ms timeout counter expired (Modified TS are used)
48		CONFIG_LANENUM_ACC_EP	8'd2	Received the TS ordered sets with no change in the transmitted lane numbers
49		TX_ELEC_IDLE_ST	8'd3	2 TS1 received with PAD or 2 ms timeout counter expired
50	CONFIG_LANENUM _ACC_EP	CONFIG_LANENUM_ACC_EP_1	8'd0	Always transitions

Table 14.48: Configuration

S.No.	Previous State	Current State	Trans Cause	Transition Reason
51	CONFIG_LANENUM _ACC_EP_1	CONFIG_COMPLETE	8'd0	If two consecutive TS2 Ordered Sets are received with non-PAD Link and non-PAD Lane numbers that match all non-PAD Link and non-PAD Lane numbers that are being transmitted in Upstream Lane TS1 Ordered Sets, the next state is Configuration.Complete (Modified TS are used)
52		CONFIG_LANENUM_WAIT_EP	8'd1	If lane number configured and not received on all the active lanes (Modified TS are used)
53		TX_ELEC_IDLE_ST	8'd2	2 TS1 received with PAD or 24 ms timeout counter expired (Modified TS are used)
54			8'd3	2 TS1 received with PAD or 24 ms timeout counter expired (Modified TS are used)
55		CONFIG_COMPLETE	8'd4	If two consecutive TS2 Ordered Sets are received with non-PAD Link and non-PAD Lane numbers that match all non-PAD Link and non-PAD Lane numbers that are being transmitted in Upstream Lane TS1 Ordered Sets, the next state is Configuration.Complete
56		CONFIG_LANENUM_WAIT_EP	8'd5	If lane number configured and not received on all the active lanes
57		TX_ELEC_IDLE_ST	8'd6	2 TS1 received with PAD or 24 ms timeout counter expired
58			8'd7	2 TS1 received with PAD or 24 ms timeout counter expired
59	CONFIG_COMPLETE	CONFIG_COMPLETE_1	8'd0	Always transitions
60	CONFIG_COMPLETE_1	CONFIG_COMPLETE_2	8'd0	Valid modified TS2 received (Modified TS are used)
61		TX_ELEC_IDLE_ST	8'd1	2ms timeout counter expired and gen3 idle to rlock transitioned is FFh (Modified TS are used)
62		CONFIG_IDLE	8'd2	2ms timeout counter expired and tx_state = TX_COM (Modified TS are used)
63		CONFIG_COMPLETE_2	8'd3	Valid modified TS2 received
64		TX_ELEC_IDLE_ST	8'd4	2ms timeout counter expired and gen3 idle to rlock transitioned is FFh
65		CONFIG_IDLE	8'd5	2ms timeout counter expired and tx_state = TX_COM

Table 14.48: Configuration

S.No.	Previous State	Current State	Trans Cause	Transition Reason
66	CONFIG_COMPLETE_2	DETECT_QUIET	8'd0	CXL retimer mismatch occurred or CXL flex bus enable bits mismatch occurred (Modified TS are used)
67		CONFIG_IDLE	8'd1	CXL retimer and flex bus enable are aligned and 8 error free modified TS are received (Modified TS are used)
68		TX_ELEC_IDLE_ST	8'd2	2ms timeout counter expired and gen3 idle to rlock transition is FFh (Modified TS are used)
69		CONFIG_IDLE	8'd3	2ms timeout counter expired and tx_state = TX_COM (Modified TS are used)
70			8'd4	8 error free TS2 received
71		TX_ELEC_IDLE_ST	8'd5	2ms timeout counter expired and gen3 idle to rlock transitioned
72		CONFIG_IDLE	8'd6	2ms timeout counter expired and tx_state = TX_COM
73	CONFIG_IDLE	CONFIG_IDLE_1	8'd0	Tx TS last cycle or Tx skip last cycle
74		TX_ELEC_IDLE_ST	8'd1	2ms timeout counter expired and gen3 idle to rlock transitioned is FFh
75		RECOVERY_RCVR_LOCK	8'd2	If the idle_to_rlock_transitioned variable is less than FFh, the next state is Recovery.RcvrLock.
76	CONFIG_IDLE_1	L0_STATE	8'd0	CXL mode enabled and CXL link up acquired
77		TX_ELEC_IDLE_ST	8'd1	2ms timeout counter expired and gen3 idle to rlock transitioned is FFh
78			8'd2	2ms timeout counter expired and gen3 idle to rlock transitioned is FFh
79		RECOVERY_RCVR_LOCK	8'd3	If the idle_to_rlock_transitioned variable is less than FFh, the next state is Recovery.RcvrLock.

Table 14.49: L0

S.No.	Previous LTSSM State	Current LTSSM State	LTSSM Debug Cause	Description
1	L0_STATE(7'd41)	RECOVERY_RCVR_LOCK(7'd42)	0	RP initiated retrain from CFG "Retrain Link"
2	L0_STATE	RECOVERY_RCVR_LOCK	1	EP Initiated Speed Change Retrain Link
3	L0_STATE	RECOVERY_RCVR_LOCK	2	EP 8GT Request EQ Retrain Link
4	L0_STATE	RECOVERY_RCVR_LOCK	3	EP 16GT Request EQ Retrain Link
5	L0_STATE	RECOVERY_RCVR_LOCK	4	EP 32GT Request EQ Retrain Link
6	L0_STATE	RECOVERY_RCVR_LOCK	5	RP Initiated Disable Link Request
7	L0_STATE	RECOVERY_RCVR_LOCK	6	RP Initiated Hot Reset Request
8	L0_STATE	RECOVERY_RCVR_LOCK	7	RP Initiated Loopback Request
9	L0_STATE	RECOVERY_RCVR_LOCK	8	NFTS Timeout
10	L0_STATE	RECOVERY_RCVR_LOCK	9	RP Auto Equalization Speed Change to Gen3 and higher
11	L0_STATE	RECOVERY_RCVR_LOCK	10	RP Auto Speed Change to Gen1

Table 14.49: L0

S.No.	Previous LTSSM State	Current LTSSM State	LTSSM Debug Cause	Description
12	L0_STATE	L1_ENTRY (7'd96)	11	Upper Layer initiated L1 entry
13	L0_STATE	L2_ENTRY(7'd103)	12	Upper Layer Initiated L2 entry
14	L0_STATE	RECOVERY_RCVR_LOCK	13	Error case: RX ELEC IDLE detected without EIOS
15	L0_STATE	RECOVERY_RCVR_LOCK	14	Error case: PIPE_RX_VALID de-asserted in L0
16	L0_STATE	RECOVERY_RCVR_LOCK	15	TS1 Received in L0 state
17	L0_STATE	RECOVERY_RCVR_LOCK	16	DL Replay Rollover Retrain Link Request
18	L0_STATE	RECOVERY_RCVR_LOCK	17	Framing Error Retrain Link
19	L0_STATE	RECOVERY_RCVR_LOCK	18	FC Timeout Retrain Link
20	L0_STATE	RECOVERY_RCVR_LOCK	19	EP/RP Link Upconfigure Retrain Link
21	L0_STATE	L0S_ENTRY (7'd58)	20	Tx.L0s Entry due to Tx inactivity
22	L0_STATE	RECOVERY_RCVR_LOCK	21	LM initiated IORECAL Request Retrain Link
23	L0_STATE	RECOVERY_RCVR_LOCK	22	PHY Initiated IORecal Request

Table 14.50: Loopback

S.No.	Previous State	Current State	Trans Cause	Transition Reason
1	LOOPBACK_SLAVE__ENTRY	LOOPBACK_SLAVE_GEN2_1	8'd0	Current speed is not equal to the loopback rate
2		LOOPBACK_SLAVE_GEN2_4	8'd1	Current speed is equal to the loopback rate
3	LOOPBACK_SLAVE _ENTRY_FROM_RECOVERY	LOOPBACK_SLAVE_ACTIVE	8'd0	Wait till SDS is completely transmitted in Gen3 or idle symbols in Gen1, Gen2 transition when tx_ts_last_cycle is asserted during SDS implement a timeout of 24 cycles when end_sds_ordered_set==1, and 8 cycles when send_sds_ordered_set==0
4		LOOPBACK_SLAVE_GEN2_4	8'd1	tx_state = TX_COM or TX_EIEOS
5	LOOPBACK_SLAVE_GEN2_1	LOOPBACK_SLAVE_GEN2_2	8'd0	Second idle seq sent
6	LOOPBACK_SLAVE_GEN2_2	LOOPBACK_SLAVE_GEN2_3	8'd0	Always transitions
7	LOOPBACK_SLAVE_GEN2_3	LOOPBACK_SLAVE_ACTIVE	8'd0	Next state is Loopback.Active if the Compliance Receive bit of the TS1 Ordered Sets that directed the slave to this state was asserted. The slave's transmitter does not need to transition to transmitting looped-back data on any boundary, and it is permitted to truncate any Ordered Set in progress.
8		LOOPBACK_SLAVE_GEN2_4	8'd1	tx_state = TX_COM or TX_EIEOS

Table 14.50: Loopback

S.No.	Previous State	Current State	Trans Cause	Transition Reason
9	LOOPBACK_SLAVE_GEN2_4	RECOVERY_EQUALIZATION _PHASE0	8'd0	Next state is Recovery.Equalization if Loopback.Entry was entered from Configuration.Linkwidth.Start , the highest common data rate is 32.0 GT/s and the Enhanced Link Behavior Control bits of the TS1 Ordered Sets that directed the slave to this state were 01b. - The perform_equalization_for_loopback variable is set to 1b. - The transmit_modified _compliance_pattern_in_loopback variable is set to 1b if the Transmit Modified Compliance Pattern in Loopback bit is set to 1b in the TS1 Ordered Sets that directed the slave to this state.
10		LOOPBACK_SLAVE_ ACTIVE	8'd1	If Loopback.Entry was entered from Recovery. Equalization, the next state is Loopback.Active after two consecutive TS1 Ordered Sets with the Loopback bit asserted are received by the Lane under test.
11			8'd2	TS1 received.
12		LOOPBACK_SLAVE_GEN2_5	8'd3	ts1_eq_received_2_with_non_zero_ec_ all[3:2] != 'b0
13		LOOPBACK_SLAVE_ACTIVE	8'd4	Received loopback bit set
14	LOOPBACK_SLAVE_GEN2_5	LOOPBACK_SLAVE_ACTIVE	8'd0	Always transitions
15	LOOPBACK_SLAVE_ACTIVE	LOOPBACK_SLAVE_EXIT	8'd0	4 Idle sequence detected
16			8'd1	Speed gen1 and any idle sequence detected
17			8'd2	Speed gen1 and electrical idle inferred
18	LOOPBACK_SLAVE_EXIT	LOOPBACK_SLAVE_EXIT_1	8'd0	Tx data valid received
19	LOOPBACK_SLAVE_EXIT_1	TX_ELEC_IDLE_ST	8'd0	2ms timeout counter expired after asserting the tx electrical idle for one cycle after turning off loopback
20	LOOPBACK_MASTER_ENTRY	LOOPBACK_MASTER_EN-TRY_1	8'd0	Need to switch the current speed to the loopback rate
21		LOOPBACK_MASTER_EN-TRY_5	8'd1	Speed switch not required
22	LOOPBACK_MASTER_EN-TRY_1	LOOPBACK_MASTER_EN-TRY_2	8'd0	Wait for 16 TS1 to be sent.
23	LOOPBACK_MASTER_EN-TRY_2	LOOPBACK_MASTER_EN-TRY_3	8'd0	Second idle seq sent
24	LOOPBACK_MASTER_EN-TRY_3	LOOPBACK_MASTER_EN-TRY_4	8'd0	Always transitions
25	LOOPBACK_MASTER_EN-TRY_4	LOOPBACK_MASTER_EN-TRY_5	8'd0	1ms timeout counter expired after changing the current speed to the loopback rate

Table 14.50: Loopback

S.No.	Previous State	Current State	Trans Cause	Transition Reason
26	LOOPBACK_MASTER_ENTRY_5	RECOVERY_EQUALIZATION_PHASE0	8'd0	Next state is Recovery.Equalization if the data rate was changed to 32.0 GT/s and 16 consecutive TS1 Ordered Sets were sent on any Lane with the Enhanced Link Behavior Control bits set to 01b. The perform_equalization_for_loopback variable is set to 1b.
27		LOOPBACK_MASTER_ACTIVE	8'd1	Next state is Loopback.Active if Loopback.Entry was entered from Recovery. Equalization and the Lane under test receives two consecutive TS1 Ordered Sets with the Loopback bit asserted.
28			8'd2	Received 2 TS with loopback bit set
29		LOOPBACK_MASTER_EXIT	8'd3	48ms timeout counter expired
30	LOOPBACK_MASTER_ENTRY_FROM_RECOVERY	LOOPBACK_MASTER_ENTRY_5	8'd0	TX TS last cycle or TX Skip last cycle
31	LOOPBACK_MASTER_ACTIVE	LOOPBACK_MASTER_EXIT	8'd0	Loopback request removed
32	LOOPBACK_MASTER_EXIT	LOOPBACK_MASTER_EXIT_1	8'd0	Always transitions
33	LOOPBACK_MASTER_EXIT_1	LOOPBACK_MASTER_EXIT_2	8'd0	Idle seq sent
34	LOOPBACK_MASTER_EXIT_2	TX_ELEC_IDLE_ST	8'd0	2ms timeout counter expired after driving SERDES to electrical idle

Table 14.51: Recovery

S.No.	Previous State	Current State	Trans Cause	Transition Reason
1	RECOVERY_RCVR_LOCK	RECOVERY_EQUALIZATION_PHASE0	8'd0	Speed at Gen3 or higher and start equalization with preset set
2		RECOVERY_RCVR_LOCK_1	8'd1	Start equalization with preset is not set

Table 14.51: Recovery

S.No.	Previous State	Current State	Trans Cause	Transition Reason
3	RECOVERY_ RCVR_LOCK_1	RECOVERY_EQUALIZATION_PHASE0	8'd0	If 8 consecutive error-free EQ TS1s are received at 8G speed with speed change bit = 0 and Eq Control != 00, transition to Recovery. Equalization.
4		RECOVERY_RCVR_CFG	8'd1	If Recovery was entered from L1, wait for T_COMMON_MODE before transitioning to Recovery.RcvrCfg
5			8'd2	Any 8 error free TS ordered set received
6		RECOVERY_SPEED	8'd3	Gen2 speed changed recovery set
7		CONFIG_LW_START_RC	8'd4	Next state is Configuration if The directed_speed_change variable is equal to 0b and the speed_change bit on the received TS1 or TS2 Ordered Set is equal to 0b, Or The current data rate of operation is 2.5 GT/s and 2.5 GT/s data rate is the highest commonly advertised data rate among the transmitted TS1 Ordered Sets and the received TS1 or TS2 Ordered Set(s). Root port selected
8		CONFIG_LW_START_EP	8'd5	Next state is Configuration if The directed_speed_change variable is equal to 0b and the speed_change bit on the received TS1 or TS2 Ordered Set is equal to 0b, Or The current data rate of operation is 2.5 GT/s and 2.5 GT/s data rate is the highest commonly advertised data rate among the transmitted TS1 Ordered Sets and the received TS1 or TS2 Ordered Set(s). End point selected
9		TX_ELEC_IDLE_ST	8'd6	No valid TS received
10	RECOVERY_ RCVR_CFG	RECOVERY_RCVR_CFG_1	8'd0	Always transitions
11	RECOVERY_ RCVR_CFG_1	RECOVERY_SPEED	8'd0	Any 8 error free TS ordered set received
12			8'd1	8 error free TS received with the equalization request set
13		RECOVERY_IDLE	8'd2	
14		CONFIG_LW_START_RC	8'd3	8 TS1 received with the link or lane number mismatch and RP mode selected
15		CONFIG_LW_START_EP	8'd4	8 TS1 received with the link or lane number mismatch and EP mode selected
16		RECOVERY_SPEED	8'd5	Received electrical idle or received idle sequence or TS not received in recovery rcvr cfg state
17		RECOVERY_IDLE	8'd6	48ms timeout counter expired and gen3 idle to rlock transitioned to FFh
18		TX_ELEC_IDLE_ST	8'd7	48ms timeout counter expired

Table 14.51: Recovery

S.No.	Previous State	Current State	Trans Cause	Transition Reason
19	RECOVERY_IDLE	CONFIG_LW_START_RC	8'd0	Link upconfigure in progress and Root port mode
20		CONFIG_LW_START_EP	8'd1	Link upconfigure in progress and Endpoint mode
21		HOT_RESET	8'd2	Hot reset requested
22		DISABLE_LINK	8'd3	Link disable requested
23		LOOPBACK_MASTER_ENTRY_FROM _RECOVERY	8'd4	Loopback requested
24		RECOVERY_IDLE_1	8'd5	Go to recovery idle 1 set
25		DISABLE_LINK	8'd6	Assert link disable received
26		HOT_RESET	8'd7	Received any 2 request to assert hot reset
27		CONFIG_LW_START_RC	8'd8	Any 2 valid TS1 received. Root port mode
28		CONFIG_LW_START_EP	8'd9	Any 2 valid TS1 received. Endpoint mode
29		LOOPBACK_SLAVE_ENTRY_FROM_RE-COVERY	8'd10	Received loopback bit set
30		TX_ELEC_IDLE_ST	8'd11	2ms timeout counter expired and gen3 idle to rlock transitioned to FFh
31		RECOVERY_RCVR_LOCK	8'd12	2ms timeout counter expired and gen3 idle to rlock transitioned less than FFh
32	RECOVERY_IDLE_1	L0_STATE	8'd0	Idle condition met recovery and sds seq re-ceived all and CXL mode enable
33		DISABLE_LINK	8'd1	Received disable link request
34		HOT_RESET	8'd2	Received hot reset request
35		CONFIG_LW_START_RC	8'd3	Any 2 valid TS1 received. Root port mode
36		CONFIG_LW_START_EP	8'd4	Any 2 valid TS1 received. Endpoint mode
37		LOOPBACK_SLAVE_ENTRY_FROM_RE-COVERY	8'd5	Loopback requested
38		TX_ELEC_IDLE_ST	8'd6	2ms timeout counter expired and gen3 idle to rlock transitioned to FFh
39		RECOVERY_RCVR_LOCK	8'd7	2ms timeout counter expired and gen3 idle to rlock transitioned less than FFh
40	RECOVERY _SPEED	RECOVERY_SPEED_1	8'd0	Always transitions
41	RECOVERY_ SPEED_1	RECOVERY_SPEED_2	8'd0	Second idle sequence sent
42	RECOVERY_ SPEED_2	RECOVERY_SPEED_3	8'd0	Rx lanes enter electrical idle
43		TX_ELEC_IDLE_ST	8'd1	48ms timeout counter expired before rx lanes entered electrical idle
44	RECOVERY_ SPEED_3	RECOVERY_RCVR_LOCK	8'd0	Wait for delay done If start_equalization is set, then link_eq_tx_coefficients received If no_eq_needed is set, then no_eq_coefficients restored
45		TX_ELEC_IDLE_ST	8'd1	48ms timeout counter expired.

Table 14.51: Recovery

S.No.	Previous State	Current State	Trans Cause	Transition Reason
46	RECOVERY_ EQUALIZATION _PHASE0	RECOVERY_EQUALIZATION_PHASE1	8'd0	Root port mode and speed is Gen5
47			8'd1	Endpoint mode and TS1 eq received with non zero ec all
48		LOOPBACK_SLAVE_ENTRY_FROM_RE-COVERY	8'd2	12ms timeout counter expired and gen5 perform equalization for loopback set
49		RECOVERY_SPEED	8'd3	12ms timeout counter expired and gen5 perform equalization for loopback is not set
50	RECOVERY_ EQUALIZATION _PHASE1	RECOVERY_EQUALIZATION_PHASE2_1	8'd0	If the data rate is 32.0 GT/s and perform_ equalization_for_loopback is 0b
51		LOOPBACK_SLAVE_ENTRY_FROM_RE-COVERY	8'd1	Next state is Loopback.Entry if perform_ equalization_for_loopback is 1b and The Lane under test receives eight consecutive TS1 Ordered Sets with EC=00b
52		RECOVERY_RCVR_LOCK	8'd2	Loopback bit not set
53		LOOPBACK_SLAVE_ENTRY_FROM_RE-COVERY	8'd3	Next state is Loopback.Entry if perform_ equalization_for_loopback is 1b and A 12 ms timeout
54		RECOVERY_SPEED	8'd4	Loopback bit not set
55		RECOVERY_RCVR_LOCK	8'd5	Next state is Recovery.RcvrLock if all configured Lanes receive two consecutive TS1 Ordered Sets with EC=01b, perform_equalization_for_loopback is 0b and the Downstream Port does not want to execute Phase 2 and Phase 3.
56		LOOPBACK_MASTER_ENTRY_FROM_ RECOVERY	8'd6	Next state is Loopback.Entry if perform_ equalization_for_loopback is 1b and one of the following conditions is satisfied: a. The Lane under test receives two consecutive TS1 Ordered Sets with EC=01b and the Downstream Port does not want to execute Phase 2 and Phase 3.
57		RECOVERY_EQUALIZATION_PHASE3_1	8'd7	Loopback bit is not set and phase 2 and phase 3 of equalization is not bypassed
58		LOOPBACK_MASTER_ENTRY_FROM _RECOVERY	8'd8	Next state is Loopback.Entry if perform_ equalization_for_loopback is 1b and one of the following conditions is satisfied: a. 24ms timeout counter expired
59		RECOVERY_SPEED	8'd9	Loopback bit is not set and 24ms timeout counter expired

Table 14.51: Recovery

S.No.	Previous State	Current State	Trans Cause	Transition Reason
60	RECOVERY_ EQUALIZATION_ PHASE2_1	RECOVERY_EQUALIZATION_PHASE3_1	8'd0	Not downstream port
61		LOOPBACK_MASTER_ENTRY_FROM _RECOVERY	8'd1	Next state is Loopback.Entry if the data rate of operation is 32.0 GT/s, perform_equalization_for_loopback is 1b and one of the following conditions is satisfied: a. The Lane under test is operating at its optimal setting and all Lanes receive two consecutive TS1 Ordered Sets with retimer Equalization Extend bit set to 0b are received.
62		RECOVERY_RCVR_LOCK	8'd2	Loopback bit not set
63		RECOVERY_EQUALIZATION_PHASE2_2	8'd3	Link equalization is not completed.
64		LOOPBACK_MASTER_ENTRY_FROM_ RECOVERY	8'd4	Next state is Loopback.Entry if the data rate of operation is 32.0 GT/s, perform_equalization_for_loopback is 1b and one of the following conditions is satisfied: a. A 24 ms timeout with a tolerance of -0 ms and +2 ms. Root port mode
65		LOOPBACK_SLAVE_ENTRY_FROM_RECOVERY	8'd5	Next state is Loopback.Entry if the data rate of operation is 32.0 GT/s, perform_equalization_for_loopback is 1b and one of the following conditions is satisfied: a. A 24 ms timeout with a tolerance of -0 ms and +2 ms. Endpoint mode
66		RECOVERY_SPEED	8'd6	Loopback bit not set
67	RECOVERY_ EQUALIZATION _PHASE2_2	RECOVERY_EQUALIZATION_PHASE2_1	8'd0	wait for remote phy to change its coefficients
68			8'd1	wait for 2ms to receive per TX setting in phase2
69		LOOPBACK_MASTER_ENTRY_FROM_ RECOVERY	8'd2	Next state is Loopback.Entry if the data rate of operation is 32.0 GT/s, perform_equalization_for_loopback is 1b and one of the following conditions is satisfied: a. A 24 ms timeout with a tolerance of -0 ms and +2 ms. Root port mode
70		LOOPBACK_SLAVE_ENTRY_FROM_RECOVERY	8'd3	Next state is Loopback.Entry if the data rate of operation is 32.0 GT/s, perform_equalization_for_loopback is 1b and one of the following conditions is satisfied: a. A 24 ms timeout with a tolerance of -0 ms and +2 ms. Endpoint mode
71		RECOVERY_SPEED	8'd4	Loopback bit not set

Table 14.51: Recovery

S.No.	Previous State	Current State	Trans Cause	Transition Reason
72	RECOVERY_ EQUALIZATION _PHASE3_1	RECOVERY_EQUALIZATION_PHASE3_2	8'd0	link_eq_intfc_ack_received = 1'b1
73		LOOPBACK_SLAVE_ENTRY_FROM_RE-COVERY	8'd1	Next state is Loopback.Entry if perform_ equalization_for_loopback is 1b and one of the following conditions is satisfied: a. The Lane under test receives two consective TS1 Ordered Sets with EC=00b.
74		RECOVERY_RCVR_LOCK	8'd2	Loopback bit not set
75		RECOVERY_EQUALIZATION_PHASE2_1	8'd3	TS1 eq received with ec all and root port mode
76		LOOPBACK_MASTER_ENTRY_FROM _RECOVERY	8'd4	Next state is Loopback.Entry if perform_ equalization_for_loopback is 1b and one of the following conditions is satisfied: a. A time-out of 32 ms with a tolerance of -0 ms and +4 ms. Root port mode
77		LOOPBACK_SLAVE_ENTRY_FROM_RE-COVERY	8'd5	Next state is Loopback.Entry if perform_ equalization_for_loopback is 1b and one of the following conditions is satisfied: a. A time-out of 32 ms with a tolerance of -0 ms and +4 ms. Endpoint mode
78		RECOVERY_SPEED	8'd6	Loopback bit not set and 32ms timeout counter expired
79	RECOVERY_ EQUALIZATION _PHASE3_2	RECOVERY_EQUALIZATION_PHASE3_1	8'd0	LTSSM Transition is 1'b0

Table 14.52: Low power

S.No.	Previous State	Current State	Trans Cause	Transition Reason
1	L0S_ENTRY	L0S_1	8'd0	Training sequence grant
2	L0S_1	L0S_2	8'd0	Second idle sequence sent
3	L0S_2	L0S_3	8'd0	Always transitions
4	L0S_3	L0S_4	8'd0	Tx ts cycle count is more than L0s Electrical Idle delay defined.
5	L0S_4	L0S_5	8'd0	Phy status asserted.
6	L0S_5	WAIT_FOR_LINK_TX	8'd0	Tx ts cycle count is more than L0s min delay defined.
7	WAIT_FOR_ LINK_TX	TX_FTS_ENTRY	8'd0	If link tx active detected or request to enter L1 or L2 received or retraining requested
8	TX_FTS_EN-TRY	TX_FTS_1	8'd0	tx_state != TX_TS_ID_SEND_SKIP_GEN3_2
9	TX_FTS_1	TX_FTS_2	8'd0	If speed less than Gen3 and tx_state = TX_TS_ID _SEND_SKIP
10			8'd1	If speed Gen3 or higher and tx_state = TX_SDS
11	TX_FTS_2	L0_STATE	8'd0	If speed less than Gen3 and tx_state = TX_IDLE
12			8'd1	If speed is Gen3
13	L1_ENTRY	L1_1	8'd0	Training sequence grant
14	L1_1	L1_2	8'd0	Second idle sequence sent

Table 14.52: Low power

S.No.	Previous State	Current State	Trans Cause	Transition Reason
15	L1_2	L1_3	8'd0	Always transitions
16	L1_3	L1_4	8'd0	EIOS received and delay of 2 clocks done to exit the state when speed is less than Gen3
17			8'd1	EIOS received and speed is Gen3 or higher
18			8'd2	Internal PIPE width is not 32 or 64 or 128
19		RECOVERY_RCVR_LOCK	8'd3	RX EI detected/inferred without receiving EIOS
20			8'd4	If the link retrains while waitinf ro EIOS, move to Recovery.
21	L1_4	L1_IDLE	8'd0	Tx ts cycle count is more than L1 Electrical Idle delay defined.
22	L1_IDLE	L1_EXIT	8'd0	Exit immediately if L1.x has not started. If it has started, wait for it reach exit state At non-Gen1 speeds wait for a min of 40 ns
23	L1_EXIT	RECOVERY_RCVR_LOCK	8'd0	Phy status asserted.
24	L2_ENTRY	L2_1	8'd0	Training sequence grant
25	L2_1	L2_2	8'd0	Second idle sequence sent
26	L2_2	L2_3	8'd0	Always transitions
27	L2_3	L2_4	8'd0	EIOS or electrical idle received
28	L2_4	L2_IDLE	8'd0	8 clks after receiving EIOS before entering L2_IDLE
29	L2_IDLE	DETECT_QUIET_ENTRY	8'd0	Tx ts cycle count is more than L2 electrical idle delay defined or did not received electrical idle on any of the active lanes.
30			8'd1	

Table 14.53: Disabled, hot reset and others

S.No.	Previous State	Current State	Trans Cause	Transition Reason
1	TX_ELEC_IDLE_ST	TX_ELEC_IDLE_1	8'd0	Always transitions
2	TX_ELEC_IDLE_1	TX_ELEC_IDLE_2	8'd0	Second idle sequence sent
3	TX_ELEC_IDLE_2	TX_ELEC_IDLE_3	8'd0	Current speed higher than Gen1
4		DETECT_ QUIET_ENTRY	8'd1	Current speed Gen1 and tx idle count more than 2
5	TX_ELEC_IDLE_3	DETECT_QUIET_EN-TRY	8'd0	Wait for the recovery speed electrical idle
6	DISABLE_LINK	DISABLE_LINK_1	8'd0	Always transitions
7	DISABLE_LINK_1	DISABLE_LINK_2	8'd0	Always transitions
8	DISABLE_LINK_2	DISABLE_LINK_3	8'd0	tx_state = TX_COM
9	DISABLE_LINK_3	DISABLE_LINK_4	8'd0	Tx ordered set count more than 24
10	DISABLE_LINK_4	DISABLE_LINK_5	8'd0	Speed is Gen2 and 2nd idle seq sent
11			8'd1	Speed is not Gen2

Table 14.53: Disabled, hot reset and others

S.No.	Previous State	Current State	Trans Cause	Transition Reason
12	DISABLE_LINK_5	DISABLE_LINK_6	8'd0	Tx idle count is more than 16 and received electrical idle on the active lanes. Endpoint mode
13			8'd1	2ms timeout counter expired before receiving electrical idle on the lanes. Endpoint mode
14			8'd2	Link disable requested
15	DISABLE_LINK_6	DETECT_ QUIET_ENTRY	8'd0	Current speed is Gen1
16		TX_ELEC_IDLE_3	8'd1	Current speed is higher than Gen1
17	HOT_RESET	HOT_RESET_1	8'd0	tx_state = TX_COM
18	HOT_RESET_1	HOT_RESET_2	8'd0	Root port mode. Hot reset asserted.
19		TX_ELEC_IDLE_ST	8'd1	Root port mode and 2ms timeout counter expired
20			8'd2	Endpoint mode and 2ms timeout counter expired.
21	HOT_RESET_2	TX_ELEC_IDLE_ST	8'd0	Hot reset request de-asserted

14.1.12 LTSSM State Encoding

14.1.12.1 External LTSSM State Encoding

The following table provides the encoding of the LTSSM states visible on the *dbg_ltssm_state[5:0]* output of the Controller and the LTSSM state read from the Physical Layer Debug Status Register 0.

Table 14.54: External LTSSM state encoding

External LTSSM State Encoding	State Value[5:0]
LTSSM_DETECT_QUIET	6'h0
LTSSM_DETECT_ACTIVE	6'h1
LTSSM_POLLING_ACTIVE	6'h2
LTSSM_POLLING_COMPLIANCE	6'h3
LTSSM_POLLING_CONFIGURATION	6'h4
LTSSM_CONFIGURATION_LINKWIDTH_START	6'h5
LTSSM_CONFIGURATION_LINKWIDTH_ACCEPT	6'h6
LTSSM_CONFIGURATION_LANENUM_ACCEPT	6'h7
LTSSM_CONFIGURATION_LANENUM_WAIT	6'h8
LTSSM_CONFIGURATION_COMPLETE	6'h9
LTSSM_CONFIGURATION_IDLE	6'ha
LTSSM_RECOVERY_RCVR_LOCK	6'hb
LTSSM_RECOVERY_SPEED	6'hc
LTSSM_RECOVERY_RCVR_CFG	6'hd
LTSSM_RECOVERY_IDLE	6'he
LTSSM_L0_STATE	6'h10
LTSSM_RX_L0S_ENTRY	6'h11
LTSSM_RX_L0S_IDLE	6'h12
LTSSM_RX_L0S_FTS	6'h13

Table 14.54: External LTSSM state encoding

External LTSSM State Encoding	State Value[5:0]
LTSSM_TX_L0S_ENTRY	6'h14
LTSSM_TX_L0S_IDLE	6'h15
LTSSM_TX_L0S_FTS	6'h16
LTSSM_L1_ENTRY	6'h17
LTSSM_L1_IDLE	6'h18
LTSSM_L2_IDLE	6'h19
LTSSM_L2_TRANSMIT_WAKE	6'h1a
LTSSM_DISABLED	6'h20
LTSSM_LOOPBACK_ENTRY_MASTER	6'h21
LTSSM_LOOPBACK_ACTIVE_MASTER	6'h22
LTSSM_LOOPBACK_EXIT_MASTER	6'h23
LTSSM_LOOPBACK_ENTRY_SLAVE	6'h24
LTSSM_LOOPBACK_ACTIVE_SLAVE	6'h25
LTSSM_LOOPBACK_EXIT_SLAVE	6'h26
LTSSM_HOT_RESET	6'h27
LTSSM_RECOVERY_EQUALIZATION_PHASE0	6'h28
LTSSM_RECOVERY_EQUALIZATION_PHASE1	6'h29
LTSSM_RECOVERY_EQUALIZATION_PHASE2	6'h2A
LTSSM_RECOVERY_EQUALIZATION_PHASE3	6'h2B

14.1.12.2 Internal LTSSM State Encoding

The following table provides the encoding of the internal LTSSM states visible on the *dbg_debug_data_out[9:3]* output of the Controller.

Table 14.55: Internal LTSSM state encoding

Internal LTSSM State Encoding	State Value[6:0]
DETECT_QUIET	7'd0
DETECT_QUIET_ENTRY	7'd1
DETECT_ACTIVE	7'd2
DETECT_ACTIVE_1	7'd3
DETECT_ACTIVE_2	7'd4
DETECT_ACTIVE_3	7'd5
RCVR_DETECTED_ST	7'd6
RCVR_DETECTED_1	7'd7
POLLING_ACTIVE	7'd8
POLLING_ACTIVE_1	7'd9
POLLING_ACTIVE_2	7'd10
POLLING_ACTIVE_3	7'd11
POLLING_COMPLIANCE	7'd12

Table 14.55: Internal LTSSM state encoding

Internal LTSSM State Encoding	State Value[6:0]
POLLING_COMPLIANCE_1	7'd13
POLLING_CONFIG	7'd14
POLLING_CONFIG_1	7'd15
POLLING_CONFIG_2	7'd16
CONFIG_LW_START_RC	7'd17
CONFIG_LW_START_RC_1	7'd18
CONFIG_LW_START_RC_2	7'd19
CONFIG_LW_ACC_RC	7'd20
CONFIG_LANENUM_WAIT_RC	7'd21
CONFIG_LANENUM_WAIT_RC_1	7'd22
CONFIG_LANENUM_ACC_RC	7'd23
CONFIG_LW_START_EP	7'd24
CONFIG_LW_START_EP_1	7'd25
CONFIG_LW_START_EP_2	7'd26
CONFIG_LW_ACC_EP	7'd27
CONFIG_LANENUM_WAIT_EP	7'd28
CONFIG_LANENUM_WAIT_EP_1	7'd29
CONFIG_LANENUM_ACC_EP	7'd30
CONFIG_LANENUM_ACC_EP_1	7'd31
DUMMY_STATE_1	7'd32
CONFIG_COMPLETE	7'd33
CONFIG_COMPLETE_1	7'd34
CONFIG_COMPLETE_2	7'd35
CONFIG_IDLE	7'd36
CONFIG_IDLE_1	7'd37
DUMMY_STATE_2	7'd38
DUMMY_STATE_3	7'd39
DUMMY_STATE_4	7'd40
L0_STATE	7'd41
RECOVERY_RCVR_LOCK	7'd42
RECOVERY_RCVR_LOCK_1	7'd43
RECOVERY_RCVR_CFG	7'd44
RECOVERY_RCVR_CFG_1	7'd45
RECOVERY_IDLE	7'd46
RECOVERY_IDLE_1	7'd47
DISABLE_LINK	7'd48
DISABLE_LINK_1	7'd49
DISABLE_LINK_2	7'd50
DISABLE_LINK_3	7'd51

Table 14.55: Internal LTSSM state encoding

Internal LTSSM State Encoding	State Value[6:0]
DISABLE_LINK_4	7'd52
DISABLE_LINK_5	7'd53
DISABLE_LINK_6	7'd54
DISABLE_LINK_7	7'd55
HOT_RESET	7'd56
HOT_RESET_1	7'd57
HOT_RESET_2	7'd58
HOT_RESET_3	7'd59
L0S_ENTRY	7'd60
L0S_1	7'd61
L0S_2	7'd62
L0S_3	7'd63
L0S_4	7'd64
L0S_5	7'd65
WAIT_FOR_LINK_TX	7'd66
TX_FTS_ENTRY	7'd67
TX_FTS_1	7'd68
TX_FTS_2	7'd69
TX_ELEC_IDLE_ST	7'd70
TX_ELEC_IDLE_1	7'd71
TX_ELEC_IDLE_2	7'd72
TX_ELEC_IDLE_3	7'd73
RECOVERY_SPEED	7'd74
RECOVERY_SPEED_1	7'd75
RECOVERY_SPEED_2	7'd76
RECOVERY_SPEED_3	7'd77
POLLING_COMPLIANCE_GEN23	7'd78
POLLING_COMPLIANCE_GEN23_1	7'd79
POLLING_COMPLIANCE_GEN23_2	7'd80
POLLING_COMPLIANCE_GEN23_3	7'd81
POLLING_COMPLIANCE_GEN23_4	7'd82
POLLING_COMPLIANCE_GEN23_5	7'd83
POLLING_COMPLIANCE_GEN23_6	7'd84
POLLING_COMPLIANCE_GEN23_7	7'd85
POLLING_COMPLIANCE_GEN23_8	7'd86
LOOPBACK_SLAVE_ENTRY	7'd87
LOOPBACK_SLAVE_ENTRY_FROM_RECOVERY	7'd88
LOOPBACK_SLAVE_EXIT_1	7'd89

Table 14.55: Internal LTSSM state encoding

Internal LTSSM State Encoding	State Value[6:0]
LOOPBACK_SLAVE_EXIT	7'd90
LOOPBACK_SLAVE_GEN2_1	7'd91
LOOPBACK_SLAVE_GEN2_2	7'd92
LOOPBACK_SLAVE_GEN2_3	7'd93
LOOPBACK_SLAVE_GEN2_4	7'd94
LOOPBACK_SLAVE_GEN2_5	7'd95
LOOPBACK_SLAVE_ACTIVE	7'd96
L1_ENTRY	7'd97
L1_1	7'd98
L1_2	7'd99
L1_3	7'd100
L1_4	7'd101
L1_IDLE	7'd102
L1_EXIT	7'd103
L2_ENTRY	7'd104
L2_1	7'd105
L2_2	7'd106
L2_3	7'd107
L2_4	7'd108
L2_5	7'd109
L2_IDLE	7'd110
LOOPBACK_MASTER_ENTRY	7'd111
LOOPBACK_MASTER_ENTRY_1	7'd112
LOOPBACK_MASTER_ENTRY_2	7'd113
LOOPBACK_MASTER_ENTRY_3	7'd114
LOOPBACK_MASTER_ENTRY_4	7'd115
LOOPBACK_MASTER_ENTRY_5	7'd116
LOOPBACK_MASTER_ENTRY_FROM_RECOVERY	7'd117
LOOPBACK_MASTER_ACTIVE	7'd118
LOOPBACK_MASTER_EXIT	7'd119
LOOPBACK_MASTER_EXIT_1	7'd120
LOOPBACK_MASTER_EXIT_2	7'd121
RECOVERY_EQUALIZATION_PHASE0	7'd122
RECOVERY_EQUALIZATION_PHASE1	7'd123
RECOVERY_EQUALIZATION_PHASE2_1	7'd124
RECOVERY_EQUALIZATION_PHASE2_2	7'd125
RECOVERY_EQUALIZATION_PHASE3_1	7'd126
RECOVERY_EQUALIZATION_PHASE3_2	7'd127

Chapter 15: Verification

15.1 Verification Environment

The Cadence® PCIe® Verilog Delivery Testbench provides a simple Verilog-based test environment to enable you to generate and observe inbound and outbound traffic through a PCIe controller and PHY, if operating in a Subsystem mode.

It is a self-contained environment which does not required any VIP (*Verification IP*).

- The traffic and scenarios generated with this testbench are limited when compared to an environment utilising PCIe and ARM® AMBA® VIP.
- The verilog delivery testbench is not intended to verify the functionality of the controller, PHY, or the integration of the controller with the PHY.

15.2 Verification Overview

The testbench instances to the PCIe subsystems (Controller and PHY) are connected at the serial link. Verilog Bus Functional Models (BFMs) are connected to the application level interfaces – HLS/AXI master and HLS/AXI slave ports and APB slave port.

The APB BFM connects to the configuration ports for the PCIe Controller, the PHY and the configuration registers within the PCIe subsystem. The HLS/AXI BFM connects to the application interface of the PCIe controller. The SRAM represents the PCIe controller memories used by the IP.

For additional information refer to the *PCIe Verilog Delivery Testbench User Guide*.

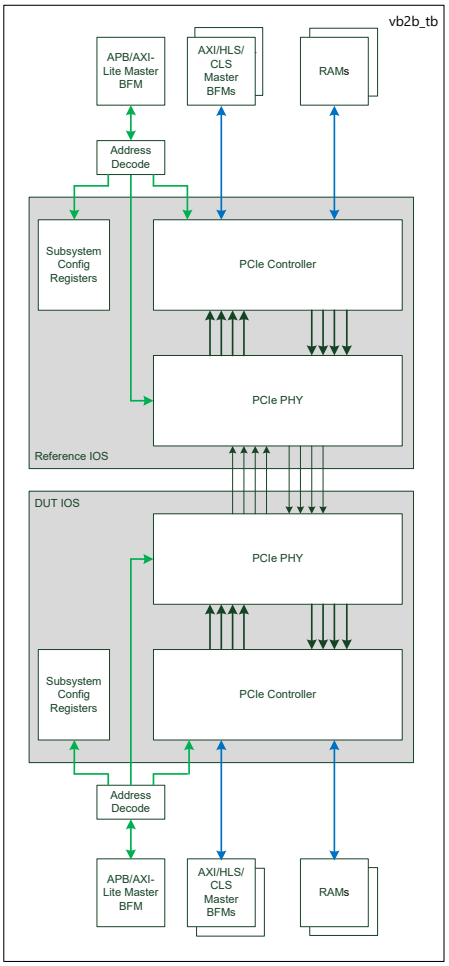


Figure 15.1: HPA Subsystem testbench block diagram

Note: The figure is for representative purpose only, and the blocks are applicable based on the configuration selected.

Chapter 16: Appendix C: PCIe Configuration Capabilities Linked List

16.1 PCIe Controller Capabilities Linked List Implementation

This section describes the PCIe Configuration Capabilities Linked List for the PCIe Controller. The following table shows the PCIe Controller Capabilities Linked List implementation for all the Capabilities present in the PCIe Controller. If any of the capabilities are not present or valid for the chosen controller, the capability before that would point to the next capability present in the controller.

Capability Name	Capability Address	Capability Next pointer	Condition to Skip	Mode Capability Valid
PCI PM Capability	0x80	0x88	-	EP & RP
VPD Capability	0x88	0x90	-	EP
MSI Capability	0x90	0xB0	MSI Not selected	EP & RP
MSIX Capability	0xB0	0xC0	MSIX Not Selected	EP & RP
PCIE Capability	0xC0	0x00 (End of PCI Base)	-	EP & RP
AER Capability	0x100	0x144	First Extended Cap , always present	EP & RP
ARI Capability	0x144	0x150	ARI not Selected	EP
Device Serial Num Capability	0x150	0x160	Device serial number not selected	EP & RP
Power Budgeting Capability	0x160	0x180	Power Budgeting not selected	EP
Resizable BAR Capability	0x180	0x1B8	Resizable BAR not enabled	EP
LTR Capability	0x1B8	0x1C0	LTR not selected	EP
DPA Capability	0x1C0	0x200	DPA not selected	EP
SRIOV Capability	0x200	0x274	num_vfs is zero	EP
TPH Capability	0x274	0x300	TPH not selected	EP & RP
PCIE Secondary Ext Capability	0x300	0x400	strap_pcie_rate_max < 2	EP & RP
Vendor Specific Capability	0x400	0x440	-	EP
PASID Capability	0x440	0x4C0	PASID not selected	EP
VC Capability	0x4C0	0x5C0	num_vcs == 1	EP & RP
ATS Capability	0x5C0	0x640	ATS not selected	EP
ATS Page Request Capability	0x640	0x900	ATS not selected	EP
L1 PM Substates Capability	0x900	0x910	L1 PM substate not selected	EP & RP
DL Feature Capability	0x910	0x920	strap_pcie_rate_max < 3	EP & RP
RX Margining Capability	0x920	0x9C0	strap_pcie_rate_max < 3	EP & RP
PL 16.0 GT/s Capability	0x9C0	0xA40	strap_pcie_rate_max < 3	EP & RP

Capability Name	Capability Address	Capability Next pointer	Condition to Skip	Mode Capability Valid
PL 32.0 GT/s Capability	0xA40	0xAA0	strap_pcie_rate_max < 4	EP & RP
PL 64.0 GT/s Capability	0xAA0	0xAD0	strap_pcie_rate_max < 5	EP & RP
Flit Logging Capability	0xAD0	0xB10	strap_pcie_rate_max < 5	EP & RP
Flit Performance Measurement Capability	0xB10	0xB40	strap_pcie_rate_max < 5	EP & RP
Flit Error Injection Capability	0xB40	0xA20	strap_pcie_rate_max < 5	EP & RP
PTM Capability	0xA20	0xA30	PTM not selected	EP & RP
NPEM Capability	0xA30	0xA70	NPEM not selected	EP & RP
Alternate Protocol Negotiation Capability	0xA70	0xA90	Alternate Protocol not selected	EP & RP
Device 3 Capability	0xA90	0xB70	Deferred MWR not supported and strap_pcie_rate_max < 5	EP & RP
IDE Extended Capabiltity	0xB70	0xDE0	IDE not selected or not PF0	EP & RP
DOE Extended Capability	0xDE0	0xE00	IDE not selected or not PF0	EP
Designated Vendor Specific Capability	0xE00	0x000	CXL not selected or not PF0	EP & RP

Figure 16.1 shows PCIe Controller Capabilities Linked List implementation for all the Capabilities present in the PCIe Controller

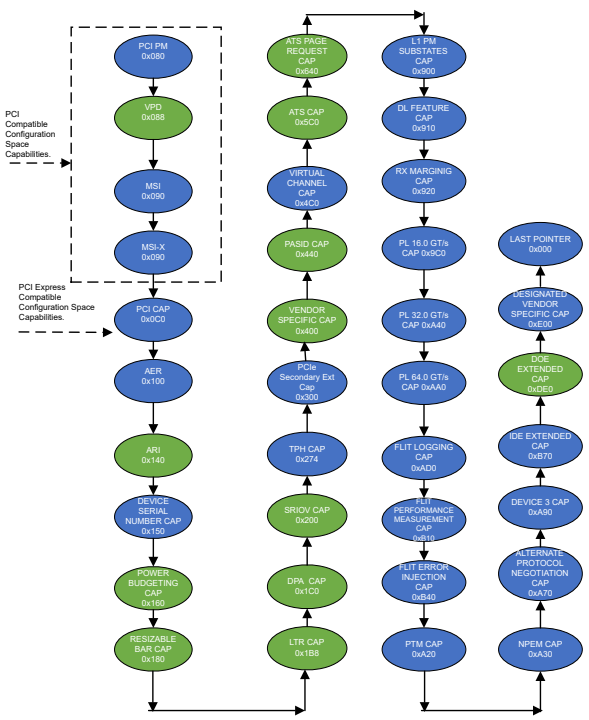


Figure 16.1: PCIe Controller Capabilities Linked List implementation

Chapter 17: Appendix G

17.1 Differences between the Released Versions

The following table shows the differences between the released versions of the *Cadence Controller IP for PCIe 5.0 HPA User Guide*.

Version	Release Date	Comments
0.7	07 December 2020	Initial Release
0.71	03 March 2021	Added AXI Master section
0.72	11 November 2021	Added Skew Components Appendix
		Added IP Registers Access through the PCIe Link
		Added Appendix: PCIe Configuration Capabilities Linked List
		Added Appendix: LTSSM State Names
		Added Synthesis Strategy Chapter
0.73	08 March 2022	In Chapter, Overview of the IP • updated Overview section • updated Core Configuration section
		In Chapter, Application Interfaces AXI4 • updated AXI Slave Write Operations section • updated AXI Slave Read Operations section • updated topics under AXI Slave (Error) Response section - added topic, Errors generated by link down event • AXI Link-Down Handling section • AtomicOp on AXI Slave section
		In Chapter, IP Register Interface • updated Overview section
		Updated Appendix C: PCIe Configuration Capabilities Linked List
		In section, Synthesis Stratergy • added Standard Cell Replacement
		Generic and editorial updates
		0.74
0.75	25 April 2022	• In section, Request Transformation - updated table, Example AWLEN and AWSIZE signalization for 512-bit data path width. Register. • Updated section, AXI Slave (Error) Response.

Version	Release Date	Comments
0.76	7 July 2022	• In Chapter, Application Interfaces AXI4 - AXI Slave Descriptor through Sideband section - AXI Slave Descriptor Example (Normal Message) section - AXI Slave Write Operations section - AXI Slave section - AXI Slave Access Note section - AXI Slave Read Operations section - Tag Management for Non-Posted Transactions • In Miscellaneous Interfaces, updated diagrams in Interrupt Interfaces. • Updated Delivery Package Information in About this Document. • In Power Management, added topics under Internal Power Shutoff section. • PCIe Controller Capabilities Linked List Implementation. • Generic and editorial updates
0.77	20 July 2022	• In Chapter, Application Interfaces AXI4, updated AXI Region Base Address Registers • Updated PCIe Controller Capabilities Linked List Implementation.
0.78	19 August 2022	• Removed AXL slave registers and provided reference to the HTML source files.
0.79	15 September 2022	In Chapter, Application Interfaces AXI4, under AXI slave section, added *Application Layer Read Size*
1.01	01 December 2022	• Updated *Table 5.2* for addr[25:24] decode value. • Updated the figure *Figure 5.1*
1.02	24 January 2023	• Updated reset sequence. See *Reset Sequence* • Updated *AXI Subordinate Access Note* for the ATS Translation Requests support.
1.03	27 February 2023	Updated *Table 8.6* and *Table 8.7*
1.04	29 March 2023	• Added *Vital Product Data (VPD) Integration*. • Updated *AXI Subordinate (Error) Response*. • Updated *AXI Subordinate Access through Sideband Descriptor*
1.05	04 April 2023	• Added *Example with Non-uniform Number of Total VFs* • Added *Example with Uniform Number of Total VFs*
1.06	05 May 2023	• Non-technical update - Formatting
1.07	02 June 2023	• Updated *Table 5.2* • Updated *Reg Clock Usage*
1.08	04 September 2023	Updated the Chapter: Synthesis and Physical design guidelines.
1.09	07 November 2023	Chapters reorganization.
		Updated Reset block diagram. See *Reset Reference Block Diagram* Updated Start up sequence. See *Start-Up Sequence*Added the Hot-Plug section and subsections. See and subsections.