UG0920
User Guide
PolarFire SoC FPGA PCI Express

Microsemi
a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the current publication.

1.1 Revision 1.0

The first publication of this document.

2 Overview

PCI Express (PCIe) is a scalable, high-bandwidth serial interconnect technology that maintains compatibility with existing PCI systems. Microchip's PolarFire® SoC FPGAs contains fully integrated PCIe endpoint and root port subsystems with optimized embedded controller blocks that use the physical layer interface of the PolarFire SoC transceiver for the PCI Express (PIPE) interconnection within the transceiver block.

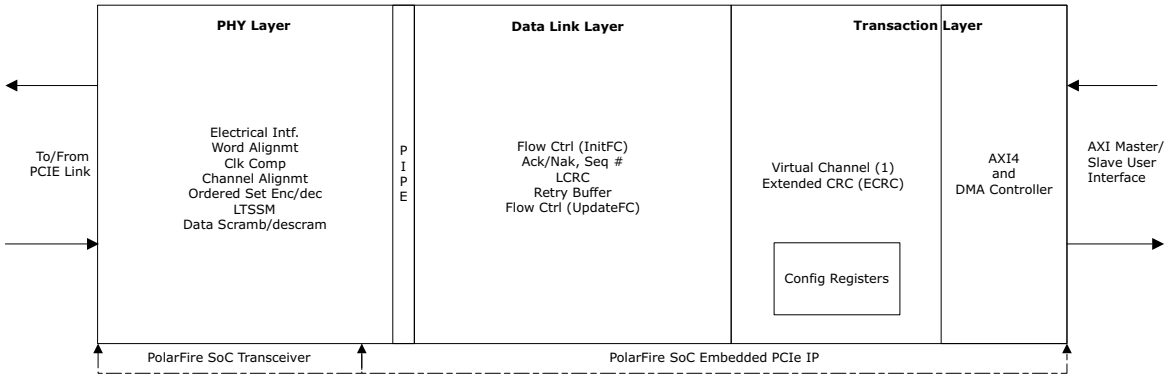
Each PolarFire SoC device includes two embedded PCIe subsystem (PCIESS) blocks that can be configured using the PF_PCIE configurator in the Libero® SoC software.

PCIESS includes the following:

- Physical layer of XCVR
- PIPE
- PCIe IP (Data link layer and Transaction layer)

The PCIESS supports AMBA AXI4 master/slave user interface functionality between the AXI4 and PCIe systems.

Figure 1 • **PCIe Functional Layers of PolarFire SoC PCIESS**



The PCIESS is compliant with the following standard:

- *PCI Express Base Specification with GEN1/2, Revision 3.0*
- *PCI Express Card Electromechanical (CEM) 2.0*
- *PCI Industrial Computer Manufacturers Group (PICMG) 3.4*
- *PCI Power Management Specification, v1.2*
- *AMBA AXI Protocol Specification, Version 2.0 – ARM, March 2010*

2.1 Features

The PCIESS is a hard block and supports the following features:

- PCIE 3.0 compliant with 2.5 and 5.0 Gbps line speeds
- x1, x2, and x4 lane-support[1]
- Endpoint support for up to six 32-bit or three 64-bit base address register (BAR)
- Root port support for up to two 32-bit or one 64-bit BAR
- Two fully-independent DMA engines with SGDMA support
- One virtual channel (VC)
- Single-function capability
- Maximum payload size (MPS) of up to 256 bytes
- Advanced error reporting (AER) support
- Integrated clock domain crossing (CDC) to support user-selected frequency
- Lane reversal/polarity inversion
- Legacy PCI power management
- End point hot-plug capability (not supported for root port)
- Native active-state power management L0 and L1 support
- Power management event (PME) message
- Latency tolerance reporting (LTR)
- 64-bit AXI master and slave interface to the FPGA fabric
- End-to-end data integrity

1. Each PolarFire SoC device supports two PCIESS blocks, which shares four lanes within a transceiver block. This permits two x1 PCIESS or two x2 PCIESS that can run simultaneously, or one PCIESS that can run as x4 and the other PCIESS is left unused.

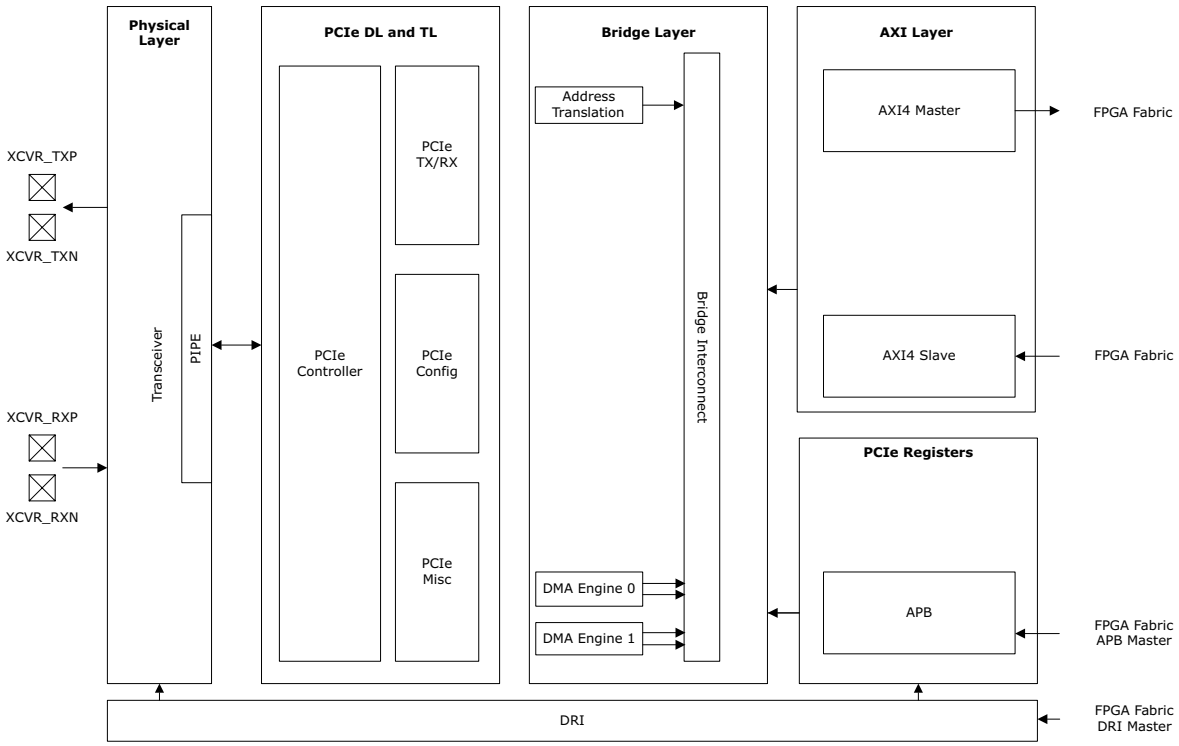
3 Functional Descriptions

The PCIe subsystem uses several built-in features such as PolarFire SoC transceivers, embedded PCIe controller, and programmable FPGA resources. The functional details of the PCIe subsystem are described in this chapter.

As shown in the following figure, the PCIESS is composed of the following four sub-modules:

- Physical layer
- PCIe DL and TL
- Bridge layer
- AXI layer
- Configuration interface (DRI and APB)

Figure 2 • **PolarFire SoC Embedded PCIESS Architecture**



3.1 Physical Layer Interface

A PCIe lane consists of a pair of differential transmit signals and a pair of differential receive signals. The lanes are organized as listed in the following table.

Table 1 • **Lane Configuration**

X1	X2	X3
PCIe0 (Lane0) and PCIe1 (Lane2)	PCIe0 (Lane0, Lane1) and PCIe1 (Lane2, Lane3)	PCIe1 (Lane 0, Lane1, Lane2, and Lane3)

3.1.1 Physical Layer Functions

The physical layer is an electrical PMA required to connect to a PCIe system, and supports the following features:

- PCI Express 2.0 electrical compliance
- 2.5 and 5.0 Gbps common-mode logic (CML) electrical interface
- Signal integrity programmability, including differential output voltage, transmitter de-emphasis, and receiver-side continuous-time linear equalization (CTLE)
- ± 300 ppm clock-tolerance compensation
- Serializer and de-serializer
- 8b/10b symbol encoding/decoding
- Symbol alignment
- Framing and application of symbols to lanes
- Lane to lane Tx deskew

3.1.2 Receiver

The PolarFire SoC transceiver input includes all the features required to build a PCIe interface, such as input level sensitivity, signal detection, and termination. When PCIESS is used within a PolarFire SoC device, the Libero software configures the receiver with all the necessary input features. For more information, see *UG0915: PolarFire SoC FPGA Transceiver User Guide*.

3.1.3 Transmitter

The PolarFire SoC transceiver output includes features such as output swing, termination, and de-emphasis. When PCIESS is used within a PolarFire SoC device, the Libero software configures the transmitter with all the necessary output features. For more information, see *UG0915: PolarFire SoC FPGA Transceiver User Guide*.

Note: In PolarFire SoC PCIe subsystem, the receiver detection circuitry within the transmitter is bypassed and the LTSSM state moves to "polling.compliance" state (when 50 Ω is terminated to the Transmitter) when REFCLK is available and TX PLL is locked. The link training remains unaffected when an active receiver is connected. This causes the optional key-sight PTC test (LOOPBACK_THROUGH_CONFIG) to fail. To pass the test, set the PMA_RXPLL_FLOCK_SEL bit to 0 when LTSSM is in "polling.compliance" state and set to 1 in other states.

3.1.4 Reference Clock

For PCIe applications, a differential 100 or 125 MHz reference clock with a ±300 ppm tolerance is used by the transceiver transmit PLL and CDR PLL to generate the required 125 MHZ output clock (depending on the lane speed settings) which is passed to the embedded PCIESS. The settings for the transmit PLL and the CDR PLL are automatically determined by the Libero SoC software.

The transceiver reference clock inputs accept LVDS/CML/HCSL input clock signals according to PCIe specifications. Proper termination is included as required by the specification. For more information, see *UG0915: PolarFire SoC FPGA Transceiver User Guide*.

According to PCIe specifications, upstream and downstream PCIe devices must transmit data at a rate within 600 ppm of each other at all times. This specification allows a reference clock with a ± 300 ppm tolerance. To ensure that the minimum clock period is not violated, the PCIESS uses a spread-spectrum technique that does not permit modulation above the nominal frequency.

The data rate can be modulated from 0% to 0.5% of the nominal data rate frequency at a modulation rate ranging from 30 to 33 KHz. Along with the ± 300 ppm tolerance limit, both ports require the same bit-rate clock when the data is modulated using SSC.

PolarFire SoC devices support the following clocking topologies defined by the PCIe specifications: common Refclk and separate Refclk.

- The Common Refclk is the most widely supported clocking method in open systems where the root port or root complex provides a clock to the endpoint. An advantage of this clocking architecture is that it supports spread-spectrum clocking (SSC), which is useful in reducing electromagnetic interference (EMI).
- The Separate Refclk uses two independent clock sources: one for the root and the other for the endpoint. The clock sources must maintain ± 300 ppm frequency accuracy and cannot use SSC.

3.1.5 Low-Power States

The PolarFire SoC PCIESS supports PCIe low-power operation states known as L0s, L1, and L2/P2 states. These states are available in both, root port and endpoint configurations.

L0s: Autonomous electrical idle: This state reduces power during short intervals of idle. Devices must transition to L0s independently on each direction of the link.

L1: Directed electrical idle: The L1 state reduces power when the downstream port directs the upstream ports. This state saves power in two ways:

- Shutting down the transceiver circuitry and associated PLL.
- Significantly decreasing the number of internal core transitions.

L2: In this state, a WAKE# signal is required to reinitialize the link. However, the auxiliary power is still available.

Endpoint

The endpoint automatically enters L2 state when directed to do so by its link partner. It can exit this state only when directed by the link partner. The endpoint can send a power management event to request that the link partner re-enable the link.

When the application requests a power management event and the endpoint is in L2 state, the PCIESS can request the host to wake up and re-enable the link by sending a wake-up event. An endpoint can send a wake-up event through either of the following methods:

- A beacon signal transmitted by the physical layer on the link and controlled through PIPE interface signals
- A WAKE# pin controlled by the PCIESS output signal PCIE_#_WAKE_N

Root port

When the root port detects a power management event, it sets an interrupt in the PM_MSI_INT bit of the ISTATUS_LOCAL register. The application must determine the device responsible for the event and restore it to full power.

Note: For information about Configuration registers, see *PolarFire SoC Register Map*.

The link can be enabled only by the root port. The root port enters L2 state and disables the link as soon as its link partner is ready. The application sets the “Turn Off Link” bit in the ICMD_PM register when the root port disables the link.

The root port exits L2 state and enables the link when the application clears the “Turn Off Link” bit in the ICMD_PM register.

A root port can detect a wake-up event through either of the following methods: A beacon signal detected by the physical layer and reported through PIPE interface signals

When a root port detects a wake-up event, it automatically enables the link. As soon as the link is enabled, the application receives a power management interrupt.

3.2 Data and Transaction Layers

The PCIe DL and TL includes:

- PCIe controller
  - Lane reversal
  - Link training and status state machine (LTSSM)
  - Electrical idle generation
  - Receiver detection
  - TS1/TS2 generation/detection
- PCIe transmit/receive interface between the PCIe bridge and PCIe controller
- PCIe configuration interface providing the bridge access to the PCIe configuration space
- PCIe miscellaneous interface to allow the bridge access to manage low-power, and interrupts

The PCIESS includes the data path from the transceiver to the user-defined application layer of the FPGA fabric. The AXI4 bridges the application layer to the transaction layer. The transaction layer communicates with the data link layer through FIFOs. The data link layer communicates with the physical layer through FIFOs. The data is then passed to the transaction layer blocks that manage read and write requests from the software. Finally, the data is passed to the application layer hosted in the FPGA fabric (Figure 8, page 15).

Data Link Layer – The data link layer (DLL) is responsible for link management including transaction layer packet (TLP) acknowledgment (a retry mechanism in case of a non-acknowledged packet), flow control across the link (transmission and reception), power management, CRC generation and checking, error reporting, and logging. The DLL verifies the packets sequence number and checks for errors. The DLL ensures packet integrity, and adds a sequence number and Link Cyclic Redundancy Check (LCRC) to the packet.

The replay buffer located in the data link layer stores a copy of a transmitted TLP until the transmitted packet is acknowledged by the receive side of the link. Each stored TLP includes the header, an optional data payload (the maximum size of which is determined by the maximum payload size parameter), an optional end-to-end cyclic redundancy check (ECRC), the sequence number, and the link cyclic redundancy check (LCRC) field for transaction and data integrity. The replay buffer stores the read data payload from the AXI4 master and write data payload from the AXI4 slave.

Transaction Layer – The transaction layer is responsible for the transfer of transaction layer packets (TLP). The transaction layer disassembles the transaction and transfers data to the application layer in a form that it can recognize. The transaction layer generates a TLP from information sent by the application layer. This TLP includes a header and can also include a data payload. Application communicates to the PCIe link using AXI4 master and slave interface through bridge layer.

3.2.1 Flow Control

PCIe is a flow control-based protocol. Receivers advertise the supported number of receive buffers, and transmitters are not allowed to send TLPs without ensuring that sufficient receive buffer space is available. This control is provided by the 8 KB dedicated buffers included in the PCIESS receive and transmit channels.

Maximum Payload Size – The size of the TLP depends on the capabilities of both link partners. After the link is trained, the root complex sets the maximum payload size register (MAX_PAYLOAD_SIZE) value in the device control register. The maximum allowable payload size is 256 bytes.

3.2.2 Credit Queue

PCIe credit queues are categorized into posted, non-posted, and completion queues. When the host sends writes to the PolarFire SoC PCIESS, this data consumes posted and non-posted credits. When the host needs to send a completion to a read, it consumes completion credits. As is the case for most PCIe endpoints, the completion credit count is infinite in the PCIESS.

A completion timeout is a condition where the host blocks completion due to the lack of non-posted credits. When the PCIESS does a read, if completion data is not returned, the PCIESS issues a completion timeout and the transaction is voided, triggering the AXI slave to terminate the read to avoid a deadlock.

All credit settings are automatically set according to the buffer sizes fixed in the PCIESS. As per PCIe standards, when the PCIe controller issues a read request, it ensures that the controller has the ability to sync the associated read data (completion TLP).

3.2.3 Receive Buffer

The transaction layer contains an 8-KB receive buffer to accept incoming TLPs from the link and send them to the application layer for processing. The receive buffer stores TLPs based on the transaction type (posted, non-posted, or completion). A transaction always has a header but does not always have data. The receive buffer accounts for this distinction, maintaining separate resources for the headers and data for each type of transaction. For completion transactions, data (CPLD) TLPs are stored in the received buffer in the 64-bit addressing format, with each outstanding AXI4 slave read request consuming 16 credits (totaling 128 bytes), and the headers and data consuming one credit each (totaling 16 bytes).

3.2.4 Replay Buffer

In the transmit direction, the AXI4 bridge layer first checks the credits available on the far end before sending the TLP. There must be enough credits available for the entire TLP to be sent. The AXI4 bridge layer must then check that the PCIESS is ready to send the TLP. If there is enough credit, it can proceed with the sending data. If the credit is insufficient, the TLP must wait until enough credit is available.

The replay buffer is used to fetch TLPs that are:

- Issued by the AXI before their transmission on the PCIe link
- Transmitted on the PCIe link, as long as they are not acknowledged by the opposite PCIe device, in case a replay is required

The replay buffer is 8 KB, allowing the transmit buffer to support up to 16 outstanding transmit replay data packets with a maximum payload of 256 bytes each. A maximum of 32 TLPs can be supported by the replay buffer.

Compliance

PCIe Protocol Compliance Test with LeCroy Exerciser:

PolarFire SoC device testing for PCIe compliance with a LeCroy exerciser reports a failure for a PTC test related to Replay Buffer. However, the PTC test conducted on the Key-sight exerciser passed successfully. PolarFire SoC met full compliance at the PCISIG Compliance Workshop #101 and entered into the integrators list. PTC Test 52-20 (LinkRetrainOnRetryFail) fails in the LeCroy testing suite. Root-cause is that the PCIe controller executes some of the second-round replays; prior to, rather than after, link re-training. It does not impact any application functionality as no transaction layer packet (TLP) is lost due to the built-in replay mechanism required by the PCIe base specification. Replays occur both before and after link re-training as long as TLP is not acknowledged.

3.2.5 Extended CRC

The PCIESS optionally performs automatic ECRC to ensure data integrity by default disabled. The PCIE_PEX_SPC2 bridge configuration register controls the ECRC settings.

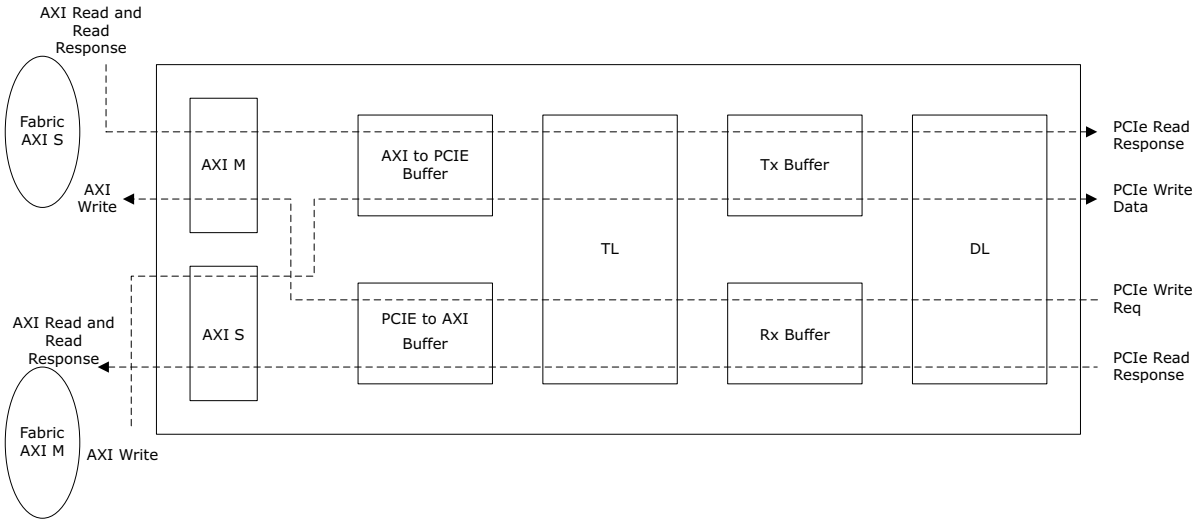
Note: For information about Configuration registers, see *PolarFire SoC Register Map*.

ECRC is enabled when PCIE Specific Capabilities Settings Register PCIE_PEX_SPC Bit[31] is set to 1 to indicate Advanced Error Reporting (AER) is enabled. ECRC error generation and checking can further be individually disabled or enabled by register PCIE_PEX_SPC2 Bit[1] and Bit[2].

3.2.6 ECC

PCIe subsystem has—AXI to PCIE, PCIE to AXI, Rx, Tx—memory buffers as shown in the following figure. These buffers support ECC capability with single bit error correction and dual bit error detection (SECDED). ECC can be disabled or enabled by configuring the ECC_CONTROL register. This register also supports error injection feature.

Figure 3 • **PCIe Subsystem Memory Buffers**



The ECC error detection flags are captured in SEC_ERROR_INT and DED_ERROR_INT registers. These are RW registers and can be cleared by the user writing 1s.

SEC_ERROR_EVENT_CNT and DED_ERROR_EVENT_CNT are counter registers that record the number of SEC and DED errors encountered by AXI to PCIe, PCIe to AXI, Rx, and Tx buffers. The system register up-counts on the internal event it monitors, saturating at the largest unsigned value the counter can hold. The count value can be cleared by writing all-ones pattern through the DRI. Writing through DRI takes momentary precedence over functional up-count. Writing zeros to the count register has no effect.

In addition to these the dual bit error detection flags are exposed to fabric as listed in the following table.

Table 2 • **Error Ports**

Signal	Direction	Description
PCIE_#_M_WDERR		Asserted to '1' when PCIe to AXI buffer reports dual error. The TLP is forwarded to AXI interface. Error during AXI master IF write transactions.
PCIE_#_S_RDERR		Asserted to '1' when PCIe to AXI buffer reports dual error. The TLP is forwarded to AXI interface. Error during AXI slave IF read transactions.
PCIE_#_M_RDERR		Reports data error if the uncorrectable error is detected in the fabric memory used by application. If this signal is asserted, the PolarFire SoC PCIe controller sets the TLP header bit EP (error forwarding field) to '1'. Report error during AXI master IF read transactions.

Table 2 • **Error Ports**

Signal	Direction	Description
PCIE_#_S_WDERR		Reports data error if the uncorrectable error is detected in the fabric memory used by application. If this signal is asserted, the PolarFire SoC PCIe controller sets the TLP header bit EP (error forwarding field) to '1'. Report error during AXI slave IF write transactions.

Table 3 • **Summary of Dual Error in Buffers**

Dual Error in Buffer	Effect at PCIe IF	Effect at Fabric IF
Rx buffer	TLP is forwarded to AXI IF.	Application requires to poll SEC_ERROR_EVENT_CNT / DED_ERROR_EVENT_CNT status register to find out the error.
Tx buffer	TLP is transmitted. Tx buffer memory error is ignored and cannot be recovered by PCIe protocol	Application requires to poll SEC_ERROR_EVENT_CNT /DED_ERROR_EVENT_CNT to find out the error.
PCIE to AXI buffer	TLP forwarded to AXI IF.	The PCIE_#_M_WDERR and PCIE_#_S_RDERR signals to fabric IF are asserted
AXI to PCIE buffer	If ECRC is enabled, PCIe interface marks this TLP as erroneous and an ECRC error is generated by PCIe controller. If ECRC is disabled, TLP is discarded. If this AXI to PCIE buffer error is during a read transaction, it results in a completion timeout.	If this error is during a write transaction, PCIESS_AXI_#_S_BRESP [1:0] == 2'b10 is returned in AXI Write Response.

Note: Rx/Tx buffer reports either SEC or DED error, when TLP is forwarded to AXI interface. The ERROR_INT and ERROR_EVENT_CNT registers indicate the SEC or DED errors.

To enable ECC after PCIe enumeration, follow:

1. Enable ECC using ECC_CONTROL register.
2. Perform greater than 16KB writes on PCIe to AXI and AXI to PCIe.
3. Clear the SEC_ERROR_EVENT_CNT, DED_ERROR_EVENT_CNT, SEC_ERROR_INT, and DED_ERROR_INT registers.
4. ECC errors can be injected using ECC_CONTROL register

3.3 Bridge Layer

The bridge layer includes:

- Two independent DMA engines
- An address translator to convert between the AXI and PCIe interfaces
- A bridge interconnect module to interconnect and arbitrate between input and output flows

3.3.1 DMA Transfers

Each PCIe controller supports the following built-in DMA features, enabling low-power and efficient data transfer to the FPGA fabric:

- Eight outstanding read and write requests
- Completion re-ordering support
- Flexible SGDMA modes, including dynamic DMA control per descriptor
- DMA engine that reports to the descriptor for easy software management
- Fetches up to three descriptors to optimize throughput

The PCIESS has two fully-independent DMA engines (DMA0 and DMA1) which helps to deliver higher performance on both, write and read functions. Each DMA engine can be configured to function as direct DMA or SGDMA. The DMA engines can be configured using PCIESS registers. For more information about DMA registers, see *PolarFire SoC Register Map*.

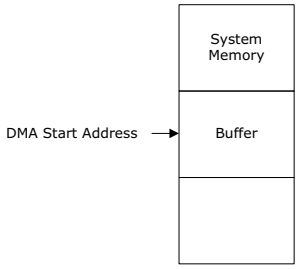
In direct DMA transfer mode, the DMA start address is a pointer to a contiguous data buffer mapped in the PCI bus address space. Data is read from and written to the buffer sequentially.

For DMA0, source is fixed to PCIe IF and destination is configurable. DMA0 is used to transfer data from:

- PCIe link (Host PC memory) to EP AXI (LSRAM/DDR in EP design).
- PCIe link (Host PC memory) to PCIe link (Host PC memory).

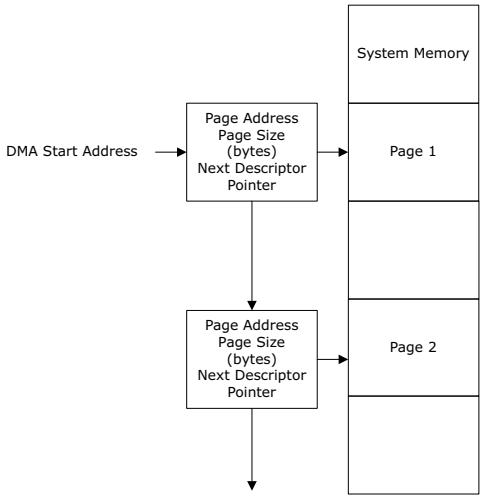
For DMA1, source is fixed to AXI IF and destination is fixed to PCIe IF. DMA1 is used to transfer data from EP AXI (LSRAM/DDR in EP design) to PCIe link (Host PC memory).

Figure 4 • **Direct DMA Transfer**



In SG transfer mode, the DMA source and/or destination start address is a pointer to a chained list of page descriptors. Each descriptor contains the address and size of a data block (page), as well as a pointer to the next descriptor block to enable circular buffers.

Figure 5 • **SGDMA Transfer**

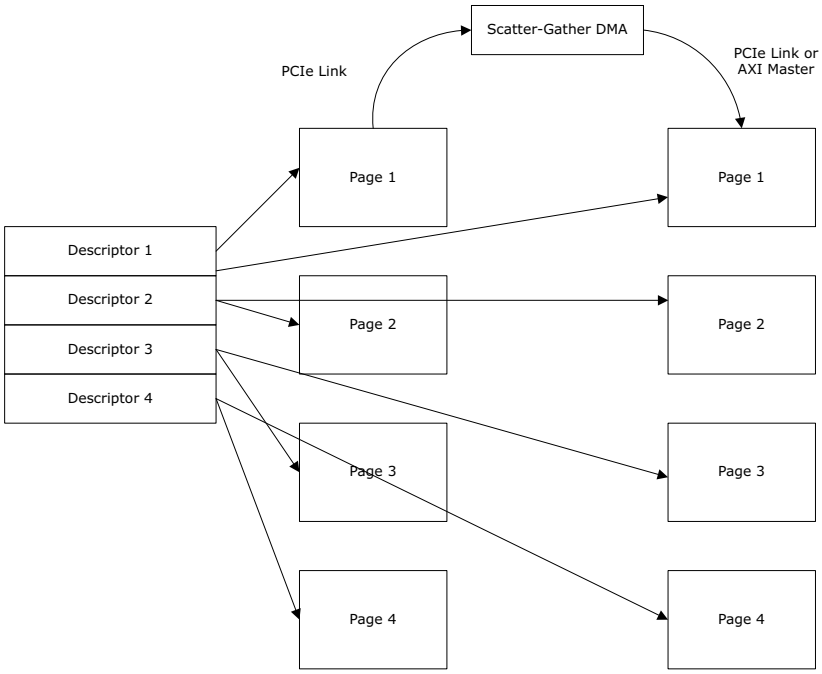


The implementation of SGDMA transfer is different in DMA0 and DMA1.

SGDMA Transfer in DMA0

Source and Destination addresses are set according to the Descriptor. DMA source is always PCIe link and destination can be configured as PCIe link or AXI Master.

Figure 6 • **DMA0 Descriptor**

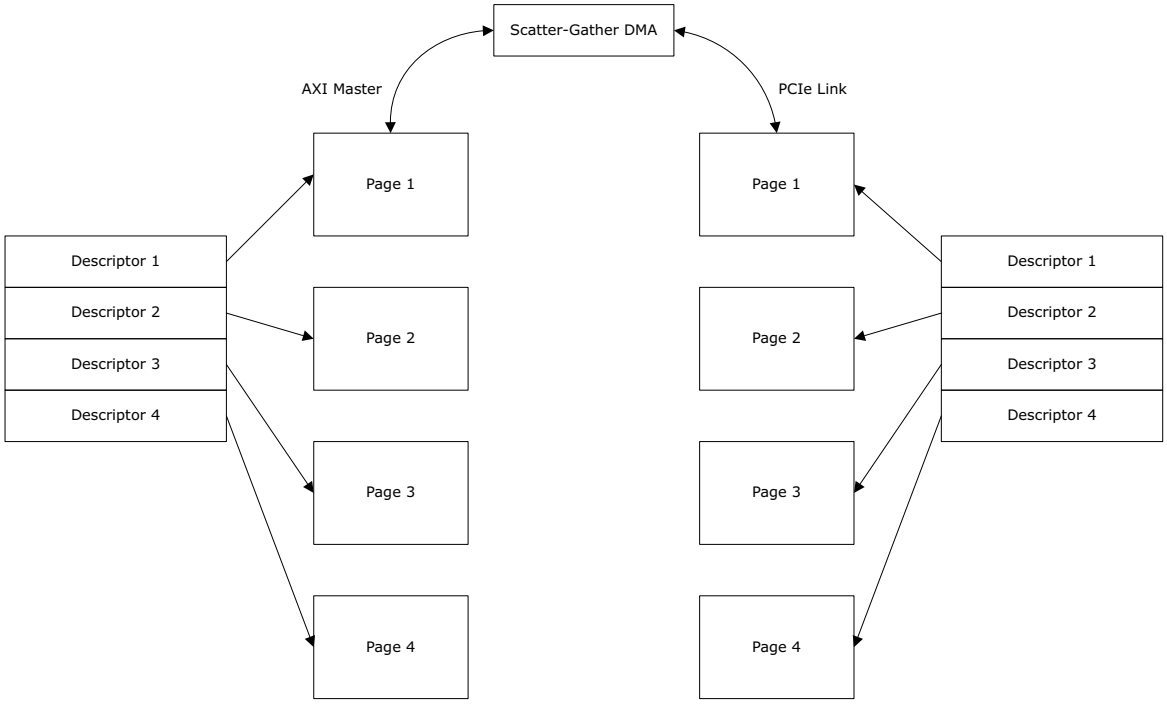


Note: For DMA0, addresses and length must be multiples of 4 bytes. Source and destination start addresses must be 32 byte aligned. For example, if the start address is 0xXX14 and the destination address can be 0xXX14, 0xXX34, 0xXX54 and so on.

SGDMA Transfer in DMA1

Source and destination addresses are set by two independent Scatter-Gather descriptors all DMA destination is always PCIe and source is AXI IF.

Figure 7 • **DMA1 Descriptor**



Note: For DMA1, addresses and length must be multiple of 32 bytes. Source and destination start addresses must be 32 byte aligned.

3.3.2 Scatter-Gather DMA Descriptors

The following table lists the Scatter-Gather DMA Descriptors.

Table 4 • **Scatter-Gather DMA Descriptors**

Name	Byte Offset	R/W	Description
DESC_STATUS	0x00 - 0x03	RW	Enables dynamic monitoring of the SG-DMA transfer (when 0th bit of DESC_CONTROL register is set) Bits [3:0]: Provides the status number of the DMA engine. This number is incremented, enabling the application to determine the last processed descriptor, which is key in streaming flow between asynchronous devices. Bits [7:4]: Bit 4: Indicates SG-DMA descriptor has been processed. Bit 5: Indicates an error occurred during the processing of the current SG-DMA descriptor. Bit 6: Indicates an End of Packet (EOP) condition has been reported by the source of the SG-DMA transfer. Bit 7: Reserved Bits [31:8]: Indicates the Processed Page Size, which is the actual written or read page size.

Table 4 • **Scatter-Gather DMA Descriptors *(continued)***

Name	Byte Offset	R/W	Description
DESC_CONTROL	0x04 - 0x07	RO	DESC_CONTROL enables dynamic control of the SG-DMA transfer. Bit [0]: Defines whether the DMA engine provides a status report by writing to DESC_STATUS when the current SG-DMA descriptor has been processed. Bits [3:1]: Reserved Bits [7:4]: Defines when an interrupt should be issued. Bit 4: Indicates an IRQ is issued when this SG-DMA descriptor has been processed. Bit 5: Indicates an IRQ is issued if an error occurs Bit 6: Indicates an IRQ is issued if the source of the transfer reports an EOP condition. Bit 7: Reserved Bits [31:8]: Provides the page size in bytes, from 1 to 16 M bytes.
DESC_NEXT_ADDR[63:32]	0x0C - 0x0F	RO	Indicates next descriptor address. This field must be aligned on a 32-byte boundary.
DESC_NEXT_ADDR[31:5]	0x08 - 0x0B	RO	Indicates next descriptor address.
DESC_NEXT_RDY[4]			Indicates if the next SG-DMA Descriptor is ready and fetch-able.
DESC_SE_COND[3:0]			Defines the Start and End conditions for SG-DMA descriptor processing. Bit 0: Indicates end the DMA transfer after this SG-DMA descriptor has been processed (equivalent to an End Of Chain). Bit 1: Indicates to abort this SG-DMA descriptor processing if an error occurs. Bit 2: Reserved Bit 3: Indicates to start this SG-DMA descriptor processing when the source of the transfer reports a SOP reception.
DESC_SRC_ADDR[31:0]	0x10 - 0x13	RO	Indicates the source address of the descriptors. It must be 32 byte aligned.
DESC_SRC_ADDR[63:32]	0x14 - 0x17	RO	
DESC_DEST_ADDR[31:0]	0x18 - 0x1B	RO	Indicates the destination address of the descriptors. It must be 32 byte aligned.
DESC_DEST_ADDR[63:32]	0x1C - 0x1F	RO	

3.3.3 PCIe/AXI4 Address Translation

There are six address table registers (ATRs) that perform address translation from the PCIe address space (BAR) to the AXI master, and six ATRs for the slave, which addresses translation from the AXI4 slave to the PCIe address space.

For the address translation PCIESS uses the following:

- Master Windows
- Slave Windows

3.3.3.1 Master Windows

Master windows translate the addresses from PCIE domain to AXI 4 domain.

When the PCIESS is in endpoint mode, there is direct mapping between the BAR and the address tables, as shown in the following figures.

Each 32-bit BAR has a address translation table (ATR). The PCIe implementation supports up to six 32-bit base address registers (BARs). Each BAR is 32 bits, but two BARs can be combined to make a 64-bit BAR. For example, BAR0 (address offset 010h) and BAR1 (address offset 014h) can be combined to form a 64bit BAR01.

For 64-bit BAR mapping, address tables are used as follows:

- BAR01 targets ATR0, and ATR1 remains unused
- BAR23 targets ATR2, and ATR3 remains unused
- BAR45 targets ATR4, and ATR5 remains unused

Figure 8 • **PCIe to AXI4 Master Address Translation Endpoint Mode for 32-Bit BAR**

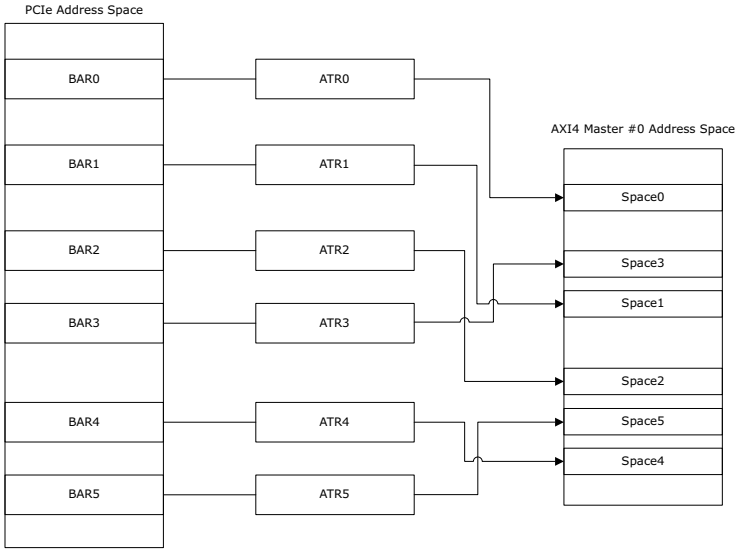
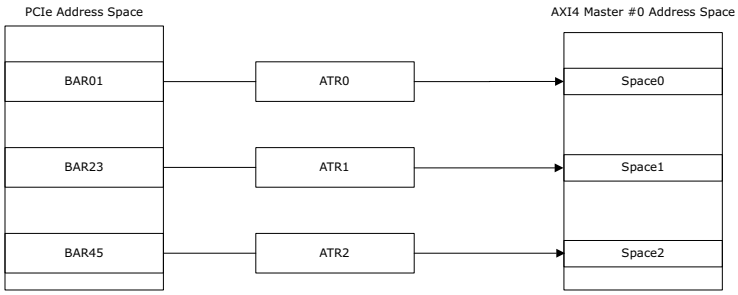


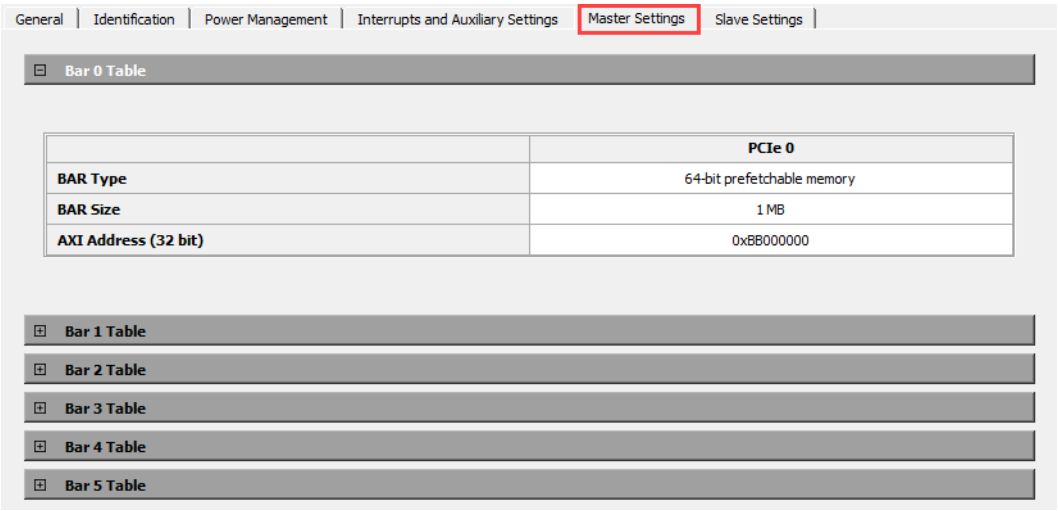
Figure 9 • **PCIe to AXI4 Master Address Translation Endpoint Mode for 64-Bit BAR**



The PCIESS Configurator in Libero SoC software provides a GUI to configure the BAR settings for an endpoint application.

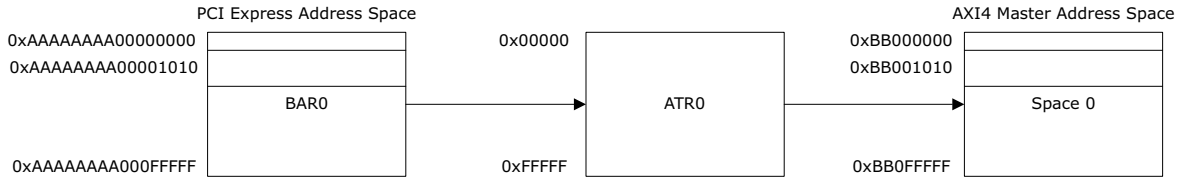
The following figure shows an example of configuring Master settings to map the BAR0 transactions to PCIe AXI master IF transactions for AXI address 0xBB000000.

Figure 10 • **Example of Configuring Master EP Settings**



If BAR0 receives write/read request with offset 1010, then ATR0 translates this address to 0xBB001010.

Figure 11 • **PCIe to AXI4 Master Address Translation Endpoint Mode Example**



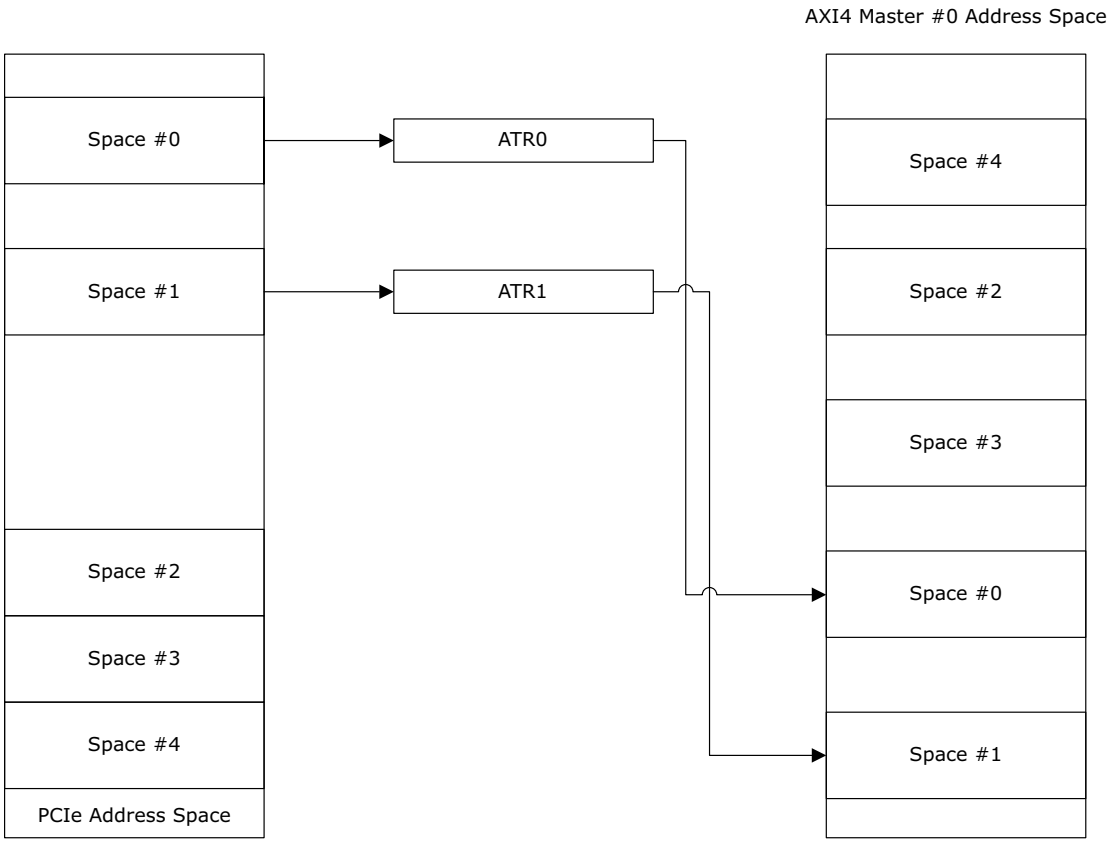
Note: In this example, BAR0 address of host is shown as 0xAAAAAAAA00000000.

When the PCIESS is in root port mode (Figure 12, page 17), up to six translation tables (currently, Libero SoC supports two translation tables) can be implemented. When transferring PCI Express receive requests to the AXI master, the bridge performs a windows match using the PCIe 64-bit address. If a match is found, the bridge forwards the request to the desired AXI4 master interface and adds the corresponding AXI base address.

When the PCI Express BAR0/1 are configured as a 64-bit prefetchable memory space of 16 K bytes. PCIe read and write requests targeting BAR0 or BAR1 are routed to the bridge configuration space. This memory space operates in the following two ways:

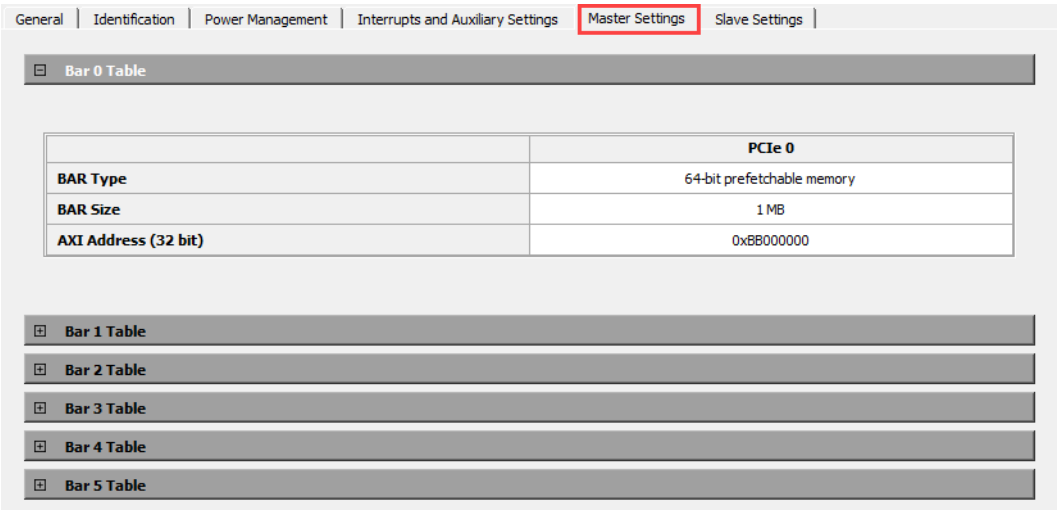
- When I/O or prefetchable memory windows are implemented, PCIe read and write requests targeting I/O or prefetchable memory windows are routed to the PCIe window address translation module.
- When I/O or prefetchable memory windows are not implemented, PCIe read and write requests that do not target BAR0 or BAR1 are routed to the PCIe window address translation module.

Figure 12 • **PCIe to AXI4 Master Address Translation Root Port Mode**



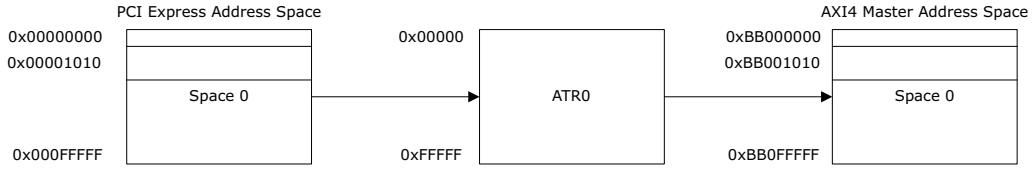
The following figure shows an example of configuring Master settings to map the PCIe RP address space to PCIe AXI master IF address space for AXI address 0xBB000000.

Figure 13 • **Example of Configuring Master RP Settings**



When PCIe receives write/read request at address 0x00001010, this address is mapped to ATR0, which removes the upper 44 bits (depending on table size) and adds the translation address 0xBB000000 to offset 0x1010.

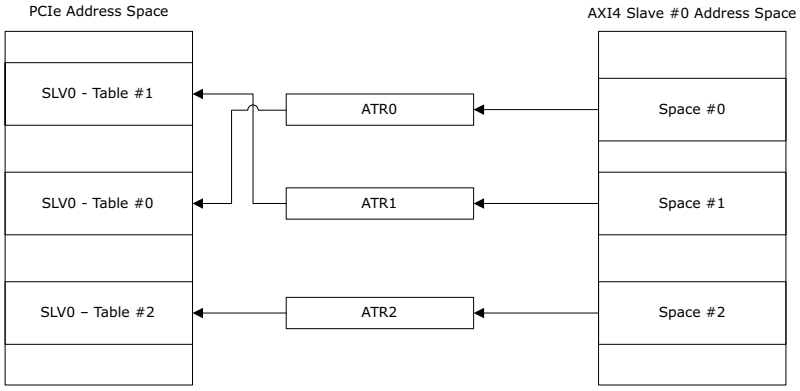
Figure 14 • **PCIe to AXI4 Master Address Translation Root Port Mode Example**



3.3.3.2 Slave Windows

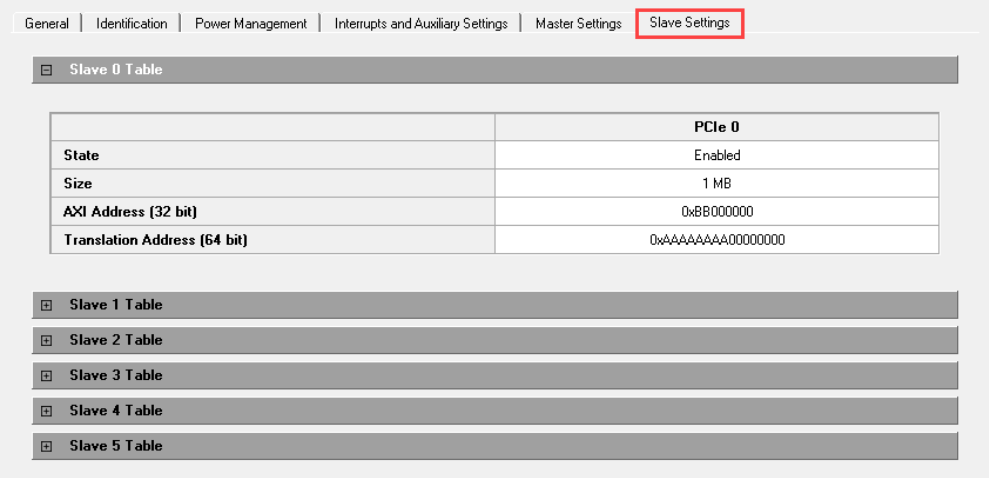
The address translation method used to transfer AXI slave receive requests to the PCIe interface is similar to that used in the endpoint mode. Up to six translation tables can be implemented for AXI4 slave interface.

Figure 15 • **AXI4 Slave to PCIe Address Translation**



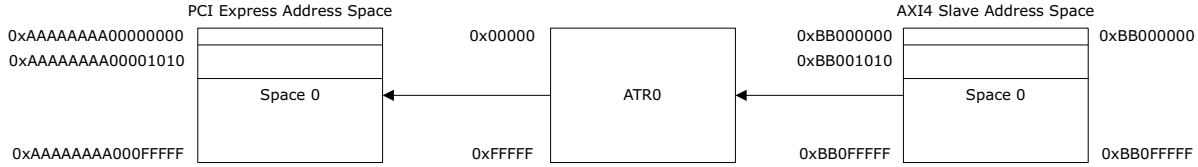
The following figure shows an example of configuring Slave settings to map AXI IF slave requests to PCIE requests for AXI address 0xBB000000 and PCIe address 0xAAAAAAAA00000000.

Figure 16 • **Example of Slave EP Settings**



When PCIe receives write/read request at address 0xBB001010, this address is mapped to Table0, which removes the upper 12 bits (depending on table size) and adds the translation address 0xAAAAAAAA0000000 to offset 0x1010.

Figure 17 • **AXI4 Slave to PCIe Address Translation Example**



3.4 AXI4 Layer

The AXI4 layer of the PCIESS provides a transaction-level translation of AXI4 commands to PCIe packets and PCIe requests to AXI4 commands. The user application in the FPGA fabric must implement an AXI4 master interface to transfer data to the PCIe link and an AXI4 slave interface to receive data from the PCIe link.

3.4.1 AXI MasterIF

A typical PCIe application interface uses an AXI master interface to respond to data read requests and an AXI slave interface to initiate requests. The AXI master performs the following functions:

- Conveys PCIe read and write transactions to the fabric in the form of AXI4 reads and writes.
- Accesses the DMA controller through the bridge and issues reads and writes that allow data pulled from the fabric to be sent over PCIe and data pulled from the PCIe to be written out to the fabric.

3.4.1.1 AXI Master Write Transactions

Write transactions are handled in big-endian order as required by the PCI Express Base Specification. The master path does not reorder transactions, but arbitrates between transactions at the AXI4 master interface. If a transaction is currently waiting for a response phase, the transaction is allowed to complete before the read transaction is forwarded to the AXI4 master interface. PCIe transactions sizes may vary up to the configurable maximum payload size (256 bytes):

- AXI4 transactions are limited to 256 bytes, and the received TLP is divided into several AXI4 transactions.
- AXI4 master receives a write transaction, processing it in 256-byte segments.
- TLP is de-constructed from the PCIe system and sent to the AXI4 interface in little-endian format.

3.4.1.2 AXI Master Read Transactions

Read transactions are handled similar to write transactions, except that before transferring the transaction to the AXI4 master read channel, the PCIESS checks the transmit buffer for available space. If the transmit replay buffer does not have sufficient place to store the PCIe completions, the PCIESS does not transfer the read transaction. The number of outstanding AXI4 master read transactions is, therefore, limited by the size of the transmit buffer.

The AXI4 master read channel can receive transactions in any order, and data can be completely interleaved. However, the PCIESS generates completions in the order they are initiated on the link.

3.4.2 AXI SlaveIF

The AXI4 slave interface forwards AXI read and write requests from the FPGA fabric to the PCIe link. The fabric application initiates PCIe transactions (memory write TLP and memory read TLP transactions) using the slave interface. The data on a read request comes back to the same interface. The slave path does not reorder transactions, but arbitrates between transactions if they occur simultaneously with master read completions. The order of priority for arbitrations is as follows:

1. Master read completions
2. Slave write requests
3. Slave read requests

3.4.2.1 AXI Slave Write Transactions

Slave write transactions support incrementing address bursts, fixed bursts, wrapping bursts, and narrow type transfers. Data interleaving, however, is not supported. Data packets of a maximum of 2 K bytes can be created. Wait states are used if the buffer is full, or has less than 128 bytes of available space. Write responses are generated as soon as the last phase is over with support for up to four outstanding write transactions.

3.4.2.2 AXI Slave Read Transactions

PCIESS generates a PCIe tag, arbitrates between write requests and completions, and then checks for available credits. An error response is generated if a timeout occurs or if a “completion with error” status is received.

3.4.3 AXI4 Limitations

Unsupported AXI4 features in the PCIESS are:

- 8, 16, and 32 bits data bus widths.
- User-defined signals.
- Low-power interface.
- Exclusive accesses are not supported.

3.4.4 Conversion Between PCIe and AXI Transactions

The PCIESS converts AXI read and write transaction as described in the following sections:

3.4.4.1 Conversion from PCIe Write to AXI Write Transactions

A single PCIe write transaction is converted into multiple AXI write transactions at the fabric interface whenever the AXI write address space crosses the negotiated maximum payload size boundary. For example, the negotiated maximum payload size is 256 Bytes, then the address boundaries are 0x000, 0x100, 0x200, 0x300, …, 0xE00, 0xF00. In this case, a PCIe write transaction writes a TLP with payload of 24 Double-Words, that is, 24 × 4 = 96 Bytes. The write transaction targets the contiguous AXI write addresses starting from 0x0005AAF0. With a payload size of 96 Bytes, the ending write address is 0x0005AAF0 + 0x50 = 0x0005AB50, which crosses the 256 Byte boundary. The bridge breaks it into two AXI write transactions—AXI write transaction starting at 0x0005AAF0 with 16 Bytes and AXI write transaction starting at 0x0005AB00 with the remaining 80 Bytes.

3.4.4.2 Conversion from PCIe Read to AXI Read Transactions

A single PCIe read transaction is converted into multiple AXI read transactions when the bridge complies with the following two constraints:

- When the PCIe read transaction targets AXI address space that crosses the negotiated maximum payload boundary, the fabric logic may still return with one single AXI burst transaction. In this case, the PCIe bridge breaks it into multiple separate PCIe read completions with AXI read data broken at the negotiated maximum payload boundary in AXI address space.

Read requests, which cross the address boundaries at integer multiples of RCB bytes may be completed using more than one completion, but the data must not be fragmented except along the following:

- The first completion starts with the address specified in the request, and ends at one of the following:
  - The address specified in the request plus the length specified by the request, that is, the entire request.
  - An address boundary between the start and end of the request at an integer multiple of RCB bytes.
- The final completion ends with the address specified in the request plus the length specified by the request.
- All completions between, but no including, the first and final completions will be an integer multiple of RCB bytes in length.

Note: For a root complex, RCB can be configured to 64 Bytes or 128 Bytes using PCIE_PEX_SPC[15] register bit. For an Endpoint, RCB is always 128 Bytes.

The following are the examples of AXI split transactions:

- Assume the negotiated maximum payload size is 256 Bytes. Memory read request with address of 0x10000 and length of 192 bytes can be completed by a root complex with a RCB value of 64 Bytes with one of the following combination of completions (bytes):
  - 192
  - 128, 64
  - 64, 128
  - 64, 64, 64
- Assume the negotiated maximum payload size is 256 Bytes. Memory read request with address of 0x10200h and length of 256 bytes can be completed by an Endpoint in one of the following combination of completions (bytes):
  - 256
  - 96, 160
  - 96, 128, 32
  - 224, 32

3.5 PCIESS Configuration Interface

Programmable FPGA Resources: The registers required for the initial configuration of the PCIESS are loaded based on the options selected in the Libero PCIESS Configurator wizard. On device power-up, these values are automatically loaded into the configuration registers from the on-chip non-volatile memory during the design initialization process. The design initialization uses the dedicated resources to bring up the PCIESS features at power-up or device reset. This does not require any programmable FPGA user resources. The PCIESS then reads and writes the configuration space registers as the link comes up and the endpoint device is enumerated into the host system.

PCIESS can be dynamically configured through the following interfaces:

- DRI – This interface is used to configure transceiver lane registers.
- APB – This interface is used to configure PCIe control registers.

3.6 PCIESS Port List

The PCIESS block is generated using the Libero PCIe Configurator. The generation of the PCIESS block includes ports based on the PCIe Configurator settings. The following table lists the port descriptions. The PCIESS also has several status signals, interrupt signals, and power management signals available to the FPGA fabric.

Table 5 • **PCIESS Port List[1, 2]**

Port Name	Direction	Description
AXI_CLK	Input	Global AXI clock. Shared for both PCIESS interfaces.
AXI_CLK_STABLE	Input	Clock lock signal. Indicates that the AXI_CLK source from the fabric is locked and ready for use.
Master		
PCIESS_AXI_#_M_ARADDR[31:0]	Output	Read address. The address of the first transfer in a read burst transaction.
PCIESS_AXI_#_M_ARBURST	Output	Read burst type. The burst type and the size of information determine how the address for each transfer within the burst is calculated.
PCIESS_AXI_#_M_ARID[3:0]	Output	Read address ID. Identification tag for the read address group of signals.
PCIESS_AXI_#_M_ARLEN[7:0]	Output	Burst length. Indicates the exact number of transfers in a burst.
PCIESS_AXI_#_M_ARREADY	Input	Read address ready. Indicates that the slave is ready to accept an address and associated control signals.
PCIESS_AXI_#_M_ARSIZE[1:0]	Output	Burst size. Indicates the size of each transfer in the burst.
PCIESS_AXI_#_M_ARVALID	Output	Read address valid. Indicates that the channel is signaling valid read address and control information.
PCIESS_AXI_#_M_AWADDR[31:0]	Output	Write address. The address of the first transfer in a write burst transaction.
PCIESS_AXI_#_M_AWBURST	Output	Write burst type. The burst type and the size of information determine how the address for each transfer within the burst is calculated.
PCIESS_AXI_#_M_AWID[1:0]	Output	Write address ID. Identification tag for the write address group of signals.
PCIESS_AXI_#_M_AWLEN[4:0]	Output	Burst length. Indicates the exact number of transfers in a burst, which determines the number of data transfers associated with the address.
PCIESS_AXI_#_M_AWREADY	Input	Write address ready. Indicates that the slave is ready to accept an address and associated control signals.
PCIESS_AXI_#_M_AWSIZE	Output	Burst size. Indicates the size of each transfer in the burst.
PCIESS_AXI_#_M_AWVALID	Output	Write address valid. Indicates that the channel is signaling valid write address and control information.
PCIESS_AXI_#_M_BID[3:0]	Input	Response ID tag. Identification tag for the write response.
PCIESS_AXI_#_M_BREADY	Output	Response ready. Indicates that the master is ready to accept a write response.
PCIESS_AXI_#_M_BRESP[1:0]	Input	Write response. Indicates the status of the write transaction. when it is asserted to 2'b10 (SLVERR/DECERR), unsupported request to PCIe is reported.

Table 5 • **PCIESS Port List[1, 2] *(continued)***

Port Name	Direction	Description
PCIESS_AXI_#_M_BVALID	Input	Write response valid. Indicates that the channel is signaling a valid write response.
PCIESS_AXI_#_M_RDATA[63:0]	Input	Read data.
PCIESS_AXI_#_M_RID[3:0]	Input	Read ID tag. Identification tag for the read data group of signals generated by the slave.
PCIESS_AXI_#_M_RLAST	Input	Read last. Indicates the last transfer in a read burst.
PCIESS_AXI_#_M_RREADY	Output	Read ready. Indicates that the master can accept the read data and associated control signals, along with response information.
PCIESS_AXI_#_M_RRESP[1:0]	Input	Read response. Indicates the status of the read transfer. • when it is asserted to 2'b10(SLVERR), completion with unsupported request status is returned. • when it is asserted to 2'b11(DECERR), completion with Completion Abort status is returned.
PCIESS_AXI_#_M_RVALID	Input	Read valid. Indicates that the channel is signaling the required read data.
PCIESS_AXI_#_M_WDATA[63:0]	Output	Write data.
PCIESS_AXI_#_M_WLAST	Output	Write last. Indicates the last transfer in a write burst.
PCIESS_AXI_#_M_WREADY	Input	Write ready. Indicates that the slave can accept the write data.
PCIESS_AXI_#_M_WSTRB[7:0]	Output	Write strobes. Indicates the byte lanes that hold valid data. There is one write strobe bit for every eight bits of the write data bus.
PCIESS_AXI_#_M_WVALID	Output	Write valid. Indicates that valid write data and strobes are available.
Slave		
PCIESS_AXI_#_S_ARADDR[31:0]	Input	Read address. The address of the first transfer in a read burst transaction.
PCIESS_AXI_#_S_ARBURST[1:0]	Input	Burst type. The burst type and the size of information determine how the address for each transfer within the burst is calculated.
PCIESS_AXI_#_S_ARID[3:0]	Input	Read address ID. Identification tag for the read address group of signals.
PCIESS_AXI_#_S_ARLEN[7:0]	Input	Burst length. Indicates the exact number of transfers in a burst.
PCIESS_AXI_#_S_ARREADY	Output	Write address ready. Indicates that the slave is ready to accept an address and the associated control signals.
PCIESS_AXI_#_S_ARSIZE[1:0]	Input	Burst size. Indicates the size of each transfer in the burst.
PCIESS_AXI_#_S_ARVALID	Input	Read address valid. Indicates that the channel is signaling valid read address and control information.
PCIESS_AXI_#_S_AWADDR[31:0]	Input	Write address. Address of the first transfer in a write burst transaction.
PCIESS_AXI_#_S_AWBURST[1:0]	Input	Burst type. The burst type and the size information determine how the address for each transfer within the burst is calculated.
PCIESS_AXI_#_S_AWID[3:0]	Input	Write address ID. Identification tag for the write address group of signals.
PCIESS_AXI_#_S_AWLEN[7:0]	Input	Burst length. Indicates the exact number of transfers in a burst, which determines the number of data transfers associated with the address.

Table 5 • **PCIESS Port List[1, 2] *(continued)***

Port Name	Direction	Description
PCIESS_AXI_#_S_AWREADY	Output	Write address ready. Indicates that the slave is ready to accept an address and associated control signals.
PCIESS_AXI_#_S_AWSIZE[1:0]	Input	Burst size. Indicates the size of each transfer in the burst.
PCIESS_AXI_#_S_AWVALID	Input	Write address valid. Indicates that the channel is signaling valid write address and control information.
PCIESS_AXI_#_S_BID[3:0]	Output	Response ID tag. Identification tag for the write response.
PCIESS_AXI_#_S_BREADY	Input	Response ready. Indicates that the master can accept a write response.
PCIESS_AXI_#_S_BRESP[1:0]	Output	Write response. Indicates the status of the write transaction. It is asserted to 2'b10(SLVERR), when: • an AXI Write Completion Timeout (128 ms in hardware and 128 µs in simulation) cannot be transferred to the PCIe link due to an AXI write transaction, because the link is down or in low-power mode. • ECRC error occurs in TLP/Poisoned TLP It is asserted to 2'b11(DECERR), when no address translation table is matched (Unsupported Address).
PCIESS_AXI_#_S_BVALID	Output	Write response valid. Indicates that the channel is signaling a valid write response.
PCIESS_AXI_#_S_RDATA[63:0]	Output	Read data.
PCIESS_AXI_#_S_RID[3:0]	Output	Read ID tag. Identification tag for the read data group of signals generated by the slave.
PCIESS_AXI_#_S_RLAST	Output	Read last. Indicates the last transfer in a read burst.
PCIESS_AXI_#_S_RREADY	Input	Read ready. Indicates that the master can accept the read data and response information.
PCIESS_AXI_#_S_RRESP[1:0]	Output	Read response. Indicates the status of the read transfer. It is asserted to 2'b10(SLVERR), when: • PCIe Completion TLP with Unsupported Request (UR) Status • ECRC error/Poisoned TLP occurs. It is asserted to 2'b11(DECERR), when: • no address translation table is matched (Unsupported Address) • PCIe Completion TLP with Completer Abort (CA) status.
PCIESS_AXI_#_S_RVALID	Output	Read valid. Indicates that the channel is signaling the required read data.
PCIESS_AXI_#_S_WDATA[63:0]	Input	Write data.
PCIESS_AXI_#_S_WLAST	Input	Write last. Indicates the last transfer in a write burst.
PCIESS_AXI_#_S_WREADY	Output	Write ready. Indicates that the slave can accept the write data.
PCIESS_AXI_#_S_WSTRB[7:0]	Input	Write strobes. Indicates the byte lanes that hold valid data. There is one write strobe bit for each eight bits of the write data bus.
PCIESS_AXI_#_S_WVALID	Input	Write valid. Indicates that valid write data and strobes are available.
PCIe		
PCIE_#_M_RDERR	Input	Master read data error. Allows the application to report data error by asserting this signal during AXI read data phase.

Table 5 • **PCIESS Port List[1, 2] *(continued)***

Port Name	Direction	Description
PCIE_#_M_WDERR	Output	Master write data error. Asserted when an uncorrectable error is detected by the memory's ECC logic when reading data from the buffer. The error is reported on the same clock cycle as the affected data, that is, AXI write data phase.
PCIE_#_S_RDERR	Output	Slave read data error. Asserted when an uncorrectable error is detected by the memory's ECC logic when reading data from the buffer. The error is reported on the same clock cycle as the affected data, that is, AXI read data phase.
PCIE_#_S_WDERR	Input	Slave write data error. Allows the application to report data error by asserting this signal during AXI read data phase.
PCIE_#_L2_EXIT	Output	L2 exit. Asserted for one clock cycle when the link training and status state machine (LTSSM) exits the L2 state. Prompts the application layer to perform a global reset.
PCIE_#_HOT_RST_EXIT	Output	Hot reset exit. Asserted for one clock cycle when the LTSSM exits hot reset state. Prompts the application layer to perform a global reset.
PCIE_#_DLUP_EXIT	Output	DL-up exit. Indicates transition from DL_UP to DL_DOWN. dlup_exit = 1'b0, data link layer is down dlup_exit = 1'b1, data link layer is up only valid when LTSSM is in L0 state
PCIE_#_INTERUPT[7:0]	Input	Local interrupt input ports.The fabric logic can drive up to eight interrupt sources by generating a pulse (high) on the ports. [7:0] can be used for MSI. [0] is also available for INTx. Used only for endpoints. PCIESS uses TL_CLK to monitor this signal. MSI offsets are [negotiated interrupt-1:negotiated interrupt-8]
PCIE_#_INTERUPT_OUT	Output	Local interrupt output port. Indicates that one of the possible interrupt sources was detected and the user can read the interrupt through the DRI. It is a level sensitive signal whenever an interrupt described in ISTATUS_LOCAL, SEC_ERROR_INT register, DED_ERROR_INT register, and PCIE_EVENT_INT register is active, the PCIE_#_INTERUPT_OUT signal gets asserted and remains high. It is low when the corresponding interrupts in the registers are cleared.

Table 5 • **PCIESS Port List[1, 2] *(continued)***

Port Name	Direction	Description
PCIE_#_LTSSM[4:0]	Output	LTSSM state encoding: 0x0: LTSSM_DET_QUIET 0x1: LTSSM_DET_ACT 0x2: LTSSM_POL_ACT 0x3: LTSSM_POL_COMP 0x4: LTSSM_POL_CFG 0x5: LTSSM_CFG_LWSTR 0x6: LTSSM_CFG_LWACC 0x7: LTSSM_CFG_LWAIT 0x8: LTSSM_CFG_LNACC 0x9: LTSSM_CFG_CPLT 0xa: LTSSM_CFG_IDLE 0xb: LTSSM_RCV_RLOCK 0xc: LTSSM_RCV_EQL 0xd: LTSSM_RCV_SPEED 0xe: LTSSM_RCV_RCFG 0xf: LTSSM_RCV_IDLE 0x10: LTSSM_L0 0x11: LTSSM_L0S 0x12: LTSSM_L1_ENTRY 0x13: LTSSM_L1_IDLE 0x14: LTSSM_L2_IDLE 0x15: LTSSM_L2_XMIT/WAKE 0x16: LTSSM_DISABLED 0x17: LTSSM_LOOPBACK_ENTRY 0x18: LTSSM_LOOPBACK_ACTIVE 0x19: LTSSM_LOOPBACK_EXIT 0x1a: LTSSM_HOTRESET
PCIE_#_WAKE_N	Output	Wake-up signal. L2/P2 implementation: L2 exit request to RP. Wake (WAKE#) is connected to any I/O on the PolarFire SoC device. It is required on any add-in card or system board that supports wake-up functionality compliant with the PCIE CEM specification. WAKE# can use any general purpose IO.
PCIE_#_PERST_N	Input	Asynchronous. Input signal used for L2/P2 implementation: L2 exit request from RP. PERST_N can use any general purpose IO.
PCIE_#_TL_CLK_125MHz	Input	125 MHz (maximum) clock input. Continuous running clock is required for PCIe core transaction layer. Connects to the DIV_CLK output from the TX_PLL. TL_CLK is available only after PCIe initialization. User have to derive from the on-chip oscillator to drive the TL_CLK during PCIe initialization. An NGMUX can be used to switch this clock to the required DIV_CLK after PCIe initialization. For more information, see *DG0756: PolarFire FPGA PCIe Endpoint, DDR3,and DDR4 Memory Controller Data Plane Demo Guide*.
Transceiver		
CLKS_FROM_TXPLL_TO_PCIE_#	Input	PCIE_#_TX_BIT_CLK_#: This port must be driven by the BIT_CLK output of the Tx PLL. Gen1 2.5 G, Gen2 5 G, and mix of Gen 1 and Gen 2 is 2.5 G.
		PCIE_#_TX_PLL_REF_CLK_#: Reference clock from TX_PLL.
		PCIE_#_TX_PLL_LOCK_#: Lock status input to PCIESS. Connects to the lock output of the TX_PLL.

Table 5 • **PCIESS Port List[1, 2] *(continued)***

Port Name	Direction	Description
PCIESS_LANE#_CDR_REF_CLK_#	Input	Reference clock to lane CDR. Connects to the REF_CLK input of the TX_PLL.
PCIESS_LANE_TXDn_P	Output	Transceiver differential output transmit data. n = 0, 1, 2, 3.
PCIESS_LANE_TXDn_N		
PCIESS_LANE_RXDn_P	Output	Transceiver differential input receive data. n = 0, 1, 2, 3.
PCIESS_LANE_RXDn_N		

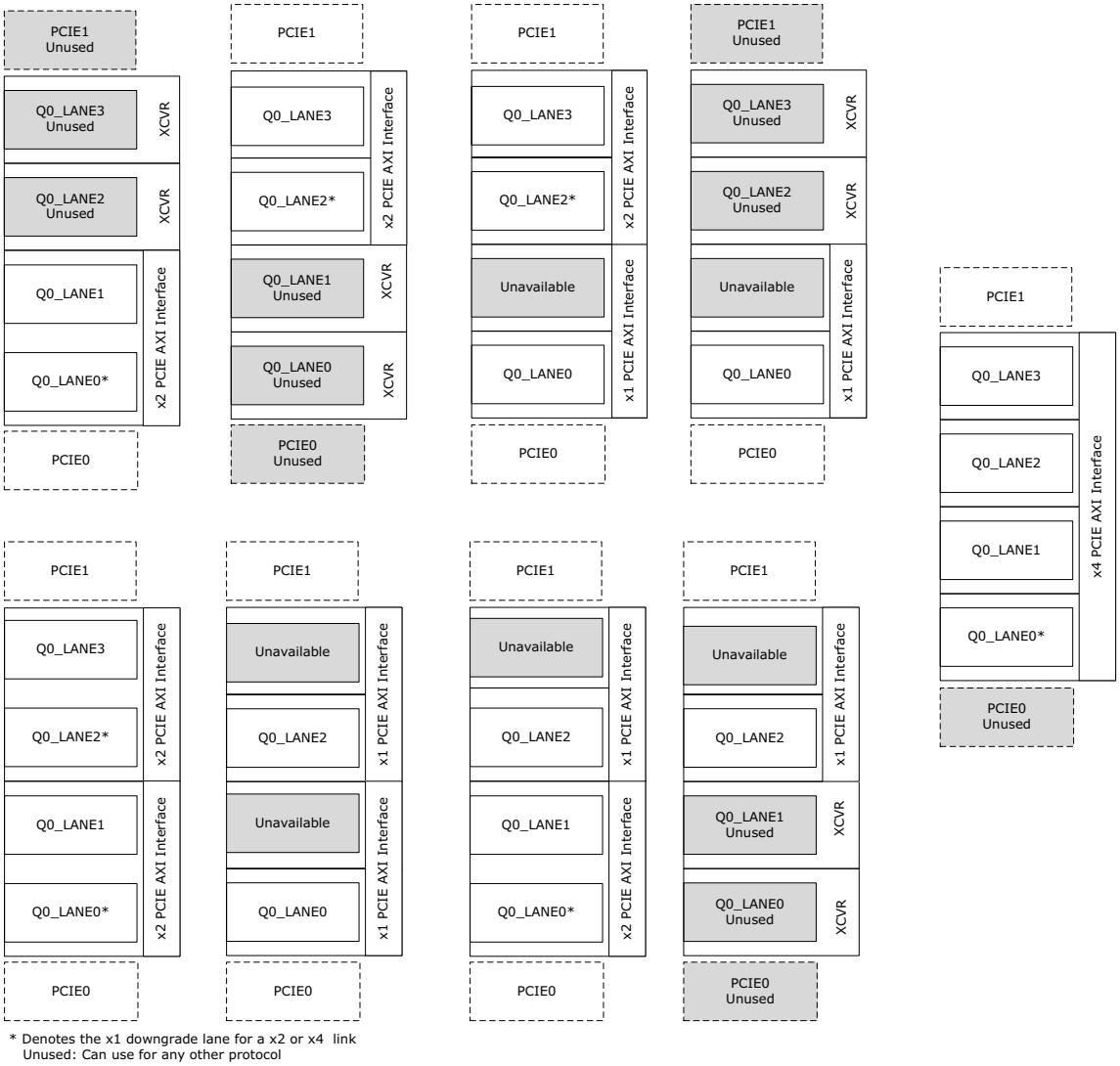
1. Unless otherwise indicated, all signals are active high.
2. # = 0, 1.

4 Implementation

The PolarFire SoC PCIe core uses several embedded blocks that are built using Libero configurators. PCIESS functionality is reserved for the Quad0 transceiver block, and this functionality allows up to two x1 or x2 PCIe endpoint/rootports links or one x4 PCIe endpoint/rootport link. PCIe 0 and PCIe 1 blocks can be used in any combination of x1 and x2 links within Quad0. A PCIe x4 link is only supported using PCIe 1, PCIe 0 is unused. The Libero configurator allows the setting of the reference clock and data rates for the PCIe block. This information is used to generate the configuration settings for the PMA as well as associated interface logic. The configurators build the components that are used to instantiate/configure specific hardware macros including the PMA and PCS blocks using the Libero SmartDesign software.

Lane[0:1] and Lane[2:3] share on-chip hardware resources that create inter-dependency between the physical lanes. The possible combinations for implementing and mixing the PCIe controllers on four physical XCVR lanes within QUAD0 are shown in the following figure.

Figure 18 • **Legal Combinations of PCIe and XCVR Protocols**



4.1 Libero Configurators

The PolarFire SoC FPGA transceiver configurator is the preferred tool for the wrapper generation needed to instantiate the transceiver primitive macros called PF_XCVR_REF_CLK, PF_TXPLL, and PF_PCIE. The configurator is part of the Libero SoC design tools and is available when the PolarFire SoC macros are downloaded into the Libero catalog. The following table provides details of three Libero configurator modules used by the Libero FPGA design when the blocks are implemented in the design. These three blocks must be instantiated and configured in the PCIe design.

Table 6 • **Libero Configurators in Libero Software**

Libero Configurator	Macro	Details
Transceiver Reference Clock	PF_XCVR_REF_CLK	Generates the reference clock based on the input to the GUI—differential or single-ended input buffer and single or dual-clock input to the transmit PLL clock network. Reference clocks for PCIESS systems use differential HCSL/LVDS. However, this can vary according to the system application.
Transmit PLL[1]	PF_TXPLL	Generates the TxPLL/TxPLL_SSC based on the input to the GUI. Typically a 100 MHz clock (Refclk) with greater than ± 300 ppm frequency stability is used for PCIe applications. For Refclk flexibility, the PCIESS block accepts 100 MHz or 125 MHz or 156.25 MHz input and translates for PCIe Gen1 or Gen2 speeds.
PCI Express	PF_PCIE	Configures the requested number of lanes with the same PMA and PCS settings—location of each lane and CDRPLL settings. The configurator has presets for all the supported protocols.

1. It is not advisable to share the TxPLL with other serial protocols that have a tight transmit jitter specification.

Each transceiver module configurator guides the user through a sequential selection of choices and defaults. Each configurator maintains a macro diagram that displays module ports based on the current configuration. When all of the choices are made, the configurator generates a macro specific to the requirements of the design. Only the relevant ports appear in the generated macro.

This section describes how to enter these configuration parameters in the Libero configurator GUIs.

A PCIe design requires the transceiver reference clock and transceivers transmit PLL blocks to be configured and instantiated in the design. For more information on TX_REF_CLK and TX_PLL block configurators, see *UG0915: PolarFire SoC FPGA Transceiver User Guide*.

4.2 PCIe Configurator

The PCIe configurator is used to build a PCIe endpoint or rootport PCIe block. The configurator sets up the correct PCIESS registers and ports based on the user inputs.

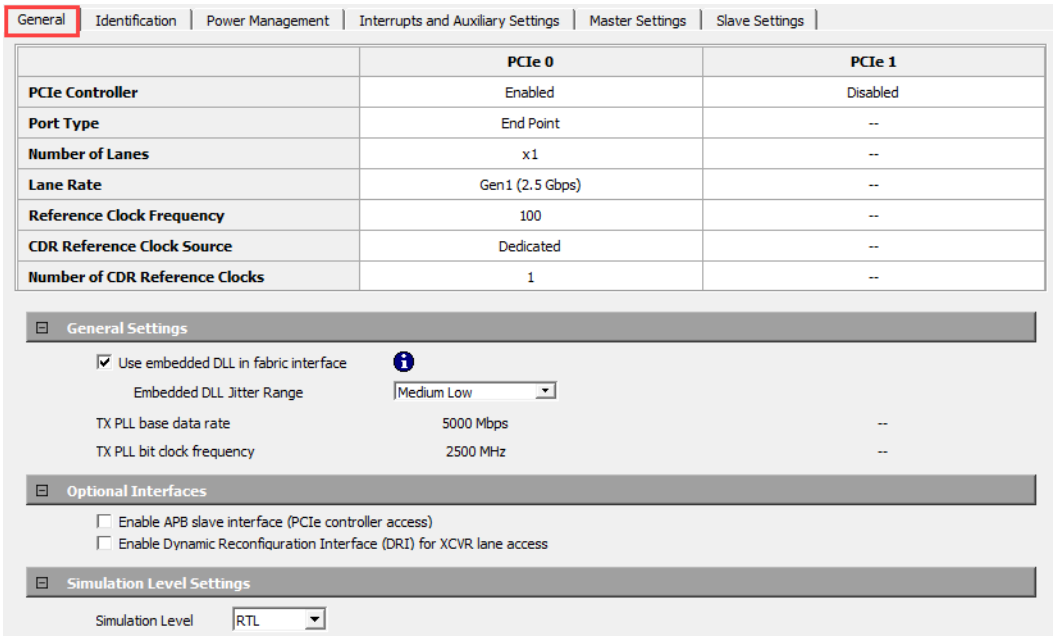
To create a configured PCIe component, follow these steps:

1. Access the PCIe module in the PolarFire SoC Features from the catalog as shown in the following figure.

Figure 19 • **PCIe Selection from Catalog**

2. Double-click **PCI Express**, **Create Component** dialog box pops-up. Enter the name of the component and then click **OK** to launch the configurator, as shown in the following figure. The GUI allows the user to select the related PCIe properties.

Figure 20 • **PCIe General Settings**



The following table lists the options available in the General tab.

Table 7 • **PCIe General Settings**

PCIE General Settings (PCIe 0 and PCIe 1)	Options	Default
PCIe Controller	Enabled and Disabled	PCIe 0 = Enabled PCIe 1 = Disabled
Port Type	End Point and Root Port	End Point
Number of Lanes	x1, x2, and x4	x1
Lane Rate	Gen1 (2.5 Gbps) and Gen2 (5.0 Gbps)	PCIe 0 = Enabled PCIe 1 = Disabled
Reference Clock Frequency (MHz)	100, 125, and 156.25	100
CDR Reference Clock Source	Dedicated and Fabric	Dedicated
Number of CDR Reference Clocks	1 and 2	1
Embedded DLL in fabric interface[1]	Enabled and Disabled	Enabled
Embedded DLL Jitter Range	Low, Medium-Low, Medium-High and High	Medium-Low
Optional Interfaces		
APB Slave Interface[2]	Enabled and Disabled	Disabled
DRI Slave Interface[3]	Enabled and Disabled	Disabled

1. AXI clock frequency must be greater than or equal to 125 MHz when the embedded DLL is enabled.
2. APB interface is used to configure PCIe control registers.
3. DRI interface is used to configure lane related registers.

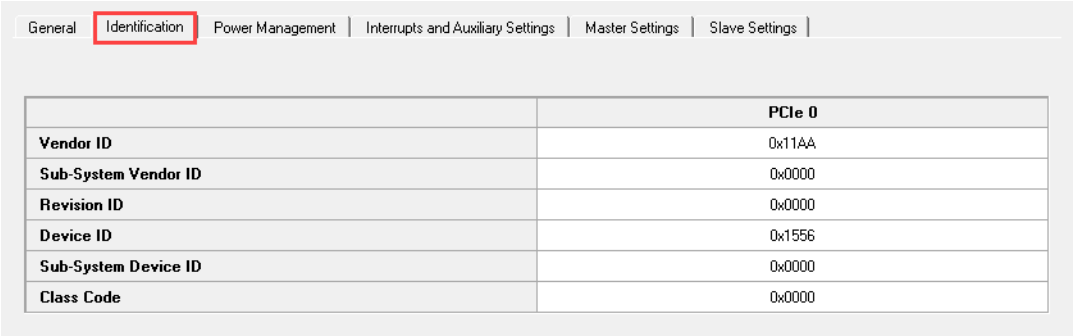
Number of Lanes: PCIe requires selection of the initial lane width. Wider lane-width cores are capable of training down to smaller lane widths if attached to smaller lane-width devices. The configurations, x2 and x4 support automatic lane reversal, allowing the PCIe link to permit board interconnections with reversed lane numbers, and the PCIESS continues to link train successfully and operate normally.

Reference Clock Frequency: PCIESS requires a 100 MHz or 125 MHz or 156.25 MHz clock input. The specified clock frequency must match with the TXPLL clock frequency.

Optional Interfaces (APB Slave/DRI Slave): Enabling these options, exposes the particular bus on the PCIESS component for connecting to the FPGA fabric of the APB and DRI.

The following figure shows the options available in the Identification tab.

Figure 21 • **PCIe Identification Settings**



The following table lists the options available in the Identification tab.

Table 8 • **PCIe Identification Settings**

PCIE Identification Settings (PCIe 0 and PCIe 1)	Options	Default
Vendor ID	User Input	0x11AA
Sub-System Vendor ID	User Programmable	0x0000
Revision ID	User Input	0x0000
Device ID	User Input	0x1556
Sub-System Device ID	User Input	0x0000
Class Code	User Input	0x0000

Vendor ID: It identifies the manufacturer of the device or application. The default value (0x11AA) is the vendor ID of Microsemi and is registered with PCI-Sig. Customized vendor identification IDs can also be used.

Sub-System Vendor ID: This ID further qualifies the manufacturer of the device or application. The default value is 0x0000 matching the vendor ID. Customized vendor identification IDs can also be used.

Note: 0x0000 is not recommended for vendor IDs or sub-system vendor ID.

Note: SSVID/SSDID value of 0x0000 causes the Endpoint to fail the PCIECV compliance tests when non-zero SSVID/SSDID is a requirement by the PCIe specification.

Revision ID: This indicates the revision of the device or application; an extension of the device ID. The default value is 0x0000. Customized revision IDs can also be used.

Device ID: A unique identifier for the application. This can be any value based on the input.

Sub-System Device ID: This is similar to sub-system vendor ID and further qualifies the device application.

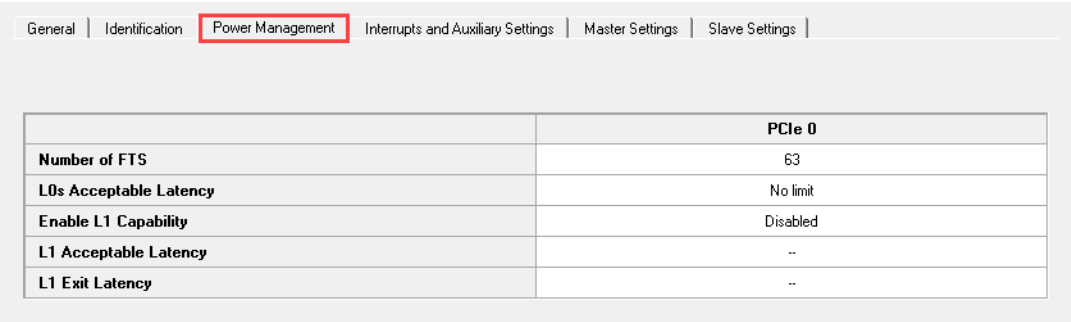
Class Code: The class code identifies the general function of a device, and is divided into three byte-size fields:

- Base Class: Broadly identifies the type of the function performed by the device.
- Sub-Class: More specifically identifies the device function.
- Interface: Defines a specific register-level programming interface.

Class code encoding details can be found at *www.pcisig.com*.

The following figure shows the options available in the Power Management tab. By selecting the power management option, allows loading settings to the PCIe config space headers.

Figure 22 • **PCIe Power Management Settings**



The following table lists the options available in the Power Management tab.

Table 9 • **PCIe Power Management Settings**

PCIE Power Management Settings (PCIe 0 and PCIe 1)	Options	Default
Number of Fast Training Sequences (FTS)	User entered	63
L0 Standby (L0s) Acceptable Latency	No Limit, Maximum of 64 ns, Maximum of 128 ns, Maximum of 256 ns, Maximum of 512 ns, Maximum of 1 µs, Maximum of 2 µs, and Maximum of 4 µs	No Limit
Enable L1 Capability	Disabled and Enabled	Disabled
L1 Acceptable Latency	No Limit, Maximum of 1 µs, Maximum of 2 µs, Maximum of 4 µs, Maximum of 8 µs, Maximum of 16 µs, Maximum of 32 µs, and Maximum of 64 µs	No Limit
L1 Exit Latency	Less than 1 µs, 1µs less than 2 µs, 2µs less than 4 µs, 4µs less than 8 µs, 8µs less than 16 µs, 16µs less than 32 µs, 32µs to 64µs, and more than 64 µs	16 µs to less than 32 µs

The PCIe base specification defines two levels of active state power management (ASPM) that are designed to provide options for trading off increased power conservation with rapid recovery to the L0 state.

Number of fast training sequences (FTS): The specific number to be repeated is defined by the receiving device and broadcast during training sequences at the link up time. The more FTS transmitted, the easier it is to obtain a receiver lock on the transmitted signal. The user can specify an input value for the number of FTS required.

L0s Acceptable Latency: This state is required by all the PCIe devices and applies to a single direction on the link. The latency to return to L0 from L0s is specified to be very short. When entering L0s, the device moving into the power saving state sends an electrical idle ordered set (EIOS) to the receiving device, and then turn off the power to its transmitter. When returning from L0s to L0, the device must first generate a specific number of small ordered known as FTS. However, the purpose of L0s is to regain receiver lock and be able to receive traffic as quickly as possible, so the receiving device selects the lowest number of FTS that ensure clock recovery based on its specific design. This selection is used to choose a time interval to achieve L0s.

Enable L1 Compatibility: The L1 ASPM is optionally enabled and can be entered to achieve a greater degree of power conservation. In this state, both directions of the link are placed in the L1 state. Return to L0 requires both devices to go through the link recovery process which results in a greater latency to return to L0, so that the power state can typically be used when activity on the link is not expected for some significant time period.

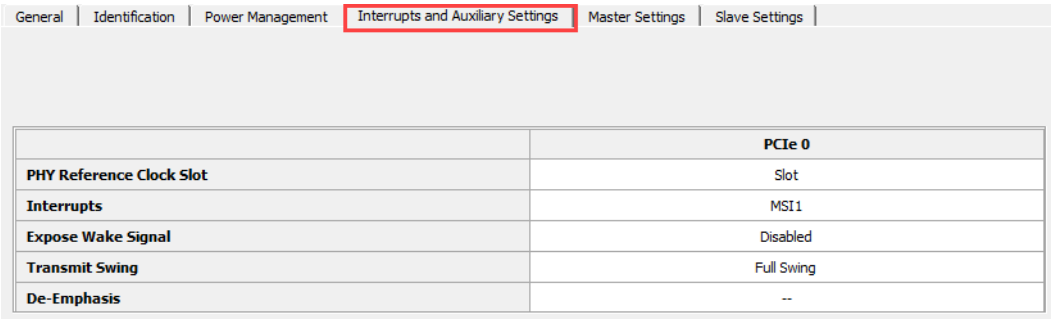
L1 Acceptable Latency: To enter the L1 state, the downstream device must first request permission from the upstream device for entering in to a deeper power conservation state. Upon acknowledgement, both devices turn off their transmitters and enter an electrical idle state. This settings gives the allowable time to wait to achieve L1.

L1 Exit Latency: Returning from L1 requires, that both devices must now go through the link recovery process. The link recovery process uses TS1 and TS2 standard ordered sets as opposed to the smaller FTSs used by L0s. This setting selects the time interval to exit from L1.

Note: When Enable L1 capability is enabled.

The following figure shows the options available in the Interrupts and Auxiliary Settings tab.

Figure 23 • **PCIe Interrupts and Auxiliary Settings**



The following table lists the options available in the Interrupt and Auxiliary Settings tab.

Table 10 • **PCIe Interrupts and Auxiliary Settings**

PCIE Interrupts and Auxiliary Settings (PCIe 0 and PCIe 1)	Options	Default
Physical layer reference clock slot	Slot and Independent	Slot
Interrupts	INTx, MSI 1, MSI 2, MSI 4, MSI 8, MSI 16, and MSI 32	INTx
Expose wake signals[1]	–	Enabled
Transmit swing	Full Swing and Half Swing	Full swing
De-Emphasis	−3.5 dB and −6 dB	−3.5dB

1. Enabled for End Point port type and disabled for Root Port by default.

PHY Reference Clock Slot: Select this option, if the PHY reference clock is either from a PCIe slot or is generated separately. Slot is a clock source shared in the PCIe system between the host and endpoint link. An Independent slot is used in a system that uses the independent clock sources on either side of the link. This setting changes the PCIe configuration space register, to advertise the used clocked topology to the system root. It makes no other functional changes to the endpoint.

Interrupts: The PCIe EP implementation supports 32 MSI interrupt and INTx interrupts. It cannot simultaneously support both the interrupts.

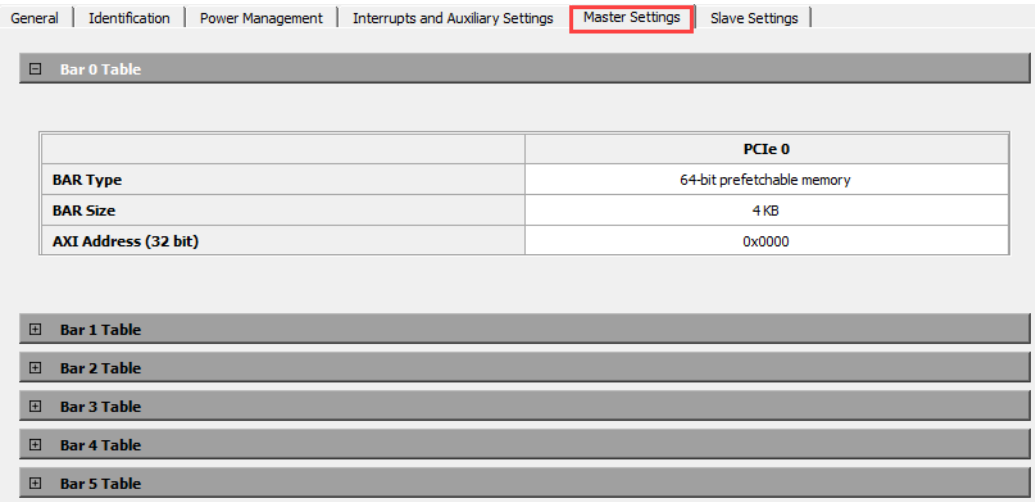
Expose Wake Signals: Enabling this option, exposes the WAKE_N input signal on the PCIESS component allowing connection to the FPGA fabric.

Transmit Swing: This sets the transmit swing for PCIe GEN 2 speed.

De-Emphasis: This sets the de-emphasis (3.5 dB and 6.0 dB) for PCIe GEN 2 speed.

The following figure shows the options available in the Master Settings tab.

Figure 24 • **PCIe Master Settings**



The following table lists the options available in the Master Settings tab.

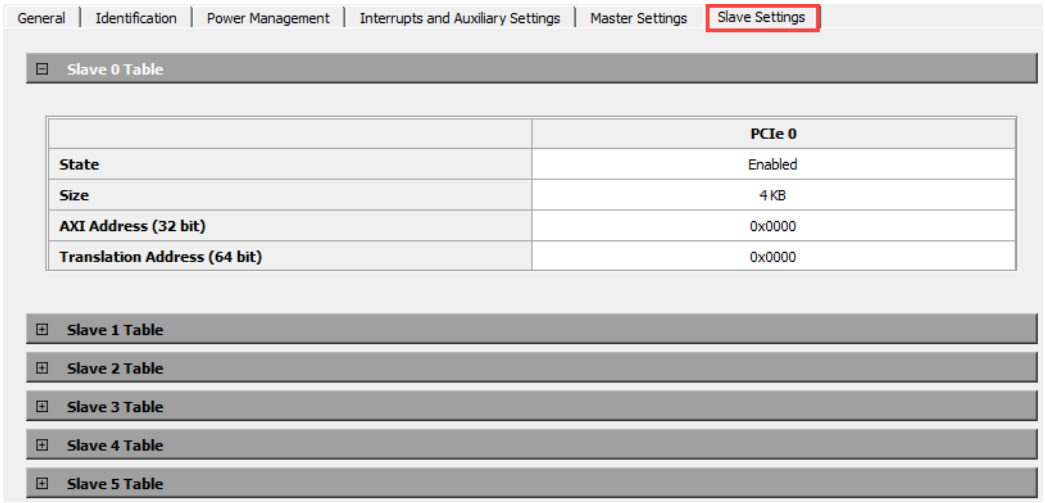
Table 11 • **PCIe Master Settings**

PCIE Master Settings (PCIe 0 and PCIe 1)	Options	Default
BAR Type	Disabled, 32-bit memory, 32-bit prefetchable memory, and 64-bit prefetchable memory	32-bit memory
BAR Size	4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB, 128 MB, 256 MB, 512 MB,1 GB, and 2 GB.	4 KB
AXI Address [32 bit]	User Input, 32-bit address to be entered, lower 12 bits must be zero.	0x0000

Note: At least one Master Bar must be enabled for PCIe 0 and PCIe 1 controllers.

The following figure shows the options available in the Slave Settings tab.

Figure 25 • **PCIe Slave Settings**



The following table lists the options available in the Slave Settings tab.

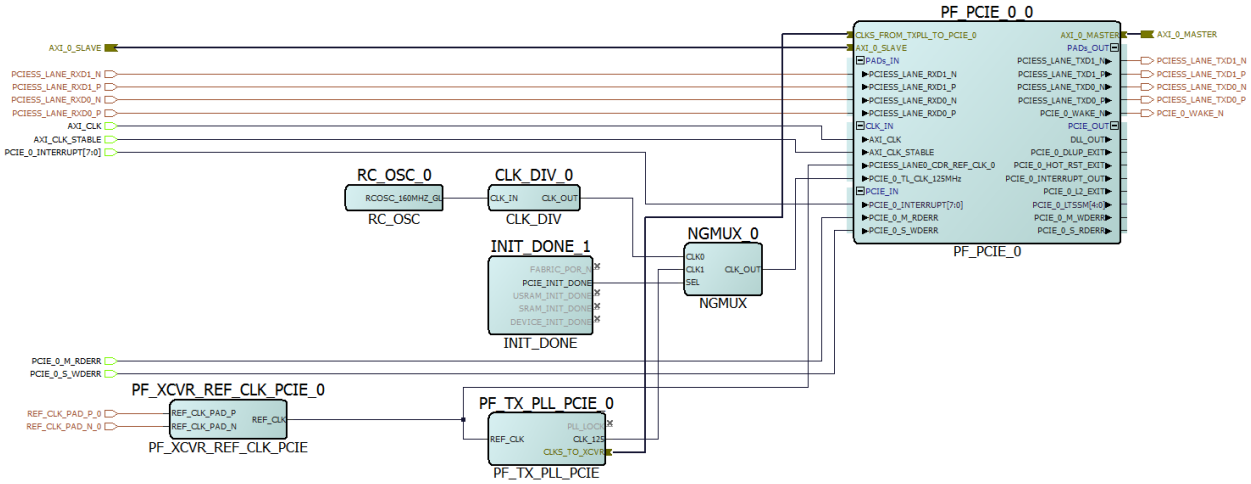
Table 12 • **PCIe Slave Settings**

PCIE Slave Settings (PCIe 0 and PCIe 1)	Options	Default
State	Disabled Enabled	Disabled
Size	4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB,128 MB, 256 MB, 512 MB, 1 GB, and 2 GB	4 KB
AXI Address [32 bit]	User Input, 32-bit address to be entered, lower 12 bits must be zero.	0x0000
Translation Address [64 bit]	User Input, 64-bit address to be entered, lower 12 bits must be zero.	0x0000

Note: When State is enabled, the size, AXI address, and translation address settings are available

3. After making all selections in the PCIe configurator, complete the generation by clicking OK.
4. The next step is to create the XCVR_REFCLK and TX_PLL modules to be instantiated and connected to the PCIe block. Typically, the REF_CLK output of the PF_XCVR_REF_CLK is connected to the respective inputs of the PF_PCIE as well as the input REF_CLK of the PF_TX_PLL. For information on XCVR block generation, see *UG0915: PolarFire SoC FPGA Transceiver User Guide*.

Figure 26 • **Complete PCIe Interface Example**



The following table lists the key connections of the SmartDesign PCIe example.

Table 13 • **Key Connections of SmartDesign**

Source	Destination
PF_XCVR_REF_CLK_PCIE_0:REF_CLK	PF_TX_PLL_PCIE_0:REFCLK
	PF_PCIE_0_0:PCIESS_LANE0_CDR_REF_CLK_0
PF_TX_PLL_PCIE_0:CLKS_TO_XCVR	PF_PCIE_0_0: CLKS_FROM_TXPLL_TO_PCIE_0
PF_TX_PLL_PCIE_0:CLK_125	PF_PCIE_0_0:PCIE_0_TL_CLK_125MHz

4.3 Design Constraints

No physical (PDC) constraints are required for PCIe. Constraints are required for PF_TXPLL and PF_XCVR_REF_CLK. The Libero software automatically places the PCIESS blocks. The PCIESS overlays the related transceiver quad 0 lanes. Therefore, when only PCIe 0 is used, it can be used as a x1 or x2 link.

Timing constraints are automatically generated by Libero SoC.

Note: Do not add user-supplied clock constraint if the paths are internal to the core.

4.4 PCIe Simulation

The PolarFire SoC PCIe supports two simulation modes:

- Bus functional model (BFM)
- Full register-transfer level (RTL) model

4.4.1 Bus Functional Model

The PCIe is simulated using the bus functional model (BFM) for the PCIESS. In this simulation mode, data transfer does not go off-chip. In the PCIe BFM simulation mode, the user can transmit/receive data from/to the fabric using the AXI4 of the PCIESS. The BFM simulation mode is selected from the Libero PCIESS configurator GUI. Libero generates the required files for BFM simulation.

The PCIe simulation mode uses the BFM commands to emulate the data that is transferred through the PCIESS block across the AXI4 bus interface to the fabric. The physical layer of the PCIe protocol is not implemented in this simulation mode. This mode is intended to validate the fabric interfaces to the PCIESS block, and the physical interface of the XCVR PMA block remains inactive.

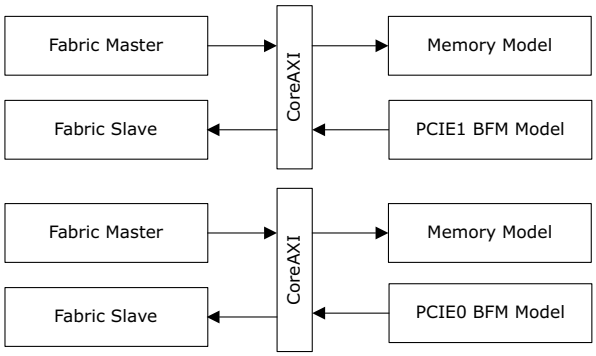
The AXI4 bus master in the PCIe BFM simulation mode enables emulating 64-bit AXI master transactions. Libero SoC generates user-customizable BFM files that instruct the model to start transactions to the fabric. The BFM allows the user to use a text file to issue the transactions from the PCIe AXI master interface to the fabric, to exercise the design. The user must include BFM instructions in the `<project>/simulation/PCIE_<0:1>_user.bfm` file. The BFM model interprets these instructions and initiates AXI transactions in sequence. The PCIE_init.bfm model is not user-editable.

The AXI bus slave available in the BFM_PCIe simulation mode provides a 64-bit slave interface for fabric communication. The user can interact with the slave by initiating write and read bus transactions using the appropriate bus master. The slave acts as a memory model, so whatever is written to the slave can be read back from the same address.

The BFM commands used in the PCIESS BFM files are similar to the BFM commands used by the bus masters.

The following figure shows the PCIESS BFM structure.

Figure 27 • **PCIESS BFM Structure**



Note: There are additional BFM commands that are only used for the PCIe AXI BFM simulation, to emulate 64-bit AXI transactions.

The command, `write64 w <base_address> <base_address_offset> <32-bit MSB> <32-bit LSB>`, makes the bus master start a 64-bit write transaction on the external bus for a slave with address given by the <base_address> and <base_address_offset>, using the data generated by <32-bit MSB> and <32-bit LSB >.

For example: `write64 w 0x00000000 0x0 0xA0A1A2A3 0xB0B1B2B3;`

The command, `readcheck64 w <base_address> <base_address_offset> <32-bit MSB> <32-bit LSB>`, makes the bus master start a 64-bit read transaction for the address given by the <base_address> and <base_address_offset>. It compares the 64-bit read data to the data.

The command **setup 0x8 <source address> <destination address>** is used to set source and destination address for DMA.

The command **setup 0xA <data>** is used to set DMA data source.

<data> =0 ==> Data increment by 1 starting from 0x1

<data> =1 ==> Random data

<data> =2 ==> Data from DMADATA.vec file

For example, **setup 0xA 0x2 data_in.vec**. In this command, DMA data source is `data_in.vec` file.

The command **setup 0x9 <DMA_Length> <Control>** is used for DMA control.

set control bit0 to '1' => start DMA

set control bit1 to '1' => sets transfer from PCIe domain to Fabric domain

set control bit2 to '1' => sets transfer from Fabric domain to PCIe domain

When control = 0x3, BFM starts DMA transfer from PCIe to Fabric and when control = 0x5, BFM starts DMA transfers from Fabric to PCIe.

4.4.2 Full Register-transfer Level (RTL) Model

In this simulation mode, the register-transfer level (RTL) model of the PCIESS is used, and the entire data path through the PCIESS is exercised. It requires a third-party verification IP (VIP) model for PCIe. The user is responsible for the VIP model for the PCIe.

When using VIP models, ensure the following:

1. Verification IP must be configured properly.
   - BFM type: indicate type of BFM (0 – Root port and 1 – End point).
   - Number of lanes: indicates number of connected lanes
   - I/O size: specifies the size of internal I/O space. The values range from 12 to 24.
   - MEM32_SIZE: specifies the size of internal 32-bit addressing memory space. The values range from 12 to 24.
   - MEM64_SIZE: specifies the size of internal 64-bit addressing memory space. The values range from 12 to 24.
   - PCLK: PIPE clock frequency depends on the signaling rate and PIPE interface width configuration.
2. The receiver pin for XCVR should not be in an unused state. The following example code snippet is used in the testbench to prevent the transmitter pin from going into an unused state.
   ```
   rxp[i] <=(tx_1b[i]===1'bX || tx_1b[i]===1'bZ) ? 1'b0 :   tx_1b[i];
   rxn[i] <=(tx_1b[i]===1'bX || tx_1b[i]===1'bZ) ? 1'b0 :  ~tx_1b[i];
   ```
3. The receiver pin for VIP model should not be in an unused state. The following example code snippet is used in the testbench to prevent the transmitter pin from going into an unused state.
   ```
   rx_1b[i] <=(txp[i]==txn[i] || txp[i]===1'bX) ? 1'bZ : txp[i];
   ```
   Where,
   i – Number of BFM lanes.
   txp and txn – Transmitter pins from XCVR
   rxp and rxn – Receiver pins from XCVR
   tx_1b – transmitter pin from VIP model
   rx_1b – receiver pin from VIP model

4.5 PCIe Subsystem Performance

Throughput is the amount of data transferred over a given period of time. The following table lists the throughput values of PCIe AXI master interface and AXI slave interface.

4.5.1 PCIe AXI Master IF Throughput

Throughput calculation is carried out at 250 MHz AXI CLK using built-in DMA (64 KB of DMA size), maximum payload of 128 Bytes. The built-in DMA initiates 32-bit AXI burst length transactions on AXI Master IF. When using the built-in DMA, eight PCIe outstanding transactions are supported.

Table 14 • **PCIe AXI Master IF Throughput**

		PC to LSRAM (Memory Read from PC)		LSRAM to PC (Memory Write to PC)		
Link width	Link speed	Maximum Throughput (MBps)	% of Theoretical Throughput	Throughput (MBps)	% of Theoretical Throughput	Maximum Theoretical Throughput (MBps)
x1	Gen1	206	82.4%	226	90.4%	250
	Gen2	411	82.2%	453	90.6%	500
x2	Gen1	410	82%	439	87.8%	500
	Gen2	814	81.4%	875	87.5%	1000
x4	Gen1	811	81.1%	830	83%	1000
	Gen2	1181	59.05%	1508	75.4%	2000

4.5.2 PCIe AXI Slave IF Throughput

Throughput calculation is carried out at 250 MHz AXI CLK using fabric AXI DMA (64KB of DMA size), maximum payload of 128 Bytes. The fabric AXI DMA 256-beat AXI burst length transactions on AXI slave IF. When using AXI slave IF, four PCIe outstanding transactions are supported.

Table 15 • **PCIe AXI Slave IF Throughput**

		PC to LSRAM (Memory Read from PC)		LSRAM to PC (Memory Write to PC)		
Link width	Link speed	Maximum Throughput (MBps)	% of Theoretical Throughput	Throughput (MBps)	% of Theoretical Throughput	Maximum Theoretical Throughput (MBps)
x1	Gen1	204	81.6%	226	90.4%	250
	Gen2	411	82.2%	453	90.6%	500
x2	Gen1	397	79.4%	440	88%	500
	Gen2	599	59.9%	877	87.7%	1000
x4	Gen1	483	48.3%	831	83.1%	1000
	Gen2	665	33.25%	1646	82.3%	2000

PCIe Throughput depends on following factors:

- PCIe uses 8b10b encoding, which causes 20% reductions in throughput.
- The maximum PCIe read throughput also depends on the supported PCIe outstanding transactions and Round-trip time (RTT).
- Maximum effective bandwidth is the rate at which valuable data is transferred at a particular point. It does not include transaction overhead, such as headers, sequence numbers, CRCs, ECRCs, and other packets like DLLPs and SKIP advanced sets.
  Maximum Effective Bandwidth = data/(data + overhead)

The following table lists the relation between maximum transaction payload size and efficiency. Increasing the transaction payload size (increasing the burst length) also improves throughput.

Table 16 • **Relation Between Maximum Transaction Payload Size and Efficiency**

Maximum Transaction Payload Size (Byte)	Efficiency
128	86%
256	92%

4.6 Initialization

When the device powers up, some registers contained in the PCIESS are automatically initialized using data stored in the non-volatile storage of the device. The values associated with these registers are set either using flash bits or based on fixed values. Some registers are also loaded at power-up through an initialization mechanism that is programmed into devices implementing PCIESS blocks. This initialization is autonomously generated by Libero and is transparent. For information about initialization, see *UG0890: PolarFire SoC FPGA Power-Up and Resets User Guide*.

5 Configuration Registers

The PCIe settings can be reconfigured through 32 bit-wide configuration space registers, which include:

- Information registers, which provide device, system, and bridge identification information.
- Bridge configuration registers, which enable configuration of the bridge functionality. These include:
  - Read-only registers that report control and status registers to the AXI4 bus
  - Bridge settings that must be configured at power-up, such as local interrupt mapping.
- Control/status registers, which can be used by the AXI4 bus to control bridge behavior during an operation.
- Power management registers, which configure the power management capabilities of the bridge.
- Address mapping registers, which provide address mapping for AXI4 master and slave windows used for address translation.
- Root port and endpoint interrupt registers.
- PCIe control and status registers, which enable the local processor to check the PCIe interface status. These read-only registers enable the local processor to detect the initialization of the bridge's PCIe interface and monitor PCI link events.

Some registers are hardwired to a fixed value within the embedded PCIESS block.

For information about Configuration registers, see *PolarFire SoC Register Map*.

6 PCIe Configuration Space

The following tables lists the layout of the PCI express configuration space and provides the mapping for each register in the space.

Table 17 • **PCI Configuration Space**

Offset	Description
0x00 to 0x03C	Type0 (endpoint) or Type1 (Root port/Bridge/Switch) Standard PCI configuration header
0x040 to 0x07C	Reserved
0x080 to 0x0B8	PCI Express Capability
0x0BC to 0x0CC	Reserved
0x0DC	Reserved
0x0E0 to 0x0F4	MSI Capability
0x0F8 to 0x0FC	PCI Power Management Capability

Table 18 • **PCI Express Capability Structure**

Byte offset	Bit Number			
	31:24	23:16	15:8	7:0
0x080	Capability Register		Next capability Pointer	Capability ID
0x084	Device capabilities			
0x088	Device status		Device control	
0x08C	Link capabilities			
0x090	Link status		Link control	
0x094	Slot capabilities			
0x098	Slot status		Slot control	
0x09C	Root capabilities		Root control	
0x0A0	Root Status			
0x0A4	Device capabilities 2			
0x0A8	Device status 2		Device control 2	
0x0AC	Link capabilities 2			
0x0B0	Link status 2		Link control 2	
0x0B4	Slot capabilities 2			
0x0B8	Slot status 2		Slot control 2	

Table 19 • **MSI Capability Structure**

Byte offset	Bit Number			
	31:24	23:16	15:8	7:0
0x0E0	Message Control		Next pointer	Capability ID
0x0E4	Message Address			
0x0E8	Message Upper Address			
0x0F0			Message Data	

Table 20 • **Power management Capability Structure**

Byte offset	Bit Number			
	31:24	23:16	15:8	7:0
0x0F8	Power management capabilities		Next item pointer	Capability ID
0x0FC	Data	PMCSR_BSE bridge support extensions	Power management control and status registers	

Table 21 • **PCI Express Extended Configuration Space**

Offset	Description
0x100 to 0x 104	Vendor-specific capability with VSECID = 1556h; RevID = 1h
0x108 to 0x10C	Latency Tolerance Reporting capability
0x200 to 0x234	Advanced Error Reporting capability

Table 22 • **Vendor Specific Extended Capability Structure**

Byte offset	Bit Number			
	31:24	23:16	15:8	7:0
0x100	Vendor-Specific Extended Capability Header			
0x104	Vendor-Specific Header			

Table 23 • **Latency Tolerance Reporting Capability Structure**

Byte offset	Bit Number			
	31:24	23:16	15:8	7:0
0x108	PCI Express Extended Capability Header			
0x10C	Max No-Snoop Latency Register		Max Snoop Latency Register	

Table 24 • **Advanced Error Reporting Capability Structure**

Byte offset	Bit Number			
	31:24	23:16	15:8	7:0
0x200	PCI Express Enhanced Capability Header			
0x204	Uncorrectable Error Status Register			
0x208	Uncorrectable Error Mask Register			
0x20C	Uncorrectable Error Severity Register			
0x210	Correctable Error Status Register			
0x214	Correctable Error Mask Register			
0x218	Advanced Error Capabilities and Control Register			
0x21C	Header Log Register			
0x22C	Root Error Command			
0x230	Root Error Status			
0x234	Error Source Identification Register			

7 Board Design Recommendations

This chapter discusses board-level implementation details of a PCIe design using PolarFire SoC FPGAs. Optimal performance requires understanding the functionality of the device pins and properly addressing issues such as device interfacing, protocol specifications, and signal integrity.

For more information, see *UG0901: PolarFire SoC FPGA Board Design User Guide*.

Various specifications from PCI-SIG apply depending on the form factor of the design. This chapter focuses on a subset of these specifications centered on chip-to-chip and add-in card implementations.

For more information, see the *PCI Express Base Specification, Revision 2.0* and *PCI Express Card Electromechanical Specification (CEM) Revision 2.0* from PCI-SIG.

7.1 AC-Coupling

PCIe electrical signals require a 75 - 200 nF AC-coupling capacitor between the transmitter and receiver. All transmitters are AC-coupled, either within the media or within the transmitting component itself. If located within the media, the AC-coupling capacitors are placed close to the transmitter. The AC-coupling capacitor is used in conjunction with internal termination for PCIe link detection.

7.2 Lane Reversal

The PCIESS supports lane reversal when required, allowing the PCIe physical I/O to be reversed with the block's logical lanes for a more flexible PCB layout. Lane reversal functionality is incorporated into the PCIESS to be layout-agnostic with respect to lane ordering and lane polarity.Using lane reversal can ease routing congestion on the PCB, leading to a cleaner interface between the PCIe host and the endpoint or root port.

7.3 Polarity Inversion

The PolarFire SoC transceiver block with PCIESS supports differential polarity inversion. Receiver polarity is automatically detected by the PCIESS during link training, as defined in the PCIe specification. The differential data received by the transceiver RX are reversed if RXP and RXN differential traces are swapped on the PCB accidentally. The transceiver RX inversion allows within the PCIESS to offset the reversed polarity of a serial differential pair.

7.4 PCIe Power-Up

The PCIe specification provides timing requirements for power-up. The PCIe connector specification specifies that the fundamental reset (PERST_N) be de-asserted at a minimum of 100 ms from the point of power being stable. The PCIe PERST_N signal release time (known as PCIe timing parameter TPVPERL) of 100 ms is used for the PCIe card electro-mechanical specification for add-in cards.

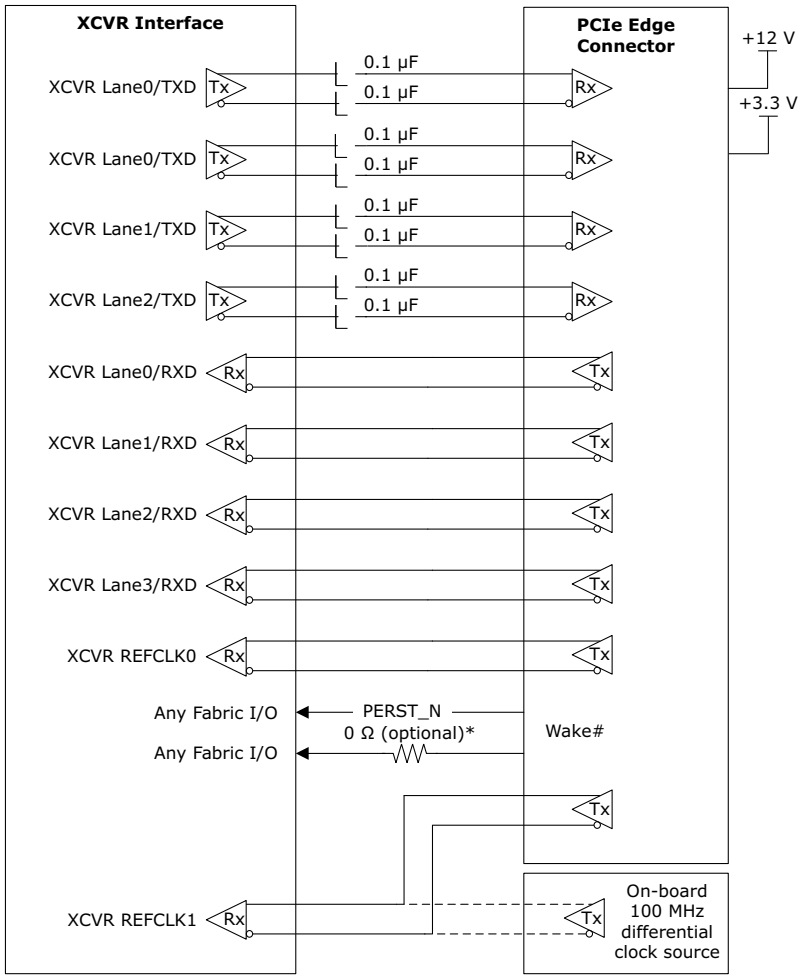
The semi-autonomous nature of the PCIESS in a PolarFire SoC device allows the device to quickly move from power-up to link detect. The PolarFire SoC transceiver initially terminates to 50 kΩ for hot-swap protection but quickly returns to 100 Ω termination so that link detection operates within the PCIe specifications. When the device is detected by the root, it proceeds to the polling state of the LTSSM. The link then cycles through the remaining LTSSM states. In cases where the root point and the endpoint power-up separately, the PERST_N signal must be used to handshake the link startups.

7.4.1 PCIe Edge Connector

PCIe is a point-to-point serial differential low-voltage interconnect supporting up to four channels. Each lane consists of two pairs of differential signals: transmit pair, receive pair, XCVR_x_TXy_P/N, and XCVR_x_RXy_P/N. Each signal has a 2.5 GHz embedded clock.

The following figure shows the connectivity between the PolarFire SoC FPGA transceiver interface and the PCIe edge connector.

Figure 28 • **Connectivity Between XCVR Interface and PCIe Edge Connector**



Note: Between the fabric I/O and the host may require additional components to match 3.3 V levels.