UG0447
User Guide
SmartFusion2 and IGLOO2 FPGA High-Speed Serial Interfaces

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 10.0

The following is a summary of the changes made in revision 10.0 of this document.

- Information about PCIe BAR was updated. See Base Address Register Settings, page 40.
- Information about SYSTEM_SER_INTERRUPT register bits was updated. See Table 191, page 164.
- Information about MAX_PAYLOAD_SIZE was updated. See Table 46, page 61 and Maximum Payload Size, page 45.
- Removed information related to AHBLite support for PCIe.
- Information about Table 184, page 162 was updated.
- Information about power supply for SERDES was added. See Calibration Resource Sharing, page 141.

1.2 Revision 9.0

The following is a summary of the changes made in revision 9.0 of this document.

- PCIe Base Specification Revision changed from v2.1 to v2.0. See Features, page 18.
- Information about PCIe Fabric Interface (AXI3), page 23 was updated.
- Information about PCIe Interrupts for Endpoints, page 48 was updated.
- Information about PCIE_ERROR_COUNTER register was updated. See Table 21, page 50.
- Information about Expansion ROM Base Address was removed from Table 100, page 79.
- Information about SerDes Startup, page 142 was added.
- Information about all the BAR SIZEs were updated. See Table 185, page 162, Table 186, page 163, and Table 187, page 163.
- Information about reset value of RXIDLE_MAX was updated. See Table 218, page 178.
- Information about AXI3 Master Block, page 30 was updated.
- Information about AXI_M_AWLEN[3:0] and AXI_M_AWBURST[1:0] port was updated. See Table 10, page 32.
- Information about PCIE_L2P2_ACTIVE port was updated. See Table 15, page 35.
- Information about User Data Throughput, page 45 was updated.
- Information about Simulating SERDESIF in EPCS Mode, page 117 was updated.
- Information about Reference Clock Inputs, page 139 was updated.
- Information about AXI_S_RRESP[1:0] was updated. See Table 11, page 33.
- Information about IP Core Status register was updated. See Table 101, page 79.
- Information about SmartFusion2 and IGLOO2 XAUI, page 89 was updated.

1.3 Revision 8.0

The following is a summary of the changes made in revision 8.0 of this document.

- Information about AXI3 Master Block was updated. PCIe supports only little-endian order, see AXI3 Master Block, page 30.
- Information about PCIe interrupts was added, see PCIe Interrupts for Endpoints, page 48.
- Information about AHBL and AXI3 limitations was added. See AXI3 Transaction and TLP Ordering Rules, page 31.
- Updated Table 15, page 35, Table 160, page 123, Table 195, page 165, Table 215, page 171, and Table 226, page 180.

1.4 Revision 7.0

The following is a summary of the changes made in revision 7.0 of this document.

- Added configuration registers access information in "Features" section. For more information, see SERDESIF Block, page 4.
- Updated section PCIe IP Block with AXI3 Interface, page 28.
- Added Table 128, page 95, Table 133, page 99, Table 163, page 133, and Table 164, page 137.
- Updated Figure 40, page 98.
- Added signal type for Table 132, page 98 and Table 134, page 99.
- Updated the bit number information for Table 141, page 108.
- Added note for Table 144, page 109.
- Added Figure 74, page 136.
- Updated section SERDES in EPCS Mode, page 144.

1.5 Revision 6.0

The following is a summary of the changes made in revision 6.0 of this document.

- Added Note to Figure 31, page 47 (SAR 80171).
- Added "Register Lock Bits Configuration" section in SERDESIF Register Access Map, page 150. (SAR 79857).
- Updated Table 1, page 3 in "About This Guide" Chapter (SAR 76047).
- Updated "AXI3 Master Block" section in PCI Express, page 18 (SAR 75336).
- Updated "AHBL Master Interface" section in PCI Express, page 18 (SAR 75336).
- Updated "Introduction" section in XAUI, page 88 (SAR 79176).
- Updated "Reset and Clocks for XAUI" section in XAUI, page 88 (SAR 79176).
- Updated Table 171, page 158 in SERDESIF Register Access Map, page 150 (SAR 74840).
- Updated Figure 84, page 150, Added Table 212, page 169, Table 213, page 170, and Table 214, page 170 (SAR 80645).
- Added note "PCIE1 is available in M2S/M2GL060 and M2S/M2GL090 devices" to Table 168, page 154, Table 173, page 158, Table 174, page 159, and Table 177, page 160 (SAR 80645).
- Updated Table 170, page 156 (SAR 80908).
- Updated Table 216, page 177 (SAR 80907).
- Updated "Acquiring Bit Lock for CDR PLL" section in Serializer/De-serializer, page 128 (SAR 80907).

1.6 Revision 5.0

The following is a summary of the changes made in revision 5.0 of this document.

- Updated Table 21, page 50 (SAR 73409).
- Updated Table 10, page 32 and Table 11, page 33 (SAR 73758).
- Updated Table 198, page 166 in SERDESIF Register Access Map, page 150 Chapter (SAR 71619).
- Updated Table 266, page 191 in SERDESIF Register Access Map, page 150 Chapter (SAR 71077).

1.7 Revision 4.0

The following is a summary of the changes made in revision 4.0 of this document.

- Updated the User Guide (SARs 49906, 55434, 54010, 50698, 50672, 64677, 65871, 67613, 67187, 63671, 65917, 69885, and 69996).
- Updated SERDESIF Block, page 4 (SARs 57190, 56868).
- Updated PCI Express, page 18 (SARs 58293, 56134, 57424, 56868).
- Updated XAUI, page 88.
- Updated EPCS Interface, page 112 (SAR 57322).
- Updated Serializer/De-serializer, page 128.
- Updated SERDESIF Register Access Map, page 150 (SAR 58405).

1.8 Revision 3.0

The following is a summary of the changes made in revision 3.0 of this document.

- SmartFusion2 and IGLOO2 User Guides are consolidated in this revision (SARs 47394, 56737, 59646, 58292, 62283, 57915, 58472, 59157, 60283, 59773, 60437, 54009, 61529, 60510, and 59645).

- Updated SERDESIF Block, page 4 (SAR 44571).
- Updated Serializer/De-serializer, page 128 (SAR 44572).
- Consolidated SmartFusion2 and IGLOO2 User Guides.

1.9 Revision 2.0

The following is a summary of the changes made in revision 2.0 of this document.

- Updated the User Guide (SARs 57190, 57466, 56868, 57466, 56134, and 57322).
- Updated "PCIe Protocol" section (SAR56868).
- Updated Table 1, page 5 (SAR 57190).

1.10 Revision 1.0

The following is a summary of the changes made in revision 1.0 of this document.

- Updated the user guide (SAR 42443).
- Updated the User Guide (SARs 49906, 55434, 54010, 50698, and 50672).
- Updated Serializer/De-serializer, page 128 (SAR 42156 and 42155).
- Updated SERDESIF Block, page 4 (SAR 42912).

2 SERDESIF Block

2.1 Introduction

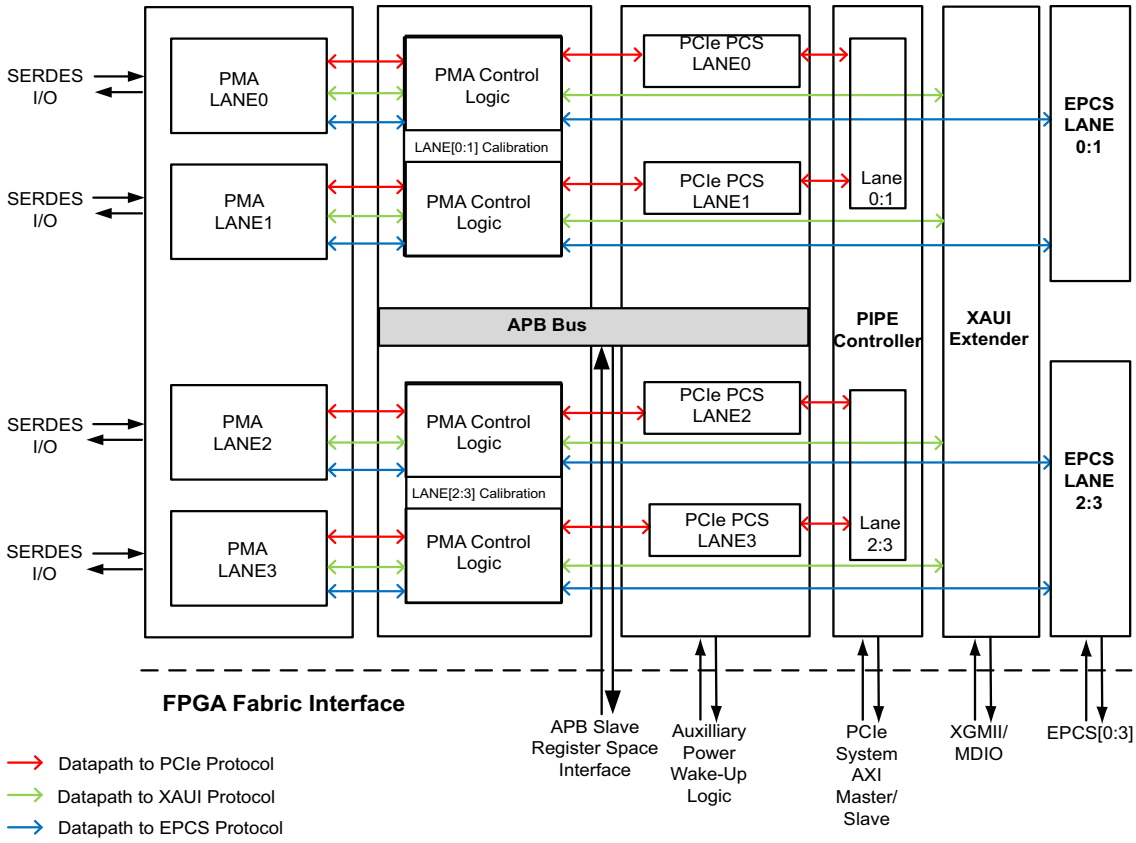
The high-speed serial interface block of SmartFusion2 and IGLOO2 FPGA devices, also known as serializer/de-serializer interface (SERDESIF), supports several serial communication standards. This module integrates several functional blocks to support multiple high-speed serial protocols in the FPGA. The only difference between the SmartFusion2 and IGLOO2 SERDES implementations is the means to initialize and configure the SERDESIF block. In the SmartFusion2 device, the embedded ARM Cortex-M3 processor within the MSS is used to perform these operations. Whereas, the IGLOO2 family performs the same functions within the HPMS.

2.2 Features

The SERDESIF block has the following features:

- PCIe protocol support.
- XAUI protocol support.
- External physical codings sub-layer (EPCS) interface supports user defined high serial protocols, such as Serial gigabit media independent interface (SGMII) 1000-BaseX and JESD204B protocol support.
- Single or multiple serial protocol modes of operation. In multiple serial protocol modes, two protocols can be implemented on the four physical lanes of the SERDESIF block.
- ARM AMBA APB-3 compliant slave interface for SERDESIF configuration registers access. Refer https://www.arm.com/products/amba-open-specifications.php.

Figure 1 • **SmartFusion2 and IGLOO2 SERDESIF Block Diagram**



2.3 Device Support

The following table shows the total number of SERDESIF blocks available in each IGLOO2 device.

Table 1 • **Available SERDESIF Blocks in SmartFusion2 and IGLOO2 Devices**

	M2S/M2GL 005	M2S/M2GL 010	M2S/M2GL 025	M2S/M2GL 050	M2S/M2GL 090	M2S/M2GL 060	M2S/M2GL 150
SERDESIF available	0	1	1	Up to 2	1	1	Up to 4
SERDES Lanes	0	4	4	8	4	4	16

Note: The specified number of SERDESIF blocks varies depending on the device package.

Note: M2S/M2GL060/090 application interfaces have dual PCIe controller capability supporting up to two x1 or x2 endpoints within a SERDESIF. It can also support one x4 endpoint.

2.4 SERDESIF Overview

SmartFusion2 and IGLOO2 device families have up to four integrated high-speed serial interface blocks (SERDESIF[3:0]). Each SERDESIF block interfaces with fabric, program control, and four duplex SERDES differential I/O pads. Figure 1, page 4 shows the inclusive high level view of the SmartFusion2 or IGLOO2 SERDESIF block. Dependent on the implemented protocol, the SERDESIF provides an AXI3, XGMII (XAUI) or native SERDES clock, and data (EPCS) along with the control plane interface APB.

SERDESIF is initially programmed at power-up with predefined parameters determined during the FPGA design flow using the Libero® SoC design software.

Each of these SERDESIF blocks includes:

- SERDESIF: Entire block implements up to four channels of high speed I/O, the physical media attachment layer (PMA), and a physical coding sub-layer (PCS) of PCIe protocols. This PCS layer is compliant to the Intel PIPE 2.0 specification. It also implements the PMA calibration and control logic. The PCIe PCS functionality can be bypassed completely in order to use the SERDES lanes for protocols other than PCIe. This allows use of the PMA in various PHY modes and implements various protocols in the SmartFusion2 and IGLOO2 devices. Refer to the Serializer/De-serializer, page 128 for more information on the SERDES block.
- PCIe system: This block implements the x1, x2, and x4 lane PCIe endpoint (regular and reverse lanes mode) with an AXI3 interface to the fabric. The SmartFusion2 and IGLOO2 PCIe is compliant with the PCIe Base Specification 1.1 for Gen1 and PCIe Base Specification 2.0 for Gen1 or Gen2. Refer to the PCI Express, page 18 for more information on the PCIe system block.
- XAUI Extender: This block is an XGMII extender to support the XAUI protocol through a FPGA IP MAC core in the FPGA fabric. Refer to the XAUI, page 88 for more information.
- EPCS: This block is a basic mode used to extend the SERDES for custom support access to the FPGA fabric. Refer to the EPCS Interface, page 112 for more information.
- SERDESIF system register: The SERDESIF system registers control the SERDES block module for single protocol or multi-protocol support implementation. These registers can be accessed through the 32-bit APB interface, and the default values of these registers are configured using Libero System On-Chip (SoC) software. See SERDESIF Register Access Map, page 150 for detailed register access descriptions. The SERDESIF is initially configured at power-up with parameters determined during the FPGA design flow using the Libero SoC software. The SERDES block

Configuration can subsequently be changed by writing the related control registers through the Advanced Peripheral Bus (APB) interface.

Table 2 • **SERDESIF Module Single Protocol Usage Overview**

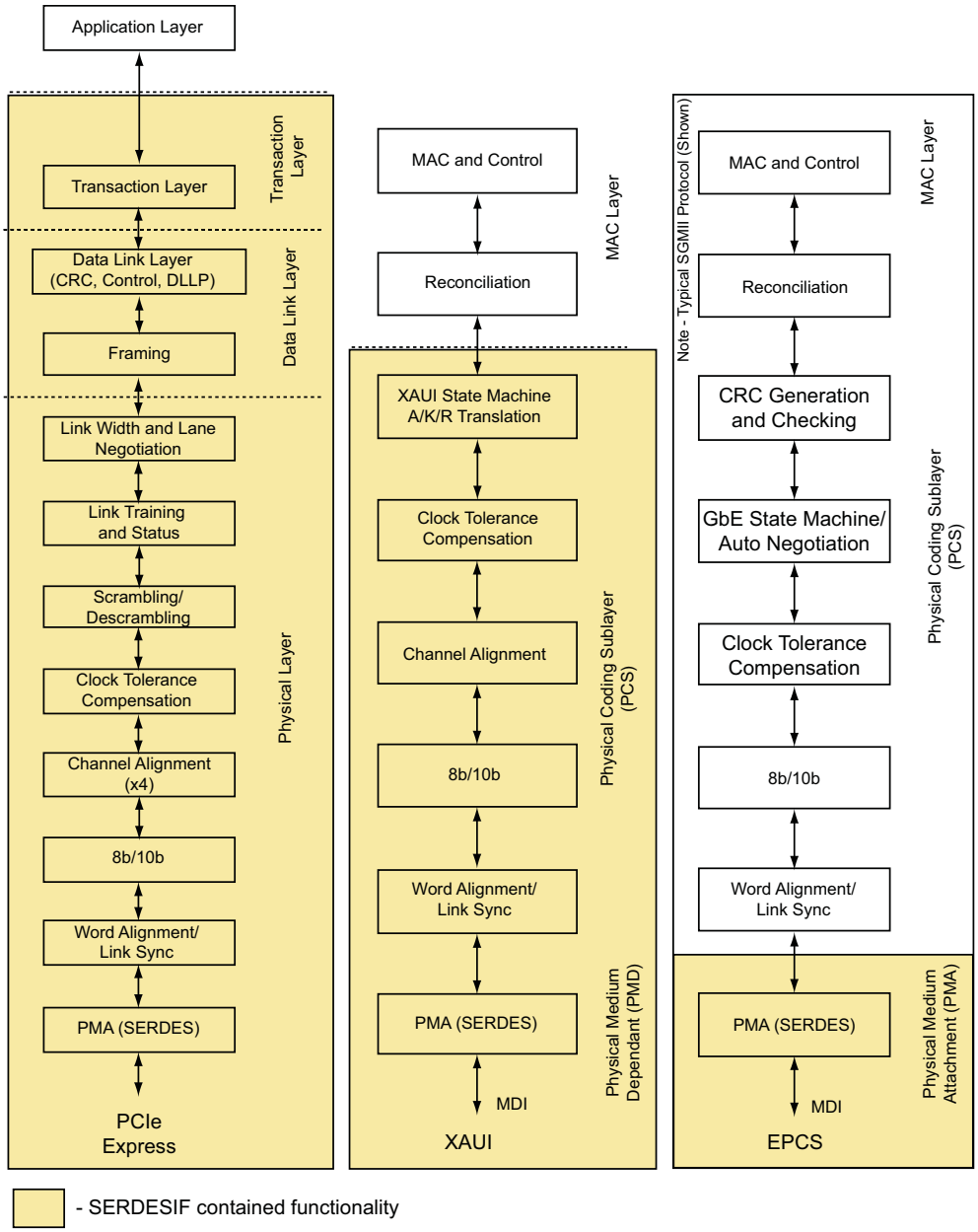
Protocol	SERDESIF Description	Data Rate (bps)	Reference Clock (typ) Input Frequency
PCIe	SERDESIF is configured to use PCIe x4, x2, and x1 link mode. The PCIe link can be configured in Regular or Reversed modes. In PCIe only mode, unused lanes are forced to RESET state and the Extender XAUI block is put in RESET state.	2.5G/5G	100 MHz
XAUI	SERDESIF is configured to use all four lanes. In XAUI mode, all lanes are used and the PCIe system is put in RESET state.	4 x 3.125G	156.25 MHz
EPCS	In EPCS mode, any serial protocol can be run though the EPCS interface to fabric using the EPCS interface. The PCIe system and XAUI blocks are put in an inactive RESET state. EPCS mode is used for implementing many other standard protocol interfaces such as JESD204B, 1000Base-X, and SGMII.	User defined	100 - 160 MHz

2.4.1 SERDESIF Serial Protocols Support

The SERDESIF block supports the implementation of multiple high speed serial protocols. Although each of the serial protocols is unique, all of them are layered protocol stacks, and the implementation can vary greatly from one layer to the next layer. Typically, the physical layer consists of fixed functionality that is common to multiple packet-based protocols, while the upper layers tend to be more customizable.

The advantage of being able to connect the FPGA logic and the SERDESIF blocks is that it allows multiple serial protocols in SmartFusion2 and IGLOO2 devices. Figure 1, page 4 shows the implementation of PCIe, XAUI, and customized protocols using the SERDESIF block and FPGA fabric. The Figure shows the fixed modules contained within the SERDESIF block per application. As shown in the example, PCIe applications include several functional blocks within the SERDESIF, whereas EPCS will require more FPGA IP blocks for complete system implementation.

Figure 2 • **Serial Protocol Using SERDESIF and FPGA Logic**



The following sections describe each of the serial protocols and their implementation in the SmartFusion2 and IGLOO2 devices using the SERDESIF block.

2.5 I/O Signal Interface of SERDESIF

The SERDESIF block interfaces with the FPGA fabric and SERDES differential I/O pad. The SERDESIF I/Os can be grouped into a number of interfaces from functional protocol. The SERDESIF I/O signals interface are listed below:

- Reset Interface
- Clock Interface
- AXI3 Master Interface
- AXI3 Slave Interface
- APB Slave Interface
- EPCS Interface

- I/O - PAD Interface (for more information, see Table 167, page 149)
- PLL Control and Status Interface
- SERDESIF Block-PCI Express Interrupt and Power Management Interface

Refer to protocol specific chapters for I/O details.

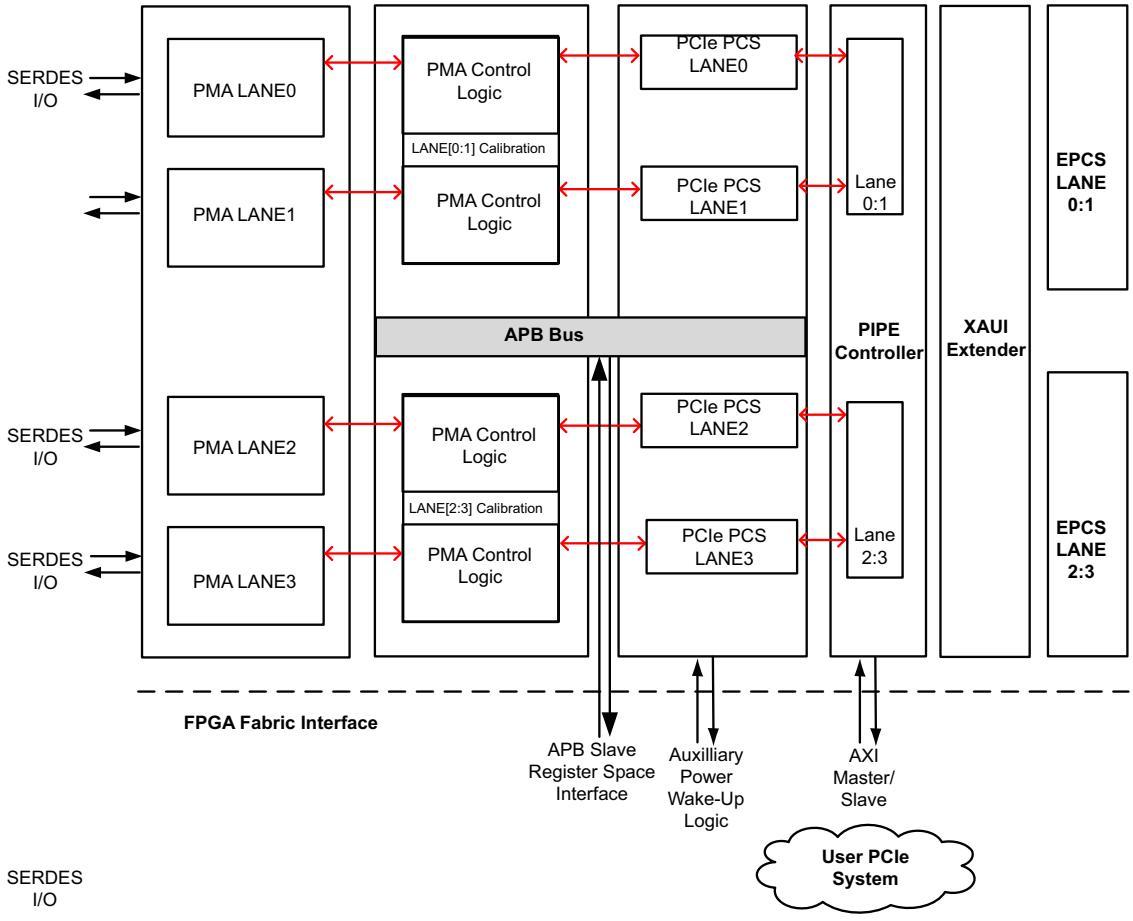
2.5.1 PCIe Protocol

The SmartFusion2 or IGLOO2 device family supports Gen1 and Gen2 PCIe endpoints. The PCIe endpoint supports PCIe base specification 1.1 (2.5 Gbps) and PCIe base specification 2.0 (5.0 Gbps) protocols with x1, x2, and x4 lane configurations. The application interface to the PCIe link is available through the FPGA fabric, and can be programmed to AXI3 master and slave interfaces. The SmartFusion2 and IGLOO2 devices have PCIe hard IP that is designed for performance and ease-of-use. The hard IP consists of the PMA, data link, and transaction layers.

Features:

- x1, x2, and x4 lane PCIe support
- Suitable for Native Endpoint
- PCIe base specification 2.0 and 1.1 compliant
- Single-function/Single virtual channel (VC)
- Three 64-bit base address registers or six 32-bit base address registers
- 2 KB Receive, 1 KB Transmit, 1 KB Retry Buffers
- 64-bit AXI3 master and slave interface to the FPGA fabric

Figure 3 • **SERDESIF Configuration for PCIe Protocol**



The following table shows the possible options for implementing the PCIe link on four physical SERDES lanes. Refer to the PCI Express, page 18 for details on PCIe protocol implementation in SmartFusion2 and IGLOO2 devices. Lane[0:1] and Lane[2:3] share on-chip hardware resources which create inter-dependency between the physical lanes. PCIe lanes are supported in Regular or Reversed modes. These modes provide the physical to logical lane implementation.

Table 3 • **Physical Interface Options for PCIe Endpoint in the SERDESIF Block**

	PHYSICAL SERDES LANES/LOGICAL LANES			
	SERDES_x_L01_REXT[4]		**SERDES_x_L23_REXT[4]**	
	LANE-0	**LANE-1**	**LANE-2**	**LANE-3**
PHY-MODE PCIe Protocol	**Protocol**	**Protocol**	**Protocol**	**Protocol**
M2S/M2GL010/025/050/150				
Single Protocol PHY -Mode (PCIe link Non-Reversed-Mode)	PCIe	*	*	*
	PCIe	PCIe	*	*
	PCIe	PCIe	PCIe	PCIe
Single Protocol PHY- Mode (PCIe link Reversed Mode)	*	*	*	
	*	*	PCIe	PCIe
	PCIe	PCIe	PCIe	PCIe
Multi Protocol PHY – Mode (PCIe link Non-Reversed-Mode)	PCIe	*	EPCS	EPCS
	PCIe	PCIe	EPCS	EPCS
Multi Protocol PHY – Mode (PCIe link Reversed-Mode)	*			
	PCIe	PCIe	EPCS	EPCS
M2S/M2GL060/090				
Dual PCIe Protocol PHY – Mode (Both PCIe link Non-Reversed-Mode)	PCIe_0	*	PCIe_1	*
	PCIe_0	PCIe_0	PCIe_1	PCIe_1
Dual PCIe Protocol PHY – Mode (Both PCIe link Reversed-Mode)	*	PCIe_0	*	PCIe_1
	PCIe_0	PCIe_0	PCIe_1	PCIe_1
Dual PCIe Protocol PHY – Mode (PCIe_0 in Reversed-Mode) (PCIe_0 in Non-Reversed-Mode)	*	PCIe_0	PCIe_1	*
	PCIe_0	PCIe_0	PCIe_1	PCIe_1
Dual PCIe Protocol PHY – Mode (PCIe_0 in Non-Reversed-Mode) (PCIe_1 in Reversed-Mode)	PCIe_0	*	*	PCIe_1
	PCIe_0	PCIe_0	PCIe_1	PCIe_1
Multi and Dual PCIe Protocol PHY – Mode (Non-Reversed-Mode)	PCIe_0	PCIe_1	EPCS	EPCS
	PCIe_0	*	EPCS	EPCS

Table 3 • **Physical Interface Options for PCIe Endpoint in the SERDESIF Block *(continued)***

	PHYSICAL SERDES LANES/LOGICAL LANES			
	SERDES_x_L01_REXT[4]		SERDES_x_L23_REXT[4]	
	LANE-0	LANE-1	LANE-2	LANE-3
PHY-MODE PCIe Protocol	**Protocol**	**Protocol**	**Protocol**	**Protocol**

Notes:

1. *PCIe 2.0 protocol is supported on SERDESIF with a maximum lane width of x4 (Single controller mode). PCIe link can be operated at GEN1 (2.5GHz) or at GEN2 (5.0GHz) speed. STD speed grade for Gen1, -1 for Gen2.*
2. *M2S/M2GL060/090 application interfaces have dual controller capability supporting up to two x1 or x2 endpoints within a SERDESIF.*
3. *Lane 2 and lane 3 EPCS interface is available in multi protocol PHY-MODE of operation.*
4. *SERDES_x_REXT: External calibration resistors are available on lane 0 and lane 2 physical lanes. Lane 1 and lane 3 need to get calibration values from adjacent lane 0 and lane 2 respectively.*
5. *Lane 2 and Lane 3 EPCS interfaces are available where noted in multi-protocol PHY modes.*
6. **: Designates Physical lane not used for any protocol. It is held in reset by device programming.*
7. *EPCS notations can be SGMII,JESD204, or any user defined serial protocol.*
8. *Non-M2S060/090 devices cannot support x1 PCIe Reverse modes.*

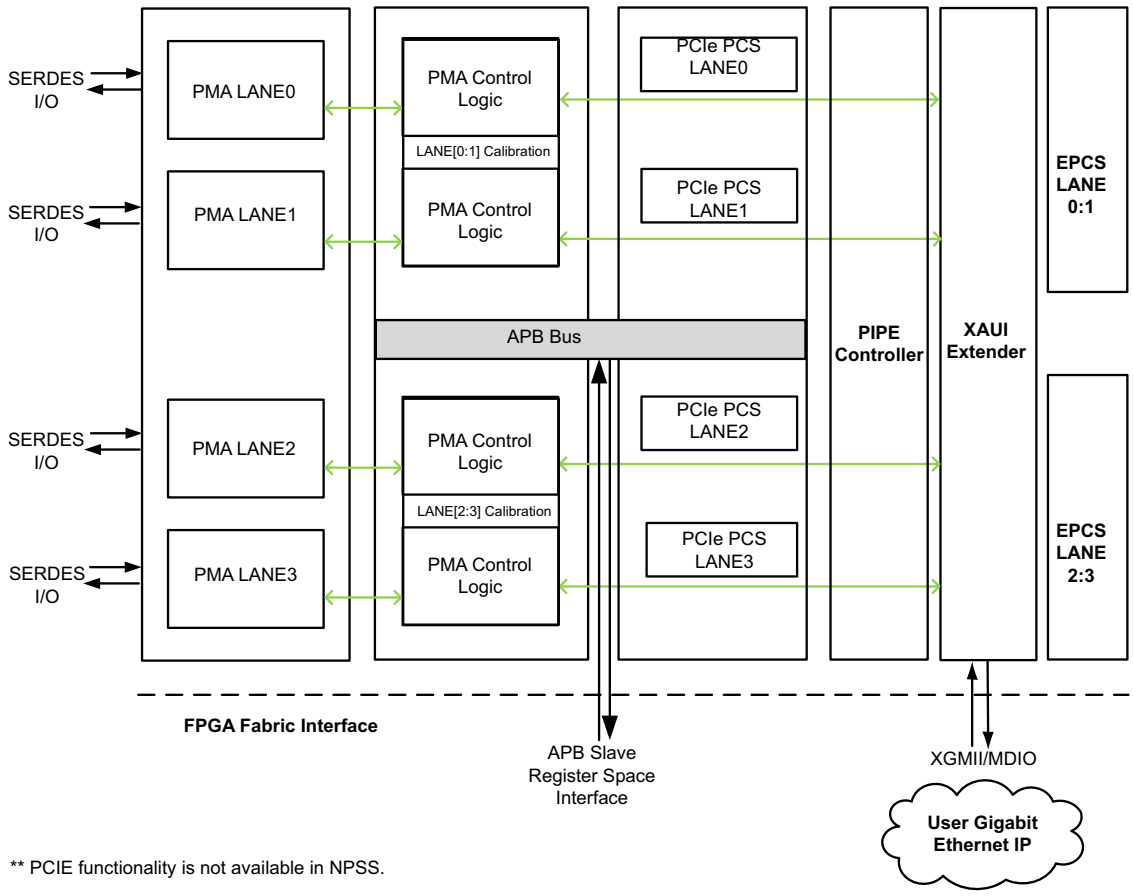
2.5.2 XAUI Protocol

The XAUI implementation is an interface extending the XGMII, 10 gigabit media independent interface. The XGMII is used to attach the Ethernet MAC to the PHY. The XAUI may be used in place of, or to extend, the XGMII in chip-to-chip applications typical of most FPGA IP Ethernet MAC-to-PHY interconnects.

Features:

- Full compliance with IEEE 802.3
- IEEE 802.3ae- clause 45 MDIO interface
- IEEE 802.3ae- clause 48 state machines
- Pseudo random idle insertion (PRBS Polynomial $X^7 + X^3 + 1$)
- Reference clock frequency of 156.25 MHz
- Double-width single data rate (SDR - 64 bit XGMII interface)
- Comma alignment function
- PHY-XS and DTE-XS loopback
- IEEE 802.3ae- annex 48A jitter test pattern support
- IEEE 802.3 clause 36 8B/10B encoding compliance
- Tolerance of lane skew up to 16 ns (50 UI)
- IEEE 802.3 PICs compliance matrix

Figure 4 • **SERDESIF Configuration for XAUI Protocol**



Note: Contact Microsemi sales for 10G MAC FPGA IP information.

The following table shows the configuration bandwidth for using XAUI in four physical SERDES lanes. Refer to the XAUI, page 88 for more information on XAUI protocol implementation in the SmartFusion2 and IGLOO2 devices.

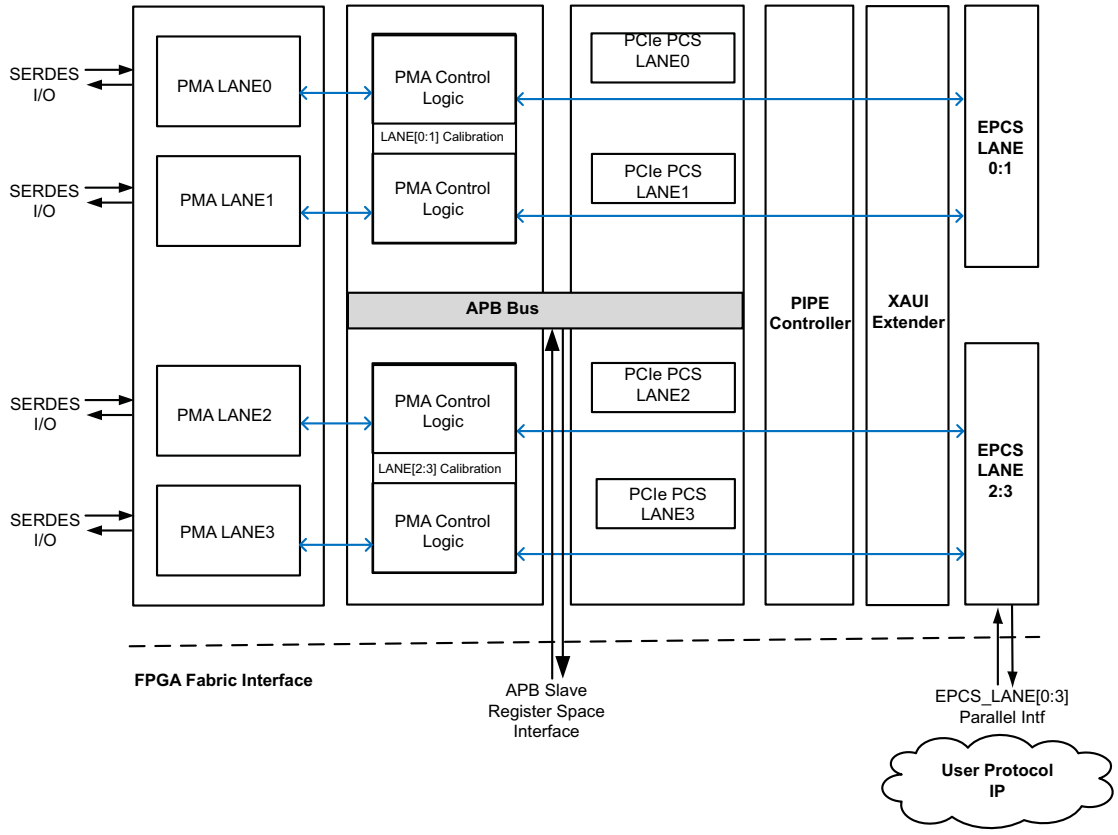
Table 4 • **Bandwidth for Implementing XAUI in SERDESIF Block**

	Lane0		Lane1		Lane2		Lane3	
XAUI Protocol	**Protocol**	**Speed (bps)**	**Protocol**	**Speed (bps)**	**Protocol**	**Speed (bps)**	**Protocol**	**Speed (bps)**
Single Protocol PHY mode	XAUI	3.125 G	XAUI	3.125 G	XAUI	3.125 G	XAUI	3.125 G

2.5.3 EPCS Protocol

By using the EPCS interface, any other serial protocol can be implemented in the SmartFusion2 or IGLOO2 device family. The SERDESIF block can be configured to bypass the embedded PCS logic in the SERDES block and expose the EPCS interface to the fabric. The user-defined IP block in the FPGA fabric can be connected to this EPCS interface.

Figure 5 • **SERDESIF Configuration for EPCS Protocol**



For more information on EPCS implementation in SmartFusion2 and IGLOO2 device families, see EPCS Interface, page 112.

The four SERDES physical lanes can be configured to run different serial protocols, resulting in different modes of operation. The following table summarizes the various modes of operation of the SERDESIF block.

Table 5 • **Serial Protocol Implementation SmartFusion2 and IGLOO2 Devices**

		PHY Physical Lanes			
		Lane0	**Lane1**	**Lane2**	**Lane3**
Serial Protocol	**Modes**	**PHY Logical Lanes Vs Logical Lanes**			
PCIe Protocol only mode	PCIe (x4)	PCIe Lane0	PCIe Lane1	PCIe Lane2	PCIe Lane3
	PCIe (x2)	PCIe Lane0	PCIe Lane1	–	–
	PCIe (x1)	PCIe Lane0	–	–	–
	PCIe Reversed mode (x4)	PCIe Lane3	PCIe Lane2	PCIe Lane1	PCIe Lane0
	PCIe Reversed mode (x2)	–	–	PCIe Lane1	PCIe Lane0
	PCIe Reversed mode (x1)	–	–	–	PCIe Lane0
	PCIe Reversed mode (x2)	PCIe Lane1	PCIe Lane0	–	–
	PCIe Reversed mode (x1)	–	PCIe Lane0	–	–
XAUI only	XAUI (x4 lane)	XAUI-0	XAUI-1	XAUI-2	XAUI-3
EPCS only	All lanes are used for user-defined protocol	EPCS	EPCS	EPCS	EPCS
Multi-protocol (PCIe and EPCS)	PCIe (x2)	PCIe Lane0	PCIe Lane1	EPCS	EPCS
	PCIe (x1)	PCIe Lane0	–	EPCS	EPCS
	PCIe Reversed mode (x2)	PCIe Lane1	PCIe Lane0	EPCS	EPCS
	PCIe Reversed mode (x1)	–	PCIe Lane0	EPCS	EPCS

Note: Lane Tx-clk is used for lane0 for PCIe protocol purposes.

Note: In Multi-protocol mode, EPCS is available only on lane2 and lane3.

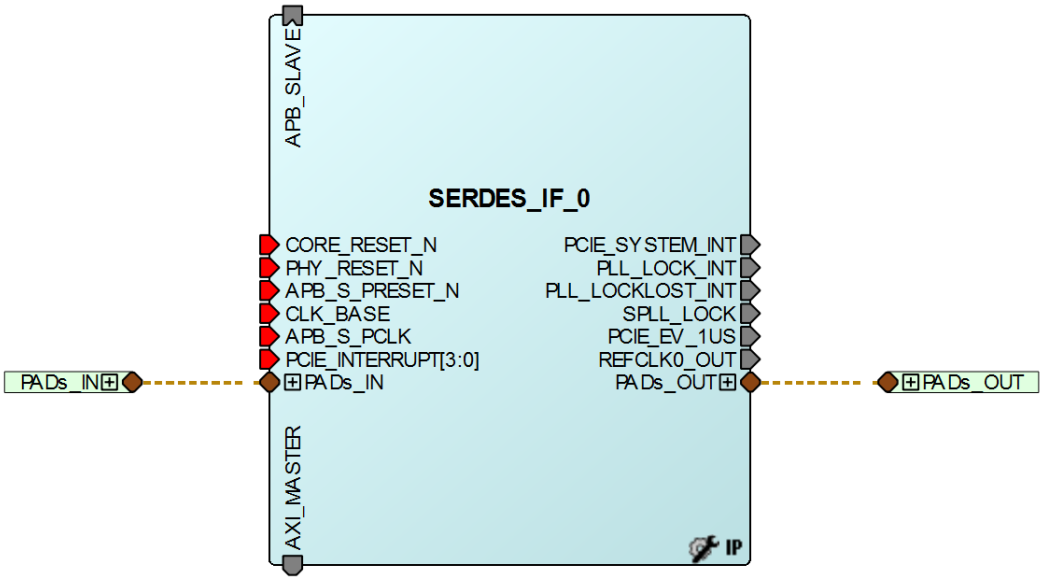
Note: For complete device listing with PCIe implementations, see Figure 4, page 11.

Note: EPCS can be used to build protocol specific interfaces such as JESD204B, 1000Base-X, and SGMII.

2.6 Getting Started with Libero SoC

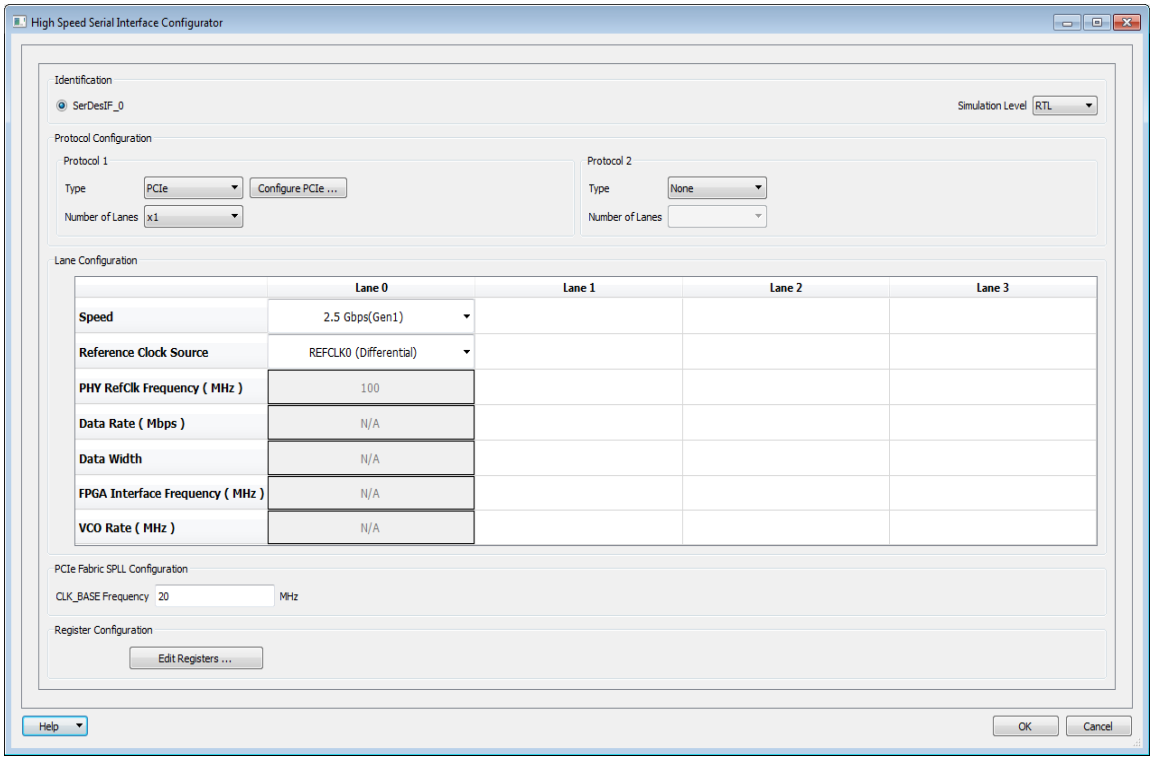
2.6.1 Using SERDESIF Macro in Libero SoC

Figure 6 • **SERDESIF Module**



The main SERDESIF GUI wizard interrogates customizable design parameters and initiates the available options. This enables the user to step through the building blocks to assemble the correct SERDESIF module.

Figure 7 • **Libero SoC SERDESIF GUI**



Note: The GUIs are examples of Libero SoC/SysBuilder GUIs and maybe enhanced over time to ease user experience.

The High Speed Serial Interface Configurator available in Libero SoC allows generation of the SERDESIF block with various protocol modes. It allows module creation of Single and Multi protocol modes. This information is detailed in the protocol sections of this chapter.

2.6.1.1 Clocking and Resets

This section describes the clocking and reset scheme for the SERDESIF block. Generally the SERDESIF block has varying reset and clocking options that are exposed for the different protocol choices. More details of these options are found in the specific protocol chapters.

2.6.1.1.1 Clocking System for SERDESIF

The clocking system in the SERDESIF block includes the following:

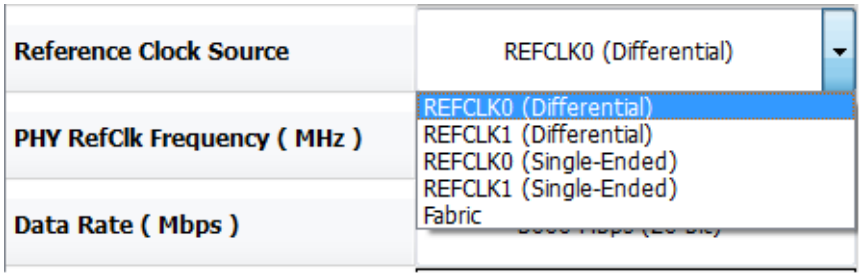
- SERDES reference clocks
  - REFCLK0 and REFCLK1 - Dedicated input pins
  - Fabric Clock available only for EPCS protocols
- Fabric/serial PLL (SPLL) clocking
  - PCIe system block clocking
  - XAUI block clocking

2.6.1.1.2 SERDES Reference Clocks

The PMA in the SERDES block needs a reference clock on each of its lanes for Tx and Rx clock generation through PLLs. Refer to the Serializer/De-serializer, page 128 for more information on Tx and Rx clock generation through PLLs. There are two dedicated reference clock (REFCLK0 and REFCLK1) inputs on the SERDESIF. The two reference clocks, REFCLK0 and REFCLK1, are connected to I/O pads REFCLK0_P/N and REFCLK1_P/N. The reference clock can also come from the fabric, but that clock source might not be optimal for certain implementations. This is discussed later in specific chapters.

The following figure shows the reference clock selection in the High Speed Serial Interface Configurator available in the Libero SoC. It sets the MUX selection, depending on the selected reference clocks.

Figure 8 • **SERDES Reference Clock Using the High Speed Serial Interface Generator**



2.6.1.1.3 Serial PLL (SPLL)

A dedicated PLL located within the SERDESIF is provided within the SERDESIF block and used transparently by the design for handling clock domain transfers between the blocks of the SERDESIF and fabric. The SPLL manages the skew between the FABRIC and SERDESIF module and is used for PCIe and XAUI protocol implementations.

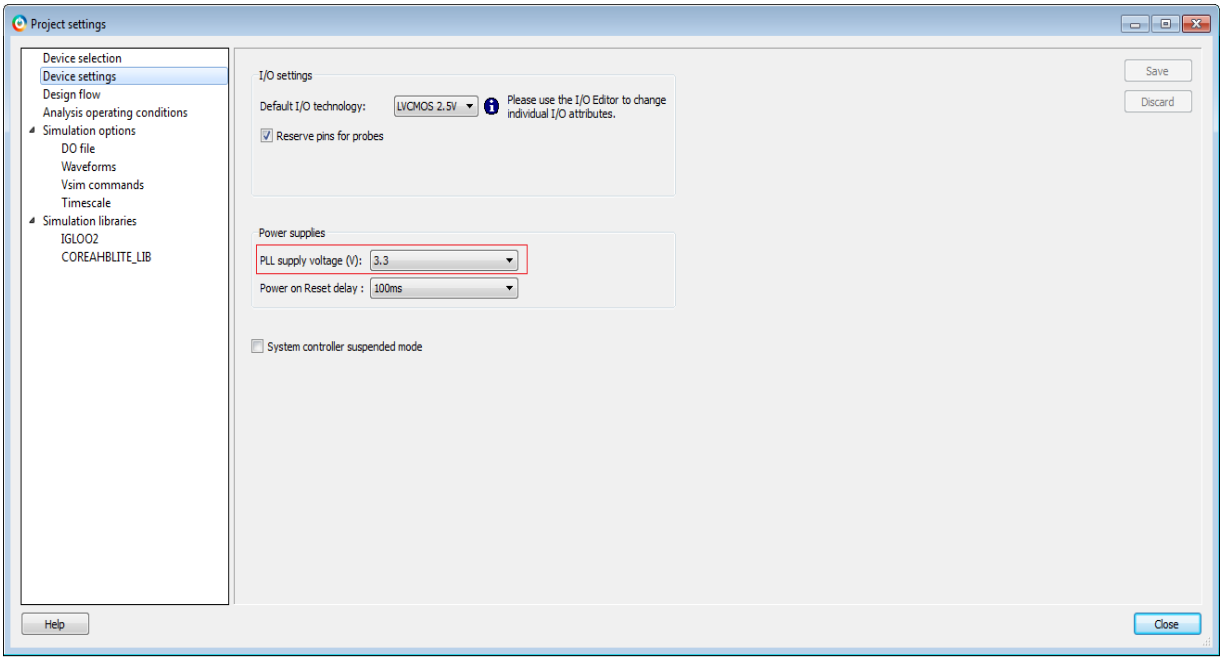
The SPLL is powered by the SERDES_[01]_PLL_VDDA supplies. This supply is selected by the user to be typically 2.5 V or 3.3 V. This selection does not impact the SPLL frequency range. Refer to the *DS0128: IGLOO2 and SmartFusion2 Datasheet* for more information on the PLL power supply requirement. The user must select this supply in the Libero software to correctly provision this supply in conjunction with the MDDR_PLL_VDDA, FDDR_PLL_VDDA, PLL0_PLL1_MDDR_VDDA, and CCC_XX[01]_PLL_VDDA.

Refer to the *UG0445: IGLOO2 FPGA and SmartFusion2 SoC FPGA Fabric User Guide* for details on fabric based PLLS.

This PLL power setting is not related to analog power for the SERDES PMA which is powered by the SERDES_[01]_L[0123]_VDDAPLL supplies.

The following figure shows the SPLL power supply UI.

Figure 9 • **SPLL Power Supply Selection in the Libero SoC New Project Wizard**



2.6.1.2 SERDESIF Reset

Refer to the PCI Express, page 18, the XAUI, page 88, and the EPCS Interface, page 112 for more information about using these reset signals.

A FPGA IP module, CORERESETP, controls the SERDESIF reset operation at initialization. The *CORERESETP* module ensures that the SERDES (PHY and CORE) reset signals remain asserted (low) until APB-based configuration has been completed. Until complete, no accesses should be initiated from the fabric to the SERDESIF through the AXI3 interface. This provides a predictable behavior for the SERDESIF at startup.
The Libero software flow provides firmware for the correct SERDESIF initialization sequence as below.

- Write PMA and System Registers
- De-assert PHY_RESET_N
- Wait 130us
- De-assert CORE_RESET_N

The following is only done for PCIe (done for each PCIe core for the 090 SERDES block)

- Select 0th lane (must take care of PCIe reverse; for x2 for instance, lane 1 is lane 0)
- Wait for PMA ready on that lane
- Write PCIe registers
- Issue INIT_DONE

Refer to specific protocol chapters for detailed pin descriptions and information on reset behaviors for each protocol.

2.6.1.3 Serial Protocols Setting Using the SERDESIF System Registers

The SERDESIF is configured to support various modes of operation. This configuration of the protocols is through high level SERDESIF system registers. These registers are configured using the APB interface. To facilitate the initial configuration, a GUI in Libero SoC is provided. The following table describes the settings for the three SERDESIF system registers to force the SERDESIF block into a specific mode of operation. Refer to SERDESIF Register Access Map, page 150 for more details.

Table 6 • **Reset Interface**

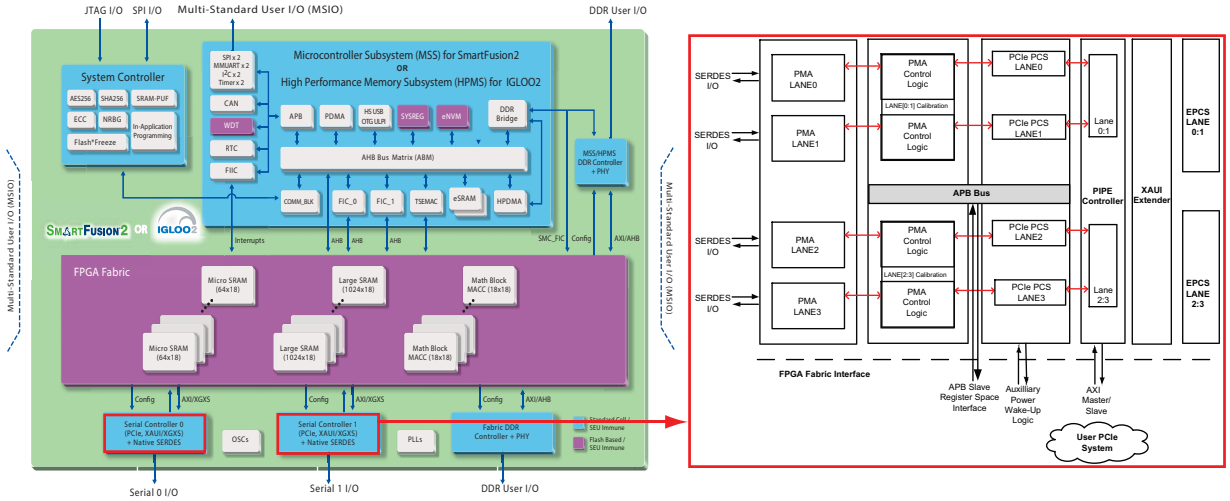
Port	Type	Connected To	Description
CORE_RESET_N	Input	Fabric	PCIe or XAUI mode: Active reset for PCIe and XAUI fundamental core
PHY_RESET_N	Input	Fabric	SERDES-PHY-Active low reset. If not, lanes are used for any serial protocol. Tie it to High.
APB_S_PRESET_N	Input	Fabric	Asynchronous set signal for APB slave interface.
EPCS_0_RESET_N EPCS_1_RESET_N EPCS_2_RESET_N EPCS_3_RESET_N	Input	Fabric	External EPCS interface mode: External PCS reset control lane0, lane1, lane2, and lane3.
EPCS_0_RX_RESET_N EPCS_1_RX_RESET_N EPCS_2_RX_RESET_N EPCS_3_RX_RESET_N	Output	Fabric	External EPCS interface mode (lane0, lane1, lane2, and lane3): Clean reset deasserted on rxclk.
EPCS_0_TX_RESET_N EPCS_1_TX_RESET_N EPCS_2_TX_RESET_N EPCS_3_TX_RESET_N	Output	Fabric	External EPCS interface mode (lane0, lane1, lane2, and lane3): Clean reset deasserted on txclk.
PLL_SERDESIF_RESET	Output	SPLL	SPLL reset output
PLL_SERDESIF_PD	Output	SPLL	SPLL power-down enable

3 PCI Express

3.1 Introduction

This chapter describes using PCIe in the SmartFusion2 and IGLOO2 FPGA devices. PCIe is a high-speed serial computer expansion bus standard designed to replace the older PCI, PCI-X, and AGP bus standards. The SmartFusion2 or IGLOO2 SERDESIF allows fully integrated PCIe endpoint (EP) implementation. This chapter provides detailed information on implementing, verifying, and debugging the PCIe EP design in the SmartFusion2 and IGLOO2 devices.

Figure 10 • **SmartFusion2 and IGLOO2 SERDESIF Block Diagram**



3.1.1 Features

The SmartFusion2 or IGLOO2 family supports up to four hard SERDESIF blocks in one device, and each block supports up to 4 serialization/deserialization (SERDES) lanes, thus allowing to have up to 16 SERDES lanes. Figure 10, page 18 shows the SmartFusion2 M2S050 or IGLOO2 M2GL050 device block diagram with PCIe implementation. Each SERDESIF block contains a integrated PCIe system block, also known as a PCIe system, which implements the PCIe transaction layer and data link layer. The SERDESIF block also has a SERDES block that implements the physical layer. The PCIe system block along with the SERDES block provide the integrated PCIe EP solution in SmartFusion2 and IGLOO2.

Following are the main features of PCIe implemented in SmartFusion2 and IGLOO2:

- x1, x2, x4 lane support
- Implements native endpoint
- Compliant with PCIe Base Specification Revision 2.0 and 1.1
- Single-function/Single virtual channel (VC)
- Receives, transmits, and retries buffer
- AXI3 master and slave interface to the SmartFusion2 or IGLOO2 FPGA fabric
- Supports design time selection of the PCIe lane reversal for flexibility of lane assignments for board layout.

Note: A PCIe Endpoint refers to the location of the connection in the PCIe topology. A PCIe Endpoint can connect to Switches Downstream Port or a Root Complex Downstream Port. As an Endpoint the PCIe can initiate and respond to transactions in the system.

3.1.2 Device Support

The SmartFusion2 and IGLOO2 families have a number of devices available. The following table lists the total number of SERDESIF Blocks available in each SmartFusion2 and IGLOO2 device that can be configured to support PCIe.

Table 7 • **SERDESIF PCIE Endpoint Blocks Available in SmartFusion2 and IGLOO2**

	M2S/M2GL 005	M2S/M2GL 010	M2S/M2GL 025	M2S/M2GL 050	M2S/M2GL 060	M2S/M2GL 090	M2S/M2GL 150
PCIE EP available	0	1	1	Up to 2	Up to 2	Up to 2	Up to 4

Note: The specified number of SERDESIF blocks varies depending on the device package.

Note: M2S/M2GL060/090 application interfaces have dual PCIe controller capability supporting up to two x1 or x2 endpoints within a SERDESIF. It can also support one x4 endpoint.

3.1.3 Overview of PCIe in SmartFusion2 and IGLOO2

The PCIe protocol is software backward-compatible with the earlier PCI and PCI-X protocols, but is significantly different from its predecessors. The performance is scalable based on the number of lanes and the generation that is implemented. The PCIe protocol specifies 2.5 giga-transfers per second for Gen1, and 5 giga-transfers per second for Gen2, because the PCIe protocol uses 8B/10B encoding, there is a 20% overhead. The following table lists the aggregate bandwidth of a PCIe link.

Table 8 • **Theoretical PCIe Throughput**

	Link Width			
	x1	x2	x4	Unit
PCI Express Gen1 Gbps (1.x compliant)	2	4	8	Gbps
PCI Express Gen2 Gbps (2.x compliant)	4	8	16	Gbps

3.2 Description

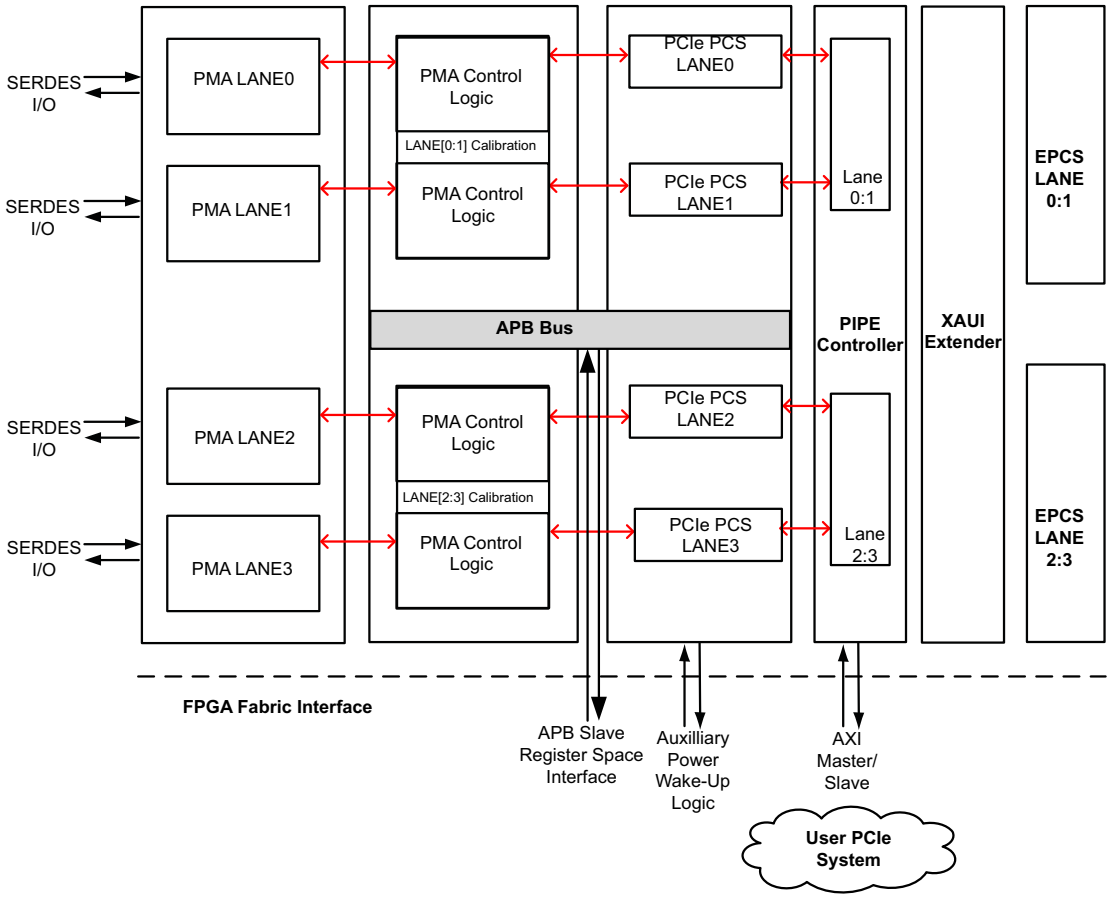
SmartFusion2/IGLOO2 support implementing PCIe Endpoints (Eps). The EP in PCIe refers to a type of function that can be the requester or the completer of a PCIe transaction. The PCIe system and SERDES high-speed serial interface (SERDESIF Block) blocks implement the PCIe EP specification for the transaction, data link, and physical layers.

Figure 11, page 20 shows a simplified view of a PCIe EP implementation in a SmartFusion2/IGLOO2 device. The PCIe system interfaces to the FPGA fabric on one side and the SERDES block on the other side. The SERDES block interfaces to the dedicated I/O in device is called a SERDES I/O. Refer to the “SERDESIF- I/O Signal Interface” listed in Table 167, page 149 for more information. The PCIe system interface to the FPGA fabric consists of an application interface and a configuration interface. In addition, it has several signals for clocking, reset, and power management.

- **Application interface**: The application interface is used to transfer transaction layer packets (TLP). It can be AXI3 master only, or AXI3 slave only, or AXI3 master plus slave interface.
  - AXI3 master interface: The master interface can be a 64-bit AXI3 master. A typical application interface uses a master interface which is used to respond to data read requests and a slave interface which is used to initiate requests. It is also possible to use a master and/or the slave interface by itself for specific applications.
  - AXI3 slave interface: The slave interface can be a 64-bit AXI3 slave interface. The SmartFusion2 or IGLOO2 fabric initiates PCIe transactions using the slave interface (that is, Memory Write TLP and Memory Read TLP). The data on a read request comes back to this same interface.

- **Configuration interface**: The configuration interface uses the APB slave interface.
  - APB interface: The APB interface has access to various registers, including PCIe configuration registers, AXI3 bridge register and SERDESIF register. The APB provides access to the memory map of the SERDESIF which includes a section for the PCIe controller.
- **Other signals**: The PCIe system also has several clocking signals, reset signals, phase-locked loop (PLL) signals, interrupts, and power management signals to the FPGA fabric.

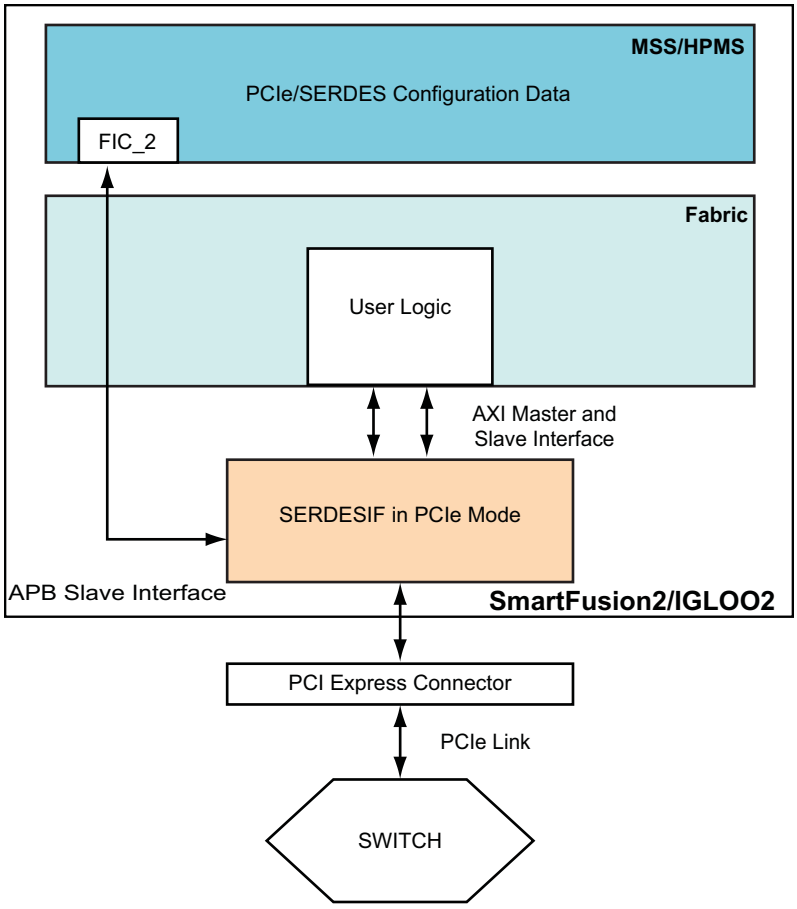
Figure 11 • **SERDESIF Configuration for PCIe Single Protocol Mode**



3.2.1 PCIe EP Application Example

The following figure shows a relatively simple application with a switch port connected to a PCIe EP implemented in a SmartFusion2 or IGLOO2 device.

Figure 12 • **SmartFusion2 and IGLOO2 PCIe EP Implementation**



3.3 Getting Started

3.3.1 Using SERDESIF Block in PCIe Mode

This section provides an overview of configuring the SERDESIF block in PCIe mode, simulating the SERDESIF block in PCIe mode, and implementing a PCIe EP in a SmartFusion2 or IGLOO2 device.

The High Speed Serial Interface Configurator in Libero SoC provides the configuration options for the PCIe EP implementation. It includes options for selecting the protocol for various SERDES lanes, serial rate settings, fabric interface, PCIe Identification registers, PCIe base address register (BAR). These setting are implemented during programming using dedicated flash bits for fast configuration or via the APB interface. These registers can be reviewed by using the advanced peripheral bus (APB) interface, which can access all the registers in the SERDESIF block.

The following sub-sections show how to instantiate PCIe in a design:

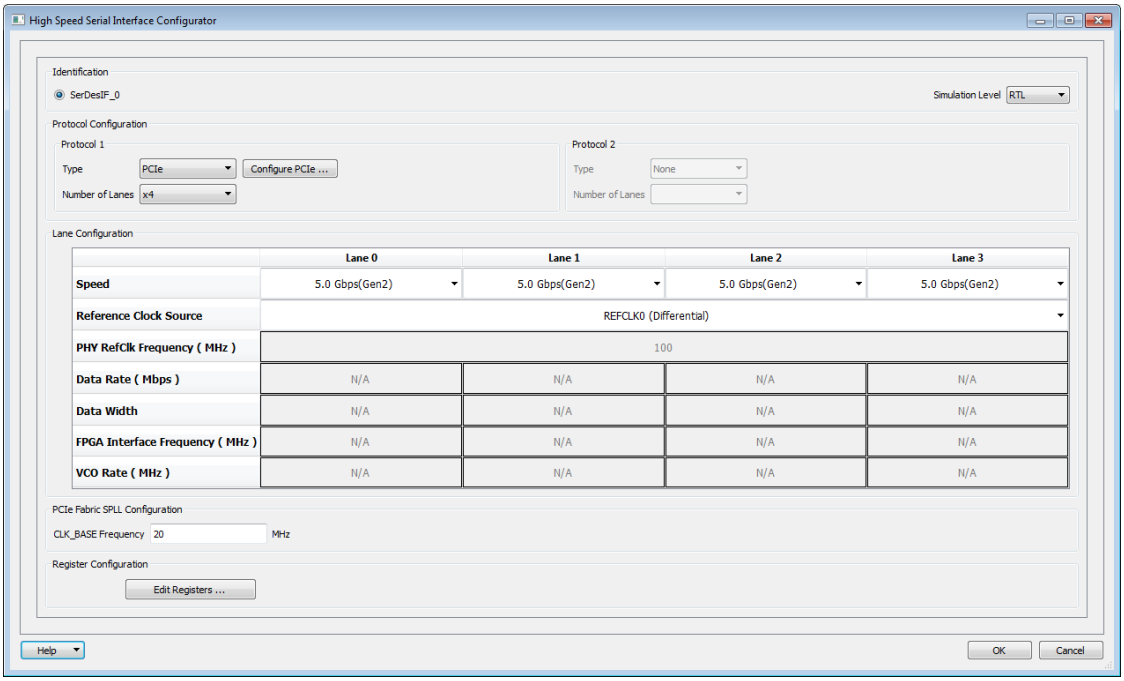
- Configuring High Speed Serial Interface Configurator for PCIe
- Simulating SERDESIF in PCIe Mode
- Adding SmartFusion2 or IGLOO2 PCIe Block to User Design

3.3.1.1 Configuring High Speed Serial Interface Configurator for PCIe

This sub-section describes configuring and generating the SERDESIF block for PCIe EP mode using the Libero SoC software.

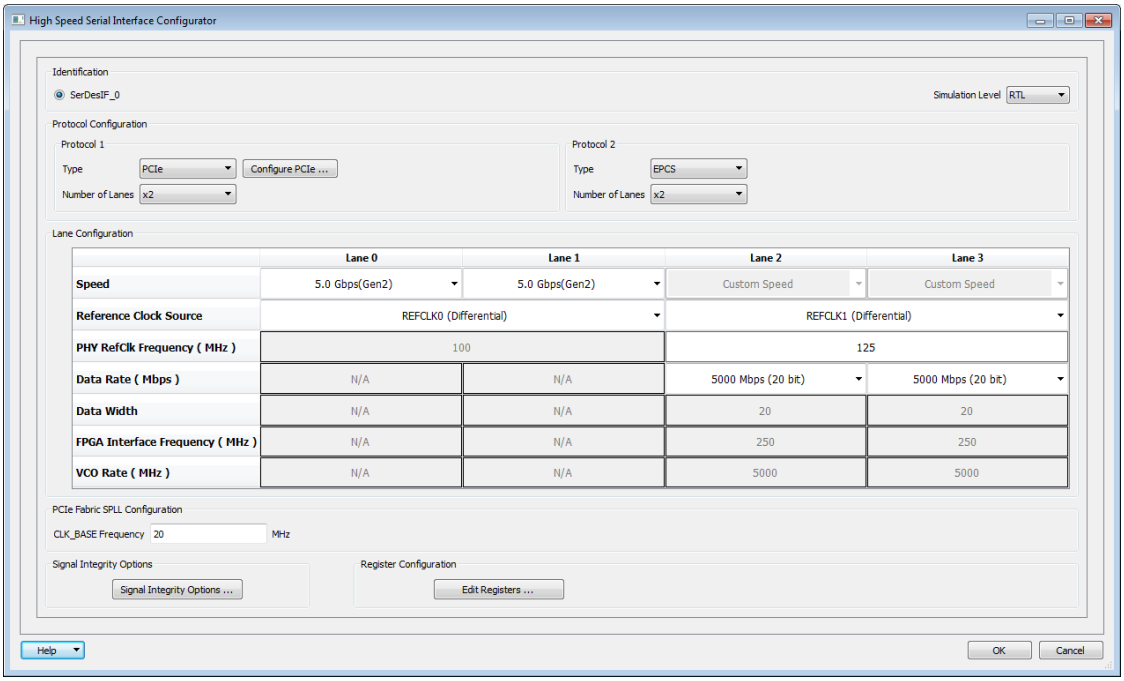
The High Speed Serial Interface Configurator (SERDESIF Configurator) in Libero allows configuration of the SERDESIF block in PCIe mode. Refer to the following figure for setting the SERDESIF Configurator in PCIe only protocol mode.

Figure 13 • **PCIe Single Protocol Mode Setting in SERDESIF Configurator**



Refer to the following figure for setting the SERDESIF Configurator in PCIe and other protocol mode.

Figure 14 • **PCIe Multi-Protocol Mode Setting in SERDESIF Configurator**



Following is a brief description of the various protocol configuration options. Refer to the SERDESIF Block, page 4 for details.

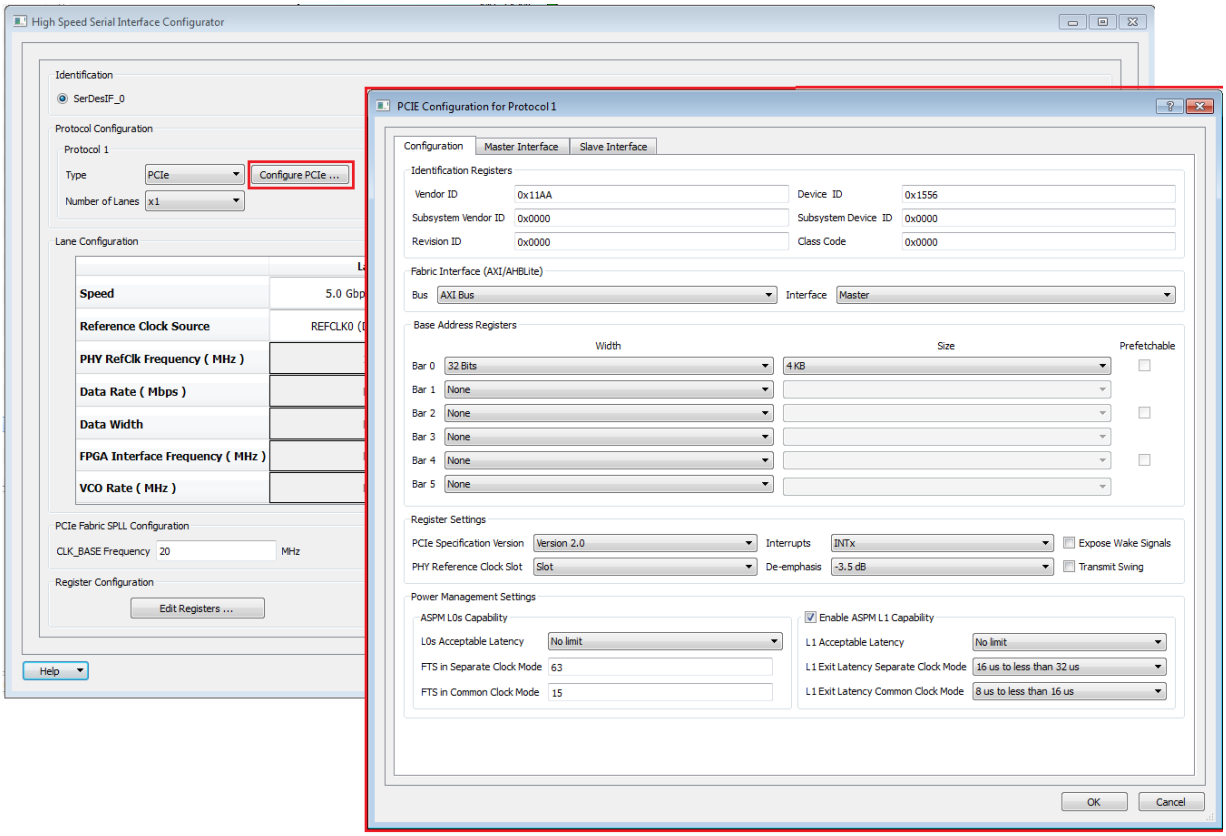
3.3.1.1.1 Protocol Selection

These settings are used for protocol selection.

- Protocol Type 1 – Protocol settings. Select **PCIe** from the drop-down menu.
- Number of Lanes – Select **number of lanes** used.
- Speed – Select **Gen1** or **Gen2** speed for the PCIe lanes.
- Protocol 1 PHY Reference Clock – Select the **inputs for the PHY reference clock** selection. Refer to the SERDES Reference Clocks Selection, page 38 for details on PHY reference clock selection.

In addition to the protocol settings, various other options for PCIe implementation can be set, as shown in the following figure.

Figure 15 • **High-Speed Serial Configurator with PCIe Implementation Options**



Following is a brief description of the various configuration options:

3.3.1.1.2 PCIe SPLL Configuration

These settings are used for SPLL that synchronizes data between the AXI3 interface to the SERDESIF block.

- CLK_BASE: This is an AXI3 clock setting.
- The SPLL is a PLL embedded in the SERDESIF to manage the clock phase used for transfers across the SERDESIF to FPGA fabric interface.

3.3.1.1.3 PCIe Fabric Interface (AXI3)

CLK_BASE frequency must be set to the same frequency of the (AXI) interface as the operating frequency.These settings select the PCIe system interfaces to the fabric. It can be AXI3 bus as master only, slave only, or both master and slave.

3.3.1.1.4 PCIe Base Address Registers

These settings are used to set six 32-bit or three 64-bit base address registers.

- Width: Width can be 32-bit or 64-bit. If an even register is selected to be 64-bits wide, the subsequent (odd) register serves as the upper half of 64 bits. Otherwise, the width of odd registers is restricted to 32 bits.
- Size: Ranges from 4 Kbytes to 2 Gbytes
- Prefetchable: Prefetchable option for memory BAR.

3.3.1.1.5 PCIe Identification Registers

These settings are used to set the six identification registers for PCIe.

- Vendor ID: 0x11AA is the Vendor ID assigned to Microsemi by PCI-SIG.
- Subsystem vendor ID: Card manufacturer's ID
- Device ID: Manufacturer's assigned part number by the vendor
- Revision ID: Revision number, if available
- Subsystem Device ID: Assigned by the subsystem vendor
- Class Code: PCIe device's generic function

3.3.1.1.6 PCIe Other Options

These settings are used to set other PCIe options:

- L2_P2 Entry/Exit: Selecting this option adds PCIE_WAKE_N, PCIE_WAKE_REQ, and PCIE_PERST_N ports to control the L2/P2 state.
- PHY Reference Clock Slot: Select this option if the PHY reference clock is coming from a PCIe slot or it is generated separately.

Note: Slot refers to a clock source that is shared in the PCIe system between both ends of the link. The other setting, Independent, is used in a system which uses independent clock sources on either side of the link. This setting changes the PCIe configuration space register to advertise to the system root which clocking topology is used. It makes no other functional changes to the endpoint.

- De-emphasis: Set the de-emphasis (3.5 dB and 6.0 dB) for PCIe GEN 2 speed.
- Transmit Swing: Set transmit swing for PCIe GEN 2 speed.
- PCIe Specification Version: Specifies the version

PCIE_WAKE_N, PCIE_WAKE_REQ, and PCIE_PERST_N ports are added optionally with L2/P2 selection. PCIE_WAKE_N is an output and PCIE_WAKE_REQ, and PCIE_PERST_N ports are inputs to the PCIE core.

Entry of L2P2 can only be requested by the host and when it is requested, the SmartFusion2 or IGLOO2 PCIE core responds to the protocol request and its state machine will go to L2 state. Its response will be in conjunction with the CFGR_L2_P2_ENABLE bit of the CONFIG_PCIE_PM register. If the bit is configured as CFGR_L2_P2_ENABLE = 0 or 1, the device goes into L2, and it needs a fundamental reset (PCIE_PERST_N) to get out of L2.

If CFGR_L2_P2_ENABLE=1, the device shows much lower power when it enters L2. In either case the host needs to perform enumeration.

PCIE_WAKE_REQ is an input into the SmartFusion2 or IGLOO2 PCIE reset controller. When PCIE_WAKE_REQ is asserted, the PCIe reset controller drives out the PCIE_WAKE to RC as WAKE# side band signal. PCIE_PERST_N is a PCIe fundamental reset. Before issuing PCIE_PERST_N, the root port must check and clear pending transactions within the SERDESIF PCIE endpoint. For more information on using PCIE_PERST_N and related reset sequences required to reset the complete PCIE Endpoint core including AXI IF, refer to the *AC437: Implementing PCIe Reset Sequence in SmartFusion2 and IGLOO2 Devices Application Note*.

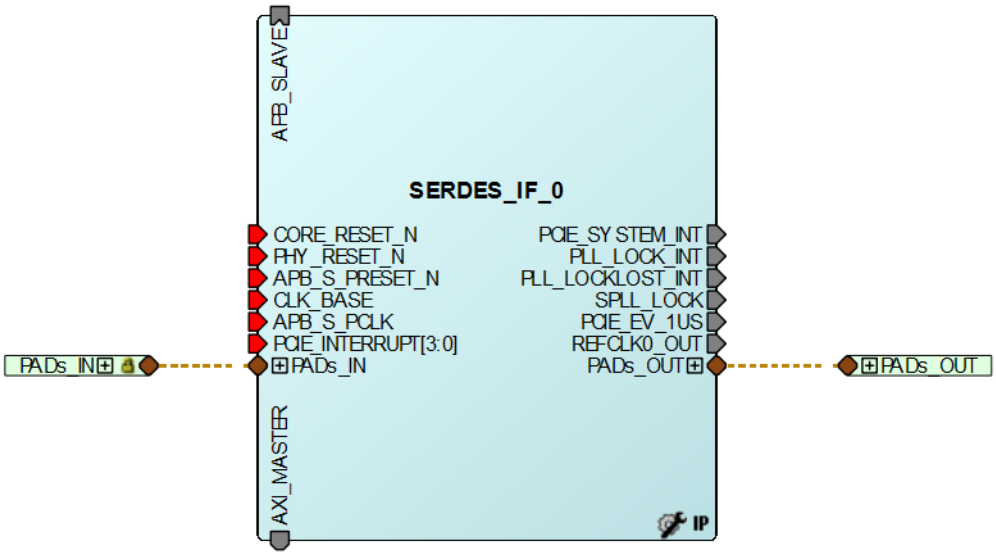
3.3.1.2 Simulating SERDESIF in PCIe Mode

Refer to the *SERDESIF BFM Simulation Guide* for more information on simulation detail.

3.3.1.3 Adding SmartFusion2 or IGLOO2 PCIe Block to User Design

To use the SmartFusion2 or IGLOO2 PCIe block in user design, the appropriate setting must be set in the SERDESIF Configurator and then generate the SERDESIF block. The following figure shows the SERDESIF block in Libero. Libero promotes the SERDES I/Os to the top level and exposes the AXI3 (based on user settings) and APB interface to the FPGA fabric. In addition, the SERDESIF block exposes the clocks, resets, PLL locks, and power management signals for PCIe implementation.

Figure 16 • **High Speed Serial Interface Block in Libero SoC**



Note: All SERDES PADs_IN[3:0] and PADs_OUT[3:0]will appear regardless of PCIE link widths

The user logic block implements an AXI3 slave interface to transfer data to the PCIe link and an AXI3 master interface to receive the data from the PCIe link. The user connects the HPMS SDIF bus to the APB3 interface of the SERDESIF. The HPMS will configure the SERDESIF and control specific resets of the SERDESIF. A FAB clock conditioning circuit (CCC) generates the clock for the AXI3 interface on the port CLK_BASE.

3.4 PCIe System Architecture

PCIe is a high speed, packet based, point-to-point, low pin count, serial interconnect bus. SmartFusion2 and IGLOO2 have a fully integrated PCIe End Point (EP) implementation. This section describes the architecture of the PCIe system that implements the main PCIe IP function.

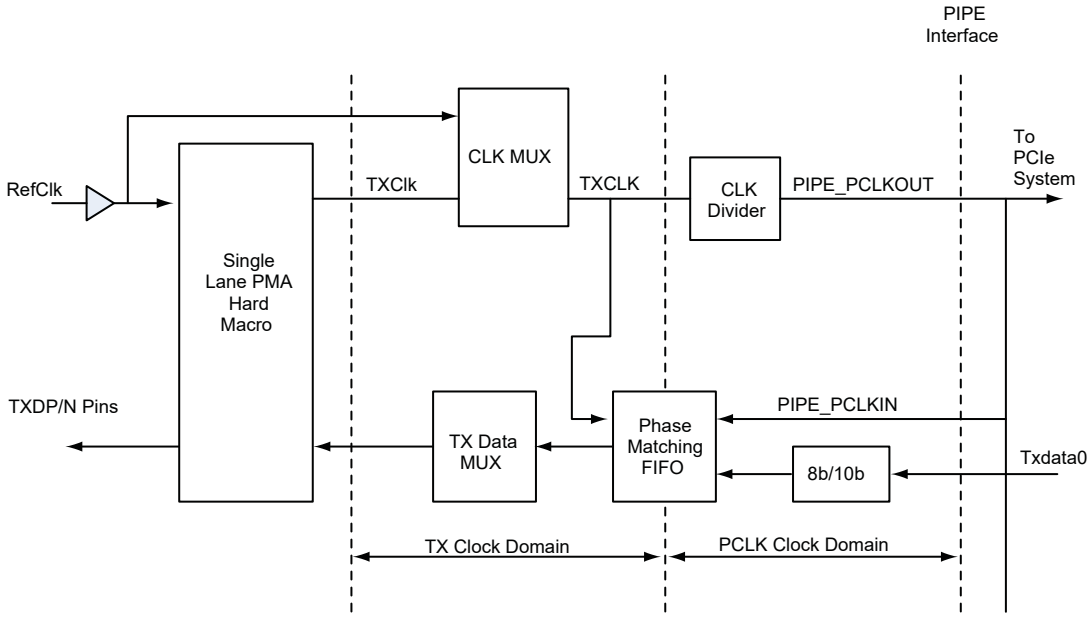
3.4.1 Physical Coding Sublayer Block

The PCS block implements 8b/10b encoder/decoder, RX detection, and an elastic buffer for the PCIe protocol. It has transmitter and receiver blocks.

3.4.1.1 Transmitter Block

The Transmitter block consists of an 8b/10b encoder and a phase matching first-in-first-out (FIFO), as shown in the following figure. The transmitter block passes the input data in the PCLK domain (PIPE clock domain) to the PMA hard macro in the TX clock domain.

Figure 17 • **Transmit Clock and Transmit Datapath**



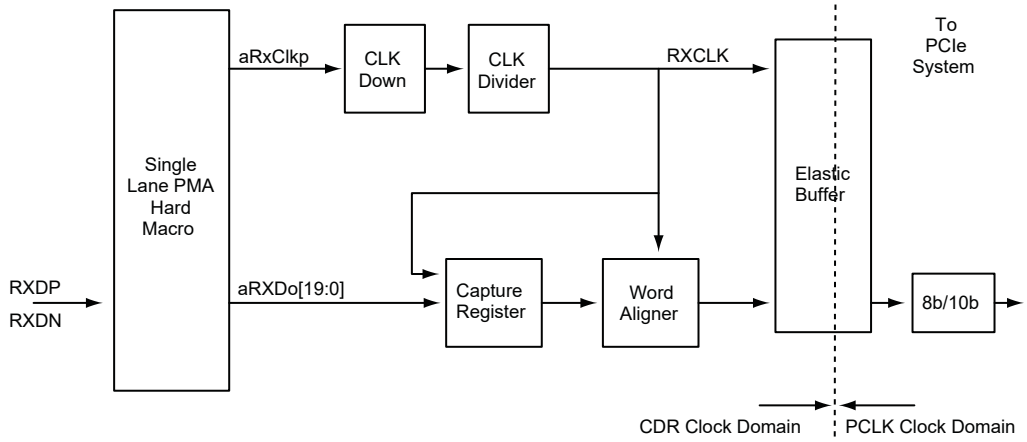
The reference clock (RefClk) is the per-lane PCIe reference clock, which is generally the PCIe 100 MHz reference clock. During multiple lane implementations, the clock is sent to each PMA single lane macro and skew between lanes is finely controlled. Effectively, each PMA macro generates a transmit clock TX clock, from which is generated the pipe clock (generated by one lane) used by the PCIe controller and also used by the PCS logic in all lanes.

- **CLK MUX Block**: is a glitchless clock multiplier for sourcing the RefClk or TXClk to the TX Clock Domain of the PCS sublayer. This MUX is used for operation at power-up and during speed changes.
- **CLK Divider Module**: is used to generate the PIPE clock (PCLK) for the PCIe controller. The PIPE_PCLKOUT signal is the output signal of the PCS and is generated on a per-lane basis.
- **Phase Matching FIFO**: is used to recapture the transmitted data generated on the PCLK clock domain back to the aTXClk domain, considering the two clocks are fully independent (asynchronous). The TX data MUX performs multiplexing between data coming from the PCIe PCS and the external PCS.
- **8b/10b Encoder**: is used to implement an 8-bit to 10-bit encoder that encodes 8-bit data or control characters in to 10-bit symbols.

3.4.1.2 Receiver Block

The Receiver block consists of receive capture logic, word alignment logic, elastic buffer, and an 8b/10b decoder, as shown in the following figure.

Figure 18 • **Receive Clock and Receive Datapath**



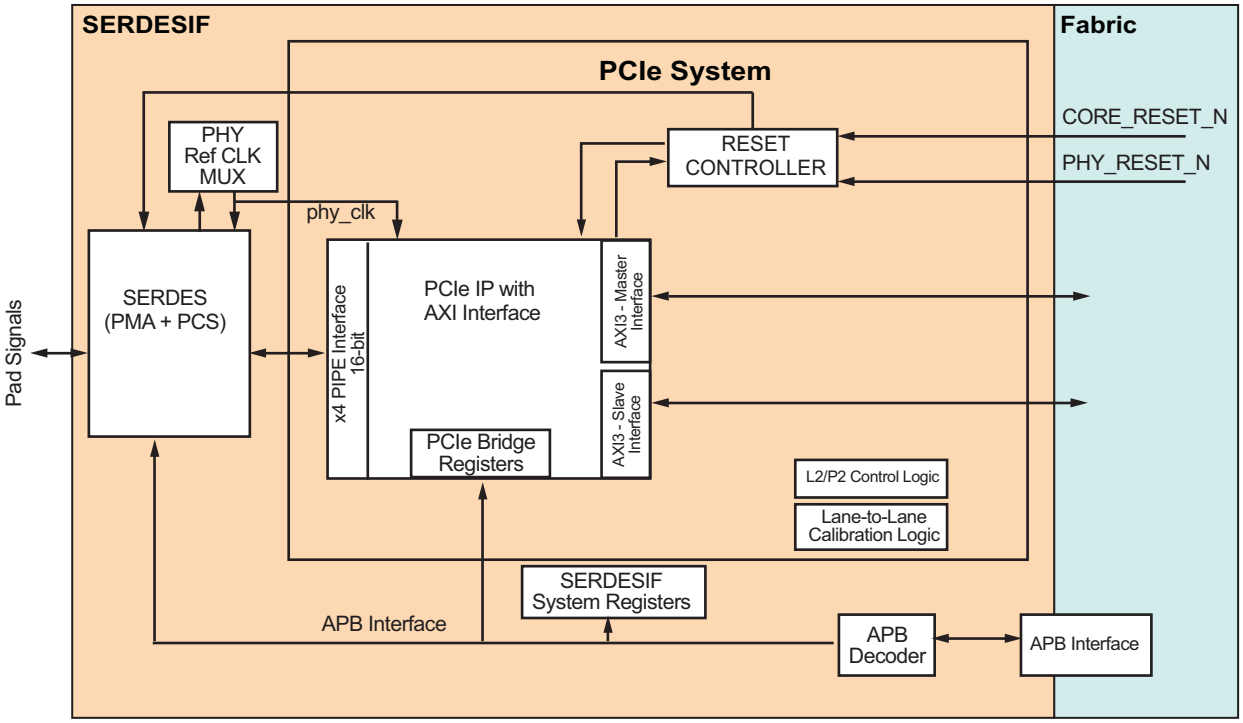
- CLK down block shuts down the receive clock when it is not stable and glitch-free.
- CLK divider function on the RX path is very similar to the one on the transmit side.
- Capture Register is clocked directly by RXCLK (output of clock divider) rising edge which is a “divide by” and delayed version of the RX clock from the PMA block.
- Elastic buffers (also known as elasticity buffers, synchronization buffers, and elastic stores) are used to ensure data integrity when bridging two different clock domains using the PCIe SKP symbol for rate monitoring. Each receiver lane incorporates a decoder, which is fed from the elastic buffer.
- 8b/10b Decoder uses two lookup tables (LUT) (the D and K tables) to decode the 10-bit symbol stream into 8-bit Data (D) or Control (K) characters plus the D/K# signal.

3.4.2 PCIe System

The PCIe system sub-block inside the SERDESIF block implements the PCIe physical layer, data link layer, and transaction layer of the PCIe specification. It interfaces with the SERDES block on one side and the FPGA fabric on the other side. The following figure shows the SmartFusion2 or IGLOO2 SERDESIF block in PCIE mode, and also shows various sub-blocks for the PCIe system block. The main sub-blocks for PCIe system include:

- PCIe IP Block with AXI3 Interface
- AXI to AXI3 Bridge
- Glue Logic Blocks

Figure 19 • **Detailed PCIe System Block Diagram**

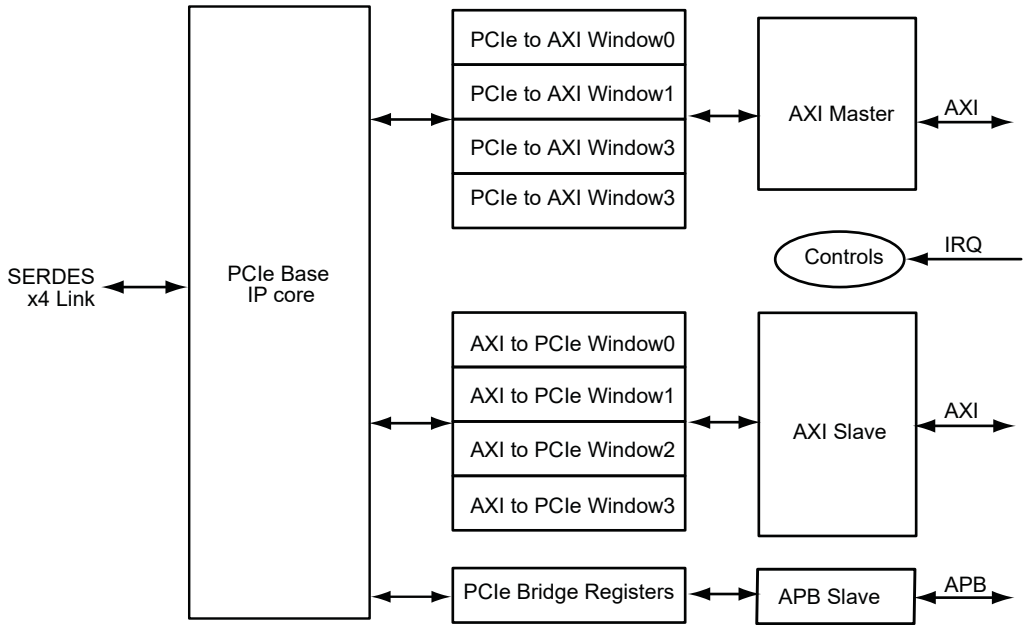


3.4.2.1 PCIe IP Block with AXI3 Interface

The PCIe IP block is a integrated block in the SmartFusion2 or IGLOO2 FPGA which implements a x1, x2, or x4 PCIe interface. On the application side, it has one master interface and one slave interface. The master interface can be a 64-bit AXI3 master. The slave interface can be a 64-bit AXI3 slave interface. The PCIe link initiates transactions to the SmartFusion2 or IGLOO2 fabric through the AXI3 master. IGLOO2 fabric initiates transactions towards the PCIe link through the AXI3 slave interface. AXI3 interfaces (master and slave) can multiplex the transmit buffer to send packets over the PCIe link. There are priority ordering rules in PCIe which mandate the scheduling of packets and this is followed by the PCIe IP block in the case of a collision. There is an APB interface that has access to the SERDESIF system registers.

The following figure shows the architecture of the PCIe IP block.

Figure 20 • **PCIe Hard Block Diagram**



The main components for the PCIe IP sub-block include the following:

- PCIe Base IP Core
- PCIe to AXI Window
- AXI3 Master Block
- AXI3 to PCIe Window
- AXI3 Slave Block
- PCIe Core Bridge Register
- APB Slave Interface

3.4.2.1.1 PCIe Base IP Core

The PCIe base IP core implements an x4 PCIe EP link, compliant to PCIe Rev. 2.0. The following sections describe the features of the SmartFusion2 or IGLOO2 PCIe IP.

General

- x1, x2, x4 PCIe core
- Supports link rate of 2.5 and 5.0 Gbps per lane
- Endpoint Topology
- PCIe Base Specification Revision 2.0 and Revision 1.1 compliant
- Single-function/Single virtual channel (VC)
- AXI3 64-bit Master and Slave Interfaces
- Advanced error reporting (AER) support
- End-to-end cyclic redundancy check (ECRC) generation, check, and forward support

Data Transfer

- Supports all base memory, configuration, and message transactions
- Implements type 0 configuration space for EP (refer to Appendix A: PCIe Configuration Space, page 78)

Configuration

- Supports three 64-bit BARs or six 32-bit BARs

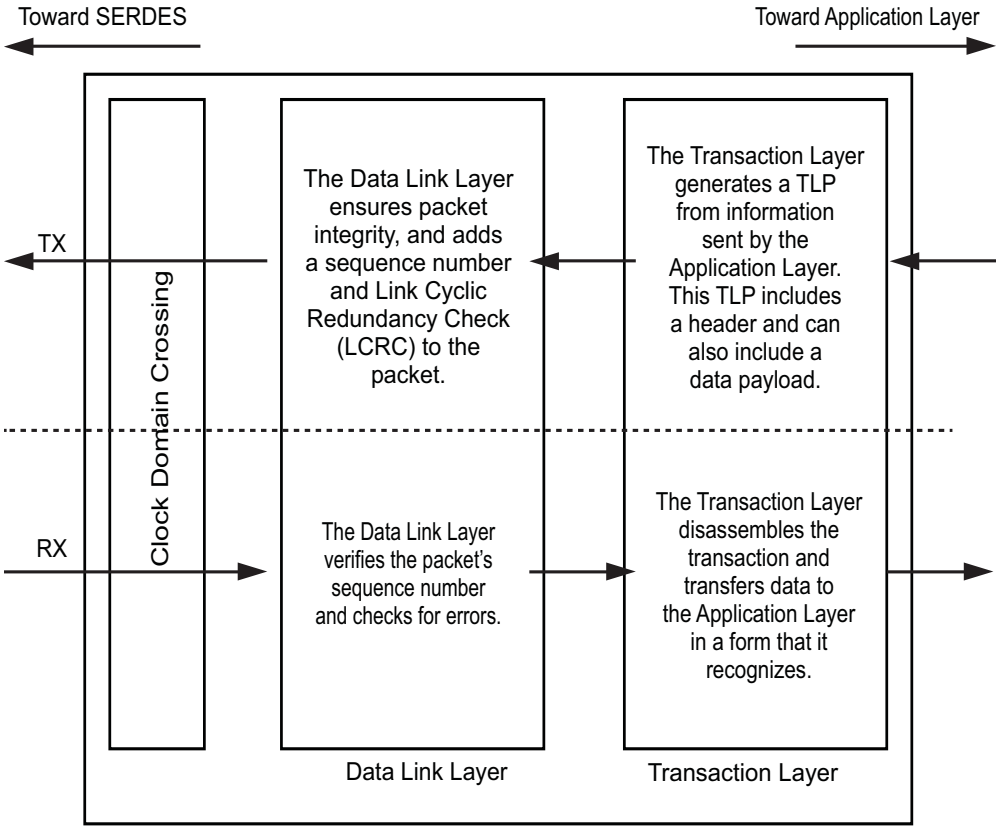
Power Management and Interrupts

- Native active state power management L0s, L1, and L2 state support
- Power management event (PME message)

The PCIe base IP core implements the transaction layer and data link layer described by the PCIe base specifications.

- Transaction layer: The transaction layer (TL) contains the configuration space, which manages communication with the user application layer: the receive and transmit channels, the receive buffer, and flow control (FC) credits.
- 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 CRC checking, error reporting, and logging.

Figure 21 • **PCIe Transaction Layer and Data Link Layer**



The PCIe IP core also utilizes a clock domain crossing (CDC) synchronizer between the DLL and the physical layer that enables the data link and transaction layers to operate at a frequency independent from that of the physical layer.

3.4.2.1.2 PCIe to AXI Window

The PCIe base IP receives both 32-bit address and 64-bit address PCIe requests, but only 32-bit address bits are provided to the AXI3 master. The PCIe to AXI3 address windows manage read and write requests from the PCIe link and are used to translate a PCIe 32-bit or 64-bit base address to a 32-bit AXI3 base address transaction.

3.4.2.1.3 AXI3 Master Block

The AXI3 master only supports memory read and write transactions. It only supports incrementing bursts of length 1.

AXI Master Write Transaction Handling

- The write transaction is handled in little-endian order, as presented at the AXI interface.
- PCIe transactions can be any size up to the configurable maximum payload size (256 bytes).
- AXI3 transactions are limited to 128 bytes, a received TLP is divided into several AXI3 transactions.

- AXI3 master receives a write transaction, it processes the transaction as 128-byte segments (aligned on a 128-byte address boundary) until the segments in the transaction have been processed.
- TLP is de-constructed and sent to the AXI3 interface and the data is presented as little endian.

AXI Master Read Transaction Handling

- Read transactions are handled the same way as write transactions, except that before transferring the transaction to the AXI3 master read channel, the PCIe IP checks the transmit buffer for available space.
- PCIe IP does not transfer the read transaction. If there is not sufficient space in the transmit replay buffer to store PCIe completions.
- The number of outstanding AXI3 master read transactions is therefore limited by the size of the Tx buffer.
- The AXI3 master read channel can receive transactions in any order, and data can be completely interleaved. However, the PCIe IP generates completions in the order they are initiated on the link.

3.4.2.1.4 AXI3 to PCIe Window

The AXI3 to PCIe address windows are used to translate a 32-bit AXI3 base address for a transaction to a PCIe 32-bit or 64-bit base address to generate a PCIe TLP.

3.4.2.1.5 AXI3 Slave Block

The AXI3 slave interface forwards AXI3 read and write requests from the FPGA fabric to the PCIe link.

AXI Slave Write Transaction Handling

- Minimum 128 bytes must be available for write transaction.
- Data interleaving is not supported.
- Wait states are used if buffer is full or has less than 128 Bytes of available space.
- Write responses are generated as soon as the last data phase is over.
- Maximum of 128 Byte data packet can be created.
- Only four outstanding write transactions are supported.
- Incrementing-address burst is supported.

AXI Slave Read Transaction Handling

- Minimum 128 bytes must be available for read transaction.
- PCIe IP generates a PCIe tag, arbitrates between write requests and completions, then checks for available FC credits.
- Response is generated if a timeout occurs or if a completion with error status is received.

Outstanding Requests

The AXI3 interface supports the following number of outstanding requests as listed in the following table.

Table 9 • **AXI3 and Outstanding Transactions**

AXI3 Transaction	Outstanding Transactions
Master Write	Limited by Tx Credits
Master Read	4
Slave Write	Limited by Rx Credits
Slave Read	4

3.4.2.1.6 AXI3 Transaction and TLP Ordering Rules

This section describes the TLP ordering rules for sending and receiving TLPs.

AXI3 Slave Interface: The slave path does not reorder transactions other than reordering using the PCIe standard ordering rules, but does arbitrate between transactions when they occur simultaneously. The order of priority for arbitrations is master read completions, slave write requests, then slave read requests.

AXI3 Master Interface: The master path does not reorder transactions other than reordering using the PCIe standard ordering rules, but does arbitrate between transactions at the AXI3 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 AXI3 master interface. All AXI3 write transactions of size less than 64-bit results in 1 DW TLPs.

3.4.2.1.7 PCIe Core Bridge Register

The PCIe core bridge registers occupy 4 KB of the configuration memory map. These registers set the PCIe configuration and status. These registers are initialized from flash and through the HPMS while configuring the high speed serial interface generator in the Libero SoC. These registers can also be accessed through the 32-bit APB interface. The physical offset location of the PCIe core registers is 0x0000-0x0FFF from the SERDESIF system memory map. Refer to the PCI Express, page 18 for more information on the PCIe core register.

3.4.2.1.8 APB Slave Interface

The APB slave interface provide APB interface to SERDESIF System registers. Refer to the Bridge Register Space, page 48 for details.

3.4.2.2 AXI3 to AHBL Bridge (Master/Slave)

There are two AHBLite IP cores available in the Libero SoC catalog to support AXI to AHBL (master) and AHBL to AXI (slave) between the SerDes block configured with an AXI3 interface and AHB hosted within the FPGA fabric. The two Direct Cores from the Libero SoC catalog are specifically developed to bridge PCIE AXI3 to AHBL transactions. SmartFusion2/IGLOO2 designs requiring SerDes configured as PCIe with AHBLite master should use the CorePCIe_AXItoAHBL DirectCore.

In addition, designs with both master and slave AHBLite interfaces are required to use the CorePCIe_AXItoAHBL and CorePCIe_AHBLtoAXI DirectCores. The Core IP Handbooks provides the implementation guidance.

3.5 Fabric Interface for PCIe System

The SmartFusion2 and IGLOO2 PCIe system block interfaces with the FPGA fabric on one side and the SERDES block on the other side. Following are the PCIe system block interface signals to the FPGA fabric:

- PCIe System AXI3 Master Interface
- PCIe System AXI3 Slave Interface
- PCIe System APB Slave Interface
- PCIe System Clock Signals
- PCIe System Reset Signals
- PCIe Interrupt and Power Management Interface

Table 10 • **PCIe System AXI3 Master Interface**

Port	Type	Description
AXI_M_AWID[3:0]	Output	AXI3 master mode: AWID (not supported)
AXI_M_AWADDR[31:0]	Output	AXI3 master mode: AWADDR
AXI_M_AWLEN[3:0]	Output	AXI3 master mode: AWLEN. HBURST signal is always 1, so INCR BURST is the only supported mode.
AXI_M_AWSIZE[1:0]	Output	AXI3 master mode: AWSIZE
AXI_M_AWBURST[1:0]	Output	AXI3 master mode: AWBURST. HTRANS signal is limited to 0 and 2, so only NONSEQ mode is supported. No BUSY mode available.
AXI_M_AWVALID	Output	AXI3 master mode: AWVALID
AXI_M_AWREADY	Input	AXI3 master mode: AWREADY
AXI_M_WID[3:0]	Output	AXI3 master mode: WID (not supported)
AXI_M_WSTRB[7:0]	Output	AXI3 master mode: WSTRB

Table 10 • **PCIe System AXI3 Master Interface *(continued)***

Port	Type	Description
AXI_M_WLAST	Output	AXI3 master mode: WLAST
AXI_M_WVALID	Output	AXI3 master mode: WVALID
AXI_M_WDATA[63:0]	Output	AXI3 master mode: WDATA
AXI_M_WREADY	Input	AXI3 master mode: WREADY
AXI_M_BID[3:0]	Input	AXI3 master mode: BID
AXI_M_BRESP[1:0]	Input	AXI3 master mode: BRESP. In response to a write the value is ignored.
AXI_M_BVALID	Input	AXI3 master mode: BVALID
AXI_M_BREADY	Output	AXI3 master mode: BREADY
AXI_M_ARID[3:0]	Output	AXI3 master mode: ARID. Used to indicate the ID of the current outstanding read completion. Read completions can be interleaved.
AXI_M_ARADDR[31:0]	Output	AXI3 master mode: ARADDR
AXI_M_ARLEN[3:0]	Output	AXI3 master mode: ARLEN
AXI_M_ARSIZE[1:0]	Output	AXI3 master mode: ARSIZE (Tied to 11)
AXI_M_ARBURST[1:0]	Output	AXI3 master mode: ARBURST
AXI_M_ARVALID	Output	AXI3 master mode: ARVALID
AXI_M_ARREADY	Input	AXI3 master mode: ARREADY
AXI_M_RID[3:0]	Input	AXI3 master mode: RID. Used to indicate the ID of the current outstanding read completion. Read completions can be interleaved.
AXI_M_RDATA[63:0]	Input	AXI3 master mode: RDATA
AXI_M_RRESP[1:0]	Input	AXI3 master mode: RRESP. In response to a read request, a SLVERR (0b10) or DECERR (0b11) response causes the PCIe core to issue an Unsupported Request back to the initiator.
AXI_M_RLAST	Input	AXI3 master mode: RLAST
AXI_M_RVALID	Input	AXI3 master mode: RVALID
AXI_M_RREADY	Output	AXI3 master mode: RREADY

Note: AXI_M_awsize[2:0] is hardwired to ‘011’, that is, fixed at 8 bytes = 64 bit only. Same applies to AXI_M_arsize[2:0].

Note: Refer to AMBA AXI3 specifications for further details.

Table 11 • **PCIe System AXI3 Slave Interface**

Port	Type	Description
AXI_S_AWID[3:0]	INPUT	AXI3 slave mode: AWID (not supported)
AXI_S_AWADDR[31:0]	INPUT	AXI3 slave mode: AWADDR
AXI_S_AWLEN[3:0]	INPUT	AXI3 slave mode: AWLEN
AXI_S_AWSIZE[1:0]	INPUT	AXI3 slave mode: AWSIZE
AXI_S_AWBURST[1:0]	INPUT	AXI3 slave mode: AWBURST
AXI_S_AWVALID	INPUT	AXI3 slave mode: AWVALID
AXI_S_AWREADY	OUTPUT	AXI3 slave mode: AWREADY
AXI_S_AWLOCK[1:0]	INPUT	AXI3 slave mode: AWLOCK

Table 11 • **PCIe System AXI3 Slave Interface *(continued)***

Port	Type	Description
AXI_S_WID[3:0]	INPUT	AXI3 slave mode: WID (not supported)
AXI_S_WSTRB[7:0]	INPUT	AXI3 slave mode: WSTRB
AXI_S_WLAST	INPUT	AXI3 slave mode: WLAST
AXI_S_WVALID	INPUT	AXI3 slave mode: WVALID
AXI_S_WDATA [63:0]	INPUT	AXI3 slave mode: WDATA
AXI_S_WREADY	OUTPUT	AXI3 slave mode: WREADY
AXI_S_BID[3:0]	OUTPUT	AXI3 slave mode: BID
AXI_S_BRESP[1:0]	OUTPUT	AXI3 slave mode: BRESP
AXI_S_BVALID	OUTPUT	AXI3 slave mode: BVALID
AXI_S_BREADY	INPUT	AXI3 slave mode: BREADY
AXI_S_ARID[3:0]	INPUT	AXI3 slave mode: ARID
AXI_S_ARADDR[31:0]	INPUT	AXI3 slave mode: ARADDR
AXI_S_ARLEN[3:0]	INPUT	AXI3 slave mode: ARLEN. PCIe AXI-Slave interface supports only single length transactions (S_ARLEN = 3'b000) when the size of the transfer is less than 64 bits.
AXI_S_ARSIZE[1:0]	INPUT	AXI3 slave mode: ARSIZE. A size value of 0b11 allows all values of burst length. A size value of 0b00, 0b01, and 0b10 allows only a burst length of 1.
AXI_S_ARBURST[1:0]	INPUT	AXI3 slave mode: ARBURST
AXI_S_ARVALID	INPUT	AXI3 slave mode: ARVALID
AXI_S_ARLOCK[1:0]	INPUT	AXI3 slave mode: ARLOCK
AXI_S_ARREADY	OUTPUT	AXI3 slave mode: ARREADY
AXI_S_RID[3:0]	OUTPUT	AXI3 slave mode: RID
AXI_S_RDATA[63:0]	OUTPUT	AXI3 slave mode: RDATA
AXI_S_RRESP[1:0]	OUTPUT	AXI3 slave mode: RRESP. The interface responds with a SLVERR under the following conditions: If a completion TLP has the EP (poisoned) bit set. Unsupported Request. If a completion with error TLP is received. If a completion timeout event happens. If an invalid AXI3 slave transaction is encountered such as invalid burst type.
AXI_S_RLAST	OUTPUT	AXI3 slave mode: RLAST
AXI_S_RVALID	OUTPUT	AXI3 slave mode: RVALID
AXI_S_RREADY	INPUT	AXI3 slave mode: RREADY

Note: Refer to AMBA AXI3 specifications for further details

Table 12 • **PCIe System APB Slave Interface**

Port	Type	Description
APB_S_PSEL	Input	APB slave select; select signal for register for reads or writes.
APB_S_PENABLE	Input	APB strobe. This signal indicates the second cycle of an APB transfer.
APB_S_PWRITE	Input	APB write or read. If High, a write occurs when an APB transfer takes place. If low, a read takes place.
APB_S_PADDR[13:0]	Input	APB address bus.

Table 12 • PCIe System APB Slave Interface *(continued)*

APB_S_PWDATA[31:0]	Input	APB write data.
APB_S_PREADY	Output	APB ready. Used to insert wait states.
APB_S_PRDATA[31:0]	Output	APB read data.
APB_S_PSLVERR	Output	APB Error.

Table 13 • PCIe System Clock Signals

Port	Type	Description
CLK_BASE	Input	Fabric source clock. This clock input is used for the Master and Slave interfaces. It is also used as the reference clock to the SPLL which is used to achieve interface timing across the fabric to SERDES block. **Note**: The frequency of this clock must match the GUI option for the CLK. BASE rate to guarantee timing is met across the fabric interface.
APB_S_CLK	Input	PCLK for APB slave interface in SERDES Block
SPLL_LOCK	Output	PLL Lock signal. High indicates that the frequency and phase lock are achieved.
PLL_LOCK_INT	Output	The SPLL Lock status register (Active High indicates locked).
PLL_LOCKLOST_INT	Output	The SPLL Lock lost status register (Active high indicates that the lock is lost).

Table 14 • PCIe System Reset Signals

Ports	Type	Description
CORE_RESET_N	Input	PCIe core active low reset. Top-level fundamental asynchronous RESET to the PCIe system. It affects only those SERDES lanes which are in PCIe mode. Lanes associated with the PCIe link must have one reset for all lanes.
PHY_RESET_N	Input	Active low – SERDES – reset. Top-level fundamental asynchronous RESET to the SERDES block.
APB_S_PRESET_N	Input	APB slave interface – PRESETN: Async set. APB asynchronous reset to all APB registers.

Note: More information about these resets is provided in Table 20, page 50.

PCIE_WAKE_N, PCIE_WAKE_REQ, and PCIE_PERST_N ports are added optionally with L2/P2 selection. PCIE_WAKE_N is an output and PCIE_WAKE_REQ, and PCIE_PERST_N ports are inputs to the PCIE core.

Table 15 • PCIe Interrupt and Power Management Interface

Port	Type	Description
PCIE_INTERRUPT[3:0]	Input	PCIe system interrupt inputs. When using INTx legacy interrupts PCIE_INTERRUPT[0] is used to assert/deassert INTA. When using MSI interrupts up to 4 MSI interrupts can be sent. Each bit sends an MSI vector with bit 0 starting at the MSI base and each bit increments the vector. These require a minimum of two clocks (CLK_BASE) for the controller to capture the PCIE_INTERRUPT.
PCIE_SYSTEM_INT	Output	PCIe system interrupt output (not supported)

Table 15 • **PCIe Interrupt and Power Management Interface** ***(continued)***

Port	Type	Description
PCIE_WAKE_REQ	Input	L2/P2 implementation: (L2 requests from fabric)[1] Asynchronous. Input to power-management state machine.
PCIE[0:1]_EV_1US	Output	1 µs Event Timer: The block outputs a pulse signal every microsecond, which can be used as a real-time counter.
PCIE_WAKE_N	Output	L2/P2 implementation: (L2 exit request to RP)[1]
PCIE_PERST_N	Input	L2/P2 implementation: (L2 exit request from RP)[1] Asynchronous. Synchronized to the internal 50 MHz RC Oscillator.
PCIE_LTSSM[5:0]	Output	Ports indicate the status of ltssm state management. Equivalent to LTSSM Register (044h) [28:24]. These bits set to LTSSM state encoding (RO)- output bits [4:0] ltssm state. Bit[5] Pulses high to indicate that a hot-reset, data link up or L2 exit condition has occurred. Synchronized to the PIPE clock.
PCIE_L2P2_ACTIVE[2]	Output	Active high output indicating the LTSSM is in low-power state.
PCIE_RESET_PHASE	Output	Active high output indicating the LTSSM is in reset state.

1. This in only available when L2/P2 option is selected in SERDESIF Configurator GUI.
2. This output is synchronized to the 50MHz OSC clock within the FPGA and should be treated as asynchronous in the fabric.

3.6 Functional Description

This section covers the functional aspects of the reset and clock circuitry inside the SERDESIF block for PCIe. It includes the following sub-sections:

- PCIe Clocking Architecture
- PCIe Reset Network

3.6.1 PCIe Clocking Architecture

The SmartFusion2 and IGLOO2 SERDESIF, when configured in PCIe mode, uses multiple clocks inside the SERDESIF block. This sub-section describes the PCIe clocking architecture inside SERDESIF in PCIe mode. Figure 22, page 37 shows the PCIe clocking architecture in the SmartFusion2 or IGLOO2 device. The two main clock inputs are a differential SERDES reference clock (100 MHz) for SERDES physical media attachment (PMA), and a CLK_BASE input for SERDESIF from the FPGA fabric. In addition, there is a APB clock input for SERDESIF from the FPGA fabric.

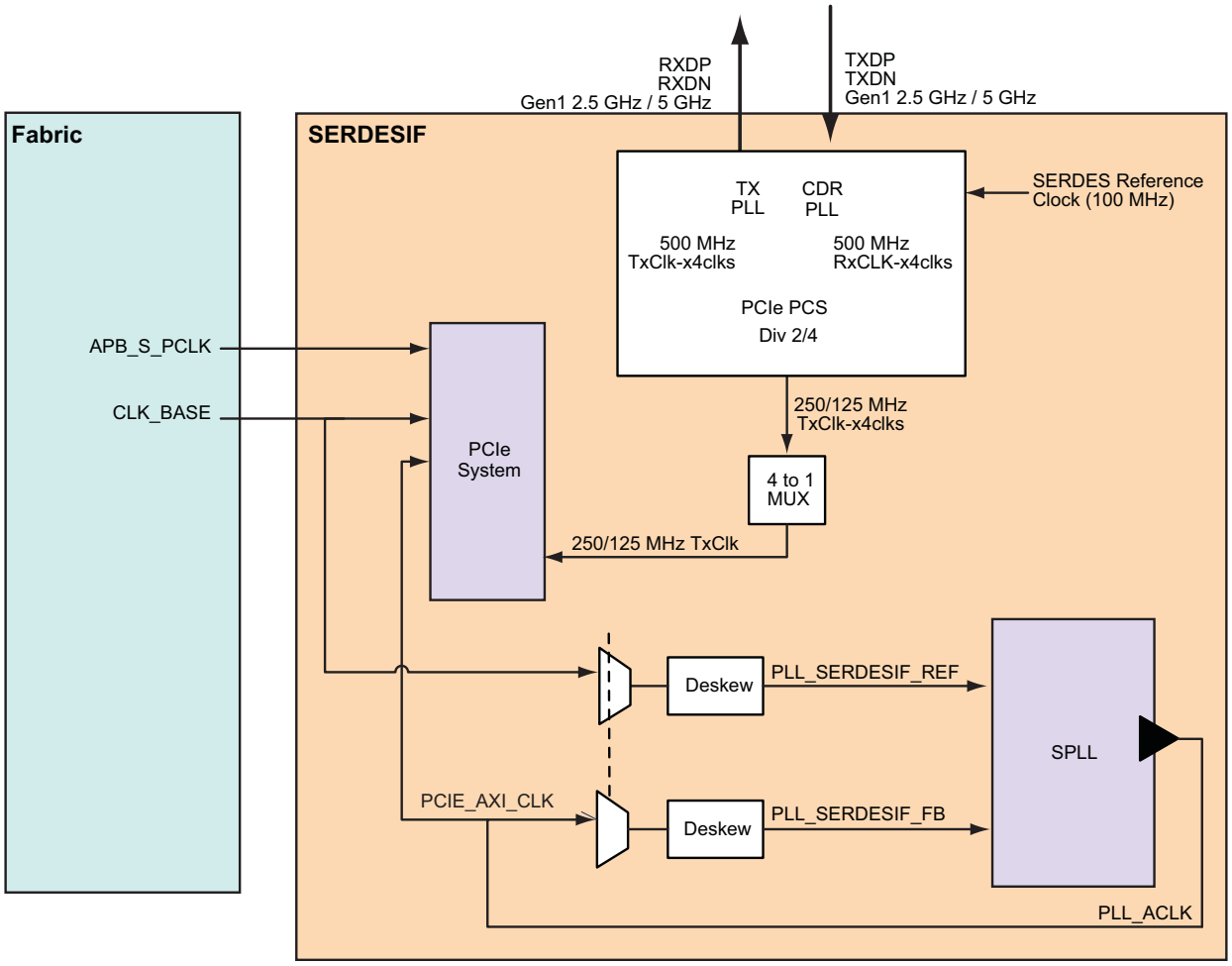
SERDES reference clock: The differential 100MHz reference clock is used by SERDES (TX PLL and CDR PLL) to generate 250 MHz or 125 MHz clock (depending on lane speed settings) and passed to PCIe System IP block. The setting for TX PLL and CDR PLL are calculated automatically by the Libero software. This 250 MHz or 125 MHz clock output from SERDES is used by PCIe system. There are several options for providing this SERDES reference clock. Refer to the SERDES Reference Clocks, page 140 in the Serializer/De-serializer, page 128 for details.

The PCIe standard specifies a 100 MHz clock (Refclk) with greater than ±300 ppm frequency stability at both the transmitting and receiving devices. SmartFusion2 and IGLOO2 support two distinct clocking topologies: Common Refclk and Separate Refclk.

Common Refclk is the most widely supported clocking method in open systems where the root provides a clock to the end point. An advantage of this clocking architecture is that it supports spread spectrum clocking (SSC) which can be very useful in reducing electromagnetic interference (EMI). SmartFusion2 and IGLOO2 support SSC clocking in common clock systems.

Separate Refclk uses two independent clock sources. One clock for the root and another clock source for the endpoint. The clock sources still must be ±300 ppm frequency accuracy and cannot use any SSC.

Figure 22 • **Various Clocks in PCIe Mode**

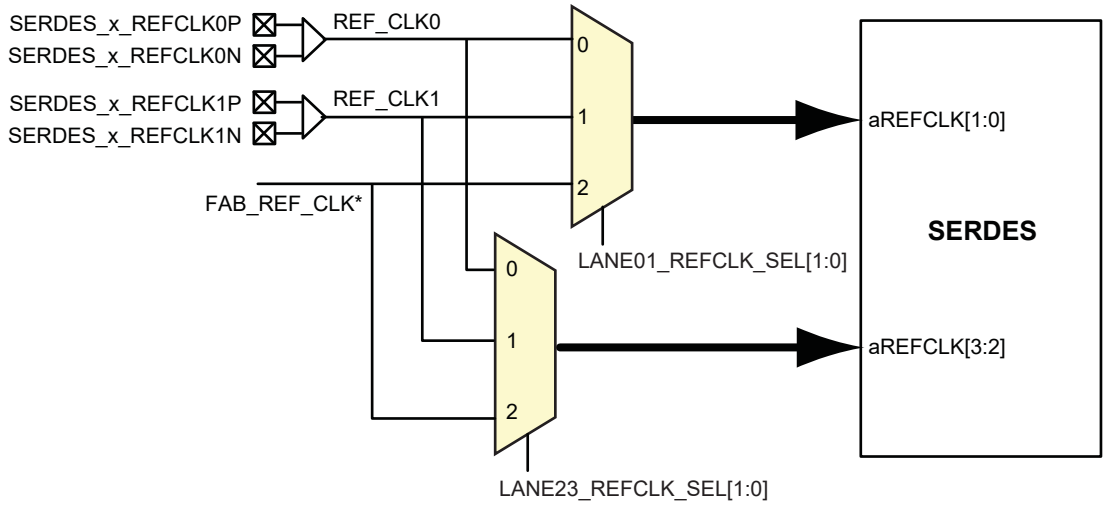


The clocking architecture also uses SPLL to synchronize data between CLK_BASE and the clock generated from SERDES (250 MHz or 125 MHz clocks). The SPLL allows the reduction of the skew between the fabric and the SmartFusion2 or IGLOO2 SERDESIF module. Table 17, page 26 summarizes the SERDESIF clock signal in PCIe mode.

3.6.1.1 SERDES Reference Clocks Selection

The PLLs in the SERDES block generate the 250/125 MHz clock for the PCIe system. REFCLK0, or REFCLK1, as the reference clock depending on Protocol mode (single or multiple protocol). Lane0 and lane1 share the same reference clock and lane2 and lane3 share the same reference clock. The reference clock pads are differential input. In Single-protocol mode, the same reference clock can be selected for all four lanes.

Figure 23 • **SERDES Reference Clock for PCIe Mode**



Note: '*' - FAB_REF_CLK not available with PCIe Mode

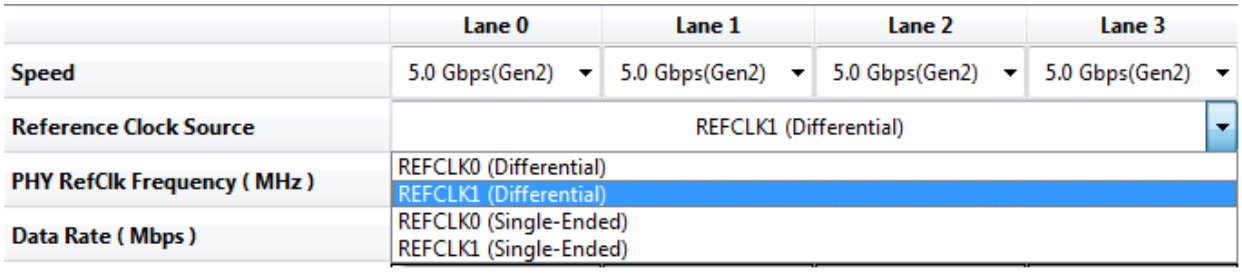
The reference clock needs to be compliant with the PCIe protocol. Refer to the *DS0128: IGLOO2 and SmartFusion2 Datasheet* for the specification. Microsemi recommends using REFCLK0 or REFCLK1 for PCIe mode.

Table 16 • **Reference Clock Signals for SERDES**

Clock Signal	Description
REFCLK0	Reference clock output of SERDES_x_REFCLK0P and SERDES_x_REFCLK0N
REFCLK1	Reference clock output of SERDES_x_REFCLK1P and SERDES_x_REFCLK1N

The following figure shows the reference clock selection in the high speed serial interface generator that is available in Libero. Libero sets the multiplexer (MUX) selection depending on the reference clocks selected.

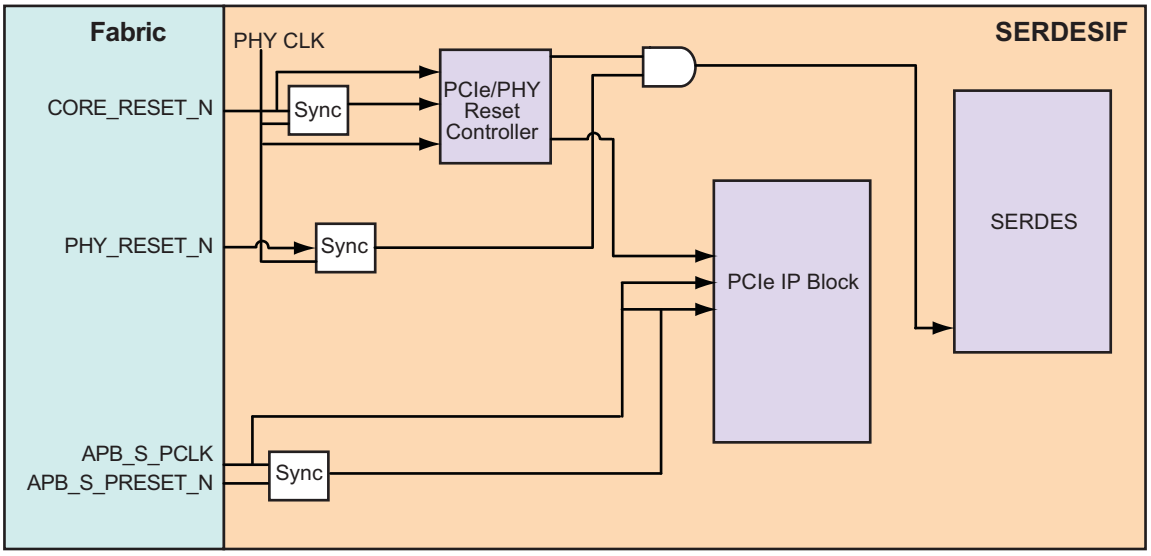
Figure 24 • **SERDES Reference Clock Using the High Speed Serial Interface Generator**



3.6.2 PCIe Reset Network

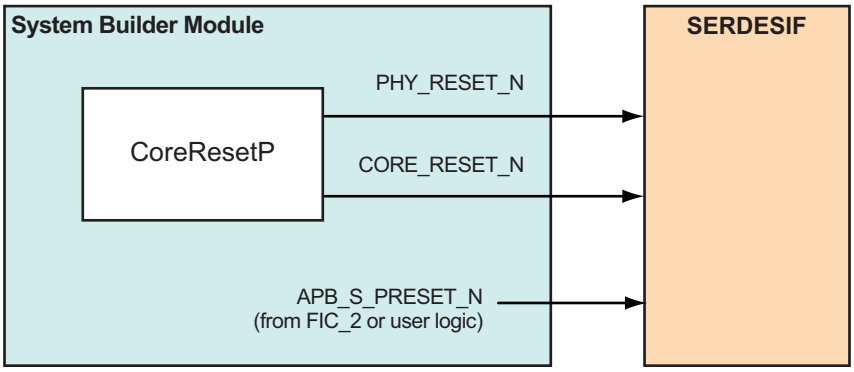
SERDESIF, when configured in PCIe mode, has different reset inputs. The following figure shows a simplified view of the reset signal in SERDESIF block in PCIe mode. Table 20, page 50 shows the reset signals and recommended connection.

Figure 25 • **Reset Signals in PCIe Mode**



Microsemi recommends using System Builder module and the embedded CoreResetP to control CORE_RESET_N and PHY_RESET_N. The System Builder module uses the recommended sequences for the various reset signals. APB_S_PRESET_N signal can be controlled from the FIC_2 interface of the HPMS or user logic, as shown in the following figure.

Figure 26 • **Reset Signals in PCIe Mode**



3.7 Designing with PCIe

This section provides instruction for using the SmartFusion2 or IGLOO2 PCIe EP implementation user design. It includes the following sub-sections:

- Base Address Register Settings
- Address Translation on AXI3 Master Interface
- AXI3 Slave Interface Address Translation
- PCIe System Credit Settings
- Setting up Lane Reversal
- PCIe Power Management

3.7.1 Base Address Register Settings

The PCIe implementation supports up to six 32-bit BARs or three 64-bit BARs. The BARs can be one of two sizes:

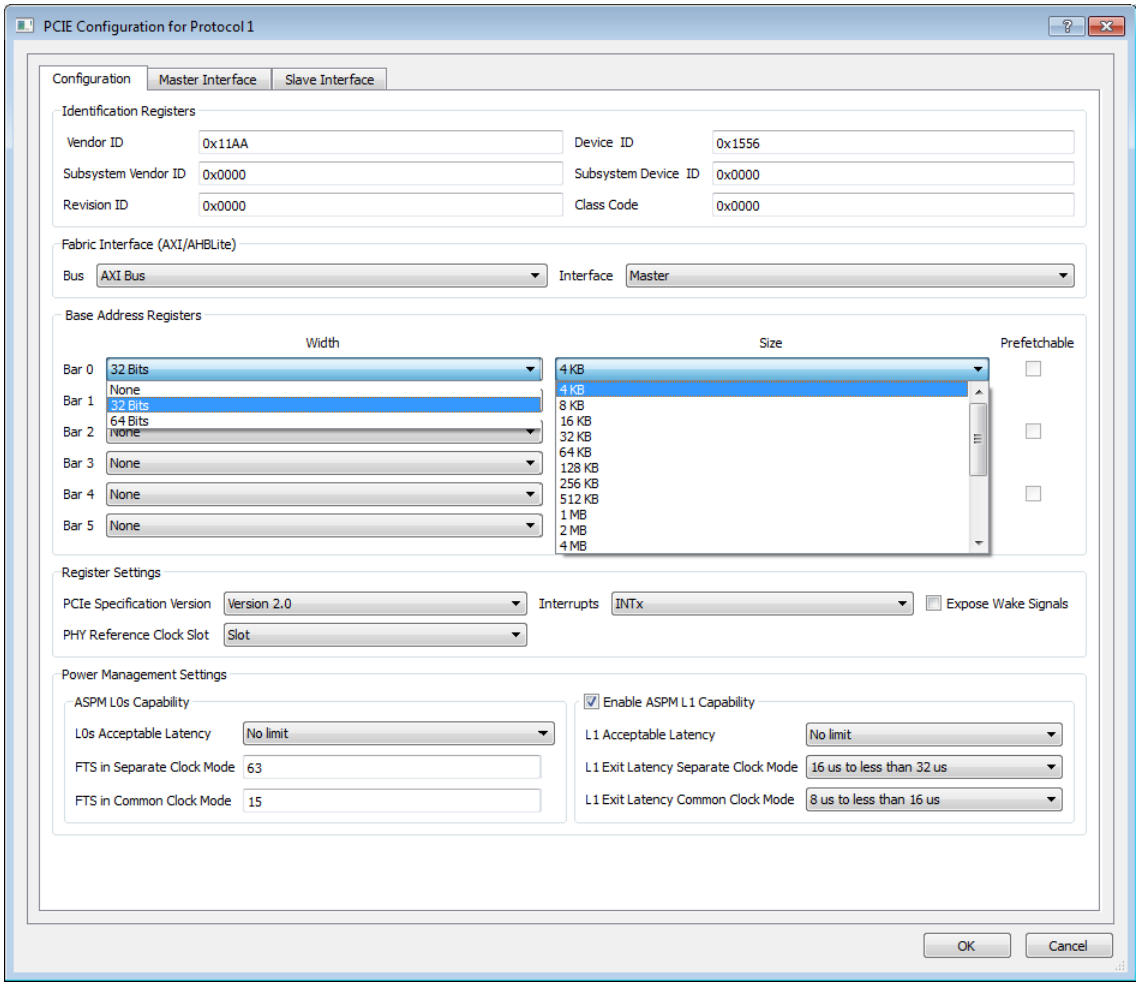
- 32-bit BAR: The address space can be as small as 16 bytes or as large as 2 gigabytes.
- 64-bit BAR: The address space can be as small as 128 bytes or as large as 8 gigabytes. Used for memory only.

Each BAR register is 32 bits, but BARs can be combined to make a 64-bit BAR. For example, BAR0 (address offset 010h) and BAR1 (address offset 014h) define the type and size of BAR01 of the PCIe native endpoint. BAR01 can be memory-mapped prefetchable (64-bit BAR) or non-prefetchable (32-bit BAR). PCIe BAR does not support I/O mapped space and support only the memory mapped space.

The SERDESIF Configurator in Libero provides a GUI to configure the BAR settings for the EP application. Refer to the following figure for details. The BAR registers share the options below:

- Width: Width can be 32-bit or 64-bit. If an even register is selected to be 64-bit wide, then the subsequent (odd) register serves as the upper half of 64 bits. Otherwise, the width of odd registers is restricted to 32-bit.
- Size: Ranges from 4 Kbytes to 1 Gbyte.
- Prefetchable: Prefetchable option for memory BAR.
    - A PCI Express Endpoint requesting memory resources through a BAR must set the BAR's prefetchable bit unless the range contains locations with read side-effects or locations in which the device does not tolerate write merging. It is strongly encouraged that memory-mapped resources be designed as prefetchable whenever possible. For a PCI Express Endpoint, 64-bit addressing must be supported for all BARs that have the prefetchable bit set. 32-bit addressing is permitted for all BARs that do not have the prefetchable bit set.

Figure 27 • **Configuring Base Address Register**



The BAR setting is read by the PCIe bridge register inside the SERDESIF block using the APB interface.

3.7.2 Address Translation on AXI3 Master Interface

The address space for PCIe is different from the AXI3 address space. To access one address space from another address space requires an address translation process.

The PCIe IP can receive both 32-bit address and 64-bit address PCIe requests, but only 32-bit address bits are provided to the AXI3 master. In order to manage address translation, the PCIe IP can implement up to 4 AXI3 master address windows, which can be mapped to 3 BARs in the main PCIe IP core.

The address mapping registers (AXI_MASTER_WINDOWx[x], where x can be 0, 1, 2, or 3) are shown in the following table and are used to set the address mapping.

Table 17 • **AXI_MASTER_WINDOW Registers**

Bit Number	Name	Description
[31:12]	AXI_MASTER_WINDOWx[0]	Base address AXI3 master window x
[11:0]		Reserved

Table 17 • **AXI_MASTER_WINDOW Registers *(continued)***

Bit Number	Name	Description
[31:12]	AXI_MASTER_WINDOWx[1]	Size of AXI3 master window x
[11:1]		Reserved
0		Enable bit of AXI3 master window x
[31:12]	AXI_MASTER_WINDOWx[2]	LSB of base address PCIe window x
[11:6]		Reserved
[5:0]		These bits set the BAR. To select a BAR, set the following values: 0x01: BAR0 (32-bit BAR) or BAR0/1 (64-bit BAR) 0x02: BAR1 (32-bit BAR) only 0x04: BAR2 (32-bit BAR) or BAR2/3 (64-bit BAR) 0x08: BAR3 (32-bit BAR) only 0x10: BAR4 (32-bit BAR) or BAR4/5 (64-bit BAR) 0x20: BAR5 (32-bit BAR) only
[31:0]	AXI_MASTER_WINDOWx[3]	MSB of base address PCIe window x

Note: x = 0, 1, 2, 3

Each AXI3 master address window implemented can be mapped to a BAR, and several address windows can be mapped to the same BAR. When transferring PCIe receive requests to the AXI3 Master, the PCIe IP core automatically removes the decoded BAR base address, then performs a windows match using the PCIe offset address. If a match is found, the bridge then maps the corresponding AXI3 base address.

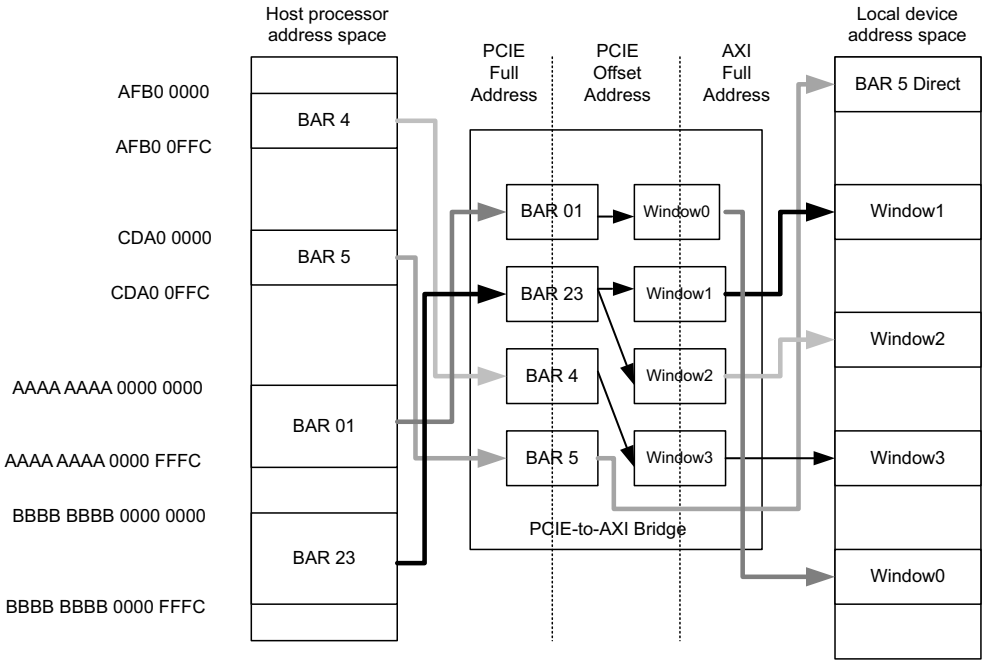
For example, in the following figure, four BARS are enabled in the bridge; two 64-bit BARS (BAR01 and BAR23), and two 32-bit BARS (BAR4 and BAR5). All AXI3 master windows are utilized: 64-bit BAR01 is mapped to AXI3 master window 0; 64-bit BAR23 is mapped to AXI3 master window 1; and AXI3 master window 2 is mapped to the upper 64 bytes of BAR23. AXI3 master window 3 is connected to BAR4. BAR5 is not mapped to an AXI3 window; its offset is passed directly to the AXI3 master and translation is not performed.

To configure AXI_MASTER_WINDOW[0], four APB write operations are performed to AXI_MASTER_WINDOW0[0], AXI_MASTER_WINDOW0[1], AXI_MASTER_WINDOW0[2] and AXI_MASTER_WINDOW0[3] registers:

- APB write to AXI_MASTER_WINDOW0[0]: APB PADDR = 100h, APB PWDATA = FFF0 0000
- APB write to AXI_MASTER_WINDOW0[1]: APB PADDR = 104h, APB PWDATA = FFF0 0001
- APB write to AXI_MASTER_WINDOW0[2}: APB PADDR = 108h, APB PWDATA = 0000 0001
- APB write to AXI_MASTER_WINDOW0[3]: APB PADDR = 10Ch, APB PWDATA = 0000 0000

The example is shown using relative SERDESIF addressing for PADDR.

Figure 28 • **16 PCIe to AXI3 Master Address Translation**



If window size is not enabled or if the PCIe offset address is located in a BAR but not in any of the windows, address translation is not performed. In this case, the PCIe base address is removed to create the AXI3 address and, for BARs larger than 4 Kbytes, MSBs are ignored.

The address translation needs to be pre-defined in the user design. This is completed using the SERDESIF configurator GUI.

The PCIe AXI3 master windows are used to translate the PCIe address domain to the local device address domain. Typically the PCIe AXI3 master windows are used to translate the address of base address registers.

3.7.3 AXI3 Slave Interface Address Translation

The bridge can configure up to four AXI3 slave address windows to handle address translation on read/write requests initiated from the FPGA fabric. The AXI3 slave address windows are used to translate a 32-bit AXI3 base address for a transaction to a PCIe 32-bit or 64-bit base address to generate a PCIe TLP. The slave address windows can also be used to generate the following PCIe parameters:

- TC selection: Indicates the PCIe traffic class in the PCIe packet header.
- RO bit selection: Generates the PCIe TLP using a selectable relaxed ordering bit.
- No snoop bit selection: Generates the PCIe TLP using a selectable no snoop bit. See Section 2.2.6.5 of the PCIE 2.0 Base specification for option details.

The address mapping registers (AXI_SLAVE_WINDOWx[x], x can be 0, 1, 2, or 3) is used to set the address mapping, refer to the following table.

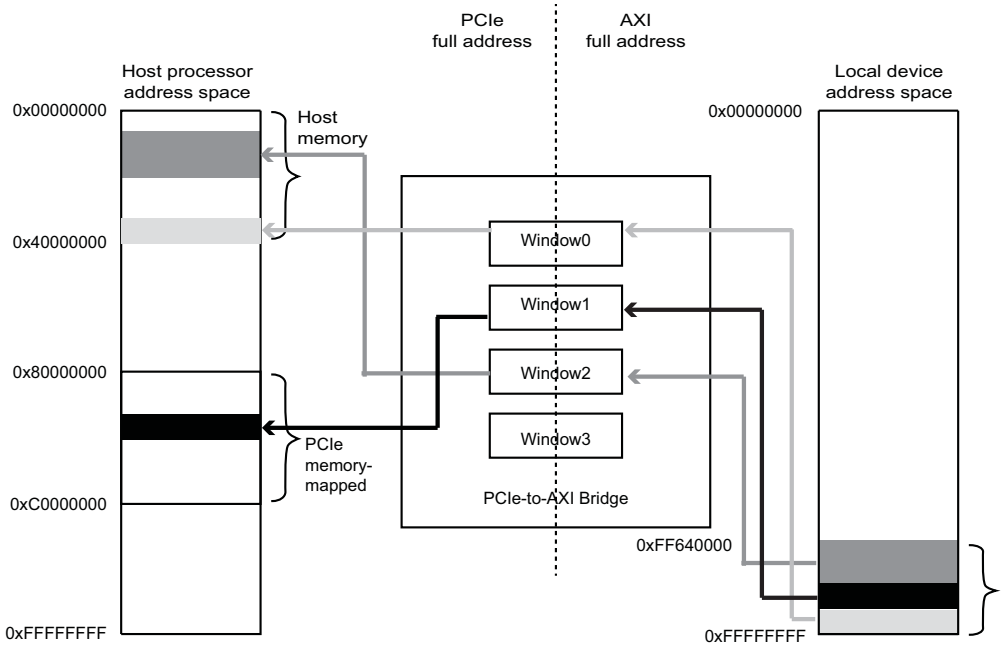
Table 18 • **AXI_SLAVE_WINDOW Registers**

Bit Number	Name	Description
[31:12]	AXI_SLAVE_WINDOWx[0]	Base address AXI3 slave window x
[11:0]		Reserved
[31:12]	AXI_SLAVE_WINDOWx[1]	Size of AXI3 slave window x
[11:1]		Reserved
0		Enable bit of AXI3 slave window x
[31:12]	AXI_SLAVE_WINDOWx[2]	LSB of base address PCIe window 0
[11:5]		Reserved
[4:2]		AXI3 slave window 0 traffic class (TC)
1		AXI3 Slave window 0 relaxed ordering (RO)
0		AXI3 Slave window 0 no snoop (NS)
[31:0]	AXI_SLAVE_WINDOWx[3]	MSB of base address PCIe window x

Note: x = 0, 1, 2, 3

The following figure shows address translation when AXI3 slave address window 0 and window 2 target two different regions of the host memory and the AXI3 slave address window 1 targets a PCIe memory-mapped device (enabling peer-to-peer transactions). Window 3 is not used in this example.

Figure 29 • **AXI3 Slave to PCIe Address Translation**



If AXI3 slave windows are not enabled, address translation is not performed and AXI3 slave requests are transferred to the PCIe IP core with defaults of TC = 0, RO = 0, and NS = 0.

3.7.4 PCIe System Credit Settings

PCIe system has 2 Kbytes of receive buffer (RAM) and 1 Kbyte of transmit and replay buffer (RAM). The following sections describe the different features that impact credit processing. All of the credit settings are set automatically by the Libero software based on the buffer sizes fixed in the SERDESIF.

3.7.4.1 Maximum Payload Size

The size of TLP is restricted by the capabilities of both link partners. After the link is trained, the root complex sets the MAX_PAYLOAD_SIZE (maximum payload size register) value in the device control register. The permitted settings for MAX_PAYLOAD_SIZE is 128 bytes (see Table 46, page 61).

3.7.4.2 Replay Buffer

The replay buffer, located in the data link layer, stores a copy of a transmitted TLP until the transmitted packet is acknowledged by the receiving side of the link. Each stored TLP includes the header, an optional data payload (of which the maximum size is determined by the maximum payload size parameter), an optional ECRC, the sequence number, and the link cyclic redundancy check (LCRC) field.

3.7.4.3 Transmit Buffer

The transmit buffer (Tx buffer) stores the read data payload from the AXI3 master as well as the write data payload from the AXI3 slave.

3.7.4.4 Receive Buffer

The receive buffer is located in the transaction layer and accepts incoming TLPs from the link and then sends them to the application layer for processing. The receive buffer stores TLPs based on the type of transaction, not the TC of a transaction. Types of transactions include posted transactions, non-posted transactions, and completion transactions. A transaction always has a header but does not necessarily have data. The receive buffer accounts for this distinction, maintaining separate resources for the header and data of each type of transaction. To summarize, distinct buffer resources are maintained for each of the following elements:

- Posted transactions, header (PH)
- Posted transactions, data (PD)
- Non-posted transactions, header (NPH)
- Non-posted transactions, data (NPD)
- Completion transactions, header (CPLH)
- Completion transactions, data (CPLD)

TLPs are stored in the received buffer in 64-bit addressing format, with each AXI3 slave read outstanding request consuming 16 credits (128 bytes), plus headers and data credits consuming 1 credit each (16 bytes).

3.7.5 User Data Throughput

PCIe uses credits to handle throughput balancing between both ends of the link. At the initial link-up, both sides of the link share their transmit and receive buffer sizes in terms of credits. As TLPs are sent across the link, credits are used. As user data is pulled out of the TLPs stored in the receive buffers credits are released. Information on the current state of the credits is continuously sent across the link using data link layer packets. All of this happens transparent to the user inside the PCIe core.

The [SmartFusion2/IGLOO2] PCIe core uses AXI3 fabric interface for user data. AXI3 slave interface will only allow a transaction when 128 byte TLP worth of credits are available to be sent. Using this method, the PCIe core can back-pressure the fabric interface when credits are not available to send a TLP. The SERDESIF PCIe core does outstanding transactions with single ID. In this case, all the transactions go in sequence. PCIe AXI slave IF accepts four outstanding transactions. AWREADY/ARREADY is de-asserted to apply back pressure.

The flow control works by releasing credits to the sender as data is pulled across to the fabric. If the user is not pulling the data out fast enough, then the sender will run out of credits. In the worst case situation where the sender is 100% writing data to the PCIe core only 1325 Mbps will be able to go through. The credit system holds the sender back from sending more data. Similar for where the sender is 100% pulling data via a read, only 1325 Mbps comes back. In this case, PCIe core is never throttled back due to a lack of credits.

Reverse the situation with the PCIe core as the sender. For 100% write data, the fabric interface is held up at 1325 Mbps. For 100% read requests the receiver wants to send them back faster than 1325 Mbps, but the fabric interface will only pull them out at 1325 Mbps. The receiver is blocked by a lack of completion credits until credits are released.

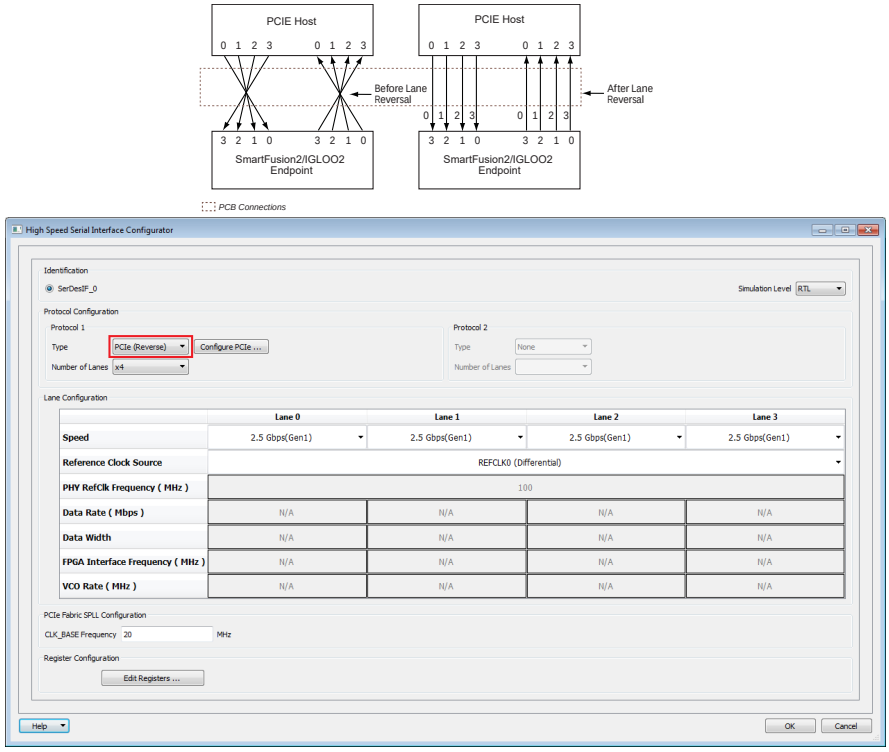
The *PCI Express Base Specification* defines a read completion boundary (RCB) parameter of the Link Control register (bit-3). The RCB parameter determines the naturally aligned address boundaries on which a read request may be serviced with multiple completions. For Smartfusion2/IGLOO2 endpoints, the RCB is fixed to 64 bytes.

The transaction size in this situation is more efficient when small (128 byte TLPs). Smaller packet sizes allow PCIe core to release credits faster compared to large packets. As a packet is pulled across the fabric interface the credit is released for the sender to send another. If this happens quickly the next packet can be sent. For large packets it takes longer to release the credit and therefore the next packet is not sent as quickly.

3.7.6 Setting up Lane Reversal

PCIe system supports lane reversal capabilities and therefore provides flexibility in the design of the board. It is possible to choose to lay out the board with reversed lane numbers and the PCIe EP continues to link train successfully and operate normally. The SERDESIF Configurator in Libero allows configuration of the SERDESIF block in reverse lane PCIe mode, as shown in the following figure. Using lane reversal allows the PCIE logical lane to be remapped to its respective physical lane. The following figure shows that using this reversal assists with routing the PCB and permits a cleaner interface between the PCIE host and Endpoint. Polarity swapping or inversion capability within a PCIE receive or transmit lane is not available in SmartFusion2 and IGLOO2.

Figure 30 • **PCIe Protocol Mode Setting in Reverse Lane Mode**



Polarity swapping or inversion capability within a PCIE receive or transmit lane is not available in SmartFusion2 and IGLOO2.

3.7.7 PCIe Power Management

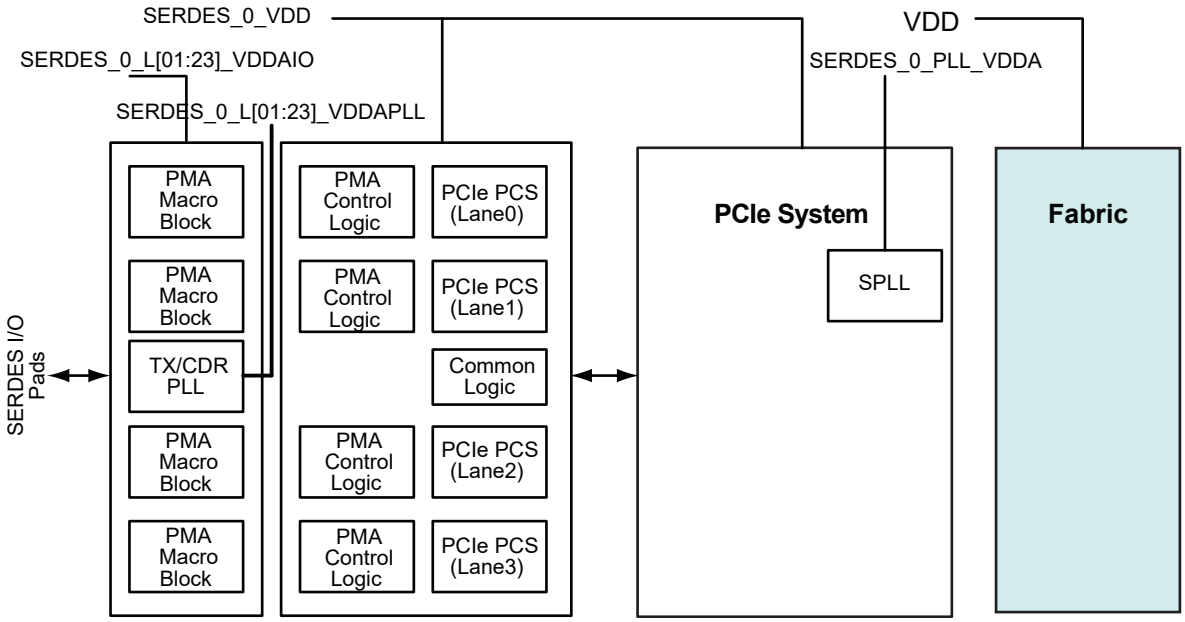
This section describes the power management scheme in the FPGA PCIe implementation. This has the following sub-sections:

- Power Domain Implementation
- Legacy Power Management
- PCIe Power Management

3.7.7.1 Power Domain Implementation

In SmartFusion2 and IGLOO2 devices, the FPGA and PCIe link (including PMA, PCS, and PCIe controller) are combined in a single chip, so they have separate power supplies. The following figure shows the SmartFusion2 and IGLOO2 power rails. Refer to the *AC393: SmartFusion2 and IGLOO2 Board Design Guidelines Application Note* for detailed power supply connections.

Figure 31 • **SmartFusion2/IGLOO2 Power Supply to the PCIe Link Implementation**



Note: Tie VDD and VDDAIO to the same 1.2 V source so they power up at the same time. The filters specified in the *AC393: SmartFusion2 and IGLOO2 Board Design Guidelines Application Note* for each supply are still required/recommended.

3.7.7.2 Legacy Power Management

The PCIe bridge register space defines the capabilities of the PCIe bridge in term of legacy power management (PME support, auxiliary current requirement and so on). The power management control and status register also contain the current power management state. The PM data and PM scale value array can define the power consumed in each power state. Refer to the Bridge Register Space, page 48 for more information.

3.7.7.3 PCIe Power Management

PCIe active state power management (ASPM) defines link power management states that a PCIe physical link is permitted to enter in response to software-driven D-state transitions or active state link power management activities.

The PCIe protocol defines the following low power link states.

Table 19 • **PCIe Low Power States**

Low Power State	Description
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.
L2/L3 ready	This state prepares the PCIe link for the removal of main power and the reference clock.

PCIe EP implementation supports L0, L1, and a special version of L2.

3.7.8 PCIe Interrupts for Endpoints

The IGLOO2 PCIe EP implementation supports 32 MSI interrupt and INTx interrupts. It cannot support both at the same time. The user can select which interrupt model to use in the High Speed Serial Interfaces Configurator. For MSI, the user has a selection of up to 32 MSI vectors. When using MSI, the first 4 interrupts can be sent using the PCIE_INTERRUPTS[3:0] port of the SERDESIF[0] for MSI0, [1] for MSI1, and so on. With respect to PCIe HDL, PCIe_Interrupt[0] has higher priority than PCIe_Interrupt[1]. When interrupts are configured as INTx, controller always send INTx message packet with message code Assert_INTA or Deassert_INTA. To send more than 4 interrupts the user must use the AXI3 Slave interface and send a memory write transaction to the specific address set by the root complex during interrupt negotiation.

To use more than four interrupts, the MSI address/data from the MSI capability structure requires the PCIe driver to gather information and write to a BAR space register in the fabric. You can use this data to perform AXI3 writes for MSI generation. The host driver needs to read the configuration space of the SmartFusion2/IGLOO2 endpoint.

The MSI capability structure contains the number of negotiated MSIs and the address location for the memory write. The host driver needs to write this information into a BAR space register in the endpoint fabric. The information in the fabric can be used to generate memory write TLPs to the correct MSI address to issue an interrupt.

3.7.9 ECRC Handling

The ECRC ensures end-to-end data integrity. The PCIe implementation transmits a TLP with ECRC from the transmit port of the application layer. When using ECRC forwarding mode, the ECRC check and generate are done in the application layer. The AER_ECRC_CAPABILITY register in bridge configuration registers sets the ECRC settings.

3.8 Bridge Register Space

The PCIe core bridge register space is used to configure the PCIe core settings at power-up and is handled by the CoreConfigP as part of the HPMS module. CoreConfig IP module facilitates configuration of the SERDESIF block in an SmartFusion2 or IGLOO2 device. For more details refer to the *UG0448: IGLOO2 FPGA High Performance Memory Subsystem User Guide*. These registers are 32 bits wide and part of the SERDESIF system register. Refer to the SERDESIF System Register, page 153 in the SERDESIF Register Access Map, page 150.

The PCIe system block registers consist of:

- Read-only registers that report control and status registers to the AXI3 side through the APB bus
- Bridge settings that must be configured at power-up, such as local interrupt mapping to MSI and test mode

- Control/status registers that can be used by the AXI3 bus to control bridge behavior during an operation

Most bridge registers are hardwired to a fixed value.

These registers are described in the next section according to their function:

- Information Registers: provide device, system, and bridge identification information (see Information Registers, page 50).
- Bridge Configuration Registers: enable configuration of bridge functionality (see Bridge Configuration Registers, page 50).
- Power Management Registers: enable configuration of the power management capabilities of the bridge (see Power Management Registers, page 52).
- Address Mapping Registers: provide address mapping for AXI3 master and slave windows. These windows are used for address translation (see Address Mapping Registers, page 53).
- EP Interrupt Registers: used in EP mode to manage interrupts (see EP Interrupt Registers, page 54).
- PCIe Control and Status Registers: read-only registers enable the local processor to check useful information related to the PCIe interface status. This enables the local processor to detect when the bridge's PCIe interface is initialized and to monitor PCI link events (see PCIe Control and Status Registers, page 54).

3.8.1 Register Initialization

The registers contained in the SERDESIF are initialized automatically when the device powers-up using data stored in non-volatile storage in the device. There are two types of initialization that are used to set the value in the registers.

3.8.2 Flash Bits

There are several flash bits that are associated with each SERDESIF block. These flash bits contain the settings for registers that need to be initialized quickly when the device powers up such as PLL and clock configurations, PCIe configuration space, and resets. The flash bits are set by the Libero configuration GUI based on the user selections and are statically set at device power-up.

3.8.3 APB

The SERDESIF supports an APB slave interface connected to the fabric interface. When the Libero System Builder assembles the SERDESIF supporting modules the APB needs to be connected to the HPMS module's APB master port. After the device powers up the HPMS and supporting modules initializes the necessary registers in the SERDESIF that use the APB interface.

It is possible that the APB initialization overwrites registers that have been initialized by the flash bits since the APB is written after the flash bit value has been loaded.

3.8.4 Fixed

These registers are read-only registers and there value is fixed based on the implementation of the device.

3.8.5 Information Registers

The registers listed in the following table provide device, system, and bridge identification information. Refer to SERDESIF Register Access Map, page 150 for further register details.

Table 20 • **Information Registers**

Register Name	Address Offset	Register Type	Initialization	Description
PCIE_VID_DEVID	000h	RO	Flash	Identifies the manufacturer of the device or application. Refer to the PCIe specification for details (see Table 26, page 55).
PCIE_CLASS_CODE_REG	008h	RO	Flash	Identifies the manufacturer of the device or application. Refer to the PCIe specification for details (see Table 28, page 55).
PCIE_CAPTURED_BUS_DEVICE_NB	03Ch	RO	NA	Reports the bus and device number of the EP device for each configuration write TLP received (see Table 39, page 58).
SUBSYSTEM_ID	02Ch	RO	Flash	Identifies the manufacturer of the device or application. Refer to the PCIe specification for details (see Table 35, page 57).
PCIE_INFO	16Ch	RO	NA	Reports the bridge version (see Table 94, page 74).

3.8.6 Bridge Configuration Registers

The registers listed in the following table enable to configure bridge functionality.

Table 21 • **Bridge Configuration Registers**

Register Name	Byte Offset	State	Initialization	Description
PCIE_PCIE_CONFIG	204h	RO	Flash	Sets the PCIe configuration (see Table 95, page 74).
BAR0	010h	R/W	Flash	Defines the type and size of BAR0 of the PCIe native endpoint. This register combines with BAR1 for defining the type and size of BAR01 of the PCIe native endpoint (see Table 29, page 55).
BAR1	014h	R/W	Flash	Defines the type and size of BAR1 of the PCIe native endpoint. This register combines with BAR0 for defining the type and size of BAR01 of the PCIe native endpoint (see Table 30, page 56).
BAR2	01 8h	R/W	Flash	Defines the type and size of BAR2 of the PCIe native endpoint. This register combines with BAR3 for defining the type and size of BAR23 of the PCIe native endpoint (see Table 31, page 56).
BAR3	01Ch	R/W	Flash	Defines the type and size of BAR3 of the PCIe native endpoint. This register combines with BAR2 for defining the type and size of BAR23 of the PCIe native endpoint (see Table 32, page 56).

Table 21 • **Bridge Configuration Registers *(continued)***

Register Name	Byte Offset	State	Initialization	Description
BAR4	020h	R/W	Flash	Defines the type and size of BAR4 of the PCIe native endpoint. This register combines with BAR5 for defining the type and size of BAR45 of the PCIe native endpoint (see Table 33, page 56).
BAR5	024h	R/W	Flash	Defines the type and size of BAR5 of the PCIe native endpoint. This register combines with BAR4 for defining the type and size of BAR45 of the PCIe native endpoint (see Table 34, page 57).
PCIE_AER_ECRC_CAPABILITY	050h	R/W	APB	Defines whether the bridge supports AER and ECRC generation/check and whether AER/ECRC is implemented. ECRC generation and check bits can only be set if AER is implemented (see Table 44, page 60).
MAX_PAYLOAD_SIZE	058h	RO	Fixed	Negotiated maximum payload size (see Table 46, page 61).
PCIE_CREDIT_ALLOCATION_0	0B0h	RO	Fixed	Provides the initial credit values for posted transactions (see Table 59, page 66).
PCIE_CREDIT_ALLOCATION_1	0B4h	RO	Fixed	Provides the initial credit values for non-posted transactions (see Table 60, page 66).
PCIE_ERROR_COUNTER_0	0A0h	R/W	NA	Has four 8-bit counters for the four error sources. To clear the register content, the bridge must perform a write transaction (any value) to this register (see Table 55, page 65). This error counter saturates when it reaches maximum value.
PCIE_ERROR_COUNTER_1	0A4h	R/W	NA	Has four 8-bit counters for the four error sources. To clear the register content, the bridge must perform a write transaction (any value) to this register (see Table 56, page 65). This error counter saturates when it reaches maximum value.
PCIE_ERROR_COUNTER_2	0A8h	R/W	NA	Has four 8-bit counters for the four error sources. To clear the register content, the bridge must perform a write transaction (any value) to this register (see Table 57, page 65). This error counter saturates when it reaches maximum value.
PCIE_ERROR_COUNTER_3	0ACh	R/W	NA	Has four 8-bit counters for the four error sources. To clear the register content, the bridge must perform a write transaction (any value) to this register (see Table 58, page 66). This error counter saturates when it reaches maximum value.

3.8.7 Power Management Registers

The registers listed in the following table enable to configure the power management capabilities of the bridge.

Table 22 • **Bridge Power Management Registers**

Register Name	Byte Offset	State	Description
PCIE_LTSSM	044h	R	Can be used to monitor the core state or to select a specific test mode on bits [31:16] and to control L2 entry on bits [15:0] (see Table 41, page 59).
PCIE_POWER_MGT_CAPABILITY	048h	R/W	Enables the local processor to configure power management capability (see Table 42, page 60).
PCIE_PM_DATA_SCALE_0	070h	R/W	There are four PM data and scale registers that define the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field (see Table 50, page 62, Table 51, page 63, Table 52, page 63, and Table 53, page 64).
PCIE_PM_DATA_SCALE_1	074h	R/W	
PCIE_PM_DATA_SCALE_2	078h	R/W	
PCIE_PM_DATA_SCALE_3	07Ch	R/W	
PCIE_ASPM_L0S_CAPABILITY	060h	R/W	Defines the EP L0s acceptable latency and the number of fast training sequences (FTS) required by the SERDES to resynchronize its receiver on incoming data, depending on clock mode configuration (separated clock or common clock). The number of FTS required in separated clock mode must be higher than that required in common clock mode. The bridge automatically computes the ASPM L0s exit latency based on these two register values, and on the maximum payload size of the control register. The selected NFTS field is that transmitted by the link training and status state machine (LTSSM) to the opposite component in order to define the number of FTS that the opposite component must send to be sure that the device receiver has re-locked onto incoming data (see Table 47, page 61).
PCIE_ASPM_L0S_GEN2	260h	R/W	Defines the EP L0s acceptable latency and the number of FTS required by the SERDES to resynchronize its receiver on incoming data, depending on clock mode configuration (separate clock or common clock) at 5.0 Gbps. The number of FTS required in separated clock mode must be higher than that required in common clock mode. The bridge automatically computes the ASPM L0s exit latency based on these two register values and the maximum payload size of the control register. The selected NFTS field is that transmitted by the LTSSM to the opposite component in order to define the number of FTS that the opposite component must send to be sure that the device receiver has re-locked onto incoming data (see Table 98, page 75).
PCIE_ASPM_L1_CAPABILITY	064h	R/W	Defines the EP L1 acceptable latency and the number of FTS required. The EP L1 acceptable latency is used to enable or disable ASPM L1 entry by comparing its value to the maximum ASPM L1 exit latency of all components in the hierarchy (plus 1 microsecond per switch). If the ASPM L1 acceptable latency is lower than the maximum ASPM L1 exit latency, ASPM L1 entry is not enabled (see Table 48, page 61).
PCIE_TIMEOUT_COMPLETION	068h	R/W	Defines four timeout ranges for the completion timeout mechanism (see Table 49, page 62).

3.8.8 Address Mapping Registers

The registers listed in the following table provide the address mapping for AXI3 master and slave windows. These windows are used for address translation.

Table 23 • **Address Mapping Registers**

Register Name	Address Offset	Register Type	Description
PCIE_AXI_SLAVE_WINDOW0_0	0C0h	R/W	There are four register sets that define the address mapping for AXI3 slave window 0 (see Table 62, page 67, Table 63, page 67, Table 64, page 67, and Table 65, page 68).
PCIE_AXI_SLAVE_WINDOW0_1	0C4h		
PCIE_AXI_SLAVE_WINDOW0_2	0C8h		
PCIE_AXI_SLAVE_WINDOW0_3	0CCh		
PCIE_AXI_SLAVE_WINDOW1_0	0D0h	R/W	There are four register sets that define the address mapping for AXI3 slave window 1 (see Table 66, page 68, Table 67, page 68, Table 68, page 68, and Table 69, page 68).
PCIE_AXI_SLAVE_WINDOW1_1	0D4h		
PCIE_AXI_SLAVE_WINDOW1_2	0D8h		
PCIE_AXI_SLAVE_WINDOW1_3	0DCh		
PCIE_AXI_SLAVE_WINDOW2_0	0E0h	R/W	There are four register sets that define the address mapping for AXI3 slave window 2 (see Table 70, page 69, Table 71, page 69, Table 72, page 69, and Table 73, page 69).
PCIE_AXI_SLAVE_WINDOW2_1	0E4h		
PCIE_AXI_SLAVE_WINDOW2_2	0E8h		
PCIE_AXI_SLAVE_WINDOW2_3	0ECh		
PCIE_AXI_SLAVE_WINDOW3_0	0F0h	R/W	There are four register sets that define the address mapping for AXI3 slave window 3 (see Table 74, page 69, Table 75, page 70, Table 76, page 70, and Table 77, page 70).
PCIE_AXI_SLAVE_WINDOW3_1	0F4h		
PCIE_AXI_SLAVE_WINDOW3_2	0F8h		
PCIE_AXI_SLAVE_WINDOW3_3	0FCh		
PCIE_AXI_MASTER_WINDOW0_0	100h	R/W	There are four register sets that define the address mapping for AXI3 master window 0 (see Table 78, page 70, Table 79, page 70, Table 80, page 71, and Table 81, page 71).
PCIE_AXI_MASTER_WINDOW0_1	104h		
PCIE_AXI_MASTER_WINDOW0_2	108h		
PCIE_AXI_MASTER_WINDOW0_3	10Ch		
PCIE_AXI_MASTER_WINDOW1_0	110h	R/W	There are four register sets that define the address mapping for AXI3 master window 1 (see Table 82, page 71, Table 83, page 71, Table 84, page 72, and Table 85, page 72).
PCIE_AXI_MASTER_WINDOW1_1	114h		
PCIE_AXI_MASTER_WINDOW1_2	118h		
PCIE_AXI_MASTER_WINDOW1_3	11Ch		
PCIE_AXI_MASTER_WINDOW2_0	120h	R/W	There are four register sets that define the address mapping for AXI3 master window 2 (see Table 86, page 72, Table 87, page 72, Table 88, page 73, and Table 89, page 73).
PCIE_AXI_MASTER_WINDOW2_1	124h		
PCIE_AXI_MASTER_WINDOW2_2	128h		
PCIE_AXI_MASTER_WINDOW2_3	12Ch		
PCIE_AXI_MASTER_WINDOW3_0	130h	R/W	There are four register sets that define the address mapping for AXI3 master window 3 (see Table 90, page 73, Table 91, page 73, Table 92, page 74, and Table 93, page 74).
PCIE_AXI_MASTER_WINDOW3_1	134h		
PCIE_AXI_MASTER_WINDOW3_2	138h		
PCIE_AXI_MASTER_WINDOW3_3	13Ch		

3.8.9 EP Interrupt Registers

The PCIe IP core can generate interrupts through the input signal. This signal may be required by a device in order to interrupt the host processor, call its device drivers, or report application layer-specific events or errors. The parameter of the bridge defines the number of interrupt bits for this signal.

Table 24 • **EP Interrupt Registers**

Register Name	Address Offset	Register Type	Description
PCIE_MSI_0	080h	R/W	Defines 8 MSI_MAP0 registers, with up to 32 possible MSI messages. 32 possible MSI messages (see Table 54, page 64).
MSI_CTRL_STATUS	040h	R/W	This register sets MSI control and status. All bits are RO except the number of MSI requested and the multiple message enable fields, which are R/W. Up to 32 MSI messages can be requested by the device, although the PCI software can allocate less than the number of MSI requested. This information can be read by the local processor through the multiple message enable field of the register (see Table 40, page 58).

3.8.10 PCIe Control and Status Registers

The following registers, as shown in the following table, are read-only registers that enable the local processor to check useful information related to the PCIe interface status, such as when the PCIe interface is initialized and monitoring of PCI link events. A complete description of these registers can be found in the PCIe specifications.

Table 25 • **PCIe Control and Status Registers**

Register Name	Address Offset	Register Type	Description
CFG_PRMSCR	004h	RO	The command and status register of PCI configuration space (see Table 27, page 55).
PCIE_DEVSCR	030h	RO	Reports the current value of the PCIe device control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system (see Table 36, page 57).
PCIE_DEV2SCR	230h	RO	Reports the current value of the PCIe device control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system. This register is used when link speed is set to 5.0 Gbps (see Table 96, page 75).
PCIE_LINKSCR	034h	RO	Reports the current value of the PCIe link control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system (see Table 37, page 57).
PCIE_LINK2SCR	234h	RO	Reports the current value of the PCIe link control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system. This register is used when link speed is set to 5.0 Gbps (see Table 97, page 75).
CFG_PRMSCR	04Ch	RO	Reports the current values of the XpressRich2 core's power management control status register (see Table 27, page 55).
PCIE_SLOTCAP	154h		Reserved
PCIE_SLOTCSR	158h		Reserved
PCIE_ROOTCSR	15Ch		Reserved

3.8.11 PCIe Bridge Registers

The following sub-section describes all PCIe bridge registers in detail.

3.8.11.1 PCIE_VID_DEVID Register (000h)

Table 26 • **PCIE_VID_DEVID**

Bit Number	Name	Reset Value	Description
[31:16]	Device ID	0x11AA	Identifies the manufacturer of the device or application. The values are assigned by PCI-SIG. The default value, 11AA, is the Vendor ID for Microsemi.
[15:0]	Vendor ID	0x1556	The field Identifies the manufacturer of the device or application. The values are assigned by PCI-SIG. The default value, 1556, is the Vendor ID for Microsemi.

3.8.11.2 PCIE_CFG_PRMSCR Register (004h)

Table 27 • **CFG_PRMSCR**

Bit Number	Name	Reset Value	Description
[31:0]	Class Code	0x00100000	The command and status register of PCI configuration space.

3.8.11.3 PCIE_CLASS_CODE Register (008h)

Table 28 • **PCIE_CLASS_CODE_REG**

Bit Number	Name	Reset Value	Description
[31:16]	PCIE_CLASS_CODE	0x0000	Identifies the manufacturer of the device or application. The values are assigned by PCI-SIG.
[15:0]	RESERVED0	0x0000	Identifies the manufacturer of the device or application. The values are assigned by PCI-SIG.

3.8.11.4 BAR0 Register (010h)

Table 29 • **BAR0**

Bit Number	Name	Reset Value	Description
[31:4]	BAR0_31_4	0x000000	Defines the type and size of BAR0 of the PCIe native endpoint.
3	BAR0_3	0x1	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR0_2_1	0x10	Set to '00' to indicate anywhere in 32-bit address space.
0	BAR0_0	0x0	Memory space indicator

3.8.11.5 BAR1 Register (014h)

Table 30 • **BAR1**

Bit Number	Name	Reset Value	Description
[31:4]	BAR1_31_4	0x000000	Defines the type and size of BAR1 of the PCIe native endpoint.
3	BAR1_3	0x0	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR1_2_1	0x00	Set to '00' to indicate anywhere in 32-bit address space.
0	BAR1_0	0x0	Memory space indicator.

3.8.11.6 BAR2 Register (018h)

Table 31 • **BAR2**

Bit Number	Name	Reset Value	Description
[31:4]	BAR2_31_4	0x000000	The register defines the type and size of BAR0 of the PCIe native EP
3	BAR2_3	0x1	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR2_2_1	0x10	Set to '00' to indicate anywhere in 32-bit address space.
0	BAR2_0	0x0	Memory space indicator.

3.8.11.7 BAR3 Register (01Ch)

Table 32 • **BAR3**

Bit Number	Name	Reset Value	Description
[31:4]	BAR3_31_4	0x000000	The register defines the type and size of BAR1 of the PCIe native EP.
3	BAR3_3	0x0	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR3_2_1	0x00	Set to '00' to indicate anywhere in 32-bit address space.
0	BAR3_0	0x0	Memory space indicator.

3.8.11.8 BAR4 Register (020h)

Table 33 • **BAR4**

Bit Number	Name	Reset Value	Description
[31:4]	BAR4_31_4	0x000000	The register defines the type and size of BAR0 of the PCIe native EP.
3	BAR4_3	0x1	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR4_2_1	0x10	Set to '00' to indicate anywhere in 32-bit address space.
0	BAR4_0	0x0	Memory space indicator.

3.8.11.9 BAR5 Register (024h)

Table 34 • **BAR5**

Bit Number	Name	Reset Value	Description
[31:4]	BAR5_31_4	0x000000	The register defines the type and size of BAR1 of the PCIe native EP.
3	BAR5_3	0x0	Identifies the ability of the memory space to be prefetched.
[2:1]	BAR5_2_1	0x00	Set to '00' to indicate anywhere in 32-bit address space.
0	BAR5_0	0x0	Memory space indicator.

3.8.11.10 SUBSYSTEM_ID Register (02Ch)

Table 35 • **SUBSYSTEM_ID**

Bit Number	Name	Reset Value	Description
[31:16]	SUBSYSTEM_ID	0x11AA	This field further qualifies the manufacturer of the device or application. This value is typically the same as the Device ID.
[15:0]	SUBSYSTEM_VENDOR_ID	0x1556	This field further qualifies the manufacturer of the device or application.

3.8.11.11 PCIE_DEVSCR Register (030h)

Table 36 • **PCIE_DEVSCR**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_DEVSCR	0x00000000	Device control and status: This register reports the current value of the PCIe device control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system.

Note: This register is READ_ONLY.

3.8.11.12 PCIE_LINKSCR Register (034h)

Table 37 • **PCIE_LINKSCR**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_LINKSCR	0x00000050	This register reports the current value of the PCIe link control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system.

3.8.11.13 TC_VC_MAPPING Register (038h)

Table 38 • **TC_VC_MAPPING**

Bit Number	Name	Reset Value	Description
[31:24]	TC_VC_MAPPING_31_24	0x0	Reserved
[23:21]	TC_VC_MAPPING_23_21	0x0	Mapping for TC7
[20:18]	TC_VC_MAPPING_20_18	0x0	Mapping for TC6
[17:15]	TC_VC_MAPPING_17_15	0x0	Mapping for TC5
[14:12]	TC_VC_MAPPING_14_12	0x0	Mapping for TC4
[11:9]	TC_VC_MAPPING_11_9	0x0	Mapping for TC3
[8:6]	TC_VC_MAPPING_8_6	0x0	Mapping for TC2
[5:3]	TC_VC_MAPPING_5_3	0x0	Mapping for TC1
[2:0]	TC_VC_MAPPING_2_0	0x0	Mapping for TC0 (always 0)

3.8.11.14 PCIE_CAPTURED_BUS_DEVICE_NB Register (03Ch)

Table 39 • **PCIE_CAPTURED_BUS_DEVICE_NB**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_CAPTURED_BUS_DEVICE_NB	0x0	This register reports the bus and device number of the EP device for each configuration write TLP received.

3.8.11.15 MSI_CTRL_STATUS Register (040h)

Table 40 • **MSI_CTRL_STATUS**

Bit Number	Name	Reset Value	Description
[31:24]	MSI_CTRL_STATUS_31_24	0x0	These RO bits are hardwired to 00000000.
23	MSI_CTRL_STATUS_23	0x0	This RO bit is hardwired to 1.
[22:20]	MSI_CTRL_STATUS_22_20	0x0	Multiple message enable. Fabric logic/MSS checks this RO APB register to see how many MSI interrupt resources are allocated from the host side.
[19:17]	MSI_CTRL_STATUS_19_17	0x0	Number of MSI requested. Fabric logic/MSS writes this RW APB register to request the number of MSI interrupt resources needed from the host.
16	MSI_CTRL_STATUS_16	0x0	MSI is enabled. Fabric logic/MSS checks this RO APB register to see if host has enabled MSI on the PCIe bus.
[15:0]	MSI_CTRL_STATUS_15:0	0x0	This bits are hardwired to 0x7805.

3.8.11.16 LTSSM Register (044h)

Table 41 • **PCIE_LTSSM**

Bit Number	Name	Reset Value	Description
[31:29]	LTSSM_31_29		Reserved
[28:24]	LTSSM_28_24	0x0	These bits set LTSSM state encoding (RO): 00000: Detect.quiet 00001: Detect.active 00010: Polling.active 00011: Polling.compliance 00100: Polling.configuration 00101: Reserved (ex polling.speed) 00110: Configuration.linkwidth.start 00111: Configuration.linkwidth.accept 01000: Configuration.lanenum.accept 01001: Configuration.lanenum.wait 01010: Configuration.complete 01011: Configuration.idle 01100: Recovery.RcvrLock 01101: Recovery.RcvrCfg 01110: Recovery.idle 01111: L0 10000: Disabled 10001: Loopback.entry 10010: Loopback.active 10011: Loopback.exit 10100: Hot reset 10101: L0s (transmit) 10110: L1.entry 10111: L1.Idle 11000: L2.idle 11001: L2.TransmitWake 11010: Recovery.speed 11011 - 11111: Reserved
[23:20]	LTSSM_23_20	0x0	Reserved
19	LTSSM_19	0x0	Forces compliance pattern (R/W).
18	LTSSM_18	0x0	Fully disables power management (R/W).
17	LTSSM_17	0x0	Sets master loopback (R/W).
16	LTSSM_16	0x0	Disables scrambling (R/W).
[15:2]	LTSSM_15_2	0x0	Reserved
1	LTSSM_1	0x0	Indicates if PME_TURN_OFF was received (RO).
0	LTSSM_0	0x0	Acknowledges PME_TURN_OFF (R/W).

3.8.11.17 PCIE_POWER_MGT_CAPABILITY Register (048h)

Table 42 • **PCIE_POWER_MGT_CAPABILITY**

Bit Number	Name	Reset Value	Description
[31:27]	POWER_MGT_CAPABILITY_31_27	0x0	These bits set PME support.
26	POWER_MGT_CAPABILITY_26	0x0	Sets D2 support. If this field is cleared, PME_SUPPORT bit 29 must also be cleared.
25	POWER_MGT_CAPABILITY_25	0x0	Sets D1 support. If this field is cleared, PME_SUPPORT bit 28 must also be cleared.
[24:22]	POWER_MGT_CAPABILITY_24_22	0x0	These bits set maximum current required.
21	POWER_MGT_CAPABILITY_21	0x0	Sets device specification initialization.
[20:19]	POWER_MGT_CAPABILITY_20_19	0x0	Reserved
18	POWER_MGT_CAPABILITY_18	0x0	Sets PCI power management interface specification version; hardwired to 011b (PCIe Spec. v1.1)
[17:0]	POWER_MGT_CAPABILITY_17_0	0x0	Reserved

3.8.11.18 PCIE_CFG_PMSCR Register (04Ch)

Table 43 • **PCIE_CFG_PMSCR**

Bit Number	Name	Reset Value	Description
31:0	CFG_PMSCR	0x0	Reports the current values of the PCIe IP core's power management control status register.

3.8.11.19 PCIE_AER_ECRC_CAPABILITY Register (050h)

Table 44 • **PCIE_AER_ECRC_CAPABILITY**

Bit Number	Name	Reset Value	Description
[31:3]	AER_ECRC_CAPABILITY_31_3		Reserved
2	AER_ECRC_CAPABILITY_2	0x0	Defines whether advanced error reporting (AER) is implemented or not.
1	AER_ECRC_CAPABILITY_1	0x0	Defines ECRC generation.
0	AER_ECRC_CAPABILITY_0	0x0	Defines ECRC check.

3.8.11.20 PCIE_VC1_CAPABILITY Register (054h)

Table 45 • **PCIE_VC1_CAPABILITY**

Bit Number	Name	Reset Value	Description
[31:0]			Reserved

3.8.11.21 PCIE_MAX_PAYLOAD_SIZE Register (058h)

Table 46 • **MAX_PAYLOAD_SIZE**

Bit Number	Name	Reset Value	Description
[31:3]			Reserved
2:0	MAX_PAYLOAD_SIZE_2_0	0x0	Negotiated max payload size. The EP sets its own max payload size to 2 KB: 000: 128 bytes 001: 256 bytes (RESERVED) 010: 512 bytes (RESERVED) 011: 1 Kbytes (RESERVED) 100: 2 Kbytes (RESERVED)

3.8.11.22 PCIE_ASPM_L0S_CAPABILITY Register (060h)

Table 47 • **PCIE_ASPM_L0S_CAPABILITY**

Bit Number	Name	Reset Value	Description
[31:24]	ASPM_L0S_CAPABILITY _31_24	0x0	NFTS_COMCLK in common clock mode
[23:16]	ASPM_L0S_CAPABILITY _23_16	0x0	NFTS_SPCLK in separated clock mode
[15:10]	ASPM_L0S_CAPABILITY _15_10	0x0	Reserved
[9:7]	ASPM_L0S_CAPABILITY _9_7	0x0	L0s exit latency for separate clock
[6:4]	ASPM_L0S_CAPABILITY _6_4	0x0	L0s exit latency for common clock
[3:1]	ASPM_L0S_CAPABILITY _3_1	0x0	EP L0s acceptable latency
0	ASPM_L0S_CAPABILITY _0	0x0	Reserved

3.8.11.23 PCIE_ASPM_L1_CAPABILITY Register (064h)

Table 48 • **PCIE_ASPM_L1_CAPABILITY**

Bit Number	Name	Reset Value	Description
[31:27]	ASPM_L1_CAPABILITY_31_27	0x0	Reserved
[26:24]	ASPM_L1_CAPABILITY_26_24	0x0	L1 exit latency common clock
[23:19]	ASPM_L1_CAPABILITY_23_19	0x0	Reserved
[18:16]	ASPM_L1_CAPABILITY_18_16	0x0	L1 exit latency separated clock mode: this value must be higher than for common clock mode
[15:4]	ASPM_L1_CAPABILITY_15_4	0x0	Reserved
[3:1]	ASPM_L1_CAPABILITY_3_1	0x0	Endpoint L1 acceptable latency
[0]	ASPM_L1_CAPABILITY_0	0x0	ASPM L1 support: If this field is not set (no ASPM L1 support), all other fields must be set to 0. ASPM L1 is mandatory for ExpressCard devices.

3.8.11.24 PCIE_TIMEOUT_COMPLETION Register (068Ch)

Table 49 • **PCIE_TIMEOUT_COMPLETION**

Bit Number	Name	Reset Value	Description
[31:4]	TIMEOUT_COMPLETION_31_4		Reserved
[3:0]	TIMEOUT_COMPLETION_3_0	0x0	Completion Timeout Ranges Supported – This field indicates device Function support for the optional Completion Timeout programmability mechanism. This mechanism allows system software to modify the Completion Timeout value. Four time value ranges are defined: Range A: 50 µs to 10 ms Range B: 10 ms to 250 ms Range C: 250 ms to 4 s Range D: 4 s to 64 s Bits are set according to the table below to show timeout value ranges supported. 0000b Completion Timeout programming not supported. –0000b is default setting. An error is not be produced if a completion does not come back. 0001b Range A 0010b Range B 0011b Ranges A and B 0110b Ranges B and C 0111b Ranges A, B, and C 1110b Ranges B, C and D 1111b Ranges A, B, C, and D All other values are reserved.

3.8.11.25 PCIE_PM_DATA_SCALE_0 Register (070h)

Table 50 • **PCIE_PM_DATA_SCALE_0**

Bit Number	Name	Reset Value	Description
[31:24]	PM_DATA_SCALE_0_31_24	0x0	Set the register that defines Data3 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[23:16]	PM_DATA_SCALE_0_23_16	0x0	Set the register that defines Data2 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[15:8]	PM_DATA_SCALE_0_15_8	0x0	Set the register that defines Data1 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[7:0]	PM_DATA_SCALE_0_7_0	0x0	Set the register that defines Data0 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.

3.8.11.26 PCIE_PM_DATA_SCALE_1 Register (074h)

Table 51 • **PCIE_PM_DATA_SCALE_1**

Bit Number	Name	Reset Value	Description
[31:24]	PM_DATA_SCALE_1_31_24	0x0	Set the register that defines Data7 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[23:16]	PM_DATA_SCALE_1_23_16	0x0	Set the register that defines Data6 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[5:8]	PM_DATA_SCALE_1_15_8	0x0	Set the register that defines Data5 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[7:0]	PM_DATA_SCALE_1_7_0	0x0	Set the register that defines Data4 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.

3.8.11.27 PCIE_PM_DATA_SCALE_2 Register (078h)

Table 52 • **PCIE_PM_DATA_SCALE_2**

Bit Number	Name	Reset Value	Description
[31:26]	PM_DATA_SCALE_2_31_26		Reserved
[25:24]	PM_DATA_SCALE_2_25_24	0x0	Set the register that defines data scale 3 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[23:18]	PM_DATA_SCALE_2_23_18		Reserved
[17:16]	PM_DATA_SCALE_2_17_16	0x0	Set the register that defines data scale 2 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[15:10]	PM_DATA_SCALE_2_15_10		Reserved
[9:8]	PM_DATA_SCALE_2_9_8	0x0	Set the register that defines data scale 1 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[7:2]	PM_DATA_SCALE_2_7_2		Reserved
[1:0]	PM_DATA_SCALE_2_1_0	0x0	Set the register that defines data scale 0 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.

3.8.11.28 PCIE_PM_DATA_SCALE_3 Register (07Ch)

Table 53 • **PCIE_PM_DATA_SCALE_3**

Bit Number	Name	Reset Value	Description
[31:26]	PM_DATA_SCALE_3_31_26		Reserved
[25:24]	PM_DATA_SCALE_3_25_24	0x0	Set the register that defines data scale 7 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[23:18]	PM_DATA_SCALE_3_23_18		Reserved
[17:16]	PM_DATA_SCALE_3_17_16	0x0	Set the register that defines data scale 6 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[15:10]	PM_DATA_SCALE_3_15_10		Reserved
[9:8]	PM_DATA_SCALE_3_9_8	0x0	Set the register that defines data scale 5 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.
[7:2]	PM_DATA_SCALE_3_7_2		Reserved
[1:0]	PM_DATA_SCALE_3_1_0	0x0	Set the register that defines data scale 4 of the PM data value of the device for each possible power state defined by the PCI power management specification, used in conjunction with the PM scale field.

3.8.11.29 PCIE_MSI_0 Register (080h)

Table 54 • **PCIE_MSI_0**

Bit Number	Name	Reset Value	Description
[31:27]	MSI0_31_27	0x0	These bits set MSI_Offset[4] of MSI_MAP0.
[26:24]	MSI0_26_24	0x0	These bits set MSI_TC[4] of MSI_MAP0.
[23:19]	MSI0_23_19	0x0	These bits set MSI_Offset[3] of MSI_MAP0.
[18:16]	MSI0_18_16	0x0	These bits set MSI_TC[3] of MSI_MAP0.
[15:11]	MSI0_15_11	0x0	These bits set MSI_Offset[2] of MSI_MAP0.
[10:8]	MSI0_10_8	0x0	These bits set MSI_TC[2] of MSI_MAP0.
[7:3]	MSI0_7_3	0x0	These bits set MSI_Offset[1] of MSI_MAP0.
[2:0]	MSI0_2_0	0x0	These bits set MSI_TC[1] of MSI_MAP0.

3.8.11.30 PCIE_ERROR_COUNTER_0 Register (0A0h)

Table 55 • **PCIE_ERROR_COUNTER_0**

Bit Number	Name	Reset Value	Description
[31:24]	ERROR_COUNTER0_31_24	0x0	8-bit counter that reports the following error source: A3: DLLP error
[23:16]	ERROR_COUNTER0_23_16	0x0	8-bit counter that reports the following error source: A2: TLP error
[15:8]	ERROR_COUNTER0_15_8	0x0	8-bit counter that reports the following error source: A1: Training error (not supported)
[7:0]	ERROR_COUNTER0_7_0	0x0	8-bit counter that reports the following error source: A0: Receiver port error

3.8.11.31 PCIE_ERROR_COUNTER_1 Register (0A4h)

Table 56 • **PCIE_ERROR_COUNTER_1**

Bit Number	Name	Reset Value	Description
[31:24]	ERROR_COUNTER1_31_24	0x0	8-bit counter that reports the following error source: A7: Poisoned TLP received error
[23:16]	ERROR_COUNTER1_23_16	0x0	8-bit counter that reports the following error source: A6: Data link layer protocol error
[15:8]	ERROR_COUNTER1_15_8	0x0	8-bit counter that reports the following error source: A5: Replay number rollover error
[7:0]	ERROR_COUNTER1_7_0	0x0	8-bit counter that reports the following error source: A4: Replay time error

3.8.11.32 PCIE_ERROR_COUNTER_2 Register (0A8h)

Table 57 • **PCIE_ERROR_COUNTER_2**

Bit Number	Name	Reset Value	Description
[31:24]	ERROR_COUNTER2_31_24	0x0	8-bit counter that reports the following error source: AB: Completer abort error
[23:16]	ERROR_COUNTER2_23_16	0x0	8-bit counter that reports the following error source: AA: Completion timeout error
[15:8]	ERROR_COUNTER2_15_8	0x0	8-bit counter that reports the following error source: A9: Unsupported request error
[7:0]	ERROR_COUNTER2_7_0	0x0	8-bit counter that reports the following error source: A8: ECRC error

3.8.11.33 PCIE_ERROR_COUNTER_3 Register (0ACh)

Table 58 • **PCIE_ERROR_COUNTER_3**

Bit Number	Name	Reset Value	Description
[31:24]	ERROR_COUNTER3_31_24	0x0	8-bit counter that reports the following error source: AF: Malformed TLP error
[23:16]	ERROR_COUNTER3_23_16	0x0	8-bit counter that reports the following error source: AE: Flow control protocol error
[15:8]	ERROR_COUNTER3_15_8	0x0	8-bit counter that reports the following error source: AD: Receiver overflow error
[7:0]	ERROR_COUNTER3_7_0	0x0	8-bit counter that reports the following error source: AC: Unexpected completion error

3.8.11.34 PCIE_CREDIT_ALLOCATION_0 Register(0B0h)

Table 59 • **PCIE_CREDIT_ALLOCATION_0**

Bit Number	Name	Reset Value	Description
[31:28]	CREDIT_ALLOCATION0_31_28	0x0	Reserved
[27:16]	CREDIT_ALLOCATION0_27_16	0x0	VC0 posted header/data credit pd_cred0
[15:8]	CREDIT_ALLOCATION0_15_8	0x2	Reserved
[7:0]	CREDIT_ALLOCATION0_7_0	0x2	VC0 posted header/data credit ph_cred0

Note: Refer to Table 17, page 71 for CREDIT ALLOCATION Details.

3.8.11.35 CREDIT_ALLOCATION_1 Register (0B4h)

Table 60 • **PCIE_CREDIT_ALLOCATION_1**

Bit Number	Name	Reset Value	Description
[31:28]	CREDIT_ALLOCATION1_31_28	0x0	Reserved
[27:16]	CREDIT_ALLOCATION1_27_16	0x0	VC0 non-posted header/data credit npd_cred0
[15:8]	CREDIT_ALLOCATION1_15_8	0x2	Reserved
[7:0]	CREDIT_ALLOCATION1_7_0	0x2	VC0 non-posted header/data credit nph_cred0

Note: Refer to Table 17, page 71 for CREDIT ALLOCATION Details.

Table 61 • **Credit Allocation Details**

Credit Type	Credit Size	Buffer Space (KB)
Posted Header (PH)	16	256
Posted Data (PD)	16	256
Non-Posted Header (NPH)	16	256
Non-Posted Data (NPD)	16	256
Completion Header (CPLH)	N/A	N/A
Completion Data (CPLD)	Infinite	512

Note: The CPLD buffer space is allocated based on the number of supported outstanding transactions. The PCIe slave interface supports up to 4 outstanding transactions therefore reserving 128 byte x 4 amount of buffer space.

3.8.11.36 PCIE_AXI_SLAVE_WINDOW0_0 Register (0C0h)

Table 62 • **PCIE_AXI_SLAVE_WINDOW0_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW00_31_12	0x0	Base address AXI3 slave window 0
[11:0]	Reserved	0x0	Reserved

3.8.11.37 PCIE_AXI_SLAVE_WINDOW0_1 Register (0C4h)

Table 63 • **PCIE_AXI_SLAVE_WINDOW0_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW0_1_31_12	0x0	Size of AXI3 Slave window 0
[11:1]	Reserved		Reserved
0	PCIE_AXI_SLAVE_WINDOW0_1_0	0x0	Enable bit of AXI3 slave window 0

3.8.11.38 PCIE_AXI_SLAVE_WINDOW0_2 Register (0C8h)

Table 64 • **PCIE_AXI_SLAVE_WINDOW0_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW0_2_31_12	0x0	LSB of base address PCIe window 0
[11:5]	Reserved		Reserved
[4:2]	PCIE_AXI_SLAVE_WINDOW0_2_4_2	0x0	AXI3 slave window 0 traffic class (TC)
1	PCiE_AXI_SLAVE_WINDOW0_2_1	0x0	AXI3 Slave window 0 relaxed ordering (RO)
0	PCIE_AXI_SLAVE_WINDOW0_2_0	0x0	AXI3 Slave window 0 no snoop (NS)

3.8.11.39 PCIE_AXI_SLAVE_WINDOW0_3 Register (0CCh)

Table 65 • **PCIE_AXI_SLAVE_WINDOW0_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_SLAVE_WINDOW0_3_31_0	0x0	MSB of base address PCIe window 0

3.8.11.40 PCIE_AXI_SLAVE_WINDOW1_0 Register (0D0h)

Table 66 • **PCIE_AXI_SLAVE_WINDOW1_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW1_0_31_12	0x0	Base address AXI3 slave window 1
[11:0]	Reserved		Reserved

3.8.11.41 PCIE_AXI_SLAVE_WINDOW1_1 Register (0D4h)

Table 67 • **PCIE_AXI_SLAVE_WINDOW1_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW1_1_31_12	0x0	Size of AXI3 slave window 1
[11:1]	Reserved		Reserved
0	PCIE_AXI_SLAVE_WINDOW1_1_0	0x0	Enable bit of AXI3 slave window 1

3.8.11.42 PCIE_AXI_SLAVE_WINDOW1_2 Register (0D8h)

Table 68 • **PCIE_AXI_SLAVE_WINDOW1_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW1_2_31_12	0x0	LSB of base address PCIe window 1
[11:5]	Reserved		Reserved
[4:2]	PCIE_AXI_SLAVE_WINDOW1_2_4_2	0x0	AXI3 slave window 0 traffic class (TC)
1	PCIE_AXI_SLAVE_WINDOW1_2_1	0x0	AXI3 slave window 0 relaxed ordering (RO)
0	PCIE_AXI_SLAVE_WINDOW1_2_0	0x0	AXI3 slave window 0 no snoop (NS)

3.8.11.43 PCIE_AXI_SLAVE_WINDOW1_3 Register (0DCh)

Table 69 • **PCIE_AXI_SLAVE_WINDOW1_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_SLAVE_WINDOW1_3_31_0	0x0	MSB of base address PCIe window 1

3.8.11.44 PCIE_AXI_SLAVE_WINDOW2_0 Register (0E0h)

Table 70 • **PCIE_AXI_SLAVE_WINDOW2_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW2_0_31_12	0x0	Base address AXI3 slave window 2
[11:0]	Reserved		

3.8.11.45 PCIE_AXI_SLAVE_WINDOW2_1 Register (0E4h)

Table 71 • **PCIE_AXI_SLAVE_WINDOW2_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW2_1_31_12	0x0	Size of AXI3 slave window 2
[11:1]	Reserved		Reserved
0	PCIE_AXI_SLAVE_WINDOW2_1_0	0x0	Enable bit of AXI3 slave window 2

3.8.11.46 PCIE_AXI_SLAVE_WINDOW2_2 Register (0E8h)

Table 72 • **PCIE_AXI_SLAVE_WINDOW2_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW2_2_31_12	0x0	LSB of base address PCIe window 2
[11:5]	Reserved		Reserved
[4:2]	PCIE_AXI_SLAVE_WINDOW2_2_4_2	0x0	AXI3 slave window 0 traffic class (TC)
1	PCIE_AXI_SLAVE_WINDOW2_2_1	0x0	AXI3 slave window 0 relaxed ordering (RO)
0	PCIE_AXI_SLAVE_WINDOW2_2_0	0x0	AXI3 slave window 0 no snoop (NS)

3.8.11.46.1 PCIE_AXI_SLAVE_WINDOW2_3 Register (0ECh)

Table 73 • **PCIE_AXI_SLAVE_WINDOW2_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_SLAVE_WINDOW2_3_31_0	0x0	MSB of base address PCIe window 3

3.8.11.47 PCIE_AXI_SLAVE_WINDOW3_0 Register (0F0h)

Table 74 • **PCIE_AXI_SLAVE_WINDOW3_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW3_0_31_12	0x0	Base address AXI3 slave window 3
[11:0]	Reserved		

3.8.11.48 PCIE_AXI_SLAVE_WINDOW3_1 Register (0F4h)

Table 75 • **PCIE_AXI_SLAVE_WINDOW3_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW3_1_31_12	0x0	Size of AXI3 slave window 3
[11:1]	Reserved		Reserved
0	PCIE_AXI_SLAVE_WINDOW3_1_0	0x0	Enable bit of AXI3 slave window 3

3.8.11.49 PCIE_AXI_SLAVE_WINDOW3_2 Register (0F8h)

Table 76 • **PCIE_AXI_SLAVE_WINDOW3_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_SLAVE_WINDOW3_2_31_12	0x0	LSB of base address PCIe window 3
[11:5]	Reserved		Reserved
[4:2]	PCIE_AXI_SLAVE_WINDOW3_2_4_2	0x0	AXI3 slave window 0 traffic class (TC)
1	PCIE_AXI_SLAVE_WINDOW3_2_1	0x0	AXI3 Slave window 0 relaxed ordering (RO)
0	PCIE_AXI_SLAVE_WINDOW3_2_0	0x0	AXI3 Slave window 0 no snoop (NS)

3.8.11.50 PCIE_AXI_SLAVE_WINDOW3_3 Register (0FCh)

Table 77 • **PCIE_AXI_SLAVE_WINDOW3_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_SLAVE_WINDOW3_3_31_0	0x0	MSB of base address PCIe window 0

3.8.11.51 PCIE_AXI_MASTER_WINDOW0_0 Register (100h)

Table 78 • **PCIE_AXI_MASTER_WINDOW0_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW0_0_31_12	0x0	Base address AXI3 master window 0
[11:0]	Reserved		Reserved

3.8.11.52 PCIE_AXI_MASTER_WINDOW0_1 Register (104h)

Table 79 • **PCIE_AXI_MASTER_WINDOW0_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW0_1_31_12	0x0	Size of AXI3 master window 0
[11:1]	Reserved		Reserved
0	PCIE_AXI_MASTER_WINDOW0_1_0		Enable bit of AXI3 master window 0

3.8.11.53 PCIE_AXI_MASTER_WINDOW0_2 Register (108h)

Table 80 • **PCIE_AXI_MASTER_WINDOW0_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW0_2_31_12	0x0	LSB of base address PCIe window 0
[11:6]	Reserved	0x0	Reserved
[5:0]	PCIE_AXI_MASTER_WINDOW0_2_5_0	0x0	These bits set the BAR. To select a BAR, set the following values: 0x01: BAR0 (32-bit BAR) or BAR0/1 (64-bit BAR) 0x02: BAR1 (32-bit BAR) only 0x04: BAR2 (32-bit BAR) or BAR2/3 (64-bit BAR) 0x08: BAR3 (32-bit BAR) only 0x10: BAR4 (32-bit BAR) or BAR4/5 (64-bit BAR) 0x20: BAR5 (32-bit BAR) only

3.8.11.54 PCIE_AXI_MASTER_WINDOW0_3 Register (10Ch)

Table 81 • **PCIE_AXI_MASTER_WINDOW0_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_MASTER_WINDOW0_3_31_0	0x0	MSB of base address PCIe window 0

3.8.11.55 PCIE_AXI_MASTER_WINDOW1_0 Register (110h)

Table 82 • **PCIE_AXI_MASTER_WINDOW1_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW1_0_31_12	0x0	Base address AXI3 master window 1
[11:0]	Reserved		Reserved

3.8.11.56 PCIE_AXI_MASTER_WINDOW1_1 Register (114h)

Table 83 • **PCIE_AXI_MASTER_WINDOW1_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW1_1_31_12	0x0	Size of AXI3 master window 1
[11:1]	Reserved		Reserved
0	PCIE_AXI_MASTER_WINDOW1_1_0	0x0	Enable bit of AXI3 master window 1

3.8.11.57 PCIE_AXI_MASTER_WINDOW1_2 Register (118h)

Table 84 • **PCIE_AXI_MASTER_WINDOW1_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW1_2_31_12	0x0	LSB of base address PCIe window 1
[11:6]	Reserved		Reserved
[5:0]	PCIE_AXI_MASTER_WINDOW1_2_5_0	0x0	These bits set the BAR. To select a BAR, set the following values: 0x01: BAR0 (32-bit BAR) or BAR0/1 (64-bit BAR) 0x02: BAR1 (32-bit BAR) only 0x04: BAR2 (32-bit BAR) or BAR2/3 (64-bit BAR) 0x08: BAR3 (32-bit BAR) only 0x10: BAR4 (32-bit BAR) or BAR4/5 (64-bit BAR) 0x20: BAR5 (32-bit BAR) only

3.8.11.58 PCIE_AXI_MASTER_WINDOW1_3 Register (11Ch)

Table 85 • **PCIE_AXI_MASTER_WINDOW1_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_MASTER_WINDOW1_3_31_0	0x0	MSB of base address PCIe window 1

3.8.11.59 PCIE_AXI_MASTER_WINDOW2_0 Register (120h)

Table 86 • **PCIE_AXI_MASTER_WINDOW2_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW2_0_31_12	0x0	Base address AXI3 master window 2
[11:0]	Reserved	0x0	Reserved

3.8.11.60 PCIE_AXI_MASTER_WINDOW2_1 Register (124h)

Table 87 • **PCIE_AXI_MASTER_WINDOW2_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW2_1_31_12	0x0	Size of AXI3 master window 2
[11:1]	Reserved		Reserved
0	PCIE_AXI_MASTER_WINDOW2_1_0	0x0	Enable bit of AXI3 master window 2

3.8.11.61 PCIE_AXI_MASTER_WINDOW2_2 Register (128h)

Table 88 • **PCIE_AXI_MASTER_WINDOW2_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW2_2_31_12	0x0	LSB of base address PCIe window 2
[11:6]	Reserved		Reserved
[5:0]	PCIE_AXI_MASTER_WINDOW2_2_5_0	0x0	These bits set the BAR. To select a BAR, set the following values: 0x01: BAR0 (32-bit BAR) or BAR0/1 (64-bit BAR) 0x02: BAR1 (32-bit BAR) only 0x04: BAR2 (32-bit BAR) or BAR2/3 (64-bit BAR) 0x08: BAR3 (32-bit BAR) only 0x10: BAR4 (32-bit BAR) or BAR4/5 (64-bit BAR) 0x20: BAR5 (32-bit BAR) only

3.8.11.62 PCIE_AXI_MASTER_WINDOW2_3 Register (12Ch)

Table 89 • **PCIE_AXI_MASTER_WINDOW2_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_MASTER_WINDOW2_3_31_0	0x0	MSB of base address PCIe window 3

3.8.11.63 AXI_MASTER_WINDOW3_0 Register (130h)

Table 90 • **PCIE_AXI_MASTER_WINDOW3_0**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW3_0_31_12	0x0	Base address AXI3 master window 3
[11:0]	Reserved		Reserved

3.8.11.64 PCIE_AXI_MASTER_WINDOW3_1 Register (134h)

Table 91 • **PCIE_AXI_MASTER_WINDOW3_1**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW3_1_31_12	0x0	Size of AXI3 master window 3
[11:1]	Reserved		Reserved
0	PCIE_AXI_MASTER_WINDOW3_1_0	0x0	Enable bit of AXI3 master window 3

3.8.11.65 PCIE_AXI_MASTER_WINDOW3_2 Register (138h)

Table 92 • **PCIE_AXI_MASTER_WINDOW3_2**

Bit Number	Name	Reset Value	Description
[31:12]	PCIE_AXI_MASTER_WINDOW3_2_31_12	0x0	LSB of base address PCIe window 3
[11:6]	Reserved	0x0	Reserved
[5:0]	PCIE_AXI_MASTER_WINDOW3_2_5_0	0x0	These bits set the BAR. To select a BAR, set the following values: 0x01: BAR0 (32-bit BAR) or BAR0/1 (64-bit BAR) 0x02: BAR1 (32-bit BAR) only 0x04: BAR2 (32-bit BAR) or BAR2/3 (64-bit BAR) 0x08: BAR3 (32-bit BAR) only 0x10: BAR4 (32-bit BAR) or BAR4/5 (64-bit BAR) 0x20: BAR5 (32-bit BAR) only

3.8.11.66 PCIE_AXI_MASTER_WINDOW3_3 Register (13Ch)

Table 93 • **PCIE_AXI_MASTER_WINDOW3_3**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_AXI_MASTER_WINDOW3_3_31_0	0x0	MSB of base address PCIe window 0

3.8.11.67 PCIE_INFO Register (016Ch)

Table 94 • **PCIE_INFO**

Bit Number	Name	Reset Value	Description
[31:12]	INFO_31_12	0x0	Bridge version
[11:0]	INFO_11_0		Reserved

3.8.11.68 PCIE_PCIE_CONFIG Register (204h)

Table 95 • **PCIE_PCIE_CONFIG**

Bit Number	Name	Reset Value	Description
[31:5]	Reserved		Reserved
4	PCIE_CONFIG_4	0x0	Selects the level of de-emphasis for an upstream component when the link speed is 5.0 Gbps.
[3:0]	PCIE_CONFIG_3_0	0x0	Sets PCIe Specification version capability: 0000: Core is compliant with PCIe Specification 1.0a or 1.1 0001: Core is compliant with PCIe Specification 1.0a or 1.1 0010: Core is compliant with PCIe Specification 2.0

3.8.11.69 PCIE_DEV2SCR Register (230h)

Table 96 • **PCIE_DEV2SCR**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_DEV2SCR_31_0	0x0	Reports the current value of the PCIe device control and status register. It can be monitored by the local processor when relaxed ordering and no snoop bits are enabled in the system. This register is used when link speed is set to 5.0 Gbps.

3.8.11.70 PCIE_LINK2SCR Register (234h)

Table 97 • **PCIE_LINK2SCR**

Bit Number	Name	Reset Value	Description
[31:0]	PCIE_INK2SCR_31_0	0x0	Reports the current value of the PCIe Link Control and Status register. It can be monitored by the local processor when Relaxed Ordering and No Snoop bits are enabled in the system. This register is used when link speed is set to 5.0 Gbps.

3.8.11.71 PCIE_ASPM_L0S_GEN2 Register (260h)

Table 98 • **PCIE_ASPM_L0S_GEN2**

Bit Number	Name	Reset Value	Description
[31:24]	PCIE_ASPM_L0S_GEN2_31_24	0x0	NFTS_COMCLK in common clock mode at 5.0 Gbps
[23:16]	PCIE_ASPM_L0S_GEN2_23_16	0x0	NFTS_SPCLK in independent clock mode at 5.0 Gbps
[15:4]	PCIE_ASPM_L0S_GEN2_15_4	0x0	Reserved
[3:0]	PCIE_ASPM_L0S_GEN2_3_0	0x0	Number of electrical idle exit (EIE) symbols sent before transmitting the first FTS

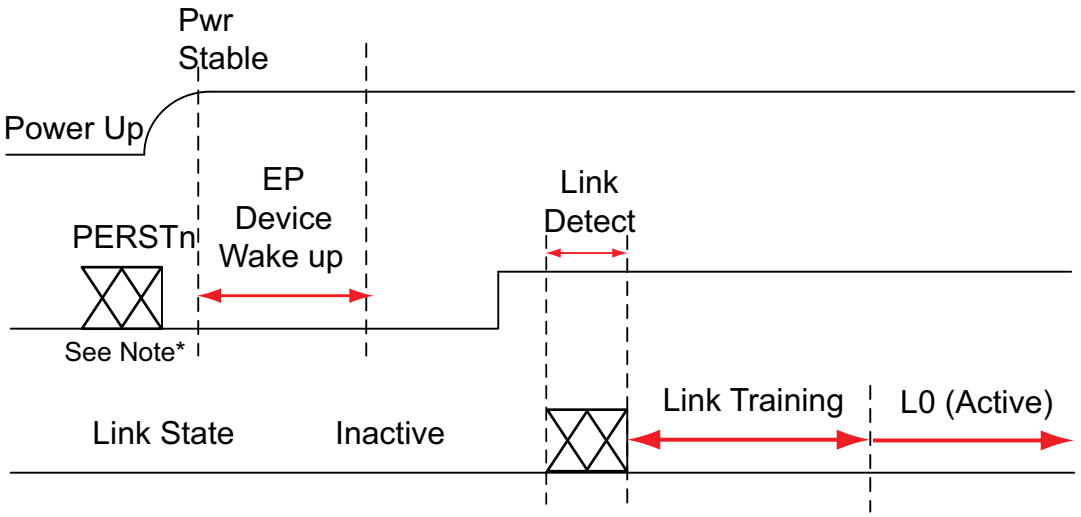
3.8.12 PCI Express Power-Up

The PCI Express (PCIe) specification provides timing requirements for power-up. As with SRAM-based FPGA endpoint devices, power-up is a concern when working within these tight specifications. The PCIe specification specifies the release of the fundamental reset (PERST#) in the connector specification. The PERSTn release time (TPVPERL) of 100ms is used for the PCIe Card Electromechanical Specification for add-in cards. From the point of power stable to at least 100 ms, the PERST# must remain asserted. Different PCIe systems hold PERSTn longer than 100 ms, but the minimum time is 100 ms.

The advantage of flash-based SmartFusion2 or IGLOO2 device is that its wake up time is very fast in contrast to SRAM FPGA endpoints. The semi-autonomous nature to the PCIe Core in the SmartFusion2 or IGLOO2 device will quickly move from power-up to link detectable allowing the device to be detected by the root. When the device is detected by the root, it proceeds to the Polling state of the LTSSM. Afterwards the device goes through Detect and then enters the Polling state. The link now cycles through the remainder of the LTSSM.

In use cases where the root and endpoint power-up separately, the PERSTn signal must be used to handshake the link startups. This is detailed in the following figure.

Figure 32 • **PCI Express Power-Up States**



1. **EP device wake up:** The internal flash loads the programming data to the device. If PERSTn is required, a fabric GPIO must be connected to the PERSTn of the Root. This GPIO must be connected in the FPGA design to the CORE_RESETn pin of the PCIe core. The embedded PCIe core will be held in reset by PERSTn, and is released afterwards to start PCIe link detection and training.
2. **Link Detection:** Out-of-band pulse looks for far-end connection.
3. The PCIe link completes the training phase and reaches the L0 state.
4. After the embedded PCIe endpoint core reaches the L0 state, the host operating system (OS) accesses the PCIe core's configuration space registers (CSR) to perform configuration write access cycles that are part of the system enumeration process.

Note: PERSTn is controlled by the root. If not connected to the EP, the EP enters Link Detect as soon as the device wake up is complete.

3.9 Hot Reset Solutions

PCI Express supports an in-band method to reset the PCIe link. A Hot Reset can be initiated by the host setting a specific bit in a training sequence ordered set. A Hot Reset resets the controller that resets the LTSSM state to the Detect state.

When the SmartFusion2 and IGLOO2 PCIe controller is reset by a Hot Reset, the configuration of the SERDESIF needs to be re-initialized. The following methods can be used to re-initialize the SERDESIF:

- Global Re-Initialization
- Stand Alone SERDESIF Re-Initialization

3.9.1 Global Re-Initialization

For SmartFusion2 and IGLOO2, there is a fully automated solution for initializing the SERDESIF by using the System Builder in Libero SoC. When the System Builder is used, all of the peripheral configuration data is written into the SERDESIF APB interface after the device is powered up. The same method can be used to re-initialize the PCIe controller in the SERDESIF after a Hot Reset.

When using global re-initialization, the entire device and all peripherals will be re-initialized. This means that all lanes of the SERDES across the device as well as memory controllers (MDDR/FDDR) will also be re-initialized. If the application can handle a complete device re-initialization, then this is the easiest method to be implemented.

To reset the System Builder generated module to re-initialize all of the peripherals, the active-low FAB_RESET_N port must be used. To identify when the PCIe controller goes into the Hot Reset, the LTSSM must be monitored for a value of 5'b10100.

3.9.2 Stand Alone SERDESIF Re-Initialization

The Stand Alone Initialization method can also be used for initialization. The Stand Alone SERDESIF initialization localizes the programming of the SERDESIF registers through the APB for each SERDESIF instance. A CoreABC programmable microcontroller is used to load the SERDESIF registers over the APB. For more information about the Stand Alone Peripheral Initialization, refer to the *SmartFusion2 Standalone Peripheral Initialization User Guide* and *IGLOO2 Standalone Peripheral Initialization User Guide*.

When using stand alone re-initialization, only the SERDESIF that is connected to the CoreABC will be re-initialized. However, all of the SERDES lanes in that SERDESIF will be disrupted by the re-initialization. For more information about developing and using the stand along re-initialization, refer to the following web-pages:

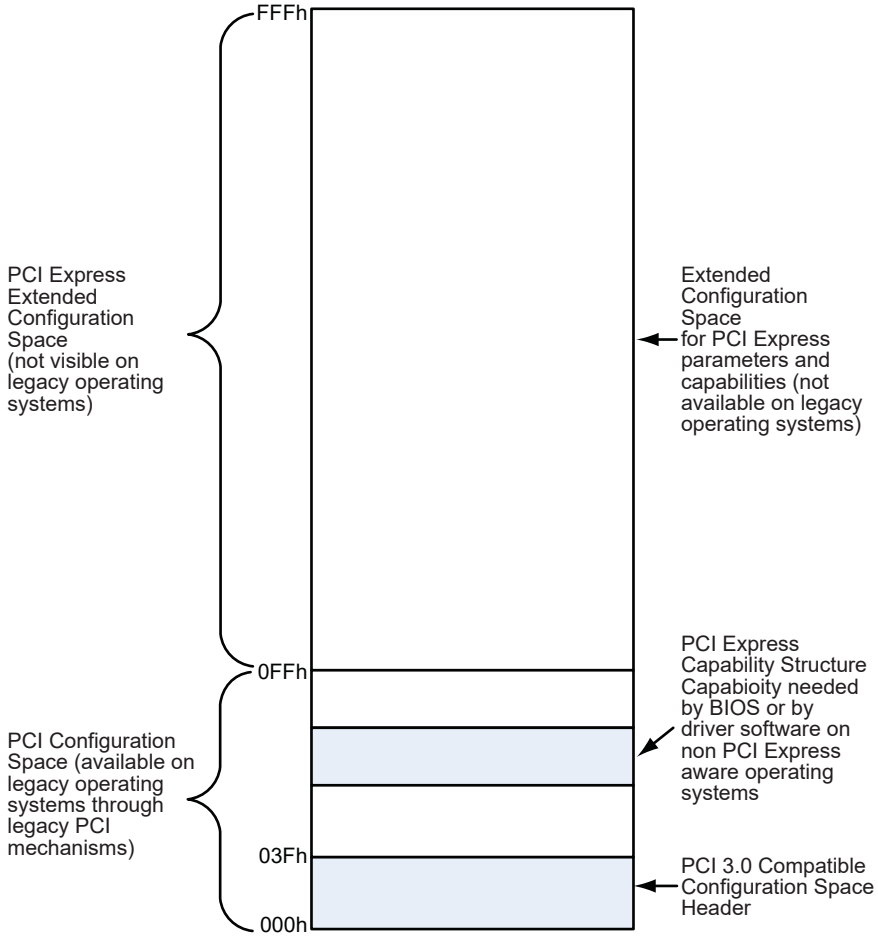
- http://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion2#design-resources
- http://www.microsemi.com/products/fpga-soc/fpga/igloo2-fpga#design-resources

Note: Re-initializing the PCIe controller in response to a Hot Reset resets all of the status bits in the PCIe controller. In the PCIe configuration space, there are a few bits that are specified as RWS (Read/Write/Sticky). These sticky bits loses their value under a Hot Reset due to the re-initialization.

3.10 Appendix A: PCIe Configuration Space

The PCIe base IP core transaction layer (TL) contains the 4Kbyte configuration space. The configuration space implements all configuration registers and associated functions. It manages BAR and window decoding, interrupt/MSI message generation, power management negotiation, and error handling. For upstream ports, the configuration space is accessed through the PCIe link using Type 0 requests. Type 1 requests are forwarded to the application layer. For downstream ports, the configuration space is accessed through the application interface using Type 0 requests. Type 1 requests are forwarded to the PCIe link. The first 256 bytes of the configuration space are the function's configuration space, and the remaining configuration space is PCIe extended configuration space (see the following figure).

Figure 33 • **PCIe Configuration Space**



3.10.1 Common Configuration Space Header

The following table lists the common configuration space header. The PCIe common configuration space includes the following registers:

- Type 0 configuration settings
- MSI capability structure
- Power management capability structure
- PCIe capability structure

For comprehensive information about these registers, refer to PCIe Base Specification Revision 1.0a, 1.1 or 2.0 specifications.

3.10.2 PCIe Extended Capability Structure

The following table lists the PCIe extended capability structure. SmartFusion2 or IGLOO2 PCIe common configuration space includes the following registers:

- PCIe advanced error reporting (AER) extended capability structure

Table 99 • **PCIe Extended Capability Structure (Function 0)**

[31:24]	[23:16]	[15:8]	[7:0]	Byte Offset
AER				800h..834h

3.10.3 Type 0 Configuration Settings

The following table lists the type 0 configuration settings.

Table 100 • **Type 0 Configuration Register**

[31:24]	[23:16]	[15:8]	[7:0]	Byte Offset
Device ID		Vendor ID		000h
Status		Command		004h
Class Code			Revision ID	008h
BIST	Header Type	Latency Timer	Cache Line Size	00Ch
Base address 0				010h
Base address 1				014h
Base address 2				018h
Base address 3				01Ch
Base address 4				020h
Base address 5				024h
Subsystem ID		Subsystem Vendor ID		02Ch
Reserved				030h
			Capabilities PTR	034h
Reserved				038h
		Int. pin	Int. line	03Ch

3.10.4 IP Core Status Register

The following table lists the content of the IP Core Status Register.

Table 101 • **IP Core Status Register**

[31:28]	[27:16]	[15:4]	[3:0]	Byte Offset
Reserved	Core version	Signature	Reserved	044h

MSI Capability Structure

The following table lists the content of the MSI capability structure.

Table 102 • **MSI Capability Structure Register**

[31:24]	[23:16]	[15:8]	[7:0]	Byte Offset
Message control		Next pointer	Cap ID	050h

Table 102 • **MSI Capability Structure Register**

Message address			054h
Message upper address			058h
Reserved	Reserved	Message data	05Ch

3.10.5 Power Management Capability Structure

The following table lists the content of the power management capability structure.

Table 103 • **Power Management Capability Structure**

[31:24]	[23:16]	[15:8]	[7:0]	Byte Offset
Capabilities register		Next cap PTR	Cap ID	078h
Data	PM control/status bridge extensions	Power management status and control		07Ch

3.10.6 PCIe Capability Structure

The following table lists the content of the PCIe capability structure.

Table 104 • **PCIe Capability Structure Register**

[31:24]	[23:16]	[15:8]	[7:0]	Byte Offset
Capabilities register		Next cap PTR	cap ID	080h
Device capabilities				084h
Device status		Device control		088h
Link capabilities				08Ch
Link status		Link control		090h
Slot capabilities				094h
Slot status		Slot control		098h
Reserved		Root control		09Ch
Root status				0A0h
Device capabilities 2				0A4h
Device status 2		Device control 2		0A8h
Link capabilities 2				0ACh
Link status 2		Link control 2		0B0h

3.10.7 PCIe AER Extended Capability Structure

The following table lists the advanced error reporting (AER) extended capability structure for Function 0. For Functions 1 - 7, the byte offset is from 100h to 134h.

Table 105 • **PCIe AER Extended Capability Structure**

[31:24]	[23:16]	[15:8]	[7:0]	Byte Offset
PCIe enhanced capability header				800h
Uncorrectable error status register				804h
Uncorrectable error mask register				808h
Uncorrectable error severity register				80Ch

Table 105 • **PCIe AER Extended Capability Structure *(continued)***

[31:24]	[23:16]	[15:8]	[7:0]	Byte Offset
Correctable error status register				810h
Correctable error mask register				814h
Advanced error capabilities and control register				818h
Header log register				81Ch
Root error command				82Ch
Root error status				830h
Error source identification register		Correctable error source ID register		834h

3.11 Appendix B: TLP Contents

The following tables describe the contents of all TLPs. The bit assignments are mapped with the MSB in the top-left and the LSB in the bottom-right of the tables.

3.11.1 B.1 Content of a TLP without a Data Payload

Table 106 • **Memory Read Request 32-bit Addressing Descriptor Format**

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

Table 107 • **Memory Read Request-Locked 32-bit Addressing Descriptor Format**

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

Table 108 • **Memory Read Request 64-bit Addressing Descriptor Format**

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

Table 109 • **Memory Read Request-Locked 64-bit Addressing Descriptor Format**

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

Table 110 • **Type 0 Configuration Read Request Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Bus Number								Device Nb.					Func			0	0	0	0	Ext. Reg.				Register Nb.						0	0
Byte 12	R																															

Table 111 • **Type 0 Configuration Read Request Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Bus Number								Device Nb.					Func			0	0	0	0	Ext. Reg.				Register Nb.						0	0
Byte 12	R																															

Table 112 • **Message (without data) Descriptor Format**

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

Table 113 • **Completion (without data) Descriptor Format**

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

Table 114 • **Completion Locked (without data) Descriptor Format**

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	1	0	1	1	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									

Table 114 • **Completion Locked (without data) Descriptor Format**

Byte 4	Completer ID																Status			B	Byte Count										
Byte 8	Requester ID																Tag								0	Lower Address					
Byte 12																															

3.11.2 B.2 Content of a TLP with a Data Payload

Table 115 • **Memory Write Request 32-bit Addressing Descriptor Format**

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

Table 116 • **Memory Write Request 64-bit Addressing Descriptor Format**

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

Table 117 • **Type 0 Configuration Write Request Descriptor Format**

	+0								**+1**								**+2**								**+3**							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Bus Number								Device Nb.					Func			0	0	0	0	Ext. Reg.				Register Nb.						0	0
Byte 12	R																															

Table 118 • **Type 0 Configuration Write Request Descriptor Format**

	+0								**+1**								**+2**								**+3**							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Bus Number								Device Nb.					Func			0	0	0	0	Ext. Reg.				Register Nb.						0	0
Byte 12	R																															

Table 119 • **Type 1 Configuration Write Request Descriptor Format**

	+0								**+1**								**+2**								**+3**							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Bus Number								Device Nb.					Func			0	0	0	0	Ext. Reg.				Register Nb.						0	0
Byte 12	R																															

Table 120 • **Completion (with data) Descriptor Format**

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

Table 121 • **Completion Locked (with data) Descriptor Format**

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

Table 122 • **Message (with data) Descriptor Format**

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

3.12 Appendix C: SERDESIF PCIe Debug Interface

SERDES block has a debug mode mostly for debugging PCIe link. A number of internal status/error signals are available to the fabric to be used for end-to-end system debug functions. To keep the number of signals interfacing between the fabric and SERDES block to a minimum, these debug signals are multiplexed on PRDATA signals of the APB bus. Debug mode is enabled only when SYSTEM_DEBUG_MODE_KEY (8'b1010_0101) is written (offset - address: A8). Once correctly written, the APB READ-BUS is multiplexed with PCIe debug data. PCIe DEBUG data is available only when APB-READ is not taking place. This feature can be activated only from the Edit Registers GUI of the SERDES Configurator.

The following table lists the condition where debug information is available.

Table 123 • **Debug Information Available Conditions**

Debug Mode	APB-Bus Operation	APB_PRDATA Bus Behavior
Enabled	Write	Debug information
Enabled	Read	APB read data
Enabled	Idle	Debug information
Disabled	Don’t care	APB read data

The following table lists the debug signals that are mapped to APB PRDATA bus.

Table 124 • **Debug Signals Mapping to APB Bus**

APB_PRDAT Signals	Debug Signal	Description
APB_S_PRDATA[31]	PHY_LOCK_STATUS	SERDES PHY related status signals. Combined status of PHY - Tx/CDR- PLL lock status. Only PHY-lanes which are used are considered for this phy_lock_status signal generation. When any used PLL's PHY lanes are locked, phy_lock_status is either 1'b1 or it is 1'b0. **Note**: Individual PHY lane's PLL information is available in SYSTEM_SERDES_TEST_OUT register in the SERDESIF system block (see Table 198, page 166).
APB_S_PRDATA[30:26]	LTSSM_R [4:0]	LTSSM state: LTSSM state encoding. Refer to LTSSM_28_24 register for more information (see Table 41, page 59).
APB_S_PRDATA[25:24]	ERR_PHY [1:0]	PHY error: Physical layer error bit0: Receiver port error bit1: Training error
APB_S_PRDATA[23:19]	ERR_DLL [4:0]	DLL error: Data link layer error bit0: TLP error bit1: DLLP error bit2: Replay timer error bit3: Replay counter rollover bit4: DLL protocol error
APB_S_PRDATA[18:10]	ERR_TRN [8:0]	TRN error: Transaction layer error bit0: Poisoned TLP received bit1: ECRC check failed bit2: Unsupported request bit3: Completion timeout bit4: Completer abort bit5: Unexpected completion bit6: Receiver overflow bit7: Flow control protocol error bit8: Malformed TLP

Table 124 • **Debug Signals Mapping to APB Bus** ***(continued)***

APB_PRDAT Signals	Debug Signal	Description
APB_S_PRDATA[9]	ERR_DL	Error ACK/NACK DLLP parameter: This signal reports that the received ACK/NACK DLLP has a sequence number higher than the sequence number of the last transmitted TLP.
APB_S_PRDATA[8]	TIMEOUT	LTSSM timeout: This signal serves as a flag, which indicates that the LTSSM timeout condition is reached for the current LTSSM state. 1'b1: Timeout condition reached 1'b0: No time condition reached
APB_S_PRDATA[7]	CRCERR	Received TLP with LCRC error: This signal reports that a TLP is received, which contains an LCRC error.
APB_S_PRDATA[6]	CRCINV	Received nullified TLP: This signal indicates that a nullified TLP is received.
APB_S_PRDATA[5]	RX_ERR_DLLP	Received DLLP with LCRC error: This signal reports that a DLLP has been received that contains an LCRC error.
APB_S_PRDATA[4]	ERR_DLLPROT	DLL protocol error at data link layer: This signal reports a DLL protocol error.
APB_S_PRDATA[3]	RX_ERR_FRAME	DLL framing error detected: This signal indicates that received data cannot be considered as a DLLP or TLP, in which case, a receive port error is generated and link retraining is initiated.
APB_S_PRDATA[2]	L2-EXIT	l2_exit information signal
APB_S_PRDATA[1]	DLUP_EXIT	dlup_exit information signal
APB_S_PRDATA[0]	HOTRST_EXIT	hotrst_exit information signal

4 XAUI

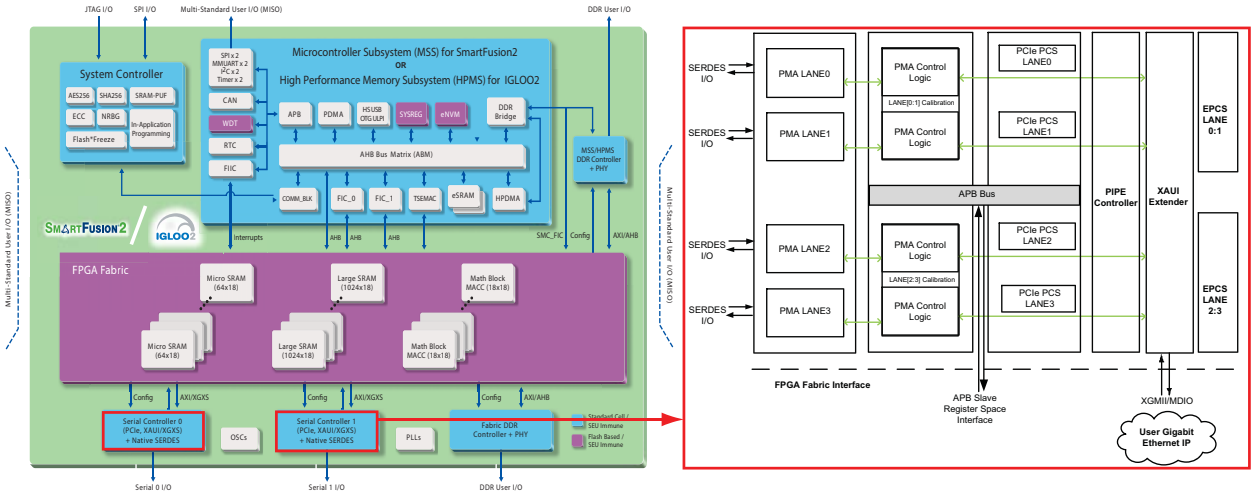
4.1 Introduction

This chapter describes implementing XAUI in SmartFusion2 and IGLOO2 FPGA devices using the XAUI extender block inside the SERDESIF block. XAUI is a standard for extending the 10 Gb media independent interface (XGMII) between the media access control (MAC) and PHY layer of 10 Gb Ethernet (10 GbE). The SmartFusion2 and IGLOO2 high speed serial block implements integrated XAUI, which can be connected to a 10 Gb Ethernet FPGA IP core in the FPGA fabric for a complete solution. XAUI is supported in -1 speed grade device and package offerings.

4.2 Overview of XAUI Implementation in SmartFusion2/IGLOO2

The IGLOO2 SERDESIF block integrates the functionality of supporting multiple high speed serial protocols, such as PCIe 2.0, XAUI, and EPCS, as shown in the following figure. The SERDESIF block can be configured in various modes, including XAUI. XAUI is a standard for extending the XGMII between the MAC and PHY layer of 10 GbE.

Figure 34 • **SmartFusion2 and IGLOO2 SERDESIF Block Diagram**



As an example for discussion, the block diagram for the M2GL050T device is shown in the preceding figure. The smaller devices have fewer SERDES channel and the larger devices have more channels.

The XAUI implementation in SmartFusion2 and IGLOO2 devices offers the following features:

- Full compliance with IEEE 802.3
- IEEE 802.3ae- clause 45 MDIO interface
- IEEE 802.3ae- clause 48 state machines
- Pseudorandom idle insertion (PRBS Polynomial $X^7 + X^3 + 1$)
- FPGA interface Clock frequency of 156.25 MHz
- Double-width 64-bit single data rate (SDR) interface
- Comma alignment function
- Low power mode
- PHY-XS and DTE-XS loopback
- IEEE 802.3ae- annex 48A jitter test pattern support
- IEEE 802.3 clause 36 8B/10B encoding compliance
- Tolerance of lane skew up to 16 ns (50 UI)
- IEEE 802.3 PICs compliance matrix

4.2.1 Device Support

The following table lists the total number of SERDESIF blocks in each SmartFusion2 and IGLOO2 device that can be configured to support XAUI.

Table 125 • **SERDESIF Blocks in SmartFusion2 and IGLOO2 FPGAs that support XAUI**

	M2S/M2GL 005	M2S/M2GL 010	M2S/M2GL 025	M2S/M2GL 050	M2S/M2GL 060	M2S/M2GL 090	M2S/M2GL 150
SERDESIF available for XAUI	0	1	1	Up to 2	1	1	Up to 4

Note: The specified number of SERDESIF blocks varies depending on the device package.

Note: XAUI uses one entire SERDESIF (4-Lanes).

4.2.2 XAUI Overview

XAUI is a standard for extending the 10 Gb media independent interface (XGMII) between the MAC and PHY layer of 10 GbE. XGMII provides a 10 Gbps pipeline; the separate transmission of clock and data coupled with the timing requirement to latch data on both the rising and falling edges of the clock results in a significant challenge in routing the bus more than the recommended short distance of 7 cm. Also, the XGMII bus puts many limitations on the number of ports that can be implemented on a system line card. To overcome these issues, IEEE 802.3ae 10 GbE Task Force developed the XAUI interface. XAUI is a full duplex interface that uses four self-clocked serial differential links in each direction to achieve 10 Gbps data throughput. Each serial link operates at 3.125 Gbps to accommodate both data and the overhead associated with 8B/10B coding. The self-clocked nature eliminates skew concerns between clock and data, and extends the functional reach of the XGMII by approximately another 50 cm. Its compact nature and robust performance makes it ideal for chip to chip, board to board, and chip to optics module applications. The XAUI standard is fully specified in clauses 47 and 48 of the 10 GbE specification IEEE Std. 802.3-2008.

XAUI has the following features:

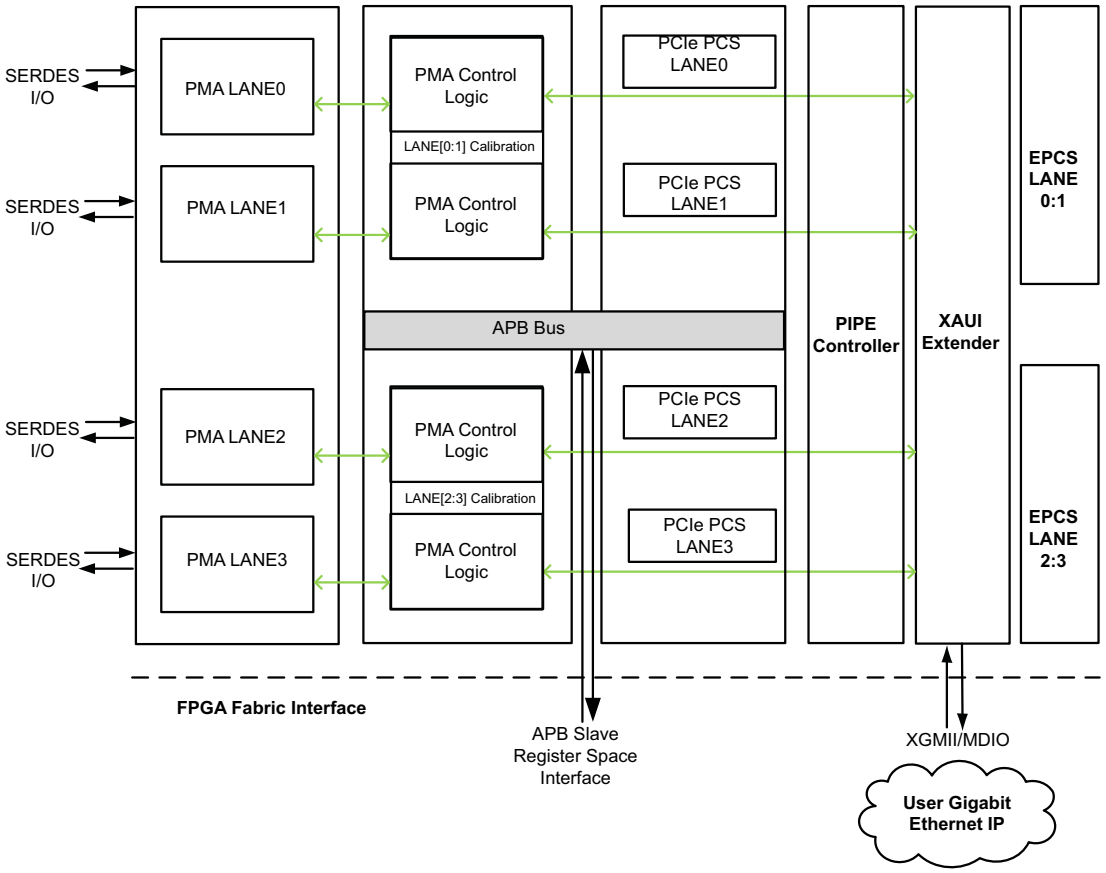
- Simple signal mapping to the XGMII
- Independent transmit and receive data paths
- Four lanes conveying the XGMII 64-bit data and control
- Differential signaling with low voltage swing (1600 mV (p-p))
- Self-timed interface allowing jitter control to the physical coding sublayer (PCS)
- Shared technology with other 10 Gbps interfaces
- Shared functionality with other 10 Gbps Ethernet block
- Utilization of 8b/10b encoding

The conversion between the XGMII and XAUI interfaces occurs at the XGXS (XAUI extender sublayer).

4.2.3 SmartFusion2 and IGLOO2 XAUI

The SmartFusion2 and IGLOO2 FPGA has a integrated XAUI implementation. The SmartFusion2 and IGLOO2 high speed interface (SERDESIF) has a XAUI IP block (XAUI Extender) and SERDES block. Figure 34, page 88 shows an application example; the XAUI IP is extending the 10 Gb soft IP in the fabric. The XAUI IP block in SERDESIF block provides the XGXS functionality and the SERDES block provides the physical layer. The XAUI IP block connects a 10 Gb Ethernet MAC to SERDES physical medium attachment (PMA) logic. 3rd party MAC IP hosted in the FPGA fabric is responsible for providing IEEE 802.3ae functionality including generating XAUI compliant idle sequence and insertion of ||K||, ||R||, ||A|| ordered sets at TXD/TXC interface. In addition, the XAUI IP block has a management data input/output (MDIO) interface allowing an MDIO manageable device to program the MDIO registers. The SERDES block is configured to PMA only mode and requires a reference clock of 156.25 MHz to operate at a line rate of 3.125 Gbps.

Figure 35 • **XAUI Implementation in SmartFusion2/IGLOO2**



The high speed serial interface (SERDESIF) can be configured to support multiple serial protocols. However, when using the XAUI protocol, only one protocol can be implemented because XAUI uses all four SERDES lanes. The following table lists the lane speed of four physical SERDES lanes when using XAUI.

Table 126 • **XAUI Implementation in SmartFusion2 and IGLOO2**

	Lane0		Lane1		Lane2		Lane3	
XAUI Protocol	**Protocol**	**Speed**	**Protocol**	**Speed**	**Protocol**	**Speed**	**Protocol**	**Speed**
Single Protocol PHY	XAUI	3.125G	XAUI	3.125G	XAUI	3.125G	XAUI	3.125G

The SERDESIF block in XAUI mode has a SERDES I/O pad on one side and an XGMII interface and MDIO interface on the FPGA fabric side. Table 126, page 90 lists the SmartFusion2 and IGLOO2 I/O PAD in XAUI mode.

The SmartFusion2 and IGLOO2 XAUI interface uses the SERDESIF- I/O Signal Interface, page 149 listed in Table 167, page 149. Refer to the XAUI IP Fabric Interface, page 94 for detailed information on XGMII and MDIO interfaces on the fabric side.

Refer to device Pin Descriptions for other SERDES required pins.

4.3 Getting Started

This section provides an overview of how to configure a SERDESIF block in XAUI mode, and instructions for using XAUI IP in a SmartFusion2 or IGLOO2 device.

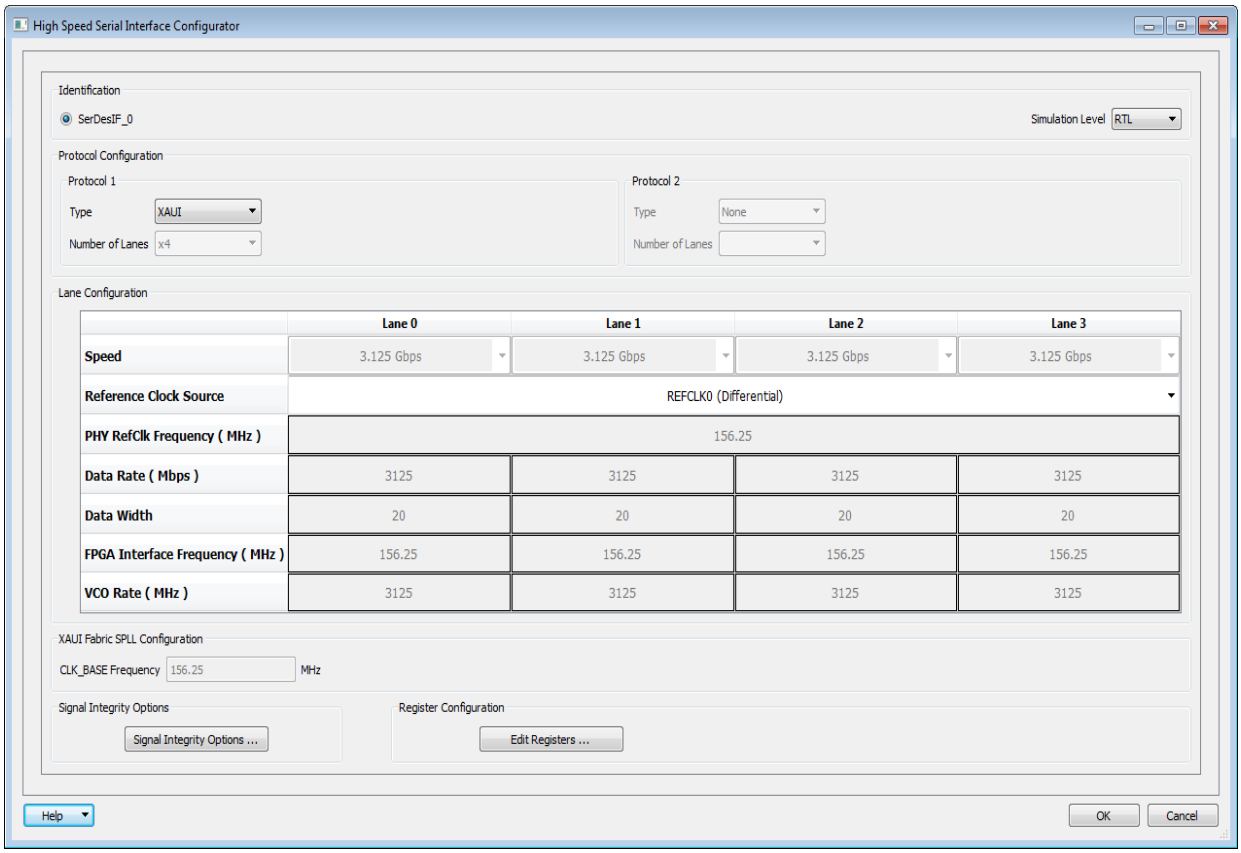
The following sections show how to instantiate XAUI in a design by completing the following steps:

1. Using High-Speed Serial Configurator for XAUI Mode
2. Simulating SERDESIF with XAUI Mode
3. Application Example Using XAUI

4.3.1 Using High-Speed Serial Configurator for XAUI Mode

The High Speed Serial Interface Configurator in Libero SoC can be used to configure the SERDESIF block in XAUI mode. Refer to the following figure for the XAUI mode setting in the high speed serial interface configurator.

Figure 36 • **XAUI Mode Setting in High Speed Serial Interface Configurator**



Following are brief descriptions of the configuration options. For more information, see the *High Speed Serial Interface Configuration User Guide*.

4.3.1.1 Protocol Selection

These settings are used for protocol selection:

- Protocol 1 Type—This is the protocol setting. Select XAUI from the drop-down menu.
- Protocol 1 PHY Reference Clock—This is the PHY reference clock selection. Refer to the SERDES Reference Clocks Selection, page 99 for details on PHY reference clock selection.

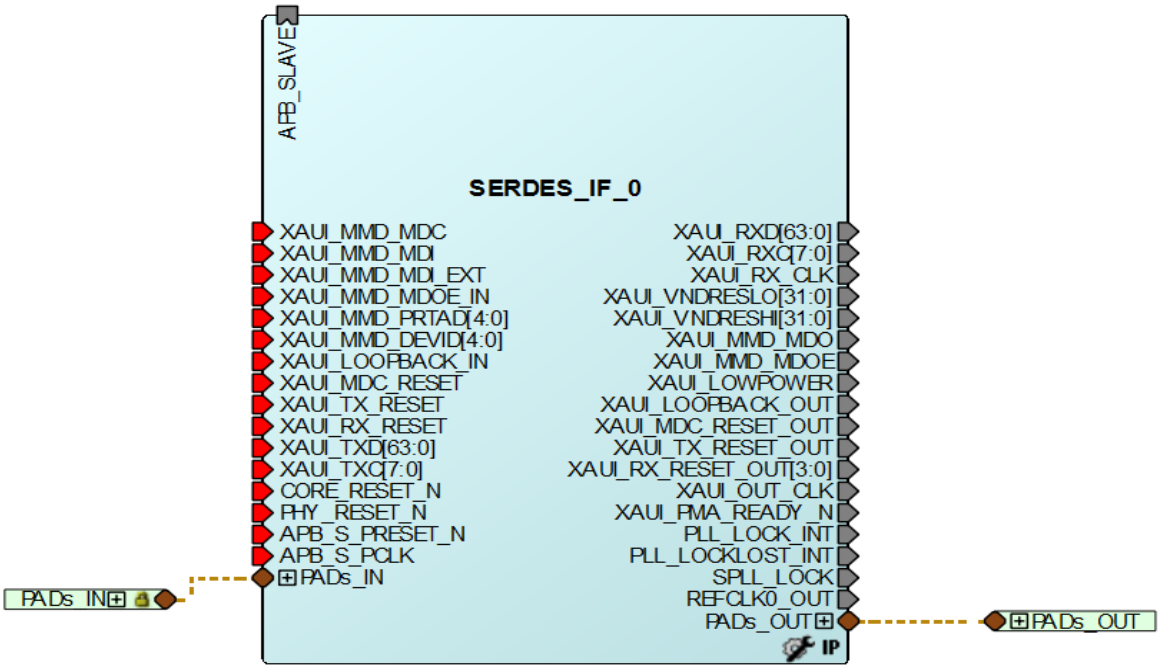
4.3.2 Simulating SERDESIF with XAUI Mode

When configured in XAUI mode, the SERDESIF block will only allow the user to run simulation using the APB3 interface. The user can read and write to the SERDESIF and SERDES register using the simulation library. Refer to the *SERDESIF BFM Simulation Guide* for details for using RTL mode for simulation. BFM mode for simulation is not supported for XAUI.

4.3.3 Application Example Using XAUI

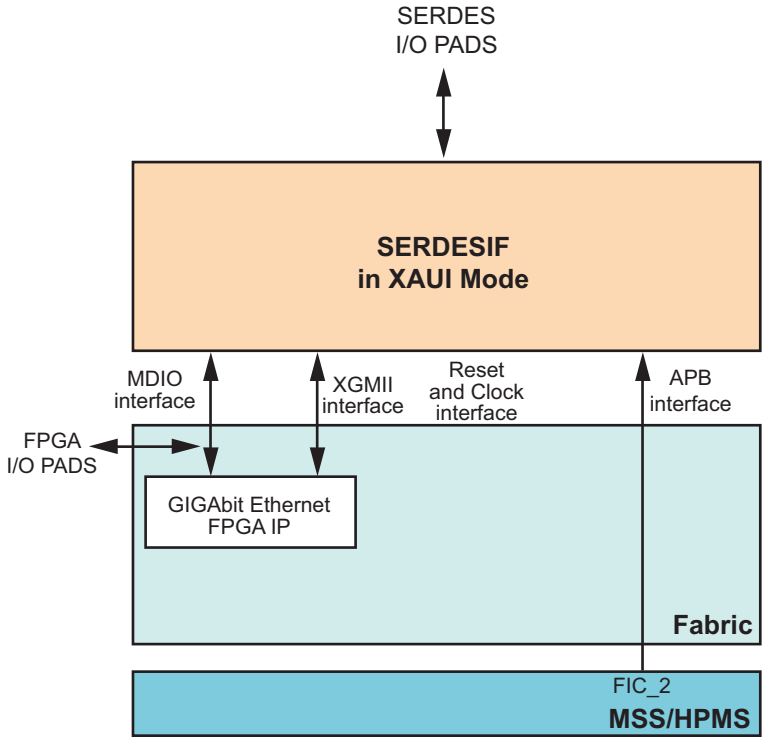
To complete an application in a SmartFusion2 or IGLOO2 device, configure the appropriate settings in the high speed serial interface generator, then generate the SERDESIF block in XAUI mode. The following figure shows the SERDESIF block in Libero SoC in XAUI mode. Libero SoC promotes the SERDES I/Os to top level and exposes XGMII, MDIO, and the APB interface to the FPGA fabric. In addition, the SERDESIF block exposes the clocks, resets, and PLL locks to the user.

Figure 37 • **High Speed Serial Interface Block in XAUI Mode in Libero SoC**



The following figure shows a complete application in a SmartFusion2 or IGLOO2 device.

Figure 38 • **An Application Example Using XAUI IP**



4.4 XAUI IP Architecture

This section provides an overview of the XAUI IP block, covering the following topics:

- Overview of XAUI IP Block
- XAUI IP Fabric Interface

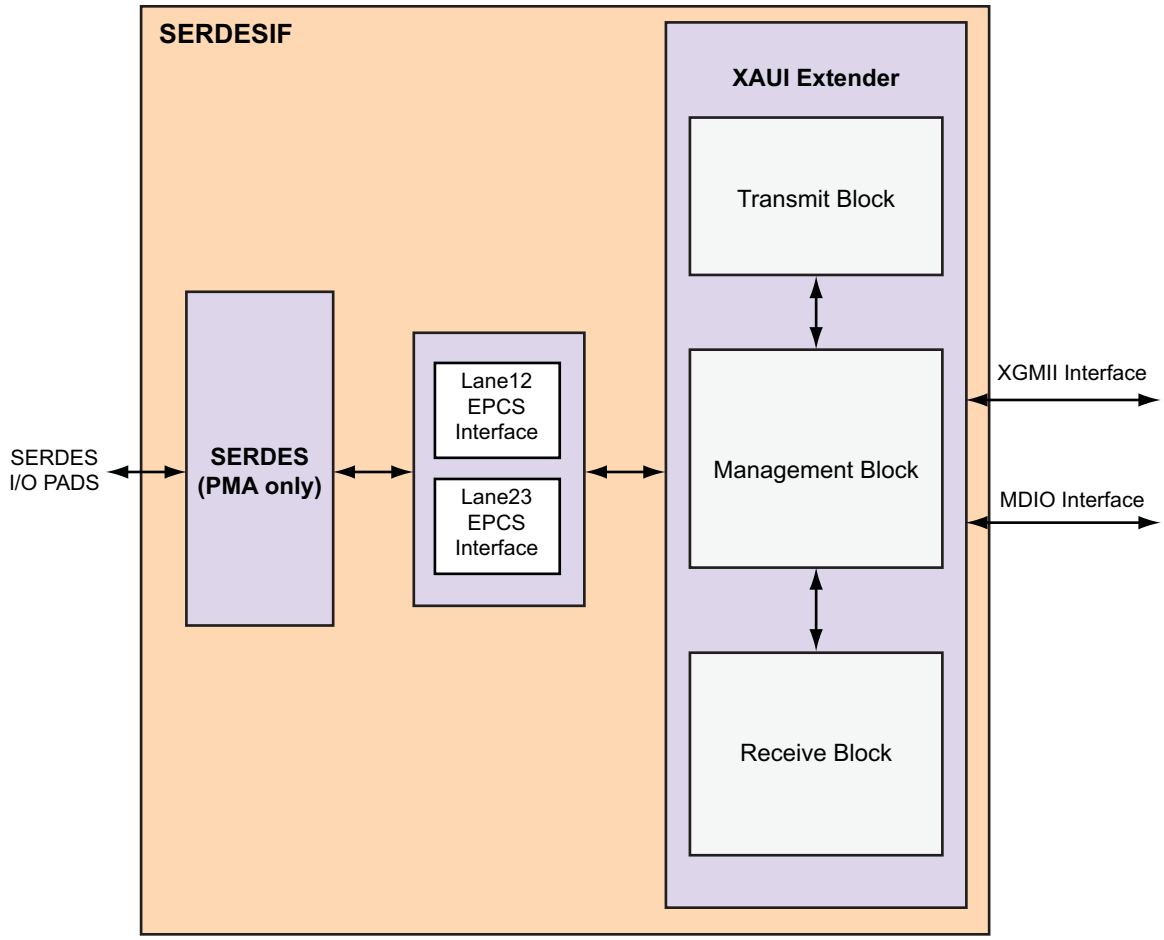
4.4.1 Overview of XAUI IP Block

The following figure shows the XAUI IP block. This module is connected to the SERDES PMA block through the two EPCS interface blocks, and to the FPGA fabric through XGMII and MDIO interfaces. There are three major blocks:

- Transmit block: This block is responsible for encoding the XGMII data (using 8B/10B). The output to the transmit block is an 80-bit interface (20 bits per lane). The PMA in the SERDES receives this 80-bit data and transmits it to the XAUI bus.
- Receive block: This block receives 8B/10B encoded data and four recovered clocks from an external XAUI SERDES PMA. The receive block performs comma alignment on the data, phase-aligns the four lanes of data, and performs the 8B/10B decode function.
- Management block: The management block is the MDIO interface to the design registers.

The transmit and receive FPGA interface frequencies are set at 156.25 MHz.

Figure 39 • **XAUI Extender Block Diagram**



4.4.2 XAUI IP Fabric Interface

The SmartFusion2 and IGLOO2 SERDESIF in XAUI mode interfaces with the fabric and differential I/O pads. The following tables describe the fabric interfaces:

- MDIO Interface Signals
- XGMII Transmit Interface Signals
- XGMII Receive Interface Signals
- XAUI Extender Block Miscellaneous Control Signal
- Clock Signals in XAUI Mode
- XAUI Extender Block Miscellaneous Control Signal

Table 127 • **MDIO Interface Signals**

Port	Type	Description
XAUI_MMD_MDC	Input	MDIO I/F clock. 40 MHz or less
XAUI_MMD_MDI	Input	MDIO data input from bidirectional pad
XAUI_MMD_MDI_ EXT	Input	Serial data output of another block that is responding to a host read transaction
XAUI_MMD_MDO	Output	MDIO data output to bidirectional pad
XAUI_MMD_ MDOE	Output	MDIO data output enable. This is used to control bidirectional pad. It is active High.

Table 127 • MDIO Interface Signals (continued)

Port	Type	Description
XAUI_MMD_ MDOE_IN	Input	MDIO data output enable input. This is used to force MMD_MDI High in an idle state. It is active High.
XAUI_MMD_PRTAD[4:0]	Input	A static signal that defines the port address of the XAUI extender block instantiated. Access to the MDIO registers is granted only if the port address specified in the MDI stream matches this input.
XAUI_MMD_ DEVID[4:0]	Input	A static signal that defines the device ID of the XAUI extender block instantiated. Access to the MDIO registers is granted only if the device ID (DEVID) specified in the MMD_MDI stream matches this input. For the PHY-XS, this value must be 04h. For the DTE-XS, this value must be 05h.
XAUI_VNDRRESLO[31:0]	Output	A general purpose register for vendor use, reset Low. The output of two 16-bit registers (address 0x8000 and 0x8001) that are set Low on reset for general purpose use.
XAUI_VNDRRESHI[31:0]	Output	General purpose register for vendor use, reset High. The output of two 16-bit registers (address 0x8002 and 0x8003) that are set High on reset for general use.

Table 128 • XAUI Block Miscellaneous/Control/Status Signals

Port	Type	Description
XAUI_PMA_READY_N	Output	This signal goes low when all 4- SerDes lanes within the XAUI block have completed the calibration sequence indicating the entire PMA is ready for operation. This pin will go high if any of the SerDes lanes go down.
PLL_LOCKLOST_INT	Output	Output of SPLL Lock lost status register (Active high indicates that the lock is lost). The SPLL manages clock domain data transfers skew between the FABRIC and XAUI block module.
SPLL_LOCK	Output	SPLL Lock signal. High indicates that the frequency and phase lock are achieved. The SPLL manages clock domain data transfers skew between the FABRIC and XAUI block module.
PLL_LOCK_INT	Output	The SPLL Lock status register (Active High indicates locked).

Table 129 • XGMII Transmit Interface Signals

Port	Type	Description
XAUI_TXD[63:0]	Input	Transmit data input from the XGMII. The signal has the following lane definitions: Lane0, row0: txd[7:0] Lane1, row0: txd[15:8] Lane2, row0: txd[23:16] Lane3, row0: txd[31:24] Lane0, row1: txd[39:32] Lane1, row1: txd[47:40] Lane2, row1: txd[55:48] Lane3, row1: txd[63:56] The row0 lanes are leading the row1 lanes in time. Refer to IEEE 802.3ae, clause 46, for a complete definition.

Table 129 • **XGMII Transmit Interface Signals *(continued)***

Port	Type	Description
XAUI_TXC[7:0]	Input	Transmit data lane control signals. The signal has the following lane definitions: Lane0, row0: txc[0] Lane1, row0: txc[1] Lane2, row0: txc[2] Lane3, row0: txc[3] Lane0, row1: txc[4] Lane1, row1: txc[5] Lane2, row1: txc[6] Lane3, row1: txc[7] The row0 lanes are leading the row1 lanes in time. Refer to IEEE 802.3ae, clause 46, for a complete definition.

Table 130 • **XGMII Receive Interface Signals**

Port	Type	Description
XAUI_RX_CLK	Output	Receive clock synchronous with Rxd, clock synchronous with the output XGMII data Rxd. Equal to the input recovered clock RX_CLKI0. This clock operates nominally at 156.25 MHz. Refer to IEEE 802.3ae, clause 46, for a complete definition.
XAUI_RXD[63:0]	Output	Receive data output to the XGMII. The signal has the following lane definitions: Lane0, row0: rxd[7:0] Lane1, row0: rxd[15:8] Lane2, row0: rxd[23:16] Lane3, row0: rxd[31:24] Lane0, row1: rxd[39:32] Lane1, row1: rxd[47:40] Lane2, row1: rxd[55:48] Lane3, row1: rxd[63:56] The row0 lanes are leading the row1 lanes in time. Refer to IEEE 802.3ae, clause 46, for a complete definition.
XAUI_RXC[7:0]	Output	Receive lane data control signals. The signal has the following lane definitions: Lane0, row0: rxc[0] Lane1, row0: rxc[1] Lane2, row0: rxc[2] Lane3, row0: rxc[3] Lane0, row1: rxc[4] Lane1, row1: rxc[5] Lane2, row1: rxc[6] Lane3, row1: rxc[7] The row0 lanes are leading the row1 lanes in time. Refer to IEEE 802.3ae, clause 46, for a complete definition.

Table 131 • **XAUI Extender Block Miscellaneous Control Signal**

Port	Type	Description
XAUI_ LOOPBACK_OUT	Output	Loopback mode enable out. This signal is asserted when the XAUI extender block is placed in loopback. Typically, this signal is shunted back into the input XAUI_ LOOPBACK_IN port. In this case, loopback is implemented in the XAUI extender block. However, this signal can be used to control the loopback function on a PMA in the SERDES block in place of the mxgxs loopback function.
XAUI_ LOOPBACK_IN	Input	Loopback mode enable in. When asserted, the XAUI PMA output data signals are shunted back into the input signals. For loopback to function appropriately, the XGMII transmit clock TX_CLK must be shunted back into the PMA recovered clock inputs.
XAUI_ LOWPOWER	Output	SERDES low power status. When set to 1, the SERDES block is placed in a low power state.

4.5 Reset and Clocks for XAUI

This section covers the functional aspects of the reset and clock circuitry inside the high speed serial interface block for XAUI mode. It includes the following sections:

- XAUI Mode Clocking
- XAUI Mode Reset Network

4.5.1 XAUI Mode Clocking

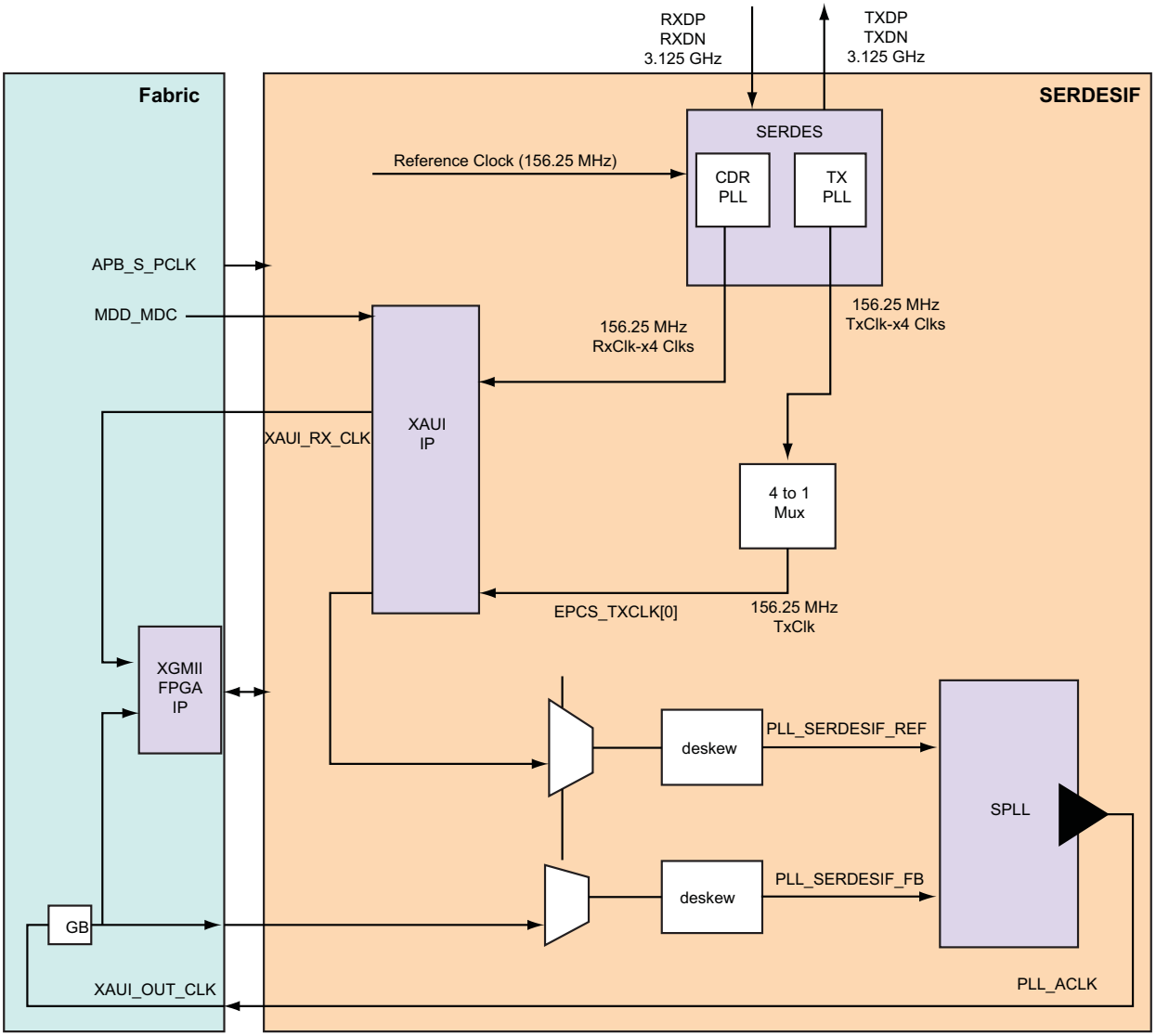
When the SERDESIF is configured in XAUI mode, it has multiple clock inputs and outputs. This section describes the XAUI clocking scheme.

4.5.1.1 SERDESIF Clock Network in XAUI

In XAUI mode, data is exchanged from FPGA IP in the fabric and XAUI IP. The following figure shows in the XAUI clocking scheme. The 156.25 MHz reference clock is used by the SERDES PMA (Tx PLL and CDR PLL). The PLLs generate 156.25 MHz clocks and send 4Rx and 4Tx clocks through the EPCS interface. The Lane0 Tx clock is fed into as reference clock of SPLL and XAUI extender block. This SPLL is used to reduce the skew between the fabric and SmartFusion2 and IGLOO2 SERDESIF module. Libero SOC automatically connects the XAUI_CLK_OUT signal with the XAUI_FDB_CLK signal in the FPGA fabric through the global network, as shown in the following figure. The 4 Rx clocks are fed into the XAUI extender block, where lane de-skewing is done and only one Rx clock is given out to the FPGA fabric. The XAUI_CLK_OUT and XAUI_RX_CLK signals are used by XGMII FPGA IP. The APB clock (APB_S_PCLK) is an asynchronous clock used for SERDESIF register access.

In XAUI only mode, the Tx clock is generated from the PMA. The lane0 Tx clock is used for this purpose. The Rx clock for all four lanes is passed to the XGXS receiver block with gating logic in between to low power operation.

Figure 40 • **SPLL Clocking in XAUI Mode**



The following table lists the various clocks in the XAUI mode.

Table 132 • **Clock Signals in XAUI Mode**

Clock Signal	Type	Description
XAUI_OUT_CLK	Output	Transmit clock to be used for the transmit data. Divided down 156.52 MHz clock from the transmit PLL.
MMD_MDC	Input	MDIO clock
XAUI_RX_CLK	Output	Receive clock synchronous with Rxd, clock synchronous with the output XGMII data Rxd. Equal to the input recovered clock RX_CLKI0. Refer to IEEE 802.3ae, clause 46, for a complete definition.
APB_S_PCLK	Input	PCLK for APB interface

Table 133 • **APB Slave Interface- APB_SLAVE**

Clock Signal	Type	Description
APB_S_PENABLE	Input	APB strobe. This signal indicates the second cycle of an APB transfer.
APB_S_PWRITE	Input	APB write or read. If High, a write occurs when an APB transfer takes place. If low, a read takes place.
APB_S_PADDR[13:0]	Input	APB address bus
APB_S_PWDATA[31:0]	Input	APB write data
APB_S_PREADY	Output	APB ready. Used to insert wait states
APB_S_PRDATA[31:0]	Output	APB read data
APB_S_PSLVERR	Output	APB Error

4.5.1.2 SERDES Reference Clocks Selection

The PMA in the SERDES block needs a reference clock on each of its lanes for Tx and Rx clock generation through PLLs. The following figure shows reference clock selection in the high speed serial interface generator available in Libero SoC. The user can choose one of the two reference clocks. The reference clock to the four lanes can come from I/O Port0 (SERDES_x_REFCLK0) or I/O Port1 (SERDES_x_REFCLK1) I/O pads. The FAB_REF_CLK option is not available in XAUI mode. Note that the reference clock pads are differential input. In XAUI mode, the user must choose one reference clock for all 4 lanes. For more information, refer to the Serializer/De-serializer, page 128.

Figure 41 • **SERDES Reference Clock for PCIe Mode**

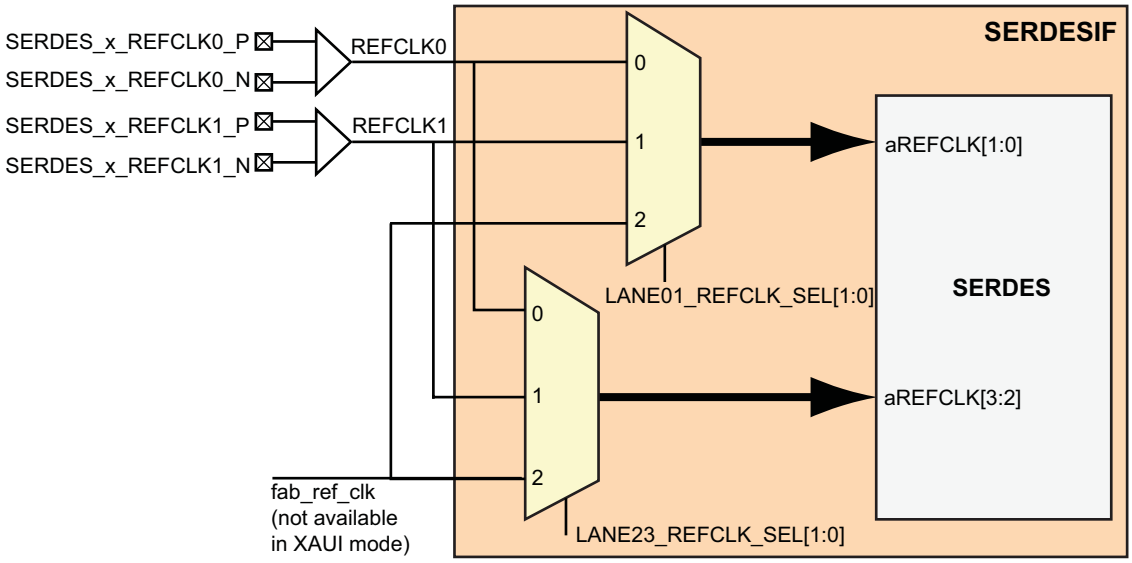


Table 134 • **Reference Clock Signals for SERDES**

Clock Signal	Type	Description
REFCLK0_P, REFCLK0_N	Input	Reference clock output of SERDES_x_REFCLK0_P and SERDES_x_REFCLK0_N
REFCLK1_P, REFCLK1_N	Input	Reference clock output of SERDES_x_REFCLK1_P and SERDES_x_REFCLK1_N
REFCLK[0:1]	Output	Optional clock output from REFCLK to fabric

The following figure shows reference clock selection in the high speed serial interface generator available in Libero SoC. I/O Port0 selects REFCLK0 and I/O Port1 selects REFCLK1.

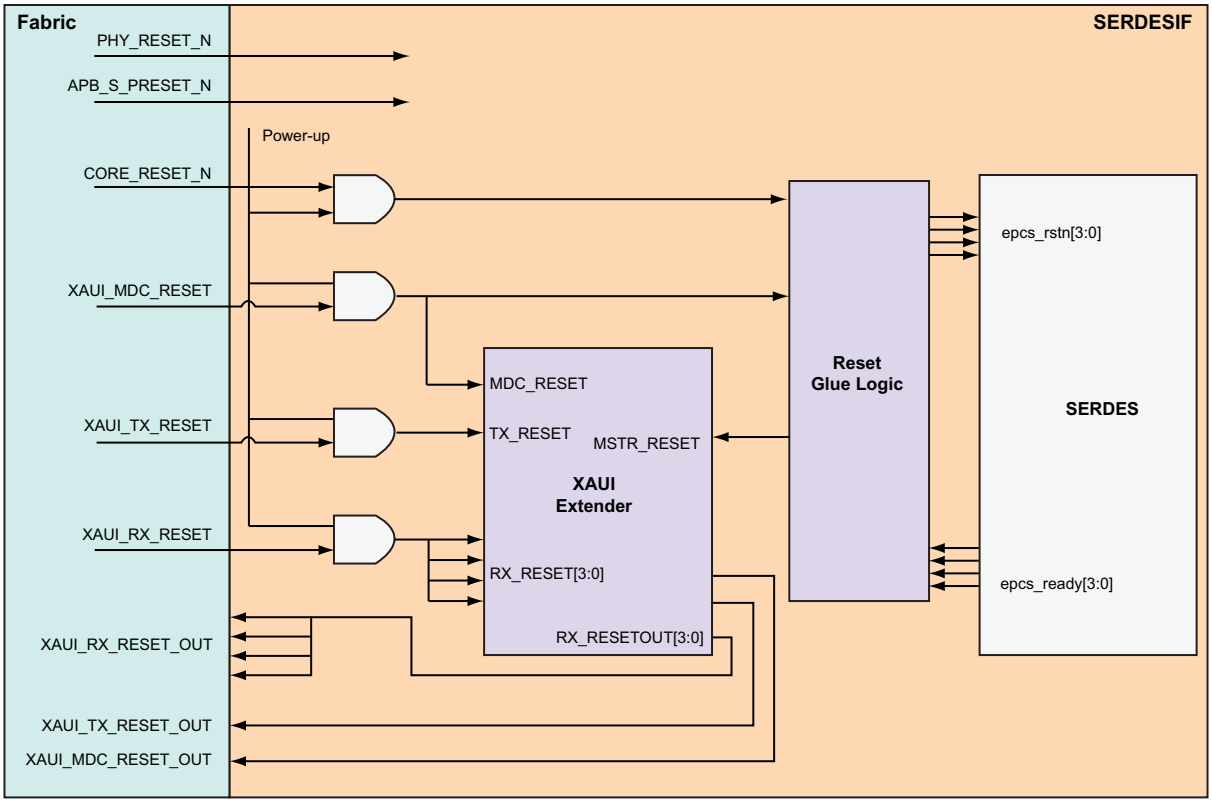
Figure 42 • **Reference Clock Selection**

	Lane 0	Lane 1	Lane 2	Lane 3
Speed	3.125 Gbps	3.125 Gbps	3.125 Gbps	3.125 Gbps
Reference Clock Source	REFCLK1 (Differential)			
PHY RefClk Frequency (MHz)	REFCLK0 (Differential)			
Data Rate (Mbps)	REFCLK1 (Differential)			
Data Width	REFCLK0 (Single-Ended)			
VCO Rate (MHz)	REFCLK1 (Single-Ended)			
FPGA Interface Frequency (MHz)	156.25	156.25	156.25	156.25

4.5.2 XAUI Mode Reset Network

The SmartFusion2 and IGLOO2 SERDESIF configured in XAUI mode has multiple reset inputs. The following figure shows the reset signals and how they are connected internally. The CORE_RESET_N input is an asynchronous reset input XAUI extender block, the XAUI_MDC_RESET input asynchronously resets all of the MDIO registers, the XAUI_TX_RESET input resets the TX block register, and the XAUI_RX_RESET input resets the RX block register. The XAUI IP generates several reset signals that are used by the FPGA IP. In addition, there are several input reset signal SERDES and SERDESIF registers.

Figure 43 • **XAUI Reset Scheme**



The following table lists the reset signals and recommended connections.

Table 135 • **XAUI Mode Reset Signals**

Port	Type	Description
CORE_RESET_N	Input	External asynchronous reset input. Must be asserted for at least two clock cycles of the host clock MMD_MDC for a full reset of the XAUI extender to occur. It is active Low.
PHY_RESET_N	Input	Active Low SERDES reset. It is synchronized with the SERDES reference clock.
XAUI_MDC_RESET_ OUT	Output	MDC synchronous reset. This reset is asynchronously asserted by MSTR_RESET and synchronously deasserted with the XAUI_MMD_MDC clock. Typically, this output is connected to the XAUI_MDC_RESET input. It is active High.
XAUI_MDC_RESET	Input	Asynchronously resets all the MDIO registers to their default values. This pin is connected directly to set/reset ports of all flops in the MMD_MDC clock domain. Typically, this input is connected to the XAUI_MDC_RESET_OUT signal.
XAUI_TX_RESET_OUT	Output	Software generated reset (MDIO Reg00, bit 15) synchronized with TX_CLK. This signal is held High whenever low power mode is enabled. This signal is asynchronously asserted by the software generated reset and synchronously deasserted with EPCS_TXCLK[0]. It is active high. Typically, this output is connected to the XAUI_TX_RESET input.
XAUI_TX_RESET	Input	Resets the XAUI Transmit block. This pin is connected directly to the set/reset ports of all flops in the EPCS_TXCLK[0] clock domain. It is active high. Typically, this is connected to the XAUI_TX_RESET_OUT signal.
XAUI_RX_RESET_ OUT[3:0]	Output	Software generated resets (register 0.15) synchronized with the XAUI_RX_CLK[3:0] clocks. These signals are held high whenever low power mode is enabled. These signals are asynchronously asserted by the software-generated reset and synchronously deasserted with the XAUI_RX_CLK[3:0]. It is active high.
XAUI_RX_RESET	Input	Resets the XAUI extender block. These pins are connected to set/reset ports of all flops in the corresponding XAUI_RX_CLK[3:0] clock domain.
APB_S_PRESET_N	Input	APB asynchronous reset to all SERDESIF APB registers.

4.6 Design Consideration

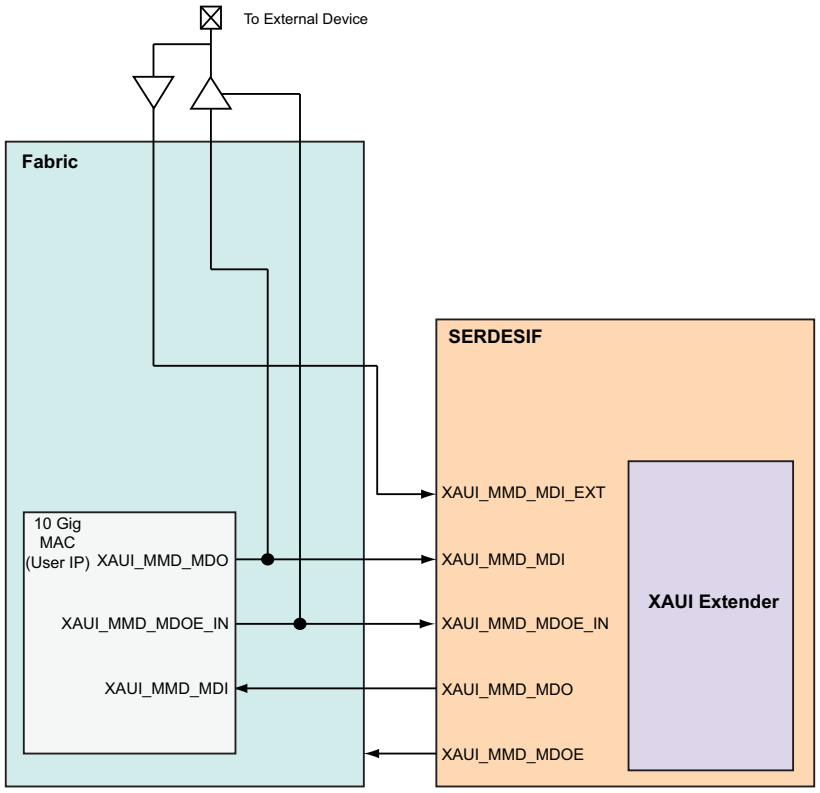
This section provides instruction for implementing XAUI in SmartFusion2 and IGLOO2 devices. It includes the following sections:

- Using the MDIO Interface
- XAUI IP Block Timing Diagram
- XAUI Mode Loopback Test Operation
- Using MMD Status Registers

4.6.1 Using the MDIO Interface

The MDIO interface allows users to access the MDIO registers. The following figure shows a system block diagram for connecting XAUI IP and an MDIO manageable device (MMD) to a station management entity (STA). In this case, the STA is A-XGMAC. The use of the MDIO interface is not required. If the register of the MDIO are not required for the user application the MDIO ports can be tied off inactive.

Figure 44 • **MDIO System Block Diagram**



4.6.2 XAUI IP Block Timing Diagram

The following sections show the timing relations between clock and data for the three interfaces of the XAUI extender.

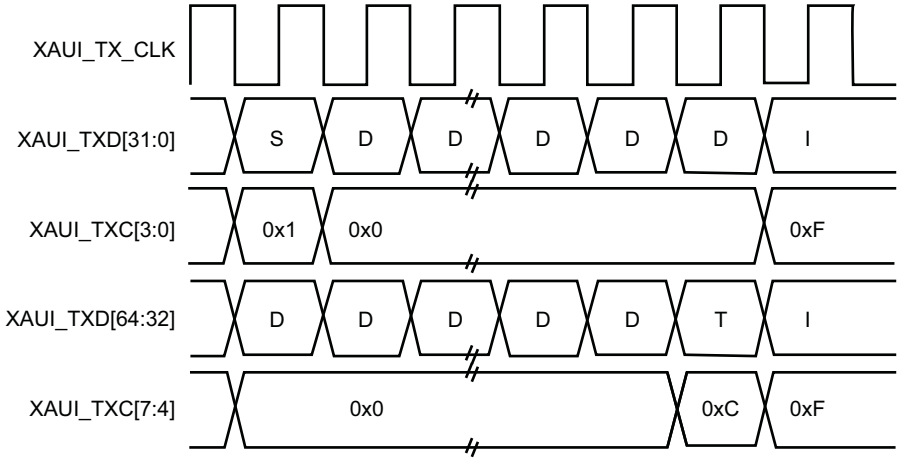
- Transmit Interface
- Receive Interface
- MMD Read Timing
- MMD Write Timing

Refer to the *DS0128: IGLOO2 and SmartFusion2 Datasheet* for the detailed timing numbers.

4.6.2.1 Transmit Interface

The following figure shows the XGMII transmit timing diagram. The transmit data and control signals are source centered on the transmit clock per requirements of IEEE 802.3ae, clause 46. Furthermore, all four lanes of data are synchronous with a common clock.

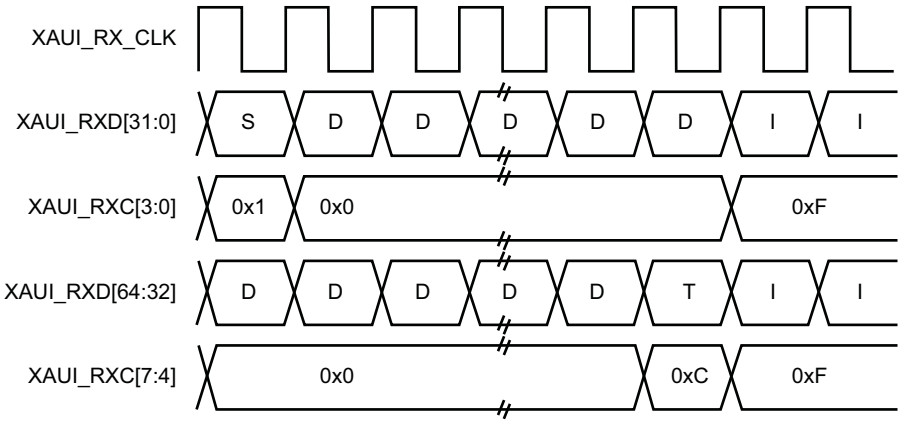
Figure 45 • **Transmit XGMII Interface Timing Diagram**



4.6.2.2 Receive Interface

The following figure shows the XGMII receive timing diagram. The receive data and control signals are edge-aligned with the receive clock XAUI_RX_CLK. To be fully compliant with IEEE 802.3ae, the data and control signals are normally source centered on XAUI_RX_CLK. However, in the SmartFusion2 and IGLOO2 FPGAs, the XAUI extender is interfaced with a FPGA 10G MAC in the fabric within the same device, eliminating the need to source center the data. All four lanes of data are synchronous with the common clock XAUI_RX_CLK. The user must check timing to make sure the data is captured by the FPGA 10G MAC in the fabric.

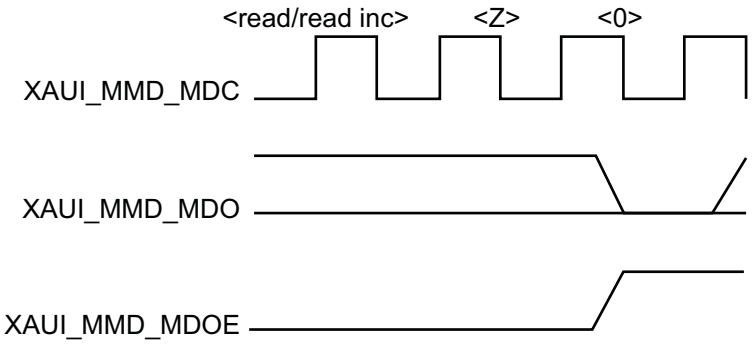
Figure 46 • **XGMII Interface Receive Timing Diagram**



4.6.2.3 MMD Read Timing

The following figure shows the timing diagram for an MDIO register read. The XAUI_MMD_MDO signal is controlled by XAUI_MMD_MDOE.

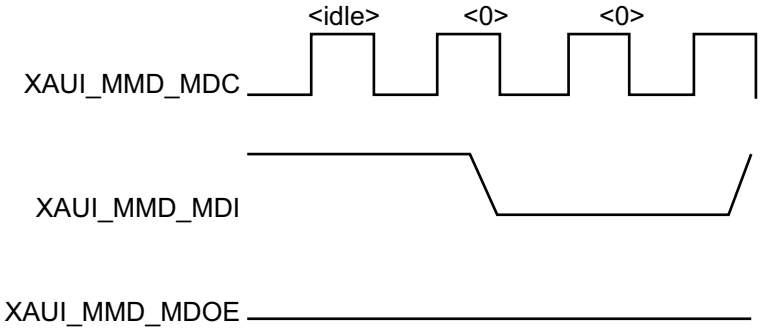
Figure 47 • **MDIO Interface Read Timing Diagram**



4.6.2.4 MMD Write Timing

The following figure shows the timing diagram for MDIO registers. XAUI_MMD_MDOE must not be asserted during a write operation. Refer to the IEEE 802.3ae specification, clause 45, for a complete definition.

Figure 48 • **MDIO Interface Read Timing Diagram**



4.6.3 XAUI Mode Loopback Test Operation

The XAUI extender block can be placed in loopback mode for testing purposes. It can also be placed in multiple loopback operations.

4.6.3.1 XAUI—Near End Loopback Test

Bit 14 of Reg00 can be used to enable the loopback. When loopback mode is enabled, the transmit output is shunted back into the receive input. For loopback mode to work appropriately, the transmit clock is also shunted back into the receive clock inputs. The loopback test data must be fed from the XGMII interface available to fabric.

4.6.3.2 XAUI—Far End Loopback Test

In the XAUI far end loopback test, the transmit interface of the XAUI extender block is connected to the EPCS interface of the SERDES block. In this case, the SERDES block is put in loopback mode, where serial data from transmit side is fed into the serial receive interface. Along with verifying the transmit and receive block of the XAUI extender, it also tests the PMA data path validity. The XAUI_ LOOPBACK_IN signal is used for this mode.

4.6.4 Using MMD Status Registers

There are two MMD status registers: XS status 1 (Table 138, page 107) and XS status 2 (Table 144, page 109).

Upon the deassertion of the reset signal on the XAUI block, the initial state of the two status registers indicate a fault condition. This initial false fault condition must be ignored, and a read operation must be performed on the XS status 1 and XS status 2 registers to clear the false fault-status. After this, when a real fault condition happens (for example, when a link is down), the fault register does indicate it properly, as expected.

4.7 MDIO Register Map

The following table lists the MDIO registers.

Table 136 • **MDIO Registers**

Register Name	Register Address	Read / Writable	Device Address	Description
Reg00	0x0000	R/W	04h/05h	XS control 1 register (see Table 137, page 106)
Reg01	0x0001	R	04h/05h	XS status 1 register (see Table 138, page 107)
Reg02	0x 0002	R	04h/05h	XS device identifier register Low (see Table 139, page 107)
Reg03	0x0003	R	04h/05h	XS device identifier register High (see Table 140, page 107)
Reg04	0x0004	R	04h/05h	XS speed ability register (see Table 141, page 108)
Reg05	0x0005	R	04h/05h	XS devices in package register Low (see Table 142, page 108)
Reg06	0x0006	R	04h/05h	XS devices in package register High (see Table 143, page 108)
–	0x0007	NA	04h/05h	Reserved
Reg07	0x0008	R	04h/05h	XS status 2 (see Table 144, page 109)
–	0x0009 to 0x000d	NA	04h/05h	Reserved
Reg08	0x000e	R	04h/05h	XS package identifier register Low (see Table 145, page 109)
Reg09	0x000f	R	04h/05h	XS package identifier register High (see Table 146, page 109)
–	0x0010 to 0x0017	NA	04h/05h	Reserved
Reg10	0x0018	R	04h/05h	10G XGXS lane status register (see Table 147, page 110)
Reg11	0x0019	R/W	04h/05h	10G XGXS test control register (see Table 148, page 110)
–	0x001a to 0x7fff	NA	04h/05h	Reserved
Reg12	0x8000	R/W	04h/05h	Vendor-specific reset Lo 1 (see Table 149, page 111)
Reg13	0x8001	R/W	04h/05h	Vendor-specific reset Lo 2 (see Table 150, page 111)

Table 136 • **MDIO Registers *(continued)***

Register Name	Register Address	Read / Writable	Device Address	Description
Reg14	0x8002	R/W	04h/05h	Vendor-specific reset Hi 1 (see Table 151, page 111)
Reg15	0x8002	R/W	04h/05h	Vendor-specific reset Hi 1 (see Table 152, page 111)
–	0x8004 to 0xffff	NA	04h/05h	Reserved

The following table lists the bit definitions for the XS Control 1 Register.

Table 137 • **Reg00**

Bit Number	Name	Reset Value	Description
15	Reset	0x0	The XAUI extender block is reset when this bit is set to 1. It returns to 0 when the reset is complete (self-clearing). 1: Block reset 0: Normal operation
14	Loopback	0x0	The XAUI extender block loops the transmit signal back into the receiver. 0: Disable loopback 1: Enable loopback
13	Speed selection	0x1	This bit is for speed selection and is set to 1'b1 for compatibility with clause 22. 0: Unspecified 1: 10 Gbps and above Any write to this bit is ignored.
12	Reserved	0x0	Reserved
11	Low power mode	0x0	When set to 1, the SERDES block is placed in a Low power mode. Set to 0 to return to normal operation. 0: Normal operation 1: Low power mode
[10:7]	Reserved	0x0	Reserved
6	Speed selection	0x1	This bit is set to 1'b1 in order to make compatible with clause 22. 0: Unspecified 1: 10 Gbps and above
[5:2]	Speed selection	0x0	The speed of the PMA/PMD may be selected using bits 5 through 2. 1 x x x: Reserved x 1 x x: Reserved x x 1 x: Reserved 0 0 0 1: Reserved 0 0 0 0: 10 Gbps Any write to this bit is ignored.
[1:0]	Reserved	0x0	–

The following table lists the bit definitions for the XS Status 1 register.

Table 138 • **Reg01**

Bit Number	Name	Reset Value	Description
2	PHY/DTE transmit/receive link status	0x0	When read as a one, the receive link is up. 0: Link down 1: Link up The receive link status bit is implemented with latching low behavior.
1	Low power ability	0x1	When read as a one, it indicates that the Low power feature in supported. 10: Low power not supported 1: Low power is supported
0	Reserved	0x0	–

The following table lists the bit definitions for the XS Device Identifier Low register.

Table 139 • **Reg02**

Bit Number	Name	Reset Value	Description
[15:0]	Organizationally unique identifier (OUI)	0x0	Reg02 and Reg03 provide a 32-bit value, which may constitute a unique identifier for a particular type of SERDES. The identifier is composed of the 3rd through 24th bits of the OUI assigned to the device manufacturer by the IEEE, a 6-bit model number, and a 4-bit revision number. Reg02 sets bits [3:18] of the OUI. Bit 3 of the OUI is located in bit 15 of the unique identifier of the register, and bit 18 of the OUI is located in bit 0 of the register.

The following table lists the bit definitions for the XS Device Identifier High register.

Table 140 • **Reg03**

Bit Number	Name	Reset Value	Description
[15:10]	OUI	0x0	Bits [19:24] of the OUI. Bit 19 of the OUI is located in bit 15 of the register, and bit 24 of the OUI is located in bit 10 of the register.
[9:4]	Manufacturer model number	0x0	Bits [5:0] of the manufacturer model number. Bit 5 of the model number is located in bit 9 of the register, and bit 0 of the model number is located in bit 4 of the register.
[3:0]	Revision number	0x0	Bits [3:0] of the manufacturer model number. Bit 3 of the revision number is located in bit 3 of the register, and bit 0 of the revision number is located in bit 0 of the register.

The following table lists the bit definitions for the XS Speed Ability register.

Table 141 • **Reg04**

Bit Number	Name	Reset Value	Description
[15:1]	Reserved	0x0	Reserved
[9:0]	10g capable	0x1	0: Not 10g capable 1: 10g capable

The following table lists the definitions for the XS Devices in Package Low register.

Table 142 • **Reg05**

Bit Number	Name	Reset Value	Description
[15:6]	Reserved	0x0	Reserved
5	DTE XS present	0x1	0: DTE XS not present in the package 1: DTE XS present in the package
4	PHY XS present	0x0	0: PHY XS not present in the package 1: PHY XS present in the package
3	PCS present	0x0	0: PCS not present in the package 1: PCS present in the package
2	WIS present	0x0	0: WIS not present in the package 1: WIS present in the package
1	PMD/PMA present	0x0	0: PMD/PMA not present in the package 1: PMD/PMA present in the package
0	Clause 22 register present	0x0	0: Clause 22 registers not present in the package 1: Clause 22 registers present in the package

The following table lists the bit definitions for the XS Devices in Package High register.

Table 143 • **Reg06**

Bit Number	Name	Reset Value	Description
15	Vendor-specific device2 present	0x0	0: Vendor-specific device 2 not present 1: Vendor-specific device 2 present
14	Vendor-specific device1 present	0x0	0: Vendor-specific device 1 not present 1: Vendor-specific device 1 present
[13:0]	Reserved	0x0	Reserved

The following table lists the bit definitions for the XS Status 2 register.

Table 144 • **Reg07**

Bit Number	Name	Reset Value	Description
[15:14]	Device present	0x0	10: Device responding at this address. 11: No device responding at this address. 01: No device responding at this address. 00: No device responding at this address.
[13:12]	Reserved	0x0	Reserved
11	Transmit fault	0x0	0: No transmit fault 1: Transmit fault Latched High, clear on read
10	Receive fault	0x0	0: No receive fault 1: Receive fault Latched High, clear on read
[9:0]	Reserved	0x0	Reserved

Note: Transmit fault and receive fault status bit roles are swapped when the MDIO device ID is set to 5'h04 (PHY equipment). The descriptions in preceding table assume that the MDIO device ID is set for DTE (Data Terminal Equipment).

The following table lists the bit definitions for the XS Package ID Low register.

Table 145 • **Reg08**

Bit Number	Name	Reset Value	Description
[15:0]	OUI	0x0	Reg08 and Reg 09 provide a 32-bit value, which may constitute a unique identifier for a particular type of package that the SERDES is instantiated within. The identifier is composed of the 3rd through 24th bits of the OUI assigned to the package manufacturer by the IEEE, plus a 6-bit model number, and a 4-bit revision number. Reg08 sets bits [3:18] of the OUI. Bit 3 of the OUI is located in bit 15 unique identifier of the register, and bit 18 of the OUI is located in bit 0 of the register.

The following table lists the bit definitions for the XS Package ID High register.

Table 146 • **Reg09**

Bit Number	Name	Reset Value	Description
[15:10]	OUI	0x0	Bits [19:24] of the OUI. Bit 19 of the OUI is located in bit 15 of the register, and bit 24 of the OUI is located in bit 10 of the register.
[9:4]	Manufacturer model number	0x0	Bits [5:0] of the manufacturer model number. Bit 5 of the model number is located in bit 9 of the register, and bit 0 of the model number is located in bit 4 of the register.
[3:0]	Revision number	0x0	Bits [3:0] of the manufacturer model number. Bit 3 of the revision number is located in bit 3 of the register, and bit 0 of the revision number is located in bit 0 of the register.

The following table lists the bit definitions for the XGXS Lane Status register.

Table 147 • **Reg10**

Bit Number	Name	Reset Value	Description
[15:13]	Reserved	–	Reserved
12	PHY/DTE XGXS lane alignment status	0x0	0: Lanes not aligned 1: Lanes aligned
11	Pattern testing ability	0x1	0: (PHY/DTE)XS is unable to generate test patterns 1: (PHY/DTE)XS is able to generate test patterns
10	PHY XGXS loopback ability	0x1	0: PHY XGXS does not has the ability to perform a loopback 1: PHY XGXS has the ability to perform a loopback
[9:4]	Reserved	–	Reserved
3	Lane3 synchronized	0x0	When read as a one, this register indicates that the receive Lane3 is synchronized. 0: Lane3 is not synchronized 1: Lane3 is synchronized
2	Lane2 synchronized	0x0	When read as a one, this register indicates that the receive Lane2 is synchronized. 0: Lane2 is not synchronized 1: Lane2 is synchronized
1	Lane1 synchronized	0x0	When read as a one, this register indicates that the receive Lane1 is synchronized. 0: Lane1 is not synchronized 1: Lane1 is synchronized
0	Lane0 synchronized	0x0	When read as a one, this register indicates that the receive Lane1 is synchronized. 0: Lane0 is not synchronized 1: Lane0 is synchronized

The following table lists the bit definitions for the XGXS Test Control register.

Table 148 • **Reg11**

Bit Number	Name	Reset Value	Description
[15:3]	Reserved	–	Reserved
2	Transmit test pattern enabled	0x0	When this bit is set to a one, pattern testing is enabled on the transmit path. 0: Transmit/receive test pattern disabled 1: Transmit/receive test pattern enabled
[1:0]	Test pattern select	0x0	The test pattern is used when enabled pattern testing is selected using these bits: 00: High frequency test pattern 01: Low frequency test pattern 10: Mixed frequency test pattern 11: Reserved

The following table lists the bit definitions for the Vendor-Specific Reset Low 1 register.

Table 149 • **Reg12**

Bit Number	Name	Reset Value	Description
[15:0]	Vendor-specific reset Lo 1	0x0000	General purpose registers that are connected to the output port. XAUI_VNDRRESLO[15:0]. Typically used for external device control.

The following table lists the bit definitions for the Vendor-Specific Reset Low 2 register.

Table 150 • **Reg13**

Bit Number	Name	Reset Value	Description
[15:0]	Vendor-specific reset Lo 2	0x0000	General purpose registers that are connected to the output port. XAUI_VNDRRESLO[31:16]. Typically used for external device control.

The following table lists the bit definitions for the Vendor-Specific Reset High 1 register.

Table 151 • **Reg14**

Bit Number	Name	Reset Value	Description
[15:0]	Vendor-specific reset Hi 1	0xFFFF	General purpose registers that are connected to the output port. XAUI_VNDRRESLI[15:0]. Typically used for external device control.

The following table lists the bit definitions for the Vendor-Specific Reset High 2 register.

Table 152 • **Reg15**

Bit Number	Name	Reset Value	Description
[15:0]	Vendor-specific reset Hi 2	0xFFFF	General purpose registers that are connected to the output port. XAUI_VNDRRESLI[31:16]. Typically used for external device control.

4.8 SERDES Block System Register Configurations for XAUI Mode

The SmartFusion2 and IGLOO2 SERDESIF block subsystem has three regions of configuration and status registers:

- SERDES Block System Register
- Bridge Register Space
- SERDES Block–I/O Signal Interface

These registers are accessed by the 32-bit APB bus. Refer to the “SERDESIF Block System Register” for details. In XAUI mode, the PCIe core registers are not used. Only the SERDES block system registers and SERDES block register are used for XAUI mode. The XAUI block also has MDIO registers, which are accessed via MDIO interface signals.

The SERDESIF block system registers occupy 1 KB of the configuration memory map. However, in XAUI mode, only subsets of the register are used. SERDES registers can be referenced in the “SERDESIF Block System Register”.These registers can be updated through the 32-bit APB interface after power-up.

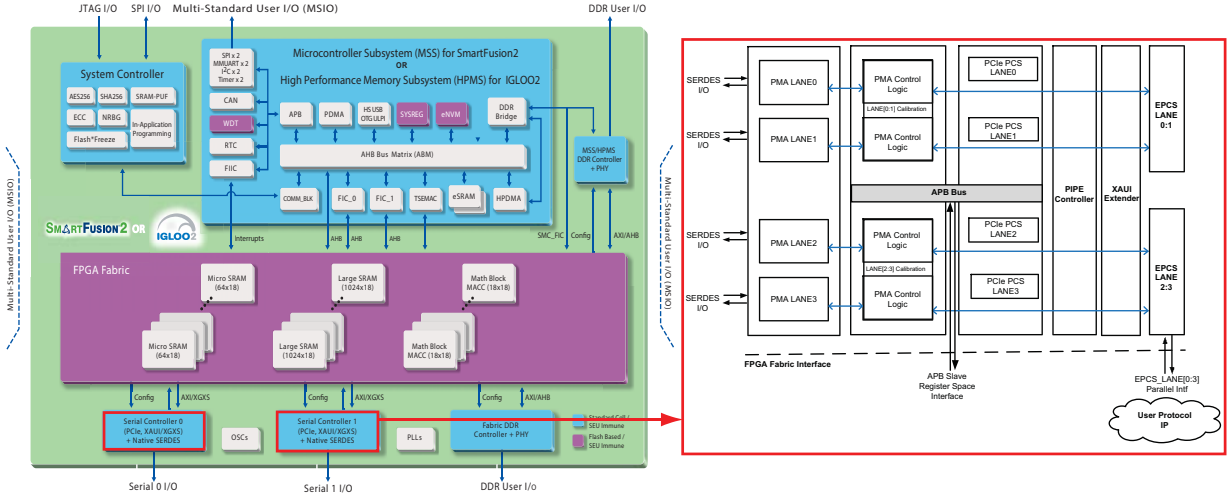
5 EPCS Interface

5.1 Introduction

This chapter describes using the EPCS interface in the SmartFusion2 and IGLOO2 FPGA SERDESIF blocks.

The SERDESIF block integrates the functionality of supporting multiple high speed serial protocols, such as peripheral component interconnect express (PCIe) 2.0, extended attachment unit interface (XAUI), and EPCS interfaces, as shown in the following figure. The SERDESIF block can be configured in various modes, including EPCS mode. In EPCS mode, Lane0 and Lane1 (L01) EPCS interface and Lane2 and Lane3 (L23) EPCS interface are exposed to the fabric and configures serializer/de-serializer (SERDES) in physical media attachment (PMA) only mode. The PCIe and XAUI PCS logic in SERDES is bypassed, however, the PCS logic can be implemented in the FPGA fabric and the EPCS interface signals of the SERDES block can be connected. This allows any user-defined high-speed serial protocol to be implemented in the SmartFusion2 and IGLOO2 device.
TU0570: Implementing a SmartFusion2 and IGLOO2 SERDES EPCS Protocol Design Tutorial demonstrates the features capabilities.

Figure 49 • **SmartFusion2 and IGLOO2 SERDESIF Block Diagram**



5.1.1 Features

The main features of the EPCS interface in SmartFusion2 and IGLOO2 are:

- Up to 20-bit Rx/Tx EPCS Interface to the FPGA fabric
- Allows the FPGA fabric to directly access the PMA block bypassing the PCIe PCS block in SERDES and thus allow implementing any serial protocol for up to four lanes using the PCS logic in the fabric
- Allows the FPGA fabric to access the SERDES register through the APB interface and allows programming various PMA settings, including programming of the SERDES Tx PLL and Rx PLLs settings

5.1.2 Device Support

The SmartFusion2 or IGLOO2 family has a number of devices available. The following table lists the total number of SERDESIF blocks available in each SmartFusion2 and IGLOO2 device that can be configured to support the EPCS interface.

Table 153 • **Available SERDESIF Blocks in SmartFusion2 and IGLOO2 Devices that support EPCS**

	M2S/M2GL 005	M2S/M2GL 010	M2S/M2GL 025	M2S/M2GL 050	M2S/M2GL 060	M2S/M2GL 090	M2S/M2GL 150
SERDESIF available for EPCS	0	1	1	Up to 2	1	1	Up to 4
SERDES Lanes	0	4	4	8	4	4	16

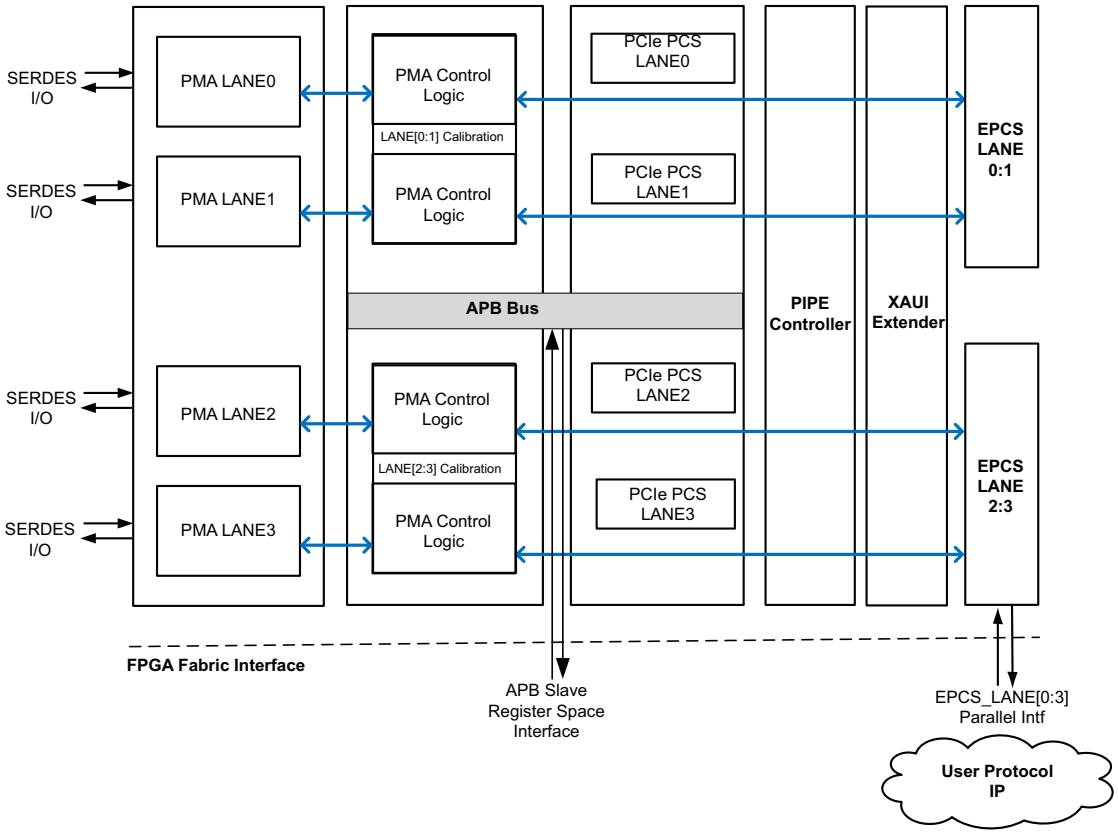
Note: The specified number of SERDESIF blocks varies depending on the device package.

Note: M2S/M2GL060/090 application interfaces have dual controller capability supporting up to two x1 or x2 PCIe endpoints within a SERDESIF.

5.1.3 SmartFusion2/IGLOO2 EPCS Interface

The SERDESIF block can be configured in EPCS mode.This allows the FPGA fabric to directly access the SERDES block. The PCS logic of SERDES is bypassed in this mode to allow user-defined protocol to be supported from the FPGA fabric. The following figure shows an application example using the EPCS interface. Refer to Figure 49, page 112 for the EPCS interface signals.

Figure 50 • **Application Using SERDESIF EPCS Interface**



The SERDESIF block can be configured to operate in two different modes: Single-protocol mode and Multi-protocol mode. In Single-protocol mode, the EPCS interface can be configured as x4 or x2 or x1 lane. In Multi-protocol mode, the SERDESIF block can operate using dedicated Lane2 and Lane3 in EPCS mode, while Lane1 and Lane2 are dedicated to the PCIe protocol link implementation. Table 154, page 114, Table 155, page 114, and Table 158, page 119 show a detailed description of the EPCS interface usage in Single-protocol and Multi-protocol mode.Multi-EPCS protocol allows up to all four Lanes to be used independently for customized protocols such as SGMII or JESD204b.

Table 154 • **EPCS Interface Usage in Single-Protocol and Multi-Protocol Mode**

Mode	Protocol	Description
Single-protocol	EPCS Protocol	Configured to use maximum 4 lanes. In EPCS mode, the user- defined serial protocol implemented within the FPGA fabric is connected though the EPCS interface.
Multi-protocol	PCIe protocol and EPCS protocol	Configured to use x2 and x1 lane in PCIe mode. (lane0 and lane1 are used for the PCIe link). Any user-defined/other serial protocol connected to the EPCS interface uses lane2 and lane3 for this purpose.
Multi-EPCS	–	Configure multiple independent EPCS protocols across Lane[0:1] and Lane[2:3].

Table 155 • **EPCS Interface and SERDES Lane Mapping in Single-Protocol Mode**

	Lane0	Lane1	Lane2	Lane3
Single Protocol EPCS Mode	EPCS	–	–	–
	EPCS	EPCS	–	–
	–	–	–	EPCS
	–	–	EPCS	EPCS
	EPCS	EPCS	EPCS	EPCS

Note: These are only examples above. Single non-bonded EPCS protocols can occupy any lanes.

Table 156 • **EPCS Interface and SERDES Lane Mapping in Multi-Protocol Mode**

PHY-MODE EPCS Multi- Protocol	PHYSICAL SERDES LANES/LOGICAL LANES			
	LANE-0	LANE-1	LANE-2	LANE-3
	Protocol	Protocol	Protocol	Protocol
M2S/M2GL010/025/050/150				
Multi Protocol PHY – Mode (PCIe link Non-Reversed-Mode)	PCIe	[1]	EPCS	EPCS
	PCIe	PCIe	EPCS	EPCS
Multi Protocol PHY – Mode (PCIe link Reversed-Mode)	[1]	PCIe	EPCS	EPCS
	PCIe	PCIe	EPCS	EPCS
M2S/M2GL060/090				
Multi and Dual PCIe Protocol PHY – Mode (Non-Reversed-Mode)	PCIe_0	PCIe_1	EPCS	EPCS

1. Refers to unused lanes.

Note: When operating the EPCS mode in x2 or x4 the lanes are not bound together. The x2 and x4 nomenclature simple indicates the number of lanes in the SERDESIF which are active. The High Speed Serial Interface Configurator allows the user to select rate of each channel independently.

5.2 Getting Started

This section provides an overview of how to configure the SERDESIF block in EPCS mode and instructions for using the EPCS interface.

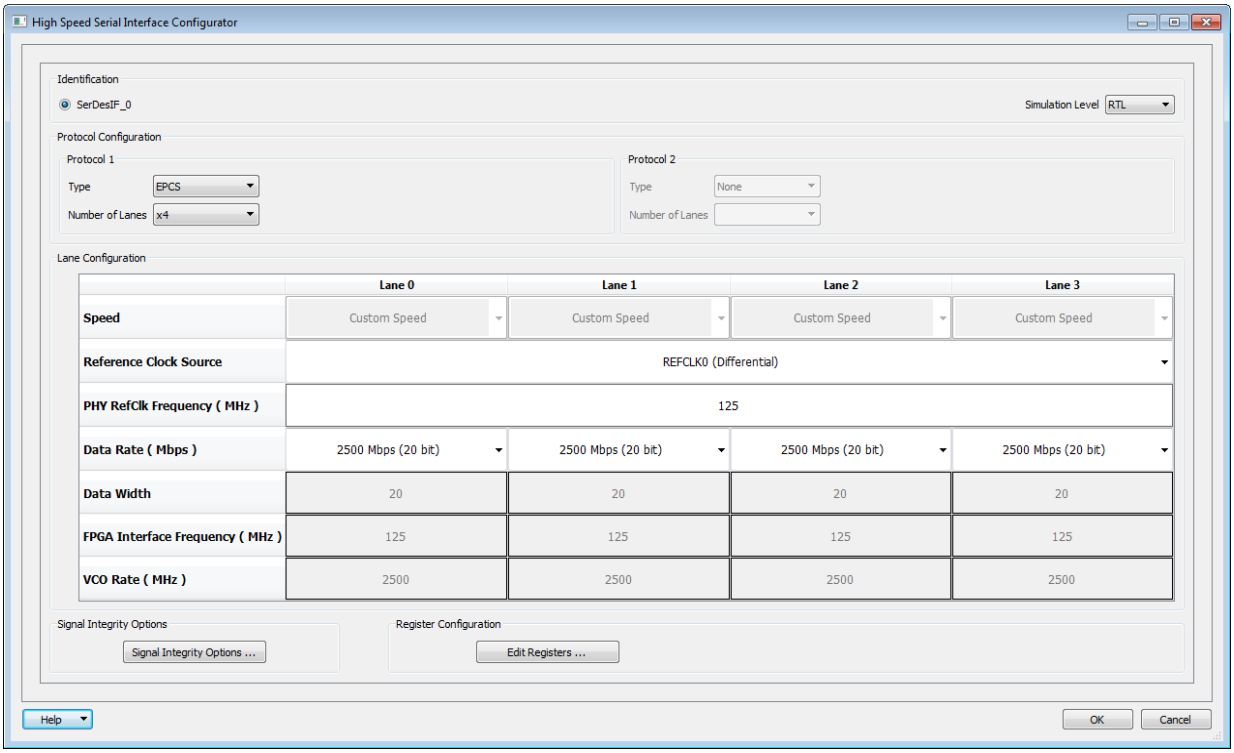
The following sections show how to use SERDESIF in EPCS mode by completing the following steps:

- Using High Speed Serial Interfaces Configurator in EPCS Mode
- Simulating SERDESIF in EPCS Mode
- Create an Application in EPCS Mode

5.2.1 Using High Speed Serial Interfaces Configurator in EPCS Mode

The high speed serial interfaces configurator (SERDESIF configurator) in the Libero SoC software allows configuring the SERDESIF block with EPCS mode in Single-protocol mode or Multi-protocol mode. Refer the following figure for setting SERDESIF configurator in the EPCS with Single-protocol mode, that shows configuring all 4 lanes in EPCS mode. Refer to Figure 52, page 116 for setting EPCS with Multi-protocol mode. EPCS data rates use a CUSTOM EPCS option within the SERDESIF configurator as shown in Figure 52, page 116. This feature allows designers to opportunity to craft customized SERDES implementations based on reference clock and data widths. This implementation allows for data rate targeted applications to be easily setup while the GUI Configurator manages the limitations of the SERDESIF block.

Figure 51 • **EPCS Mode Setting (Single-protocol mode) in SERDESIF Configurator**



SERDESIF EPCS_TX/RXDATA width supports a maximum data bus width of 20-bits. The SERDESIF supports several options for the data bus widths such as 4, 5, 8, 10, 16, and 20 bit. These are all valid options based on the data rate.

EPCS interface is always modeled as a 20-bit bus. When using less than 20-bits, the most recently received word is nearer RxDO[19] than it is RxDO[0], so PCS words are justified starting from RxDO[19], not RxDO[0]. Such is the case for all supported bus widths. Users need to split or slice the ports on the module to only connect the necessary ports. For RX bus orientation is always [n-1:0]. Whereas TX, it's orientation is upper towards lower. User must be cautious when the width is not 20 such as a bus width of 8 for instance.

Table 157 • **Data Bus Widths**

Bus Width	RX Data Bus Description	TX Data Bus Description
20-bit	[19:0]	[19:0]
16-bit	[19:4]	[15:0]
10-bit	[19:10]	[9:0]
8-bit	[19:12]	[7:0]
5-bit	[19:15]	[4:0]
4-bit	[19:16]	[3:0]

Figure 52 • **EPCS Mode Setting (Multi-protocol mode) in SERDESIF Configurator**

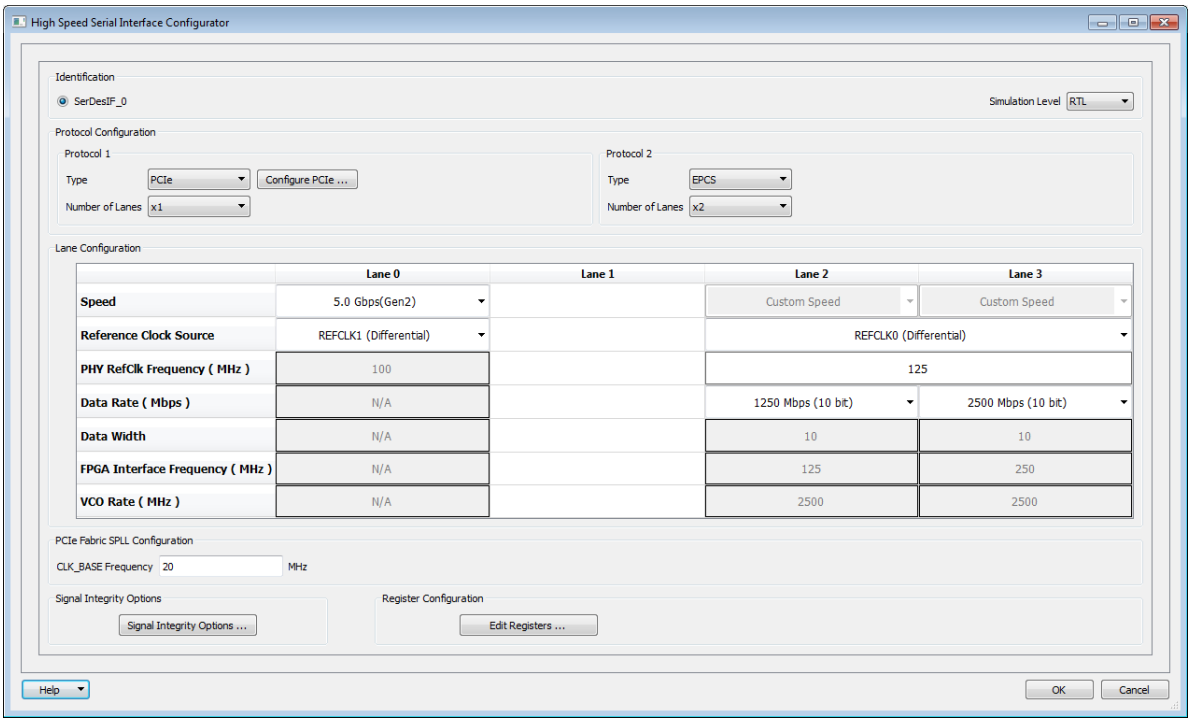
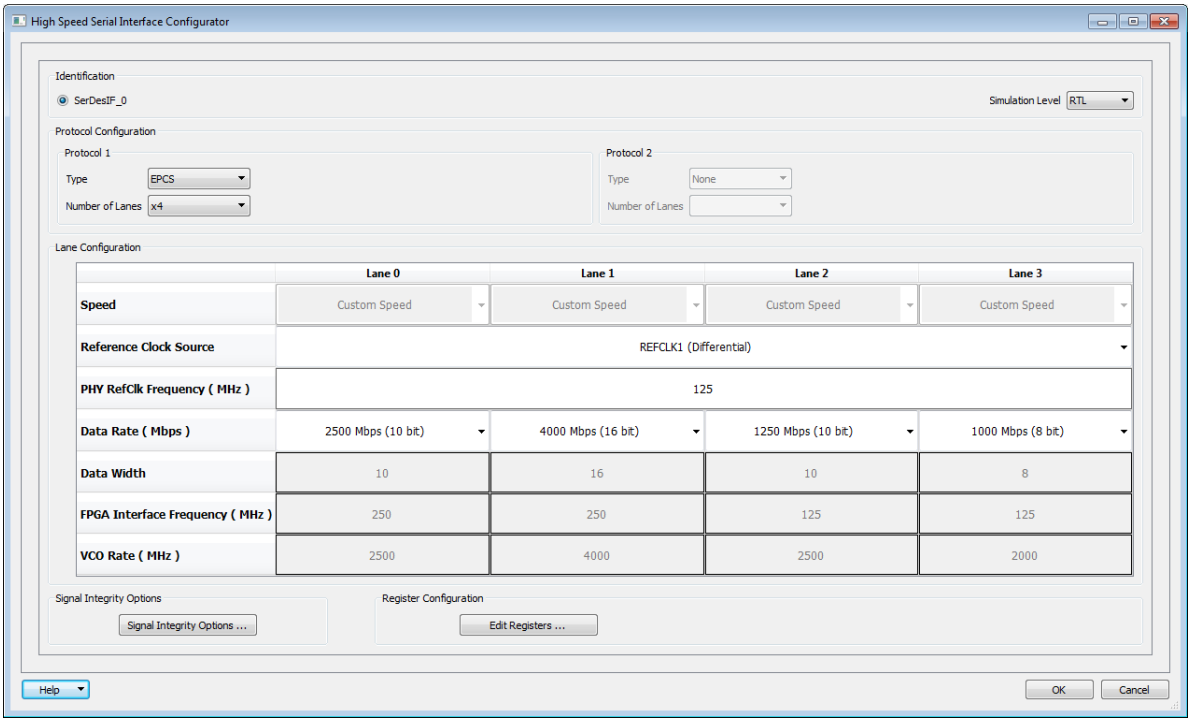


Figure 53 • **EPCS Custom Speed Mode in SERDESIF Configurator**



Following are the brief descriptions of the configuration options (refer to the SERDESIF Block, page 4 for details).

5.2.1.1 Protocol Selection

The following settings are used for protocol selection:

- Protocol 1 or Protocol 2 Type–Select protocol settings. Select **EPCS** or **PCIe** from the drop-down based on Single-protocol or Multi-protocol mode.
- Number of Lanes–Select number of lanes used.
- Speed–Select the lane speed.
- Protocol 1 or Protocol 2 PHY reference Clock–Select the inputs for the PHY reference clock selection. Refer to the SERDES Reference Clock Selection, page 122 for details on PHY reference clock selection.

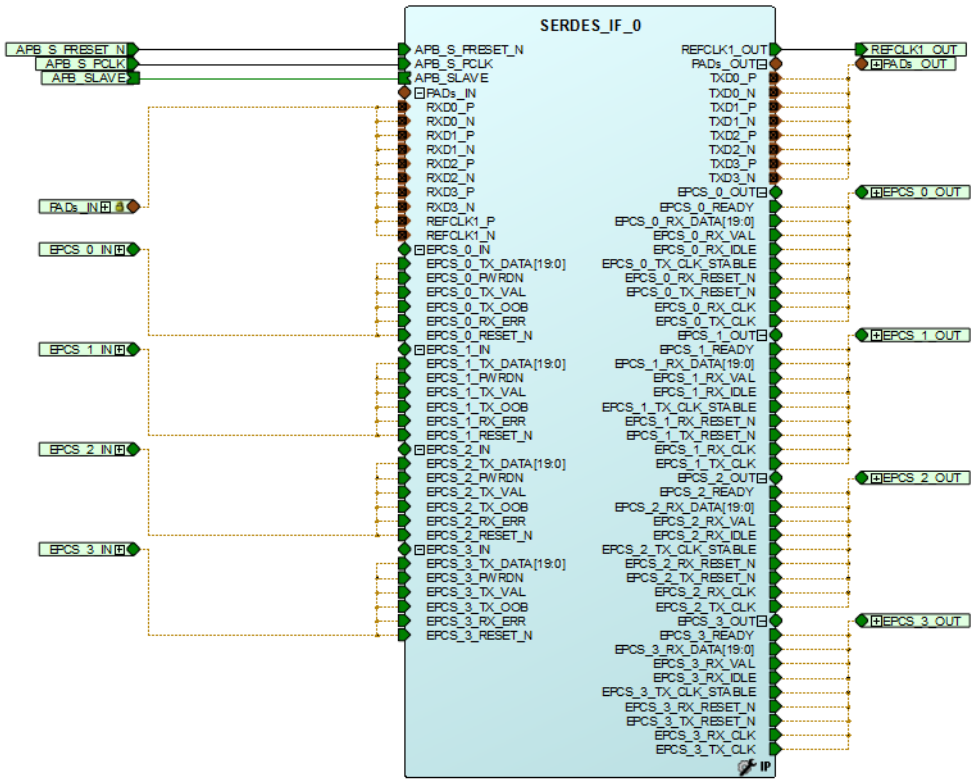
5.2.2 Simulating SERDESIF in EPCS Mode

The SERDESIF block, when configured in EPCS requires the RTL simulation model which is selectable in the High Speed Serial Interfaces Configurator. If a SERDES lane is not used, in EPCS simulations, it's data output is unknown.

5.2.3 Create an Application in EPCS Mode

A complete application in SmartFusion2 and IGLOO2 require a properly set SERDESIF Configurator and then to generate the SERDESIF block in EPCS mode. The following figure shows the SERDESIF block configured in EPCS mode in all four lanes. Libero promotes the SERDES I/Os to top level and it exposes the EPCS interface for each lane into the FPGA fabric; the SERDESIF block exposes the APB3 interface to the FPGA fabric as well.

Figure 54 • **Libero SoC Showing SERDESIF in EPCS Mode**



Fabric logic or FPGA IP is required to be connected to the EPCS interface as shown in Figure 50, page 113.

5.3 SERDESIF Architecture in EPCS Mode

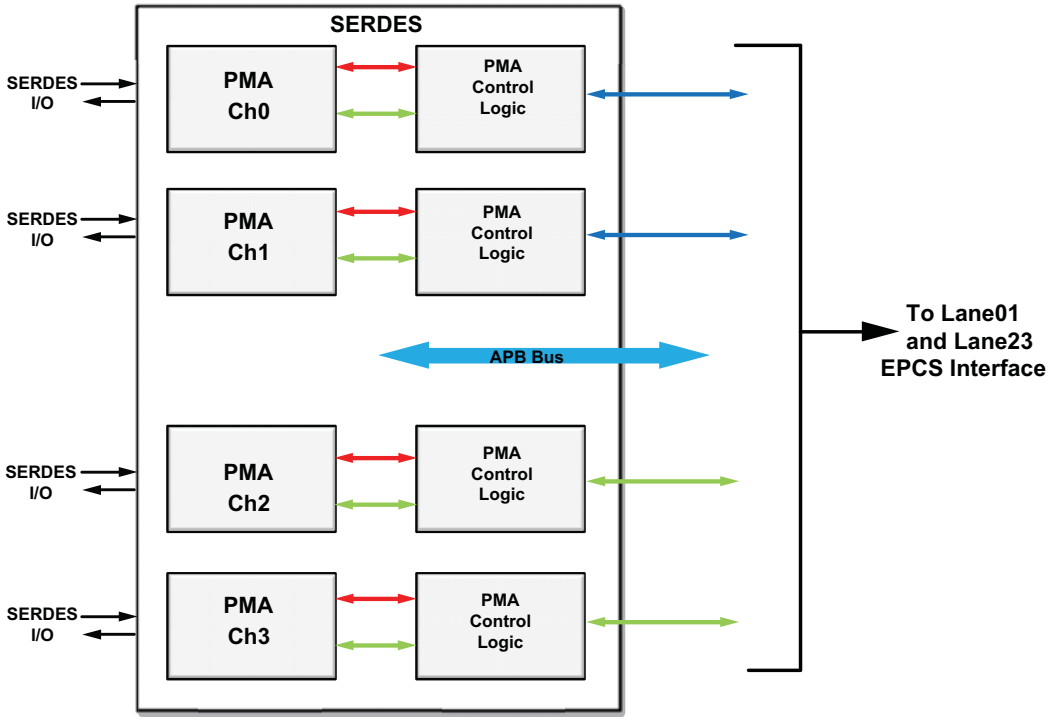
This section provides an overview of the SERDESIF in EPCS mode. The following topics are covered:

- SERDESIF Block in EPCS Mode
- SERDESIF Fabric Interface in EPCS Mode

5.3.1 SERDESIF Block in EPCS Mode

The following figure shows the SERDESIF block internal architecture during EPCS Single-protocol mode. EPCS mode facilitates the use of four lanes of the SERDES which are exposed to the FPGA fabric with a 20-bit EPCS interface per lane. EPCS mode gives full control over the PLL configurations in the SERDES using the APB interface to generate the required serial link frequencies. The following figure shows the SERDES block in EPCS mode. The EPCS interface is suitable for running any protocol, including Ethernet MAC and PCS in the FPGA fabric. The PMA macro is used for implementing any standard (SRIO, JESD204 and so on) or user-defined serial protocol.

Figure 55 • **SERDES in EPCS Mode**



5.3.2 SERDESIF Fabric Interface in EPCS Mode

The SERDESIF in EPCS mode interfaces with the fabric and differential I/O pads. The following sections list the fabric interfaces:

- EPCS interface signals
- APB interface signals

Table 158 • **SERDESIF Block – EPCS Interface**

Port	Type	Description
EPCS_0_RESET_N EPCS_1_RESET_N EPCS_2_RESET_N EPCS_3_RESET_N	Input	Active low PHY RESET. These inputs reset the associated SERDES logic for each lane. The resets are logically ordered with the power up signal.
EPCS_0_READY EPCS_1_READY EPCS_2_READY EPCS_3_READY	Output	PHY ready: This signal is asserted when the PHY has completed the calibration sequence for each specific lane. This signal can be used to release the reset for the external PCS and controller, start transmitting data to the PMA, or any other purpose.

Table 158 • **SERDESIF Block – EPCS Interface *(continued)***

Port	Type	Description
EPCS_0_PWRDN EPCS_1_PWRDN EPCS_2_PWRDN EPCS_3_PWRDN	Input	PHY power-down: This signal is used to put the PMA in power-down state where RX CDR PLL is bypassed and other low power features are applied to the PMA. When exiting power-down, no calibration is required and the link can be operational much faster than when using the EPCS_X_TX_OOB or EPCS_X_RESETN signals. These signals are active high.
EPCS_0_TX_OOB EPCS_1_TX_OOB EPCS_2_TX_OOB EPCS_3_TX_OOB	Input	PHY transmit out-of-band (OOB): This signal is used to load electrical idle III in the TX driver of the PMA macro. It can be used for serial advanced technology attachment (SATA) as part of the sequencing for transmitting very short OOB signaling. These signals are active high. Minimum transfer burst size is 23 symbols.
EPCS_0_TX_VAL EPCS_1_TX_VAL EPCS_2_TX_VAL EPCS_3_TX_VAL	Input	PHY transmit valid: This signal is used to transmit valid data. If deasserted, the PMA macro is put in electrical idle 1. It can be used for protocols requiring electrical idle (SATA) and must also be deasserted as long as EPCS_X_READY is not asserted. This signal must be generated one clock cycle earlier than corresponding EPCS_TXDATA signals. These signals are active high.
EPCS_0_TX_DATA[19:0] EPCS_1_TX_DATA[19:0] EPCS_2_TX_DATA[19:0] EPCS_3_TX_DATA[19:0]	Input	PHY transmit data: This signal is used to transmit data. This signal is always 20 bits per lane, but the SERDESIF only uses the number of bits selected in the High Speed Serial Interfaces Configurator.
EPCS_0_TX_CLK EPCS_1_TX_CLK EPCS_2_TX_CLK EPCS_3_TX_CLK	Output	PHY transmit clock: This clock signal is generated by the TX PLL in the PMA macro and must be used by the external PCS logic to provide data on EPCS_X_TX_DATA.
EPCS_0_TX_RESET_N EPCS_1_TX_RESET_N EPCS_2_TX_RESET_N EPCS_3_TX_RESET_N	Output	PHY clean active low reset on the TX clock. This signal is a clean version of the EPCS_X_RESET_N signal, which has a clean deassertion timing versus EPCS_TXCLK.
EPCS_0_RX_CLK EPCS_1_RX_CLK EPCS_2_RX_CLK EPCS_3_RX_CLK	Output	PHY receive clock: This clock signal is generated by the RX PLL in the PMA macro and must be used by the external PCS logic to provide data on EPCS_X_RX_DATA.
EPCS_0_RX_RESET_N EPCS_1_RX_RESET_N EPCS_2_RX_RESET_N EPCS_3_RX_RESET_N	Output	PHY clean active low reset on EPCS_X_RX_CLK. This signal is a clean version of the EPCS_X_RESET_N signal, which has a clean deassertion timing versus EPCS_X_RX_CLK.
EPCS_0_RX_VAL EPCS_1_RX_VAL EPCS_2_RX_VAL EPCS_3_RX_VAL	Output	PHY receive valid: This signal is used to signal receive valid data. It corresponds to the two conditions completed by the PMA control logic: Receiver detects incoming data (not in electrical idle) CDR PLL is locked to the input bit stream in fine grain state
EPCS_0_RX_IDLE EPCS_1_RX_IDLE EPCS_2_RX_IDLE EPCS_3_RX_IDLE	Output	PHY Receive Idle: This signal is used to signal an electrical idle condition detected by the PMA control logic. Note that this signal is generated on EPCS_X_TX_CLK of the selected lane.
EPCS_0_RXDATA[19:0] EPCS_1_RXDATA[19:0] EPCS_2_RXDATA[19:0] EPCS_3_RXDATA[19:0]	Output	PHY receive data: This signal is always 20 bits per lane and the external PCS can use any number of these bits for its application. The SERDESIF only uses the number of bits selected in the High Speed Serial Interfaces Configurator.

Table 158 • **SERDESIF Block – EPCS Interface *(continued)***

Port	Type	Description
EPCS_0_TX_CLK_STABLE EPCS_1_TX_CLK_STABLE EPCS_2_TX_CLK_STABLE EPCS_3_TX_CLK_STABLE	Output	Active high to signal to indicate EPCS interface Lane_X clock is stable, meaning when TX PLL is locked.
EPCS_0_RX_ERR EPCS_1_RX_ERR EPCS_2_RX_ERR EPCS_3_RX_ERR	Input	EPCS interface Lane_X receiver error is detected when using the external logic. When there are many receive errors such as, invalid 8b/10b code or disparity error, then the asynchronous signal can be used to cause the CDRPLL to switch back to the frequency lock phase. These pins can be hardwired to 0 and rely only on Electrical Idle detection to switch the CDR PLL back to frequency lock state.

5.4 Reset and Clocks

This section covers the functional aspects of the reset and clock circuitry inside the SERDESIF block in EPCS mode. The following topics are covered:

- EPCS Mode Clocking
- EPCS Mode Reset Network

5.4.1 EPCS Mode Clocking

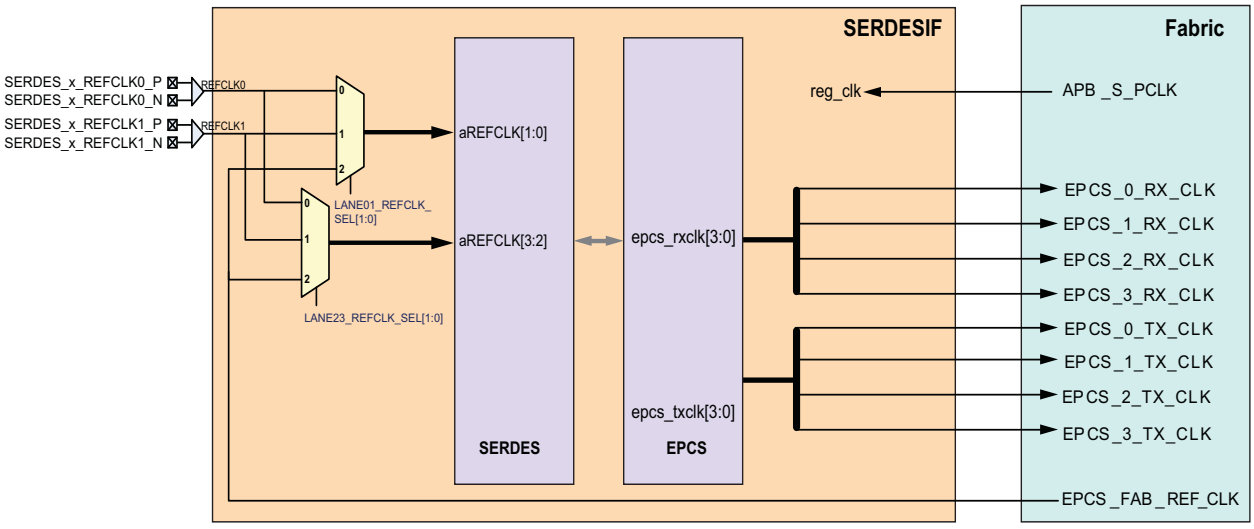
When the SERDESIF block is configured in EPCS mode, it has multiple clock inputs and outputs. This section describes the EPCS clocking scheme.

5.4.1.1 SERDESIF Clock Network in EPCS

The following figure shows the SERDESIF clock network in EPCS mode.

In EPCS mode, data is exchanged between the fabric and SERDESIF. The following figure shows the SERDESIF clock network in EPCS mode. The SERDES PMA has two PLLs (Tx PLL and CDR PLL) that generate the required clock frequency and send 4-Rx and 4-Tx clocks for each lane through the EPCS interface. User design is required to use these clocks within the FPGA fabric, in custom logic or FPGA IP, to transfer data between SERDESIF and FPGA fabric. There is an additional clock (asynchronous) dedicated to the APB interface, called APB_S_PCLK, used for accessing the SERDESIF block registers.

Figure 56 • **SERDESIF Clocking in EPCS Mode**



The following table lists the various clocks in EPCS mode.

Table 159 • **Clock Signals in EPCS Mode**

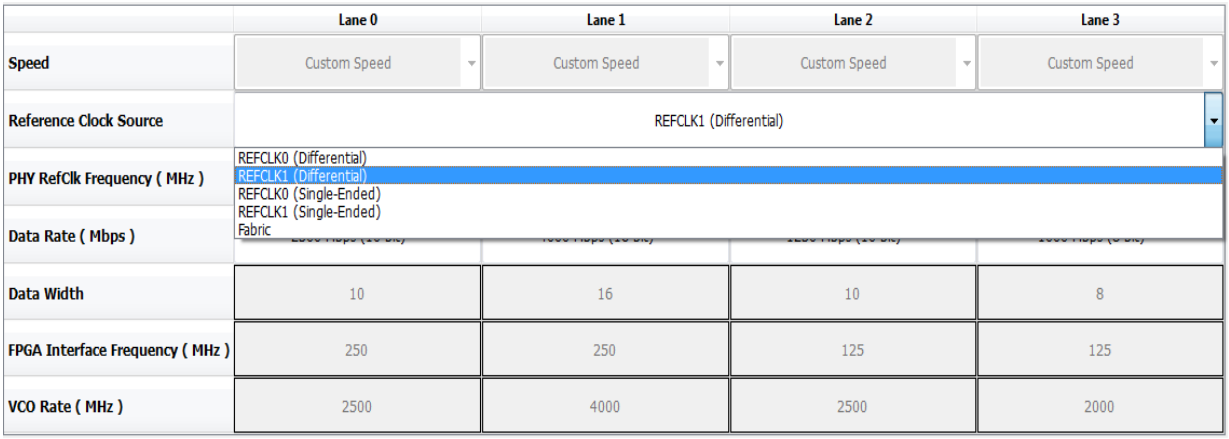
Clock Signal	Description
aREFCLK	Reference clock for SERDES
epcs_x_TX_CLK	EPCS interface LaneX (X = 0, 1, 2, 3) Tx Clock
epcs_x_RX_CLK	EPCS interface LaneX (X = 0, 1, 2, 3) Rx Clock
APB_S_PCLK	PCLK for APB interface
REFCLK0_P REFCLK0_N	Differential reference clock input I/O Port0
REFCLK1_P REFCLK1_N	Differential reference clock input I/O Port1
EPCS_FAB_REF_CLK	Fabric reference clock for SERDES PMA

5.4.1.2 SERDES Reference Clock Selection

The PMA in the SERDES block needs a reference clock on each of its lanes for Tx and Rx clock generation through the PLLs. It has three options for the reference clock. The reference clock for the four lanes comes from the I/O Port0 (REFCLK0) or I/O Port1 (REFCLK1) I/O pads or from FPGA fabric. Lane0 and lane1 share the same reference clock; lane2 and lane3 share the same reference clock, or alternatively the same clock can be shared among all four lanes. The reference clock pads are differential input.

The following figure shows the reference clock selection in the SERDESIF Configurator available in Libero. I/O Port0 selects REFCLK0 and I/O Port1 selects REFCLK1.

Figure 57 • **SERDES Reference Clock Using SERDESIF Configurator**



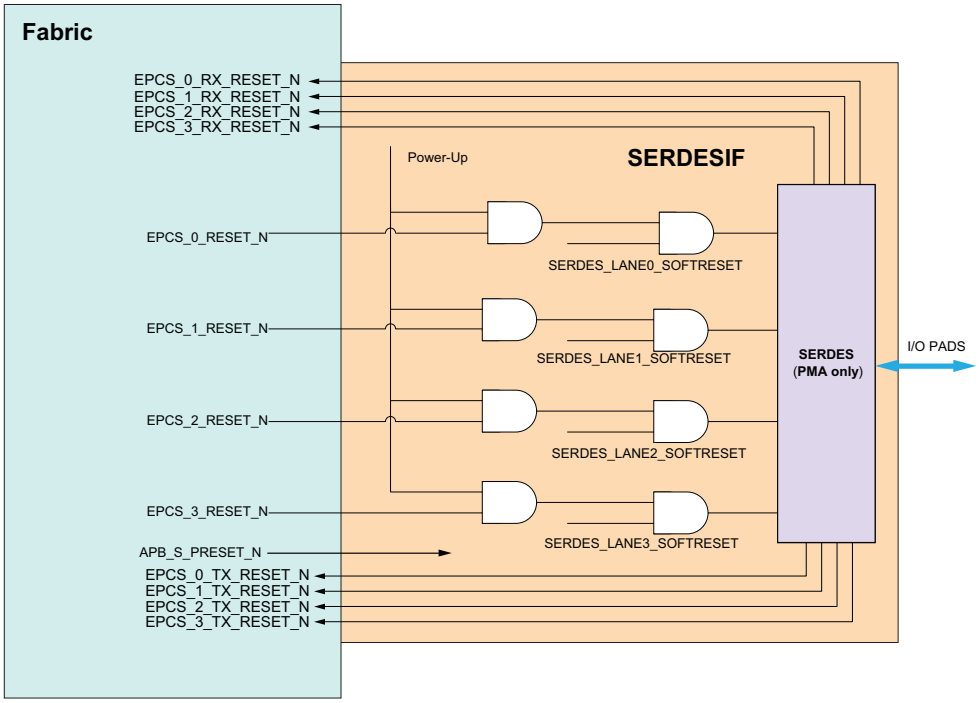
EPCS Protocol 1 and Protocol 2 can use separate reference clocks using the REFCLK0 and REFCLK1 inputs. In the case of single channel protocols, the two channels cannot be adjacent because of REF CLK sharing.

5.4.2 EPCS Mode Reset Network

The following figure shows the reset network for the EPCS interface x4 lanes implementation. The resets for all four lanes (EPCS_0_RESET_N, EPCS_1_RESET_N, EPCS_2_RESET_N, and EPCS_3_RESET_N) are gated with the power valid signal from SmartFusion2 or IGLOO2 program control and again with SERDES_LANEx_SOFTRESET (x = 0, 1, 2, 3).

The SERDES_LANEx_SOFTRESET signals are controlled by the SERDESIF System Register, page 153, SERDESIF_SOFT_RESET, (active low reset signal for each lane), which can be programmed using the APB3 interface. On the output side, the SERDESIF block in EPCS mode generates 4 sets of reset signals for each lane.

Figure 58 • **SERDESIF Reset Signals in EPCS Mode**



The following table lists the reset signals and recommended connections.

Table 160 • **SERDESIF Reset Signals in EPCS Mode**

Port	Type	Description
EPCS_X_RESET_N[1]	Input	EPCS interface LaneX (X = 0, 1, 2, 3) clean reset, de-asserted on EPCS_X_RX_CLK
APB_S_PRESET_N	Input	APB asynchronous reset to all APB registers
EPCS_X_Rx_RESET_N	Output	EPCS interface LaneX (X = 0, 1, 2, 3) reset output, de-asserted on EPCS_X_RX_CLK
EPCS_X_Tx_RESET_N	Output	EPCS interface LaneX (X = 0, 1, 2, 3) reset output, deasserted on EPCS_X_TX_CLK

1. EPCS_X_RESET_N requires a minimum pulse width two times of EPCS_X_[TX:RX]CLK time period. For example, if EPCS_X_TXCLK/EPCS_X_RXCLK is 125 MHz [8 ns], then the minimum low going pulse width for EPCS_X_RESET_N must be 16 ns.

5.5 Design Consideration

This section provides instruction for using the EPCS interface in SmartFusion2 and IGLOO2 devices. The following topics are covered:

- EPCS Interface: Timing Diagram
- EPCS SERDES Calibration and External Resistor Configuration
- Implementing SGMII Using EPCS Interface

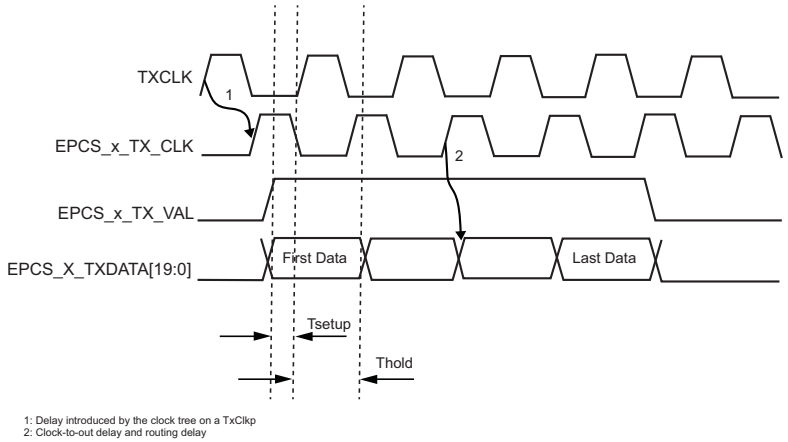
5.5.1 EPCS Interface: Timing Diagram

The Tx clock and Rx clock from SERDES are sent to the EPCS interface through the EPCS_x_TX_CLK (x = 0, 1, 2, 3) and EPCS_x_RX_CLK (x = 0, 1, 2, 3) output signals. These signals must be used as transmit clock and receive clock by the external PCS logic. It is recommended to use either a global clock (CLKINT) or a regional clock (RCLKINT) in the fabric for these signals.

Note: The EPCS_x_RX_CLK requires a clock resource which has low clock injection time into the fabric. In the smaller arrays sizes of the 010 and 025 either an RCLKINT or CLKINT can be used for the EPCS_x_RX_CLK. For the larger arrays sizes of the 050, 060, 090, and 150 an RCLKINT must be used. The CLKINT in the larger devices produces too much delay and does not allow for hold timing closure.

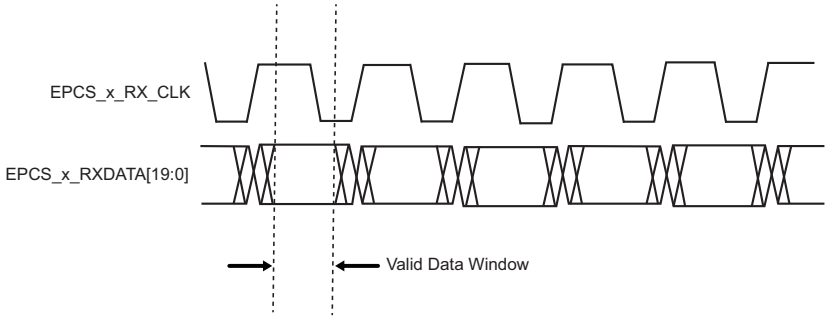
The transmit data EPCS_X_TXDATA (x = 0, 1, 2, 3) must be generated using the rising edge of EPCS_x_TX_CLK as it is sampled by the SERDES block, as shown in the following figure. To constrain the place and route of the EPCS interface, the user needs to provide a clock constraint on the EPCS_x_TX_CLK signal. This ensures that the setup and hold timing will be met across the interface.

Figure 59 • **EPCS Transmit Interface Timing Diagram**



EPCS_x_RXDATA (x = 0, 1, 2, 3) is sampled in the FPGA Fabric using the rising edge of EPCS_x_RX_CLK (x = 0, 1, 2, 3), as shown in the following figure. To constrain the place and route of the EPCS interface the user needs to provide a clock constraint on the EPCS_x_RX_CLK signal. This ensures that setup and hold timing will be met across the interface.

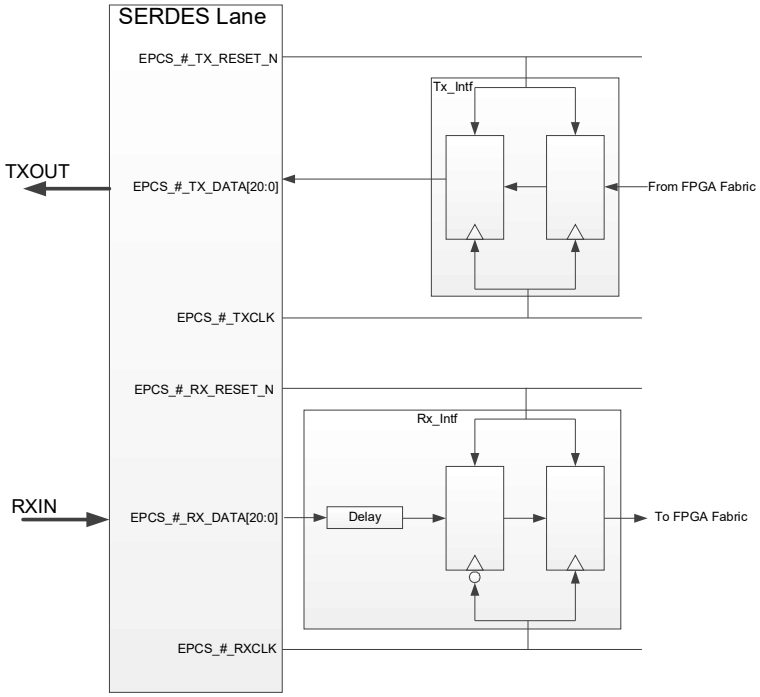
Figure 60 • **EPCS Receive Interface Timing Diagram**



All other EPCS signals are considered asynchronous with respect to the EPCS_RX_CLK or EPCS_TX_CLK clocks.The following figure shows the recommended interface circuit to achieve timing closure for high-speed designs in which the EPCS interface clocks are above 125 MHz.

Adding pipeline registers before the user logic provides the place and route tool flexibility in the placement of these registers and allow for small routes. In addition, using the proper fabric clock routing resources using the CLKINT an RCLKINT will allow for low skew and fast injection timing necessary to achieve high-speed timing closure across the EPCS interface.

Figure 61 • **Detailed EPCS Interface Diagram**



The preceding figure shows an interface diagram for using the EPCS interface to achieve both setup and hold timing closure. For more information about an example design using the EPCS interface solution as well as constraints and simulation, refer to the *TU0570: Implementing a SmartFusion2/IGLOO2 SERDES EPCS Protocol Design - Libero SoC Tutorial*.

The EPCS design should include proper clock constraints. The following table provides a good example to properly constrain the design. The user needs to adjust the period and hierarchy accordingly.

Table 161 • **Example EPCS Constraints**

EPCS_0_TXCLK	create_clock -period 8.000 -name {EPCS_0_TX_CLK} [get_pins {SERDES_IF_0/SERDESIF_INST:EPCS_TXCLK_0}]
EPCS_0_RXCLK	create_clock -period 8.000 -name {EPCS_0_RX_CLK} [get_pins {SERDES_IF_0/SERDESIF_INST:EPCS_RXCLK_0}]
EPCS_1_TXCLK	create_clock -period 8.000 -name {EPCS_1_TX_CLK} [get_pins {EPCS_SERDES_0/SERDES_IF_0/SERDESIF_INST:EPCS_TXCLK_1}]
EPCS_1_RXCLK	create_clock -period 8.000 -name {EPCS_1_RX_CLK} [get_pins {EPCS_SERDES_0/SERDES_IF_0/SERDESIF_INST:EPCS_RXCLK_1}]
EPCS_2_TXCLK	create_clock -period 8.000 -name {EPCS_2_TX_CLK} [get_pins {EPCS_SERDES_0/SERDES_IF_0/SERDESIF_INST:EPCS_TXCLK[0]}]
EPCS_2_RXCLK	create_clock -period 8.000 -name {EPCS_2_RX_CLK} [get_pins {EPCS_SERDES_0/SERDES_IF_0/SERDESIF_INST:EPCS_RXCLK[0]}]

Table 161 • **Example EPCS Constraints *(continued)***

EPCS_3_TXCLK	create_clock -period 8.000 -name {EPCS_3_TX_CLK} [get_pins {SERDES_IF_0/SERDESIF_INST:EPCS_TXCLK[1]}]
EPCS_3_RXCLK	create_clock -period 8.000 -name {EPCS_3_RX_CLK} [get_pins {SERDES_IF_0/SERDESIF_INST:EPCS_RXCLK[1]}]

5.5.2 EPCS SERDES Calibration and External Resistor Configuration

An external resistor is required for the PMA hard macro in order to perform an impedance calibration (transmit, receive, and receiver equalization). In a SERDESIF (4- lanes), two lanes share a same external resistor (REXT) - so L0 & L1 share one and L2 & L3 share another REXT. Hence whenever any lane is used the respective REXT resistor must be connected on the board. The external resistor input signal needs to be connected to the SERDES_x_L01_REXT and SERDES_x_L23_REXT pads. The end of the calibration is signaled by the PMA macro through the EPCS_READY signal. Refer to the SERDESIF Block, page 4 for detail.

5.5.3 Implementing SGMII Using EPCS Interface

SmartFusion2 and IGLOO2 support implementing SGMII using SERDESIF in EPCS mode and then connecting it to a customer supplied or 3rd party soft IP MAC. SGMII is available using EPCS on all lanes when not using PCIe within the SERDESIF. Following are the two options to implement SGMII:

- Mode A:
  1. Sets Protocol mode for the lane to EPCS mode using the SERDESIF configurator in Libero SoC, and performs APB write cycles through the fabric APB interface to SGMII settings.

  2. Performs PHY reset either by asserting “EPCS_X_RX_RESET_N” or SERDES_LANEx_SOFT-RESET registers to bring the PHY up again with SGMII.
- Mode B:
  1. Sets Protocol mode for the lane to EPCS1.25G mode using the SERDESIF configurator in Libero SoC.

  2. Upon power-up the lane has same settings as those for SGMII except Tx post cursor ratio uses -3.5dB de-emphasis by default. The SGMII Tx post cursor ratio does not use any de-emphasis by default. The Tx post cursor ratio setting difference may or may not affect the transmission of SGMII depending on how lossy are the board level link connections.

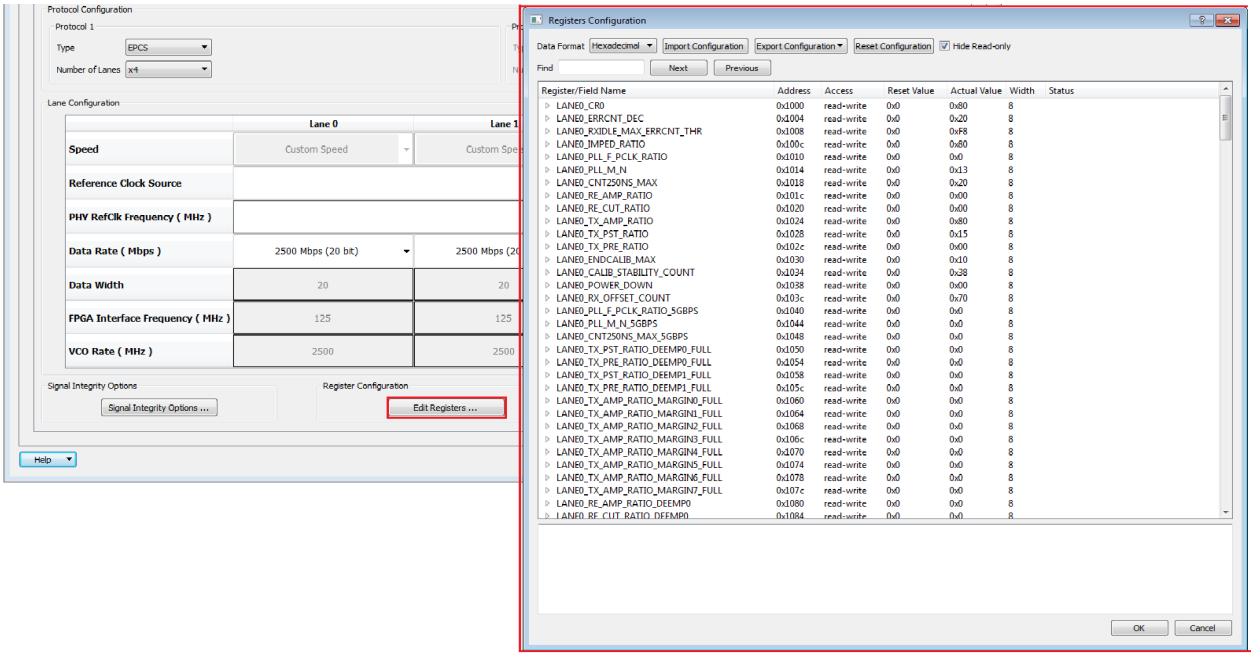
  - If instant on is not critical for SGMII application, Mode A is the preferred solution because it enables the capability to configure the settings such as Tx post-cursor ratio corresponding to the board level link connections.

  - In either options of all-four-lane SGMII, the lanes may use the same refclk source running at 125 MHz. Lane0 and lane1 can share the same calibration results from lane0, and lane2 and lane3 can share the same calibration results from lane2. When re-calibration is performed, the two lanes in the lane pair are affected at the same time, even though the SGMII application on each lane is regarded as independent.

5.6 Customized EPCS Mode Settings

While the default SERDESIF registers are pre-set by the Libero software, the designer can alter these registers using the **Edit Register** function of the SERDES generator. This feature allows customized alterations to the SERDESIF capabilities. The **Edit Register GUI** allows access to all registers of the SERDESIF. It also includes a configuration export and import capability for versatility and customization.

Figure 62 • **SERDESIF Configurator Window**



EPCS modes are all customizable, however some registers are locked and read-only using the **Edit Register** function.

6 Serializer/De-serializer

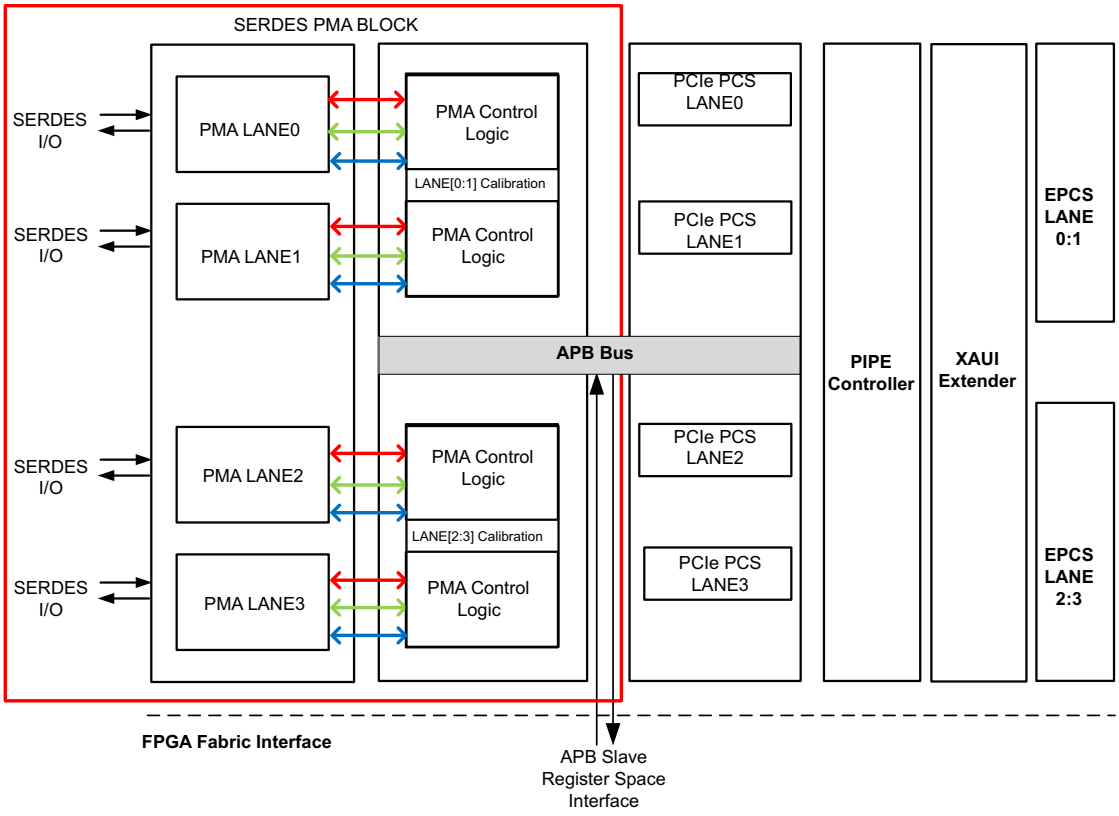
6.1 Introduction

The high speed serial/de-serializer (SERDES) hard IP block of the SmartFusion2/IGLOO2 FPGA family is included within the SERDESIF module. The details in this chapter include the physical hardware capabilities of the SERDES and a description of the fixed hardware blocks of the SmartFusion2/IGLOO2 serial physical interface (PHY). For more Information on serial protocols, refer to the previous protocol chapters in the SERDESIF Block, page 4.

6.1.1 Features

- TX macro including differential impedance matching output buffers, serializer logic, transition de-emphasis, and receiver detection circuitry
- RX macro including differential CML input buffers, de-serializer logic, on-chip termination, and continuous-time linear equalization (CTLE)
- Clock macro including clock recovery circuitry and clock management logic
- Configuration control and status register access

Figure 63 • **SmartFusion2 and IGLOO2 SERDESIF Block Diagram - SERDES/PMA Module**



6.1.2 Device Support

The following table lists the total number of SERDES channels available in each SmartFusion2/IGLOO2 device.

Table 162 • **Available SERDES Blocks in SmartFusion2/IGLOO2 Devices**

	M2S/M2GL 005	M2S/M2GL 010	M2S/M2GL 025	M2S/M2GL 050	M2S/M2GL 060	M2S/M2GL 090	M2S/M2GL 150
SERDESIF available	0	1	1	Up to 2	1	1	Up to 4
SERDES Lanes	0	4	4	8	4	4	16

Note: The specified number of SERDESIF blocks varies depending on the device package.

Note: M2S/M2GL060/090 application interfaces have dual controller capability supporting up to two x1 or x2 endpoints within a SERDESIF.

6.2 SERDES Block Overview

SmartFusion2/IGLOO2 devices include up to four integrated high-speed serial interface (SERDESIF) blocks. Details on SERDESIF can be found in SERDESIF Block, page 4. Figure 63, page 128 shows a high level diagram of the fixed SERDESIF blocks.

Each SERDESIF block has a SERDES including four full-duplex differential channels and a fully implemented physical media attachment (PMA). The PMA includes the TX and RX buffers, SERDES logic, clocking, and clock recovery circuitry.

Based on application, the datapath includes a peripheral component interface express (PCIe) physical coding sublayer (PCS), 10 gigabit attachment unit interface (XAUI) extender, or external PCS (EPCS). The PCIe PCS contains the 8b/10b encoder/decoder, RX detection, and elastic buffer for the PCI. The PCS layer interface is compliant to the Intel PHY interface for the PCIe architecture v2.00 specification (PIPE). The PCIe PCS is fully configurable in terms of number of lanes and number of links. Each link is configurable from x1, x2, or x4 with supporting power management and wakeup logic.

For XAUI applications, the datapath from the PMA passes through the 8b/10b encoder/decoder, channel aligner, clock compensation, and XAUI state machine. The XAUI path is terminated to the FPGA fabric as an XGXS interface. The XAUI extender core also includes an MDIO management interface.

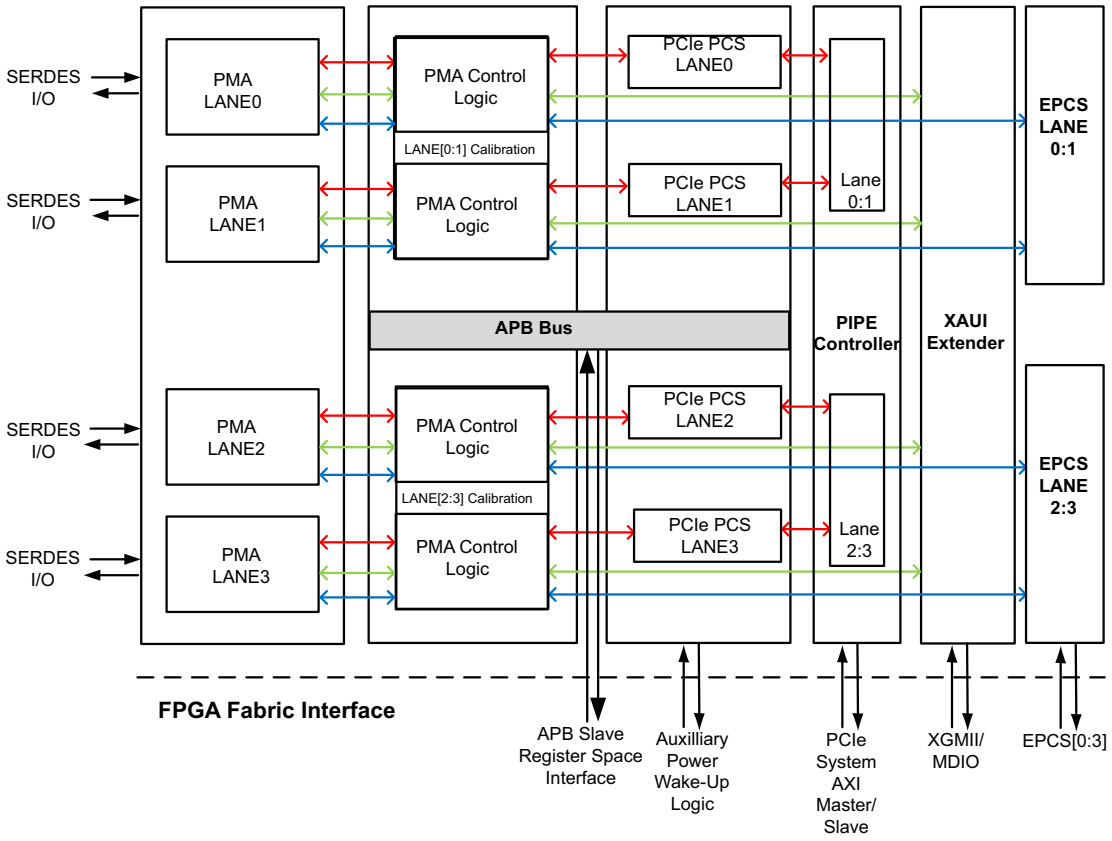
The PCIe PCS functionality can be bypassed completely in order to use the SERDES macro for protocols other than PCIe through an EPCS interface. In addition, the multi-lane instance can be disassociated at power-up to distinguish between the lanes used for PCIe and the lanes used for other protocols through the external PCS interface.

The SERDES macros terminates to the FPGA fabric by the following interfaces.

1. PIPE interface for PCIe protocol (maximum 16 bits), this also includes power management and wake-up signals.
2. XAUI XGXS with MDIO control interface
3. EPCS interface for implementing any protocol other than PCIe (maximum 20 bits)
4. Advanced peripheral bus (APB) interface for configuration

Each SERDES macro can be configured independently at power-up in a specific mode by using the SERDES macro registers. These registers are used to control the multi-function SERDES macro parameters to configure the SERDES in a specific mode. Refer to SERDESIF Register Access Map, page 150 for details about setting serial protocols. Each SERDES macro can interface to several other modules within the SERDESIF.

Figure 64 • **SERDES Macro Datapath**



As shown in the preceding figure, PCIe, XAUI, and EPCS modes include a specific connectivity through the SERDES block. The SERDES block registers allow control of the parameters corresponding to PLL frequency, baud rate, output voltage, de-emphasis, RX equalization, and parallel data path width for the PCS logic. The initial setup of these parameters is pre-configured through the Libero SoC software. These registers can be modified after power-up through the register space interface signals on a per-lane basis or all lanes together. These registers can be accessed through the APB interface and load the SERDES parameters after power-up. This run-time capability can be used to simply change specific settings such as the output voltage amplitude or de-emphasis due to a high bit error rate seen on a specific lane. The Libero software programs the appropriate registers based on the user design requirements. Any unused SERDES block resources are configured in a powered down mode via programming from the software. For unused SERDES, physical I/Os need to be properly terminated based on the recommendations in *DS0115: SmartFusion2 Pin Descriptions Datasheet* and *DS0124: IGLOO2 Pin Descriptions Datasheet*.

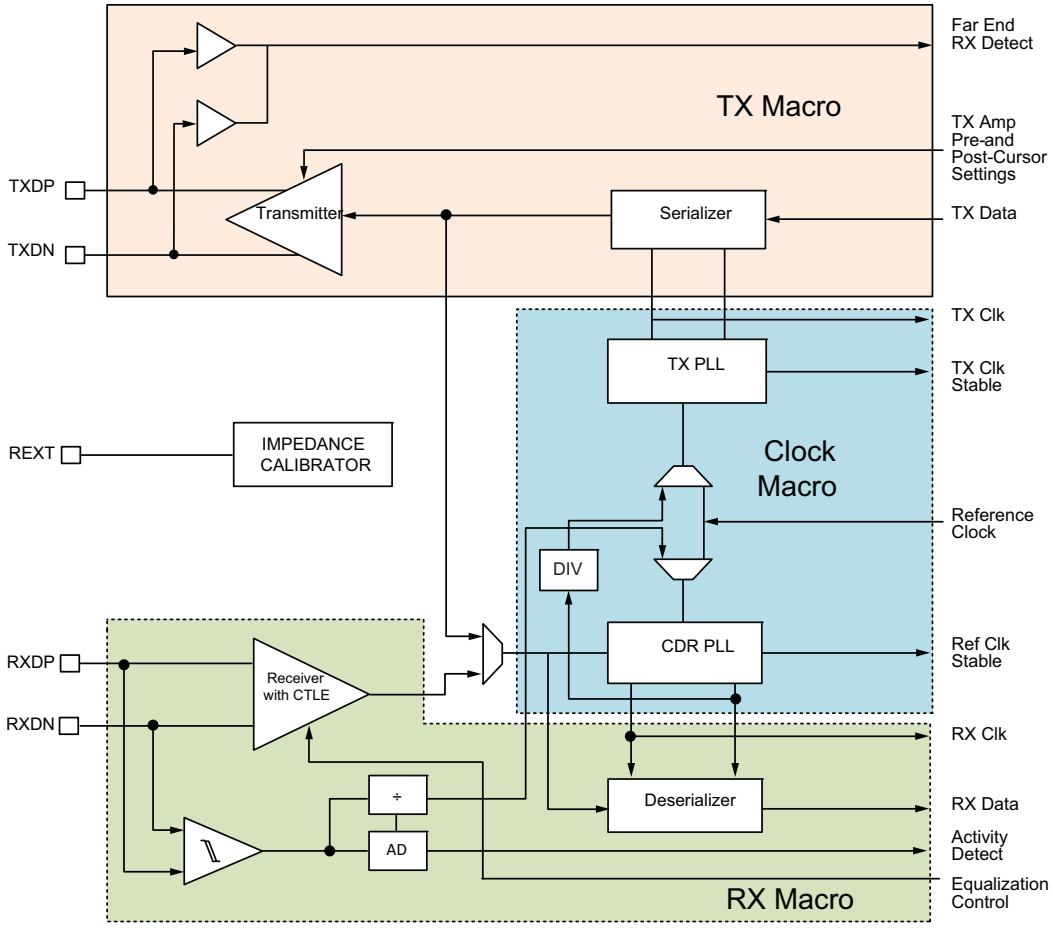
6.3 PMA Macro Block

The SERDES PMA macro contains high-speed analog front-end logic as well as TX PLL and clock and data recovery (CDR) PLL, calibration, and the voltage offset cancellation mechanism. The following figure shows a simplified functional block diagram of the PMA macro. Each of the PMA macro blocks includes the following three sub-functions:

- TX Macro
- RX Macro
- Clock Macro

The three blocks have several primary inputs and outputs as well as control and status connections. The control and status nodes are either ports or registers used in conjunction with the implementation. The following figure shows a simple diagram of the functionality fixed within the PMA.

Figure 65 • **PMA Diagram**



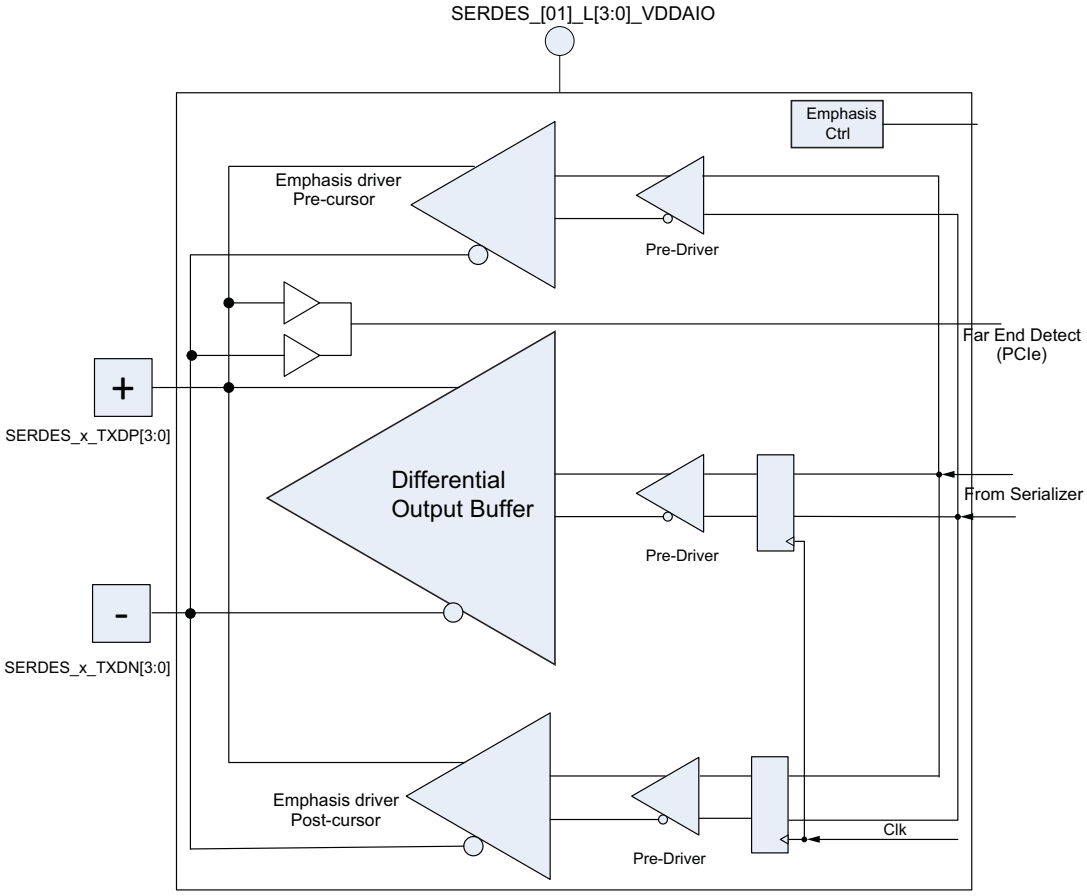
6.3.1 TX Macro

The TX macro includes a serializer which receives a 20-bit (maximum) data-word synchronous with a TX clock, serialized into a single stream of differential transmitted data transmitted to the lane. The TX macro supports multi-level output drive and multi-level transition (pre-and post-cursor) 3-tap de-emphasis while maintaining proper impedance. Refer to the SERDESIF- I/O Signal Interface, page 149 listed in Table 167, page 149 for details. The TX outputs do not support hot-swap.

6.3.1.1 TX Output Buffer

The TX macro, shown in the following figure, is a high-speed, differential impedance matching output buffer. It supports multi-level drive, pre-cursor and post-cursor transition de-emphasis, multi-level common-mode, and calibrated output impedance. These parameters are predefined by the Libero software but can be tuned by the designer.

Figure 66 • **TX Output Diagram**



The TX output voltage levels, $V_{DIFFp-p}$ and V_{CM}, are nominally set by the Libero software based on key protocol parameters. The limits of these settings are dependent on the analog SERDES I/O supply. The output voltage parameters are defined by the following equations:

- $V_{DIFFp-p} = 2*|V_{D+} – V_{D-}|$
- $V_{CM}= 0.5*(V_{D+} –V_{D-})$

Figure 67 • **TX Output Signal Parameters**

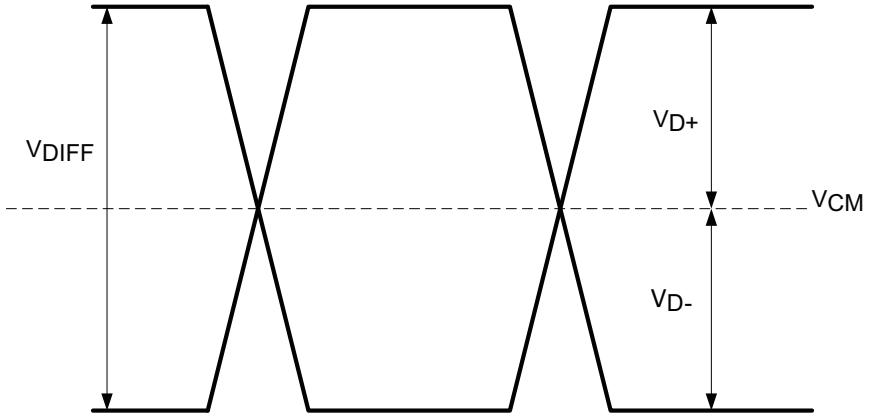
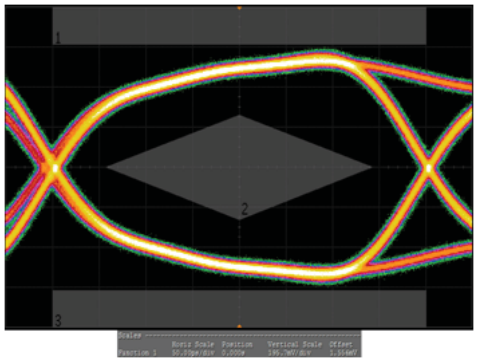
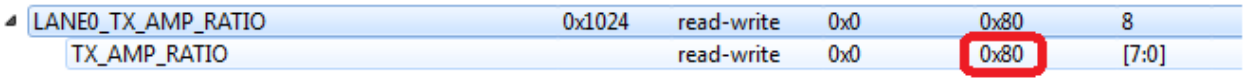


Figure 68 • **Example TX Output Signal**



The main tap of the TX output Macro is programmable and controlled by the TX_AMP_RATIO setting. The amplitude can be set as a ratio from 100%(full-swing) to 0%. The ratio is determined as a function of the initial lane calibration. Thus the device calibrates the link and determines an optimized impedance and the ratio setting uses this calibration to baseline the needed amplitude. This allows flexibility to match receiver specifications on the far-end of the link. Valid settings for TX_AMP_RATIO are between 0 and 128 whereas 128 is 100% swing.

Figure 69 • **TX AMP RATIO Setting**



The Tx amplitude is adjusted by the TX_AMP_RATIO configuration settings. The TX_AMP_RATIO setting are controlled by Libero software that adjusts the amplitude in the hardware and includes four predefined settings, as shown in the following table.

Table 163 • **Pre-defined TX Amplitude**

TX_AMP_RATIO VALUE (HEX)	MIN	MAX	Unit
0A	100.00	130.00	mV
55	740.00	850.00	mV
80	1060.00	1270.00	mV

6.3.1.2 TX De-Emphasis

The signal quality of a physical channel can be adjusted to match the interconnections and PCB using the integrated de-emphasis control. The post-curser and pre-cursor nominally spans 0 dB to 20 dB. Adjustment to these controls in conjunction with the TX amplitude allows the user to closely match the interconnect channel.

The pre-cursor and post cursor de-emphasis adjusts the magnitude of the output based on the prior bit values effectively attenuating the successive bits. This transition emphasis compensates for the channel losses and opens the signal eye at the far-end receiver. The following figure shows the de-emphasis settings and pre-cursor and post-cursor response of the transmit driver.

The pre-cursor and post-cursor attenuation is a function of the TX amplitude ratio setting (see TX_AMP_RATIO) and the TX pre and post cursor ratio setting (see TX_PRE_RATIO and TX_PST_RATIO, respectively).

The pre-cursor attenuation (de-emphasis in dB) is calculated using the formula:

dB = 20 * log (1-2*precursor ratio)

where precursor ratio is simply the TX_PRE_RATIO divided by TX_AMP_RATIO.

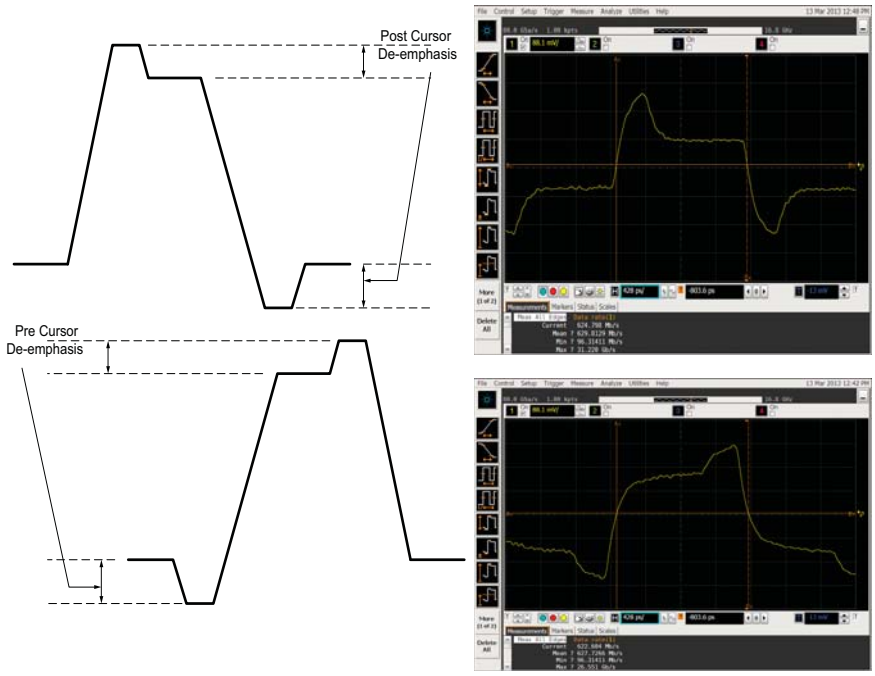
Similarly, post-cursor attenuation (de-emphasis in dB) is calculated using the formula:

dB = 20 * log (1-2*postcursor ratio)

where post cursor ratio is simply the TX_PST_RATIO divided by TX_AMP_RATIO.

For example, when TX_AMP_RATIO=128 and TX_PST_RATIO=21, Post Cursor Attenuation= 20*log(1-2*21/128). This yields -3.5dB attenuation.

Figure 70 • **Pre-Cursor and Post-Cursor De-Emphasis**



The TX Macro has many features that can be tuned dynamically when in operation mode. When changing the TX control registers in real-time, the changes are not updated until the specific UPDATE_SETTINGS register is written. Refer Table 286, page 199.

Figure 71 • **TX DEEMPHASIS RATIO Setting**

LANE0_TX_PST_RATIO	0x1028	read-write	0x0	0x15	8
TX_PST_RATIO		read-write	0x0	0x15	[7:0]
LANE0_TX_PRE_RATIO	0x102c	read-write	0x0	0x00	8
TX_PRE_RATIO		read-write	0x0	0x00	[7:0]

6.3.2 RX Macro

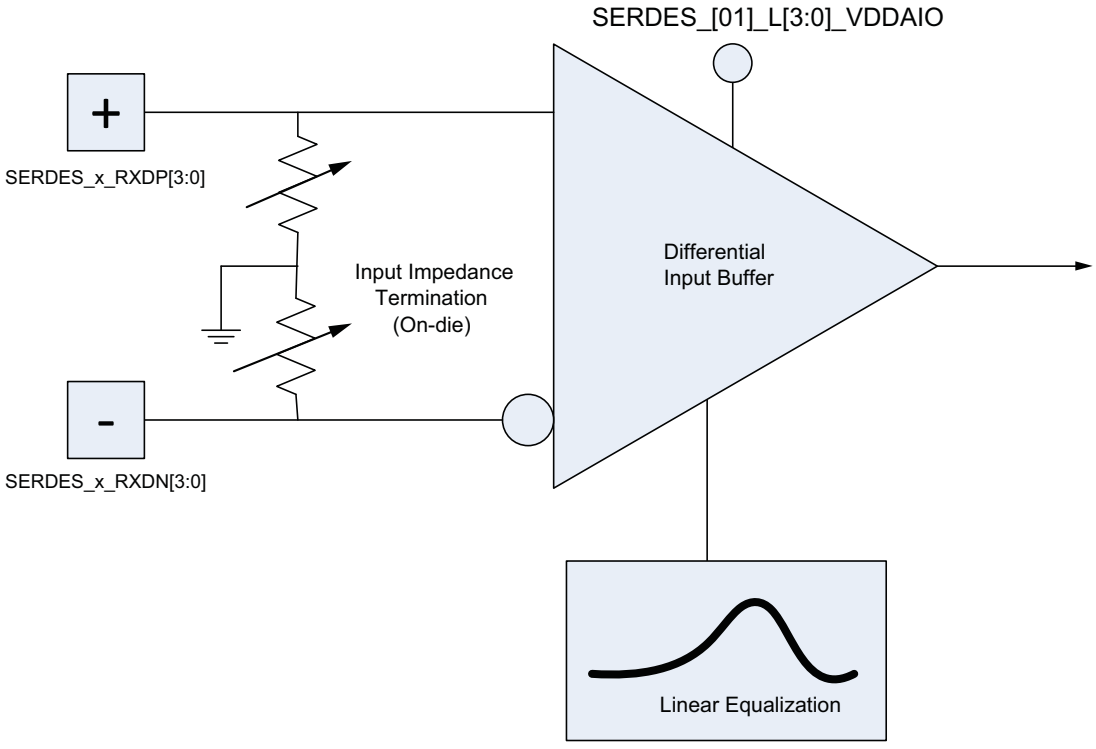
The RX macro receives the serialized data from input receivers. The CDR circuit receives the signal and extracts a properly timed bit clock from the data stream. The data signal is deserialized down to a lower speed parallel, 20-bit (maximum) digital data-word (RX data). The deserialized data is synchronous to the recovered link clock (RX clock).

6.3.2.1 RX Input Buffer

The RX macro includes an analog front-end powered by the SERDES analog power supply. It contains a current mode, input differential buffer with on-die input impedance control. The input buffer amplifier receives the incoming differential data signal. It translates the differential signal to internal logic levels, with no amplitude impairments. The input buffer amplifier rejects common mode noise. The calibrated input impedance has a typical 100-ohm differential impedance. The input impedance can be configured as needed to match the system requirements. The RX inputs do not support hot-swap.

Jitter on the incoming data signal transfers through this stage, therefore care must be taken to ensure both the incoming timing and amplitude are clean from impairments. The integrated linear equalizer filters extraneous noise from the incoming signals.

Figure 72 • **RX Input Diagram**



6.3.2.2 RX Equalization

The RX macro supports a programmable continuous time-linear equalization (CTLE). The equalizer compensates attenuated interconnections of the system printed circuit board (PCB) by effectively using a high-pass filter component which attenuates the lower frequency components to a degree greater than the higher frequency components. The equalizer circuitry can be tuned to compensate for the signal distortion due to the high frequency attenuation of the physical channel of the PCB and interconnect.

The effective receiver equalization compensates for the channel loss of the board with the added frequency response of the CTLE. The frequency response can be programmed to maximize the signal quality of the receiver for achieving the best possible bit-error rate (BER). An under-equalized channel does not adequately open the eye, whereas over-equalization can produce a channel with high jitter. Correct equalization has optimal eye opening with low noise and low jitter.

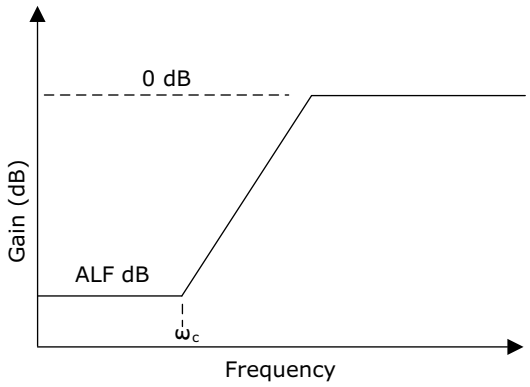
The RX Macro has many features that can be tuned dynamically when in operation mode. When changing the TX control registers in real-time, the changes will not be updated until the specific UPDATE_SETTINGS register is written. Refer Table 286, page 199.

Figure 73 • **RX EQ RATIO Setting**

LANE0_RE_AMP_RATIO	0x101c	read-write	0x0	0x00	8
RE_AMP_RATIO		read-write	0x0	0x00	[7:0]
LANE0_RE_CUT_RATIO	0x1020	read-write	0x0	0x00	8
RE_CUT_RATIO		read-write	0x0	0x00	[7:0]

The response of the CTLE is shown in the following figure. As a FIR filter, the CTLE operates on the analog input signal and is intended to equalize the incoming transmitted signal and channel by removing the ISI at the receiver. The function of the filter uses both ALF (flatband low-frequency gain) and ε_c (relative frequency of the flatband rolloff) which can be set by the user to optimize the signal quality at the receiver. The Rx Macro can set the desired level of the filter response by setting the hardware registers RE_AMP_RATIO and RE_CUT_RATIO.

Figure 74 • **Receiver Equalization Frequency Response**



The Rx equalization is adjusted by the RE_AMP_RATIO and RE_CUT_RATIO configuration settings. The Rx equalization setting is controlled by Libero software that adjusts the settings accordingly in the hardware and using predefined settings, as shown in the following table.

Table 164 • **Predefined Receiver Equalization (CTLE) Settings**

ω_c(MHz)	A_{LF} (dB)	Libero Default setting	RE_AMP_RATIO Register	RE_CUT_RATIO Register
–	–	CTLE Disabled (Default)	0x00	0x00
400	10.88	Pre-Defined Setting 1	0x77	0x20
500	13.06	Pre-Defined Setting 2	0x5B	0x2A
600	13.98	Pre-Defined Setting 3	0x51	0x2F
700	12.04	Pre-Defined Setting 4	0x32	0x80
800	13.38	Pre-Defined Setting 5	0x3E	0x3C
900	13.98	Pre-Defined Setting 6	0x39	0x40
1000	12.57	Pre-Defined Setting 7	0x70	0x4F
1100	11.6	Pre-Defined Setting 8	0x7D	0x58
1200	10.88	Pre-Defined Setting 9	0x37	0xFE
1300	9.95	Pre-Defined Setting 10	0x22	0xFC
1400	8.52	Pre-Defined Setting 11	0x52	0xFE
1500	7.47	Pre-Defined Setting 12	0x5B	0xFE
1600	7.04	Pre-Defined Setting 13	0x73	0xFE
1700	6.66	Pre-Defined Setting 14	0x78	0xFE

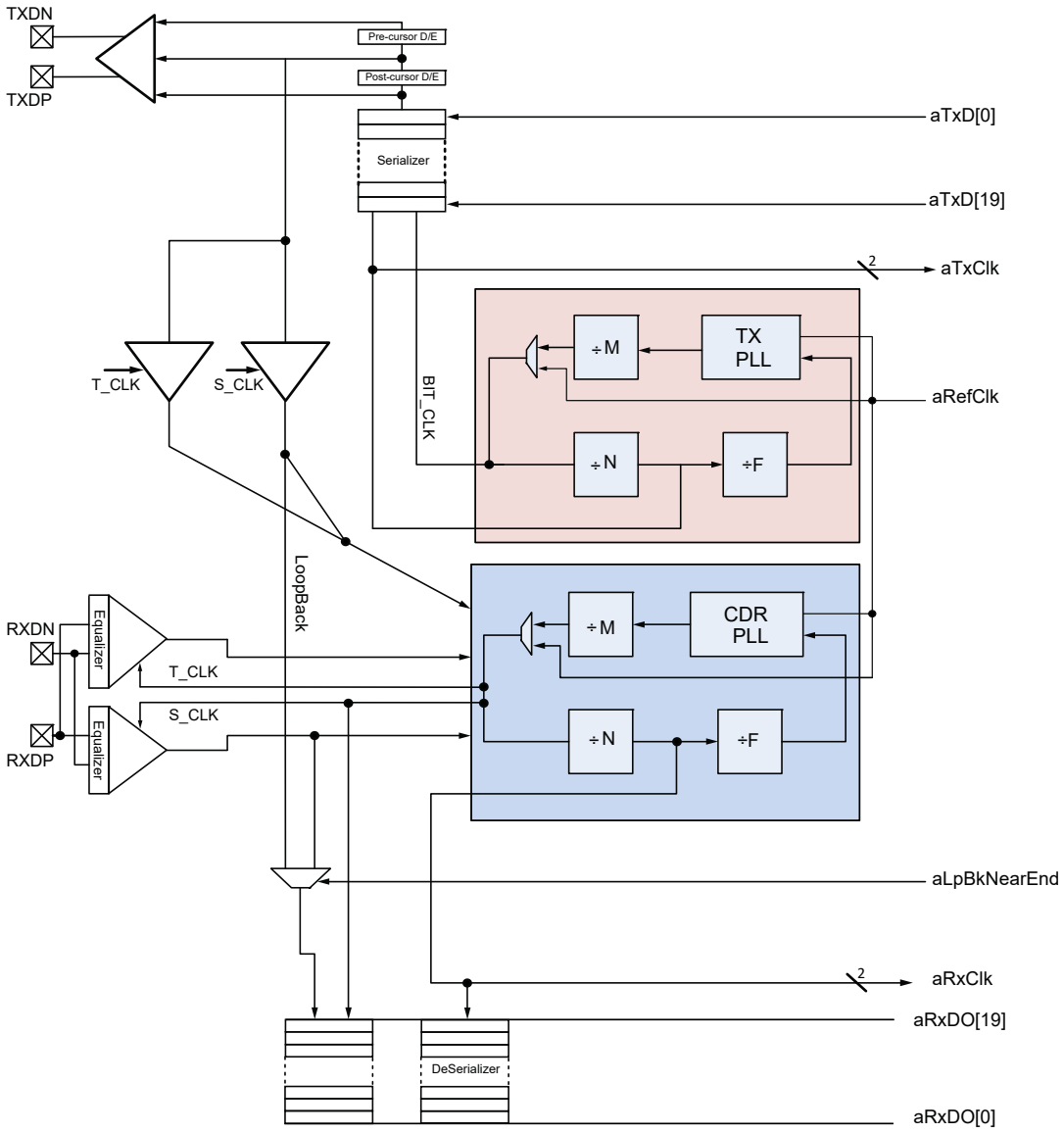
6.3.2.3 AC Coupling

Each channel must be AC-coupled to remove common mode dependencies. However, AC coupling generates baseline wander if the high-speed serial data transmission is not DC-balanced. 8B/10B encoded data is an example of DC-balanced signaling used with PCIe and XAUI protocols. The addition of external capacitors for AC coupling requires careful consideration. The designer should select a capacitor knowing the requirements of the system. It is important to minimize the pattern-dependent jitter associated with the low frequency cutoff of the AC coupling network. When NRZ data containing long strings of identical 1's or 0's is applied to this high-pass filter, a voltage droop occurs, resulting in low-frequency, pattern-dependent jitter (PDJ). Off-chip AC coupling requires recommended capacitor values such as 10 nF for 8b/10b XAUI and 75-200 nF for PCIe. These example values need to be reviewed based on specific system requirements. Refer to the *AC393: SmartFusion2 and IGLOO2 Board Design Guidelines Application Note* for further details.

6.3.3 Clock Macro

The Clock macro in the PMA contains one transmit PLL (TX PLL) and one CDR PLL. The following figure shows the overview of the clock macro with some associated signals. The TXPLL and RXCDR use a common input pin “aRefClk”. The power supply for the TXPLL and CDR PLL is supplied from a dedicated 2.5 V supply. Figure 78, page 141 shows the required power supply connections for SERDES_x_L01_VDDAPLL and SERDES_x_L23_VDDAPLL. These supplies are specified and used separately from the SPLL which was mentioned in the PCIE and XAUI sections of this document.

Figure 75 • **Clock Macro Diagram**



Each of the PLLs (TX PLL and CDR PLL) contains the necessary dividers and output high frequency (BitClk, S_Clk, and T_Clk) and low frequency (aTXClk and aRXClk) clocks. The TX clock (aTXclk) and RX clock (aRXClk) are divided down and pipelined versions of the high frequency clocks BitClk and S_Clk. The TX clock and RX clock are complementary. The exact frequencies of the clocks are determined by the reference clock (RefClk) and divide ratio settings (M, N, and F). The divide ratio settings—M, N, and F—can be programmed from the APB interface on the SERDESIF block. Refer to the *DS0128: IGLOO2 and SmartFusion2 Datasheet* for the RefClk operating ranges.

The relationships between FREF (from RefClk clock input), FBaudClock, FBusClock, and bus width are as shown in the following equations.

FVCO = (FREF) * M * N * F

EQ 6-1

FBaudClock = FVCO / M = (FREF) * N * F

EQ 6-2

FBusClock = FBaudClock / N = (FREF) * F

EQ 6-3

Bus width = FBaudClock / FBusClock = N

EQ 6-4

Note: FBaudClock in TX PLL is the EPCS_TX_CLK for EPCS mode, FBaudClock in CDR PLL is the EPCS_RX_CLK for EPCS mode, and bus width is the EPCS bus width.

TX clock will only be present and at the correct frequency if all the following are true:

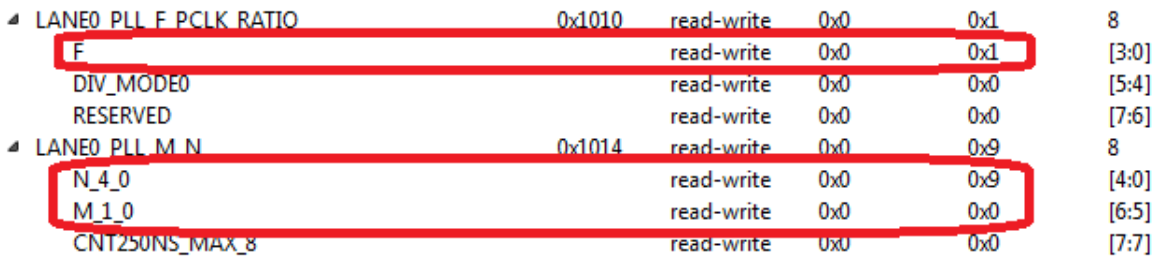
- Reference clock is present and at correct frequency
- M, N, and F are correctly set
- TX PLL is on
- TX clock trees are on (internal)
- Power-down mode is off and initialization is done

The RX clock will only be present and at the correct frequency (with high frequency internal S and T clocks aligned to the bitstream) if all of the following are true:

- Reference clock is initially present and at the correct frequency
- M, N, and F are correctly set
- RX PLL is on, at correct frequency and TX clock is present (PMA controlled mode)
- Serial bitstream is present and valid
- De-serializer circuitry is on
- Receivers are on

Refer to the TX PLL and CDR PLL Operation, page 142 for more information on using the TX and CDR PLL.

Figure 76 • **M, N, and F Setting**



6.3.3.1 Reference Clock Inputs

Each SERDESIF consists of reference clock input pads (SERDES_x_REFCLKn_P/N). The REFCLK is multiplexed in the clock macro, It optionally allows the reference clock to be sourced to the TX PLL and/or the CDR PLL These are dual purpose I/Os; as they can be alternatively used as generic I/O to MSIOD fabric only if the SERDESIF is not activated. If unused for either SERDES REFCLK or MSIOD, they can be left floating. For more information about MSIOD, see *DS0128: IGLOO2 and SmartFusion2 Datasheet*.

The MSIOD supported REFCLKs can be used as differential or single-ended I/Os. When used differentially, the SERDES_x_REFCLKnP/N is operational to receive clock signaling from LVDS and HCSL type clock drivers. These input signals must be dc-coupled (no series capacitors) from the SERDES_x_REFCLKnP/N pins to the clock driver device. SmartFusion2 and IGLOO2 reference clock inputs support ODT features as available in the FPGA IO. The ODT termination provides a good signal

integrity, saves board space, and reduces external components on PCB. For information about ODT configurations, see *UG0445: SmartFusion2 SoC FPGA and IGLOO2 FPGA Fabric User Guide*.

Note: The HCSL inputs are supported directly with LVDS IOSTD inputs from Libero. There is no specific HCSL IOSTD available in Libero and designs requiring HCSL are supported by using the LVDS IO standard. Internal ODT on REFCLK can be used. Designers are recommended to use IBIS modeling with their specific reference clock drive and the Smartfusion2/IGLOO2 reference clock input.

If used with LVPECL clock type drivers, the signal requires ac-coupling capacitor and termination resistors. The termination resistors must be placed closest to the SERDES_x_REFCLKnP/N pins and used to properly bias the inputs to the correct voltage input common-mode (VICM) and voltage input differential (VID).

AC-coupling of the SERDES_x_REFCLKnP/N cannot be used without the correct re-biasing terminations. Refer to the *DS0128: IGLOO2 and SmartFusion2 Datasheet* for REFCLK input specifications. The REFCLK inputs do not support any 3.3 V input standards and only operate at up to 2.5 V nominal. The inputs do not support hot-plug.

Table 165 • **Reference Clock Specifications (Typical)**

Reference Clock Parameter	Min	Max	Unit
Ref Clock Speed	100	160	MHz

Note: *Refer to the DS0128: IGLOO2 and SmartFusion2 Datasheet for detailed specifications*

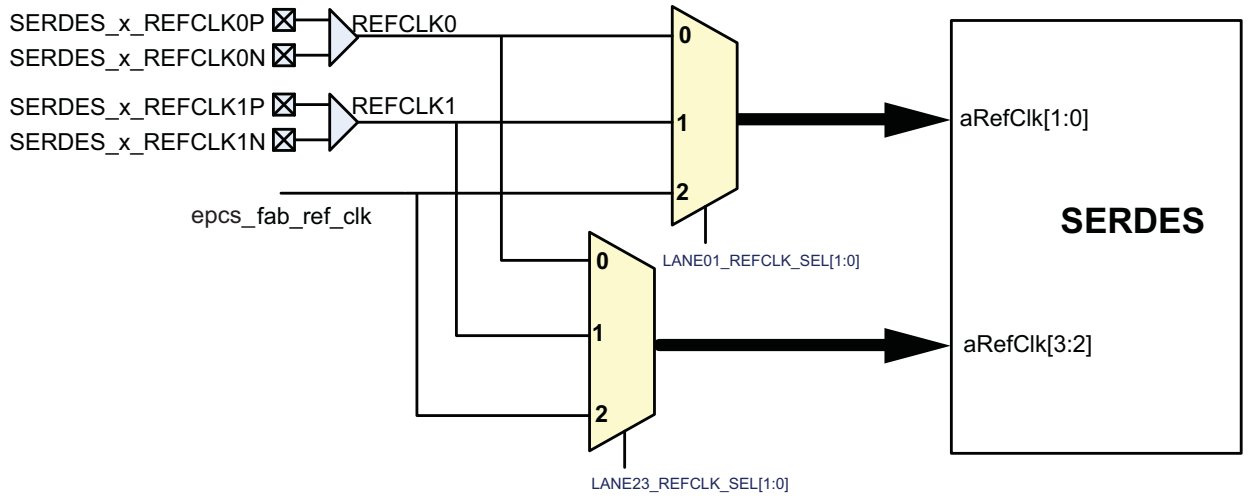
Selection of a quality clock source is important for the best performance of the SERDES. The SERDES reference clock phase noise contributes to the transmit output phase noise and can decrease receiver jitter tolerance. Refer to the *DS0128: IGLOO2 and SmartFusion2 Datasheet* for more specific information.

6.3.3.2 SERDES Reference Clocks

The PMA in the SERDES block needs a reference clock on each of its lanes for Tx and Rx clock generation through the PLLs. The two reference clock inputs ports (REFCLK0 and REFCLK1) are optionally connected to I/O pads, as previously discussed, or an additional reference clock source, fab_ref_clk, coming from the fabric. The dedicated clock input pins are recommended to achieve optimal performance. The fabric reference clock is only available for EPCS modes. The following figure shows the reference clock selection. The fabric clock should not be sourced from any of the on-die oscillators. The user programmed clock selection is routed to the SERDES Clock Macro RefClk input port shown in Figure 68, page 133.

The SERDES has four lanes, the two adjacent SERDES lanes share the same reference clock, as shown in the following figure. In this scheme, lane0 and lane1 share the same reference clock input. Similarly, lane3 and lane4 share the same reference clock.

Figure 77 • **SERDES Reference Clock Sources**

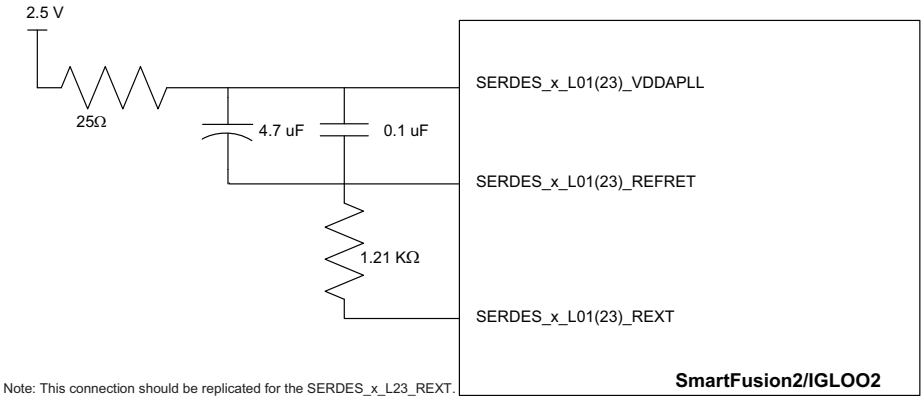


6.3.3.3 Calibration Resource Sharing

The SERDES PMA block calibration is performed to optimize the SERDES in the system. Calibration is done for the source impedance at the transmitter and the termination of the receiver. The calibration circuitry is shared across channels.

Each SERDES block contains 2 external reference resistor (REXT) signals—one for lane0 and lane1, and another for lane2 and lane3. The calibration interaction limits the combination of protocols/data rates per channel utilization since the adjacent channels are bonded to the same calibration circuitry. For example, if lane1 and lane0 operate and the PHY is reset on lane0, the recalibration function which follows the reset disrupts lane1 as a consequence of the shared REXT calibration resistor. REXT connections are required to calibrate Tx/Rx termination value and internal elements. A 1.21 k-1% resistor must be connected on the PCB, as shown in the following figure. This resistor can be a 0201 or 0402 sized component, since the power dissipation through this resistor is less than 1mW during calibration. Refer to the *AC393: SmartFusion2 and IGLOO2 Board Design Guidelines Application Note* for more details.

Figure 78 • **Calibration Resistor Connection**



Example: If lane1 and lane0 are operating and the PHY is reset on lane0, the recalibration function which follows the reset will disrupt lane1 as a consequence of the shared REXT calibration resistor.

In EPCS mode, following are the fabric sources that are used for REFCLK:

- VDDAPLL and REFRET are used to supply the PLL.
- VDDAIO and REXT are used to supply the SerDes I/O.
- SERDES_VDDI and SERDES_VREF are used for the dedicated input pins to REFCLK.

Note: The preceding pins must be terminated as recommended in *AC393: SmartFusion2 and IGLOO2 Board Design Guidelines Application Note*.

6.3.3.4 SerDes Startup

For SerDes to lock at power up, it is essential that the reference clock to the SerDes be at the target frequency and stable. If the clock frequency is lower or higher that the target, the SerDes link acquisition is not guaranteed or reliable. SERDES_VDDI of the reference clock should be at its minimum or higher. Any lower SERDES_VDDI causes higher Jitter and unpredictable locking. At system startup, the user design should hold the PCS/PMA in reset until the clock stabilizes at its target frequency. The resets signals are listed for PCIe, XAUI, and EPCS modes depending on target protocols. See Physical Coding Sublayer Block, page 25 for information about reset pin descriptions. If the PLL does not lock, a simple reset may not be sufficient and a power down (using the protocol mode related SerDes power down pin) may be required.

Note: SERDES_VDDI is not explicitly notated for Smartfusion2/IGLOO2 device pins. SERDES_VDDI is the supply associated with the inputs dedicated for SERDES of the reference clock. Refer to the related PPAT table to determine the exact VDDI bank associated with the SERDES_VDDI.

In EPCS modes, after power up and the fabric releases the EPCS_#_RESET_N starting the Tx PLL locking process. The fabric output port, EPCS_#_TX_CLK_STABLE, is the Tx PLL lock flag. A rise on this flag begins the SerDes calibration operation. Calibration time is dependent on the transmit parallel clock. Calibration determines the PMA's optimized output impedance and receiver termination that best matches the system. The end of calibration triggers the EPCS_#_READY output port to rise indicating to the fabric that the transceiver is ready for operation. Serial port outputs are held at a common-mode throughout calibration. Upon completion of calibration, the transmitter is taken out of electrical-idle. This is a serial shift operation, which adds 27 parallel transmit clock cycles to complete. After which, the serial outputs reflect the applied EPCS_#_TX_DATA pattern. When the serializer sends out differential data, it takes some time for the line to charge up because of capacitive coupling.

In PCIE or XAUI modes, the protocol layer manages the health and proper sequencing of the startup requiring no user logic intervention.

6.3.4 TX PLL and CDR PLL Operation

Each PMA Clock macro has one TX PLL and CDR PLL. This section covers how to configure and use the TX PLL and clock data CDR PLL. Operations of the PLLs are done in conjunction with reset operations. Several reset ports are available to both PLLs based on mode. Typical PLL operations are pre-configured by the Libero software based on protocol and option selections.

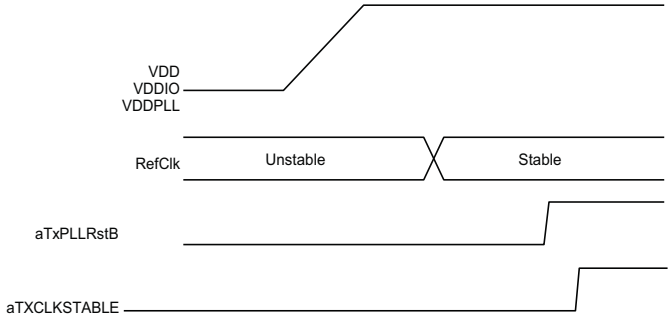
CORE_RESET_N, PHY_RESET_N, and EPCS_RESET_N[0:1] pins are defined in the SERDESIF Block, page 4. There are also lower level resets that are controlled by registers that can be accessed through the APB. These resets are used for calibration and normal operation of the TX and CDR PLLs.

6.3.4.1 Powering the TX PLL On and Off

Powering on the TX PLL from cold start is done using the aTXPLLRstB signal, which is connected to TXPLL_RST (bit 4 of the PHY_RESET_OVERRIDE register, see Table 266, page 191). In PCS driven mode (PCIe and XAUI), the PCS deasserts aTXPLLRstB after VDD and RefClk are stable, as shown in the following figure. In EPCS mode, aTXPLLRstB is deasserted using the APB interface. The PLL starts to acquire lock after the deassertion of aTXPLLRstB. During TXPLL reset, RefClk is bypassed and produced at the outputs of PLL. During bypass mode, aTXClk is a divided down version of RefClk per M, N, and F settings of the TX PLL.

Proper values for RefClk frequency and M, N, F settings of TX PLL are supplied to the PLL prior to deassertion of aTXPLLRstB. The settings must be initialized to correct values before coming out of reset. The TX PLL output is stable when the ATXCLKSTABLE signal is asserted. This signal is routed to the fabric in EPCS mode.

Figure 79 • **Transmit Clock Stabilization Timing Relationships**



If the TX PLL was powered down by asserting aTXPLLRstB for deep power savings, exit from power-down should follow the same procedure as described for powering on the TX PLL. The TXPLL is bypassed in this mode, and if RefClk was present while aTXPLLRstB was asserted, the outputs of the PLL toggle with RefClk but at a divided down frequency.

While the TX PLL is operational (and in lock) it is possible to shut down both the BitClk tree and aTXClk for intermediate power savings and faster bring-up time by the assertion of TXHF_CLKDN (bit6 of PHY_RESET_OVERRIDE register, see Table 266, page 191). The TXHF_CLKDN signal, when set, disables the TX PLL VCO by applying a static zero to the PMA aTXHfClkDnB signal. The aTXHfClkDnB signal functions to inhibit the output buffers of the PLL without interfering with the loop, hence not affecting lock.

6.3.4.2 Changing the TX PLL Mode of Operation

Once the TX PLL has acquired lock, any change of mode setting is accomplished by the suitable change of M, N, and F settings of the TX PLL. Changes to the PLL settings must be made while the PLL is held in reset. A change of mode setting does not instantaneously change the frequencies of aTXClk and BitClock, but changes the frequencies within a few aTXClk cycles, depending on the state of the internal PLL registers when mode change is applied. The TXPLL design does not guarantee that runt pulses or glitches do not occur on the clocks during mode changes. So, care should be taken when changing the PLL setting during operation mode.

6.3.4.3 Powering the CDR PLL On and Off

The sequence of operations for powering-up the CDR PLL from cold start is similar to that of the TX PLL, using the aCDRPLLRstB signal connected to RXPLL_RST (bit 5 of PHY_RESET_OVERRIDE register, see Table 266, page 191), except that proper values for CDR PLL M, N, and F have to be provided. The CDR PLL should be up and stable and a bitstream should be present at RXDP and RXDN. If the CDR PLL was powered-down for deep power savings, exit from power-down should follow the same sequence of operations as described for powering-up from system cold start. A bypass operation similar to that of the TX PLL would also result.

While the CDR PLL is operational (and in lock to RefClk) it is possible to shut down the S_CLK and T_CLK trees for intermediate power savings and faster lock to bitstream time by the assertion of RXHF_CLKDN (bit 7 of PHY_RESET_OVERRIDE register, see Table 266, page 191). The RXHF_CLKDN bit, when set, disables the RX PLL VCO settings by applying a static zero to the PMA aRXHfClkDnb signal. The aRXClk signal will still be functional in this case, but within specified bounds of accuracy linked to Refclk. Since S_Clk and T_Clk are not operational, bitstream lock cannot be achieved and the PCS will park the CDR PLL in frequency acquisition mode, which locks to RefClk.

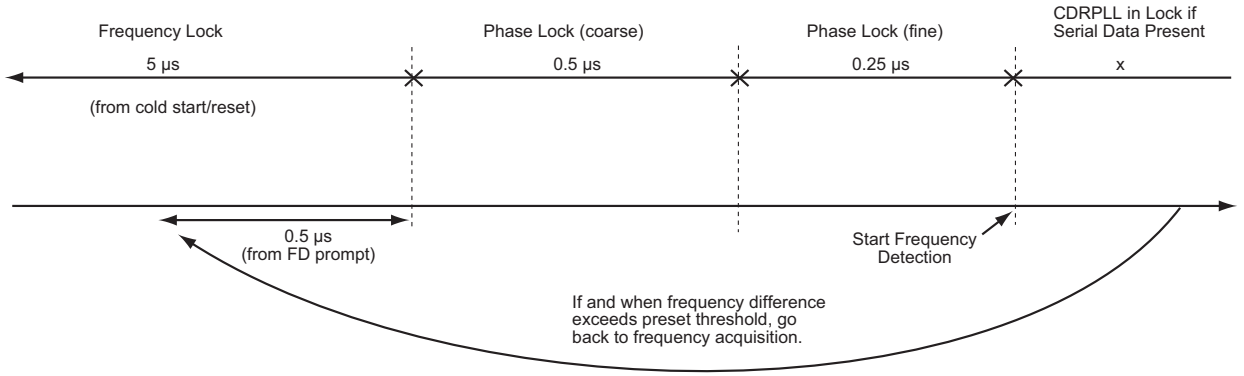
6.3.4.4 Acquiring Bit Lock for CDR PLL

PCIE, XAUI, and EPCS modes are PCS_DRIVEN modes, which trains the CDR PLL to the incoming bitstream. PMA_DRIVEN mode options are unsupported.

The steps for acquiring bit lock are similar in both modes. Bit 3 of the CR0 register (PMA driven mode, see Table 216, page 177) puts the CDR PLL in PMA driven mode or PCS driven mode. Both modes of lock require the following two steps for training the CDR PLL to the incoming bitstream for the lock:

1. Frequency lock: The frequency lock (FL) operation whereby the CDR PLL locks to the reference clock. The sampling clock at the receiver is not aligned to the center of the data eye during this step.
2. Phase lock: The phase lock (PL) operation whereby CDR PLL acquires phase and small frequency deviation lock to the bitstream. The sampling clock at the receiver will be aligned to the center of the data eye after this step. It is imperative that the bitstream be valid upon entering phase lock. There are two further steps for phase lock:
   - Coarse phase lock, which has a higher range of frequency acquisition (± 5000 ppm of static frequency difference). This step is always a transient step before embarking on fine phase lock.
   - Fine phase lock, which has a lower range of frequency acquisition (± 300 ppm static frequency difference).

Figure 80 • **CDR Bit Locking**



6.3.4.5 Changing CDR PLL Mode of Operation

Once the CDR PLL has acquired lock, any change of mode settings is accomplished by the suitable change of CDR PLL M, N, and F settings. A valid bitstream has to be present for the CDR PLL to correctly bit-lock. If a change of mode setting is desired with no change in VCO frequency, a certain amount of time is required for the CDR PLL to reacquire bit-lock. This kind of change of mode setting does not disturb the PLL frequency lock significantly, but due to phase re-acquisition, the jitter specifications of the PLL may be violated for a few transient bit periods, with associated loss of received bits. If a change of mode setting is desired, resulting in a change in VCO frequency, it must be noted that the CDR PLL has to go through the entire acquisition process, including frequency lock. The CDR PLL does not guarantee that runt pulses or glitches will not occur on the clocks during mode changes. Therefore, care should be taken when changing the PLL setting during operation mode.

6.3.5 SERDES in EPCS Mode

The SERDES block can be used in modes, other than PCIe and XAUI. For this purpose, the SERDES block includes a EPCS interface that enables it to assign each implemented PHY lane to a different protocol.

The SmartFusion2/IGLOO2 SERDES can operate up to 3.2 Gbps in -1 SPD devices for PCIE Gen 2 and up to 3.2 Gbps for EPCS protocols. The EPCS Interface lends itself to timing concerns in both transmit and receive path across the boundary to the fabric. These interfaces require every lane to the fabric to use careful design considerations.

- Both setup and hold time analysis needs to be done to verify timing on this interface.
- Proper clock connections, pipelining, and floor-planning is required to place FFs at specific locations.

The native minimum speed of the SmartFusion2/IGLOO2 SERDES is 1 Gbps. Using oversampling, each data bit is sampled in multiple clock cycles before being transmitted. For example, to transmit a 400 Mbit/s data rate over a 1.2 Gbps serial link, each bit can be sampled three times and spread over three clock cycles for both transmit and receiving of data. This is called 3x oversampling. Using this technique, lower data rates can be transmitted while the SERDES PLL continues to run within its valid operating range (1 Gbps min).

6.4 SERDES Testing Operations

This section covers how to configure the SERDES in loopback to test the signal integrity of the SERDES block. It also details the internal pattern generation and checking capability to assist with system debug. Following are the sub-sections:

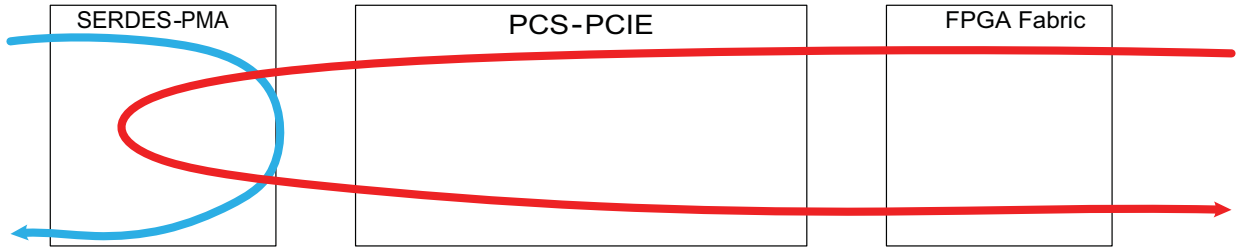
- Diagnostic Loopbacks
- Pseudo-Random Bit Sequences Pattern Generator
- Pseudo-Random Bit Sequences Pattern Checker
- Custom Pattern Generator and Checking

6.4.1 Diagnostic Loopbacks

6.4.1.1 Serial Loopbacks

Serial loopback modes are specialized configurations of the SERDES datapath where the data is folded back to the source. Typically, a specific traffic pattern is transmitted and then compared to check for errors. Loopback test modes fall into two broad categories: Near End and Far End.

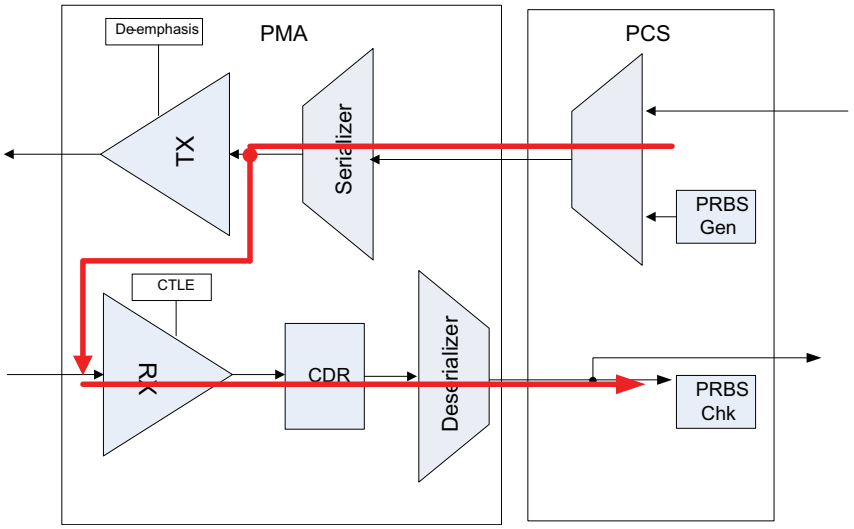
Figure 81 • **Diagnostic Loopbacks**



6.4.1.2 Near End Serial PMA Loopback

The SERDES block provides support for a serial loopback back onto itself for test purposes. When the LPBK_EN bit (bit1 of the PRBS_CTRL register) is set, the serial data is fed back to the CDR block and the CDR block extracts the clock and data. Loopback may be operated in full frequency mode (PLLs active) or bypass mode (PLLs bypassed). This mode is useful when used in conjunction with the on-die PRBS data generator and checker. In this scenario, the data can be sent and received without going off-chip.

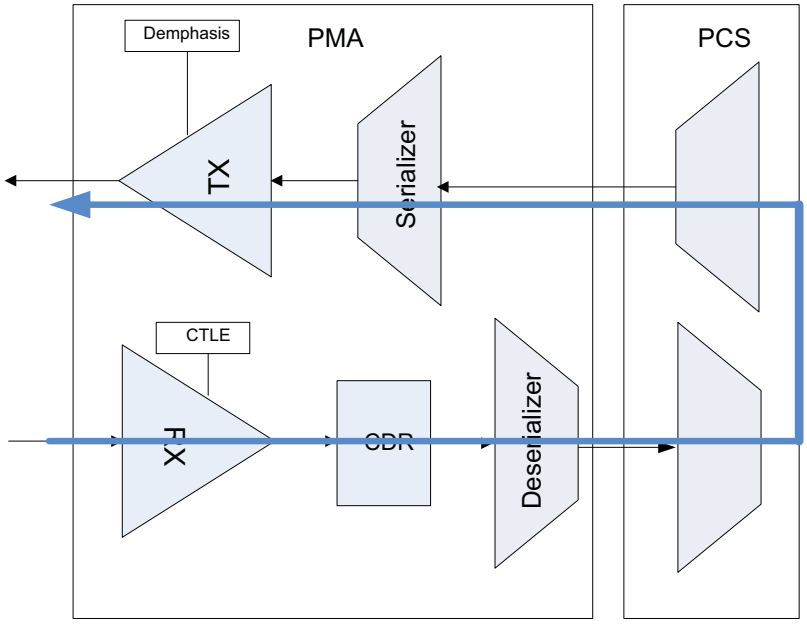
Figure 82 • **Near End Serial PMA Loopback**



6.4.1.3 PCS Far End PMA RX to TX Loopback

This loopback mode (also termed meso_lpbk, shown in blue in the following figure) is where received data is recaptured by the parallel transmit clock before being sent to the PMA transmitter. The RX is sent back to the TX and requires no PPM differences between the reference clock used by the transmit and received data. In this case, data is usually sent to the receiver from test equipment such as a BERT (Bit error-rate tester) and brought back out of the device TX to the BERT to be checked. Typically the tester will provide a reference clock to the device. There will be bit errors if there is any PPM differences.

Figure 83 • **Far End PMA RX to TX Loopback**



Note: To activate far end PMA RX to TX loopback program the PCS_LOOPBACK_CTRL[2].

6.4.2 Pseudo-Random Bit Sequences Pattern Generator

Pseudo-random bit sequences (PRBS) are commonly used to test the signal integrity of SERDES. The SERDES block allows pattern generation using the PRBS_CTRL register. This pattern can be looped back in the PMA and verified as explained in the Pseudo-Random Bit Sequences Pattern Checker, page 147. The following bits describe the PRBS pattern generation feature:

- PRBS_GEN: This signal starts the PRBS pattern transmission.
- PRBS_TYP[1:0]: This signal defines the type of PRBS pattern which is applied. PRBS7 when set to 00b, PRBS11 when set to 01b, PRBS23 when set to 10b, and PRBS31 when set to 11b.

Table 166 • **SERDES Macro PRBS Patterns**

Name	Polynomial	Length of Sequence	Descriptions
PRBS-7	$1 + X^6 + X^7$	27– 1 bits	Used to test channels which use 8b/10b encoding. Available for PCIe, XAUI, and EPCS protocols.
PRBS-15	$1 + X^{14} + X^{15}$	215 – 1 bits	ITU-T Recommendation O.150, Section 5.3. PRBS-15 is often used for jitter measurement because it is the longest pattern the Agilent DCA-J sampling scope can handle. Available for EPCS protocols.
PRBS-23	$1 + X^{18} + X^{23}$	223 – 1 bits	ITU-T Recommendation O.150, Section 5.6. PRBS-23 is often used for non-8B/10B encoding scheme. One of the recommended test patterns in the SONET specification. Available for EPCS protocols.
PRBS-31	$1 + X^{28} + X^{31}$	231 – 1 bits	ITU-T Recommendation O.150, Section 5.8. PRBS-31 is often used for non-8b/10b encoding schemes. A recommended PRBS test pattern for 10 GbE. Refer to the IEEE 802.3ae-2002 specification. Available for EPCS protocols.

Note: ITU-T Recommendation O.150 provides general requirements for instrumentation for performance measurements on digital transmission equipment.

6.4.3 Pseudo-Random Bit Sequences Pattern Checker

The SERDES block includes a built-in PRBS checker to test the signal integrity of the channel. Using the internal PMA loopback or a complete external path from the transmitter to receiver, this pattern checker allows SERDES to check the four industry-standard PRBS patterns mentioned in Table 166, page 147. The PRBS_CTRL register and PRBS_ERRCNT register allow pattern checking (see Table 264, page 190 and Table 265, page 191).

- LPBK_EN: The LPBK_EN signal, bit 1 of the PRBS_CTRL register, puts the PMA macro block in near-end loopback (serial loopback from TX back to RX). PRBS tests can be done using the near-end loopback of the PMA macro or using any far-end loopback implemented in the opposite component.
- PRBS_CHK: The PRBS_CHK signal, bit 6 of the PRBS_CTRL register, starts the PRBS pattern checker. Refer to the PRBS_CTRL register for more information.
- PRBS_ERRCNT: The PRBS_ERRCNT register reports the number of PRBS errors detected when the PRBS test is applied. Refer to PRBS_ERRCNT register for more information.

6.4.4 Custom Pattern Generator and Checking

The SERDES block allows generation of a user-defined pattern. There is no pattern checking available for custom patterns, including any of the non-PRBS patterns. The SERDES block allows pattern generation using PRBS related registers. The following bits describe the custom pattern generation feature:

- CUSTOM_PATTERN_7_0: The custom pattern registers (register offset 0X190to 0X1CC) enable programming of a custom pattern. Refer to the custom pattern registers (starting with CUSTOM_PATTERN_7_0, see Table 268, page 192) for more information.
- CUST_SEL (CUSTOM_PATTERN_CTRL[0]): This signal replaces the PRBS data transmitted on the link by the custom pattern. The PRBS_SEL register must also be set for transmitting the custom pattern on the link.
- CUST_TYP (CUSTOM_PATTERN_CTRL[3:1]): This signal defines whether the custom pattern generated on the link is generated by the custom pattern register or by one of the hard-coded patterns:
  - 00b: Custom pattern register
  - 100b: All-zero pattern (0000…00)
  - 101b: All-one pattern (1111…11)
  - 110b: Alternated pattern (1010…10)
  - 111b: Dual alternated pattern (1100…1100)
- CUST_CHK (CUSTOM_PATTERN_CTRL[4]): This bit enables the error counter.
- CUST_SKIP (CUSTOM_PATTERN_CTRL[5]): This register is used in RX Word alignment manual mode.
- CUST_AUTO (CUSTOM_PATTERN_CTRL[6]): This allows the word alignment to be performed automatically by a state machine that checks whether the received pattern is word-aligned with the transmitted pattern and to automatically use the PMA CDR PLL skip bit function to find the alignment.
- CUST_ERROR (CUSTOM_PATTERN_CTRL[3:0]): When the custom pattern checker is enabled, this status register reports the number of errors detected by the logic when the custom word aligner is in synchronization. It starts counting only after a first matching pattern has been detected.
- CUST_SYNC (CUSTOM_PATTERN_CTRL[4]): This bit reports that the custom pattern is word- aligned.
- CUST_STATE (CUSTOM_PATTERN_CTRL[7:5]): This register reports the current state of the custom pattern word alignment state machine.

6.5 Reset Requirement for Testing Operations

When performing testing operations such as Loopbacks and PRBS pattern testing, it is required that a SYSTEM_SER_SOFT_RESET (0x2008) assert/de-assert be done before and after performing these test operations. This reset operation can use the lane specific SERDES_LANE#_SOFTRESET register or by using the PCIE_CTLR_SOFTRESET or PCIE_CTLR_SOFTRESET registers. The resetting operations are embedded within the SmartDebug functions therefore will be conducted as part of the function call of the feature operations.

6.6 Using SmartDebug Utility for SERDES

The SmartDebug utility included with the Libero design software provides SERDES access that will assist FPGA and the board designers to perform SERDES real-time signal integrity testing and tuning in a system including SERDES control and test capabilities to assist with debugging high speed serial designs with no extra steps.

The SmartDebug JTAG interface extends access to configure, control and observe SERDES operations and is accessible in every SERDES design. Users simply implement their design with the Libero System Builder to incorporate the SERDESIF block enabling SERDES access from the SmartDebug tool set. This quickly enables designers to explore configuration options without going through FPGA recompilation or making changes to the board. GUI displays real-time system and lane status information. SERDES configurations are supported with TCL scripting allowing access to the entire register map for real-time customized tuning. *TU0530: SmartFusion2 and IGLOO2 SmartDebug Hardware Design Debug Tools Tutorial* demonstrates the tools capabilities.

6.7 SERDESIF- I/O Signal Interface

The SmartFusion2 and IGLOO2 SERDES block interfaces with differential I/O pads, the PCIe system, and the FPGA. The following section describes these signals.

Table 167 • **SERDESIF Block I/O – PAD Interface**

Port Name	Type	Connected to	Description
SERDES_x_RXDP0	Input	I/O Pads	Receive data. SERDES differential positive input: Each SERDESIF consists of 4 RX+ signals.
SERDES_x_RXDP1			
SERDES_x_RXDP2			
SERDES_x_RXDP3			
SERDES_x_RXDN0	Input	I/O Pads	Receive data. SERDES differential negative input: Each SERDESIF consists of 4 RX- signals.
SERDES_x_RXDN1			
SERDES_x_RXDN2			
SERDES_x_RXDN3			
SERDES_x_TXDP0	Output	I/O Pads	Transmit data. SERDES differential positive output: Each SERDESIF consists of 4 TX+ signals.
SERDES_x_TXDP1			
SERDES_x_TXDP2			
SERDES_x_TXDP3			
SERDES_x_TXDN0	Output	I/O Pads	Transmit data. SERDES differential negative output: Each SERDESIF consists of 4 TX- Signals.
SERDES_x_TXDN1			
SERDES_x_TXDN2			
SERDES_x_TXDN3			
SERDES_x_L01_REXT	Reference	I/O Pads	External reference resistor connection to calibrate TX/RX termination value. Each SERDESIF consists of 2 REXT signals—one for lanes 0 and 1 and another for lane2 and lane3.
SERDES_x_L23_REXT			
SERDES_x_REFCLK0P	Input	I/O Pads	Reference clock differential positive. Each SERDESIF consists of two signals (REFCLK0_P, REFCLK1_P). These are dual purpose I/Os; these lines can be used for MSIOD fabric, if SERDESIF is not activated.
SERDES_x_REFCLK1P			
SERDES_x_REFCLK0N	Input	I/O Pads	Reference clock differential negative. Each SERDESIF consists of two signals (REFCLK0_P, REFCLK1_P). These are dual purpose I/Os; these lines can be used for MSIOD fabric, if SERDESIF is not activated.
SERDES_x_REFCLK1N			

Note: Here, x = the SERDESIF_# where # is from 0 through 3.

7 SERDESIF Register Access Map

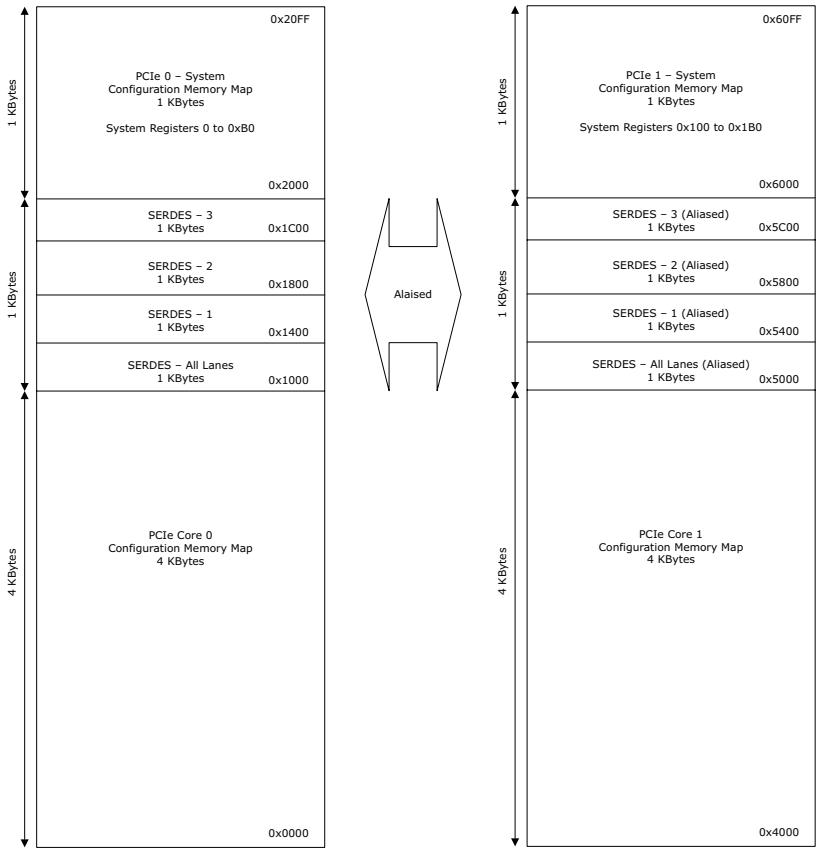
7.1 Configuration of SERDESIF

The SERDESIF contains a large number of internal registers required to properly configure the SERDESIF module. The register settings provide initial programming at power-up and most portions of the SERDESIF block can be dynamically reconfigured while in operation. A SERDESIF APB configuration interface is accessed through the FPGA fabric providing the resources to allow these programming capabilities.

In SmartFusion2, the SERDESIF is not connected directly to MSS. It can be accessed from MSS through the FPGA fabric. It contains a large number of internal registers for initialization and runtime operation. These registers are accessed through an APB configuration bus. The APB configuration interface is routed by the AHB bus matrix to the FPGA fabric interface to configure the SERDESIF. After initial programming at power-up, most portions of the SERDESIF block can be dynamically reconfigured while operating. The SERDESIF system registers can be accessed through the APB bus.

The IGLOO2 device family is supported differently than the SmartFusion2 device family for programming the SERDESIF and its supported serial protocols. The IGLOO2 uses a FPGA module to initialize peripherals and access the system controller known as the HPMS. The HPMS module provides connectivity to the AHB bus matrix allowing similar SERDESIF initialization using this FPGA IP module in the fabric. MSS/HPMS supports the SERDESIF peripheral using the Libero SoC software to correctly provision and program the user customized features.

Figure 84 • **SERDESIF Memory Map**



Note: Refer to the PCI Express, page 18 for more information on the PCIe core register.

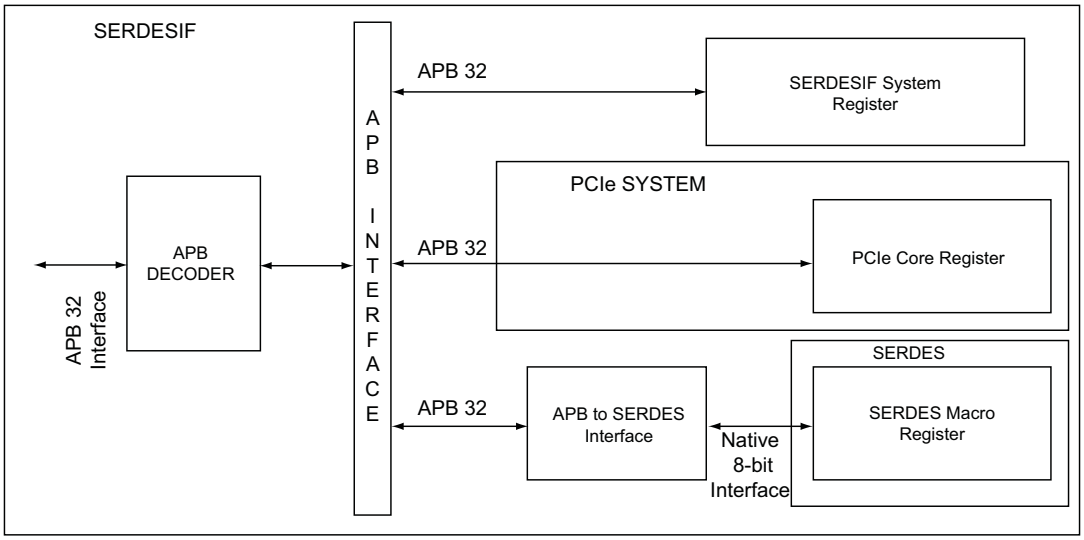
Note: Memory map offset differs from MSS and HPMS

Note: Reference *UG0331: SmartFusion2 Microcontroller Subsystem User Guide* or *UG0448: IGLOO2 FPGA High Performance Memory Subsystem User Guide.*

The Smartfusion2/IGLOO2 SERDESIF System Register Memory Map occupies (see Figure 84, page 150) 1 KBytes of configuration memory map. Figure 84, page 150 shows the physical offset location of SERDESIF system registers. For the M2S/M2GL010/025/050/150 the offset is from 0x2000 – 0x23FF for general for the PCIE0 system registers. The M2S060/090 and M2GL060/090 includes a second PCIE(PCIE1) system block which uses offset 0x6000 – 0x63FF registers for controlling PCIE1.

The following figure shows the APB implementation of three region configurations and status registers. The APB is used to interface to the FPGA fabric which enable access to these register region as an APB slave.

Figure 85 • **Address Decoder Logic Block Diagram**



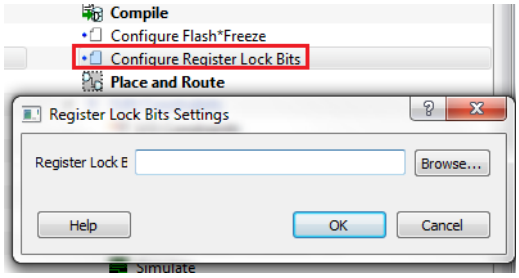
The SERDESIF block has three regions of configuration and status registers. Configuration of the SERDESIF is done through these registers. Configuration of top level functionality of the PCIe core, XAUI block, and SERDES macro is also done through these registers. The three regions of configuration and status registers shown in Figure 84, page 150 are described below.

7.2 Register Lock Bits Configuration

Register lock bits are included for users to restrict the SERDESIF configuration registers from being overwritten by the MSS (in SmartFusion2) or fabric masters that have write access to these registers. Use the Register Lock Bits Configuration tool within the Libero Design Software to lock MSS, SERDES, and FDDR configuration registers preventing them from being overwritten by masters that have access to these registers.

The Register Lock Bits Configuration tool is used to lock MSS, SERDES, and FDDR configuration registers of SmartFusion2 devices in order to prevent them from being overwritten by masters that have access to these registers. Register lock bits are set in a text (*.txt) file, which is then imported into the SmartFusion2 project. From the **Design Flow** window, click **Configure Register Lock Bits** to open the configurator. Then, click **Browse...** to navigate to the text file (*.txt) that contains the Register Lock Bit settings. (see the following figure).

Figure 86 • **Register Lock Bit Settings**



7.2.1 Lock Bit File

An initial, default lock bit file can be generated by clicking **Generate FPGA Array Data** in the **Design Flow** window.

The default file located at `<proj_location>/designer/<root>/<root>_init_config_lock_bits.txt` can be used to make the required changes.

Note: Save the file using a different name if you modify the text file to set the lock bits.

7.2.2 Lock Bit File Syntax

A valid entry in the lock bit configuration file is defined as a <lock_parameters> < lock bit value> pair format.

The lock parameters are structured as follows:

- Lock bits syntax for a register: <Physical block name>_<register name>_LOCK
- Lock bits syntax for a specific field: <Physical block name>_<register name>_<field name>_LOCK

The following are the physical block names (varies with device family and die):

- MSS
- FDDR
- SERDES_IF_x (where x is 0,1,2,3 to indicate the physical SERDES location) for SmartFusion2 and IGLOO2 (010/025/050/150) devices
- SERDES_IF2 for SmartFusion2 and IGLOO2 (060/090) devices (only one SERDES block per device)

Set the lock bit value to 1 to indicate that the register can be written to (unlocked) and to 0 to indicate that the register cannot be written to (locked).

Lines starting with # or; are comments. Empty lines are allowed in the lock bit configuration file.

The following figure shows the lock bit configuration file.

Figure 87 • **Lock Bit Configuration File**

```
# Register Lock Bits Configuration File for MSS, SERDES(s) and Fabric DDR
# Microsemi Corporation - Microsemi Libero Software Release v11.7 SP1 (Version 11.7.1.2)
# Date: Tue Mar 29 13:24:54 2016

# sb_sb_0/sb_sb_MSS_0/MSS_ADLIB_INST/INST_MSS_050_IP
 MSS_ESRAM_CONFIG_LOCK                          0
 MSS_ESRAM_MAX_LAT_LOCK                         1
 MSS_DDR_CONFIG_LOCK                            1
 MSS_ENVM_CONFIG_LOCK                           0
 MSS_ENVM_REMAP_BASE_LOCK                       1
 MSS_ENVM_FAB_REMAP_LOCK                        1
 MSS_CC_CONFIG_LOCK                             0
 MSS_CC_CACHEREGION_LOCK                        1
 MSS_CC_LOCKBASEADDR_LOCK                       1
 MSS_CC_FLUSHINDX_LOCK                          0
 MSS_DDRB_BUF_TIMER_LOCK                        1
 MSS_DDRB_NB_ADR_LOCK                           1
 MSS_DDRB_NB_SIZE_LOCK                          0
 MSS_DDRB_CONFIG_LOCK                           1
 MSS_EDAC_ENABLE_LOCK                           1
 MSS_MASTER_WEIGHT_CONFIG0_LOCK                 1
 MSS_MASTER_WEIGHT_CONFIG1_LOCK                 1
 MSS_SOFT_INTERRUPT_LOCK                        1
 MSS_SOFTRESET_ENVM0_SOFTRESET_LOCK             1
 MSS_SOFTRESET_ENVM1_SOFTRESET_LOCK             1
 MSS_SOFTRESET_ESRAM0_SOFTRESET_LOCK            1
 MSS_SOFTRESET_ESRAM1_SOFTRESET_LOCK            1
 MSS_SOFTRESET_MAC_SOFTRESET_LOCK               1
 MSS_SOFTRESET_PDMA_SOFTRESET_LOCK              1
 MSS_SOFTRESET_TIMER_SOFTRESET_LOCK             1
 MSS_SOFTRESET_MMUART0_SOFTRESET_LOCK           1
 MSS_SOFTRESET_MMUART1_SOFTRESET_LOCK           1
 MSS_SOFTRESET_G4SPI0_SOFTRESET_LOCK            1
 MSS_SOFTRESET_G4SPI1_SOFTRESET_LOCK            1
 MSS_SOFTRESET_I2C0_SOFTRESET_LOCK              1
 MSS_SOFTRESET_I2C1_SOFTRESET_LOCK              1
 MSS_SOFTRESET_CAN_SOFTRESET_LOCK               1
 MSS_SOFTRESET_USB_SOFTRESET_LOCK               1
 MSS_SOFTRESET_COMBLK_SOFTRESET_LOCK            1
 MSS_SOFTRESET_FPGA_SOFTRESET_LOCK              1
 MSS_SOFTRESET_HPDMA_SOFTRESET_LOCK             1
 MSS_SOFTRESET_FIC32_0_SOFTRESET_LOCK           1
 MSS_SOFTRESET_FIC32_1_SOFTRESET_LOCK           1
 MSS_SOFTRESET_MSS_GPIO_SOFTRESET_LOCK          1
 MSS_SOFTRESET_MSS_GPOUT_7_0_SOFT_RESET_LOCK    1
 MSS_SOFTRESET_MSS_GPOUT_15_8_SOFT_RESET_LOCK   1
 MSS_SOFTRESET_MSS_GPOUT_23_16_SOFT_RESET_LOCK  1
 MSS_SOFTRESET_MSS_GPOUT_31_24_SOFT_RESET_LOCK  1
 MSS_SOFTRESET_MDDR_CTLR_SOFTRESET_LOCK         1
 MSS_SOFTRESET_MDDR_FIC64_SOFTRESET_LOCK        1
 MSS_M3_CONFIG_LOCK                             1
```

7.2.3 Locking and Unlocking a Register

A register can be locked or unlocked by setting the appropriate lock bit value in the lock bit configuration `.txt` file,

1. Browse to locate the lock bit configuration `.txt` file.
2. Do one or both of the following:
   - Set the lock bit value to 0 for the registers you want to lock.
   - Set the lock bit value to 1 for the registers you want to unlock.
3. Save the file, and import the file into the project (**Design Flow** window > **Configure Register Lock Bits**), see Figure 86, page 152.
4. Regenerate the bitstream.

Refer to the Register Lock Bits Configuration in the *UG0331: SmartFusion2 Microcontroller Subsystem User Guide* or *UG0448: IGLOO2 FPGA High Performance Memory Subsystem User Guide* under “System Register Block” chapter.

7.3 SERDESIF System Register

The SERDES block system register controls the SERDES block module for single protocol or multi-protocol support implementation. It occupies 1 KB of the configuration memory map. The physical offset location of the SERDES block system register is shown in Figure 84, page 150. These registers can be accessed through the 32-bit APB interface, and the default values of these registers can be configured using Libero SoC. These flash bits have the settings for registers that require to be initialized quickly when the device powers up such as PLL and clock configurations, PCIe configuration space, and resets. The flash bits are set by the Libero configuration GUI based on the user selections, programmed into the

device, and are statically set at device power-up. However, the SERDES block system registers can be updated through the 32-bit APB interface, if required.

Table 168 • **SERDESIF System Registers**

Register Name	Address Offset	Register Type	Description
SYSTEM_SER_PLL_CONFIG_LOW	0x00	R/W	Sets SERDES PLL(SPLL) configuration bits (LSBs) (see Table 169, page 156).
SYSTEM_SER_PLL_CONFIG_HIGH	0x04	R/W	Sets SPLL configuration bits (MSBs) (see Table 170, page 156).
SYSTEM_SERDESIF_SOFT_RESET	0x08	R/W	PCIe controller, XAUI, and SERDES lanes soft RESET (see Table 171, page 158).
SYSTEM_SER_INTERRUPT_ENABLE	0x0C	R/W	SPLL lock interrupt enable (see Table 172, page 158).
SYSTEM_CONFIG(CONFIG2)_AXI_AHB_BRIDGE	0x10	R/W	Defines whether AXI3/AHB master interface is implemented on the master interface to fabric (see Table 173, page 158). AHB not supported.
SYSTEM_CONFIG(CONFIG2)_ECC_INTR_ENABLE	0x14	R/W	Sets ECC enable and ECC interrupt enable for PCIe memories (see Table 174, page 159).
Reserved	0x18	R/W	Reserved
Reserved	0x1C	R/W	Reserved
SYSTEM_CONFIG(CONFIG2)_PCIE_PM	0x20	R/W	Used to inform the configuration space, the slot power, PHY reference clock, Power mode etc (see Table 177, page 160).
SYSTEM_CONFIG_PHY_MODE_0	0x24	R/W	Selects the protocol default settings of the PHY (see Table 178, page 160).
SYSTEM_CONFIG_PHY_MODE_1	0x 28	R/W	Selects PCS mode, link to lane settings (see Table 179, page 161).
SYSTEM_CONFIG_PHY_MODE_2	0x2C	R/W	Sets the equalization calibration performed by the PMA control logic of the lane or use the calibration result of adjacent lane (see Table 180, page 161).
SYSTEM_CONFIG(CONFIG2)_PCIE_0	0x30	R/W	Defines PCIe vendor ID and device ID for PCIe identification registers (see Table 181, page 161).
SYSTEM_CONFIG(CONFIG2)_PCIE_1	0x34	R/W	Defines PCIe subsystem vendor ID and subsystem device ID for PCIe identification registers (see Table 182, page 162).
SYSTEM_CONFIG(CONFIG2)_PCIE_2	0x38	R/W	Defines PCIe subsystem revision ID and class code (see Table 183, page 162).
SYSTEM_CONFIG(CONFIG2)_PCIE_3	0x3C	R/W	Sets PCIe link speed (see Table 184, page 162).
SYSTEM_CONFIG(CONFIG2)_BAR_SIZE_0_1	0x40	R/W	Sets BAR0 and BAR1 of PCIe core register map (see Table 185, page 162).
SYSTEM_CONFIG(CONFIG2)_BAR_SIZE_2_3	0x44	R/W	Sets BAR2 and BAR3 of PCIe core register map (see Table 186, page 163).
SYSTEM_CONFIG(CONFIG2)_BAR_SIZE_4_5	0x48	R/W	Sets BAR4 and BAR5 of PCIe core register map (see Table 187, page 163).
SYSTEM_SER_CLK_STATUS	0x4C	R/O	This register describes SPLL lock information (see Table 188, page 164).
Reserved	0x50	R/O	–

Table 168 • **SERDESIF System Registers** ***(continued)***

Register Name	Address Offset	Register Type	Description
Reserved	0x54	R/O	–
SYSTEM_SER_INTERRUPT	0x58	R/O	SPLL/FPLL lock interrupt (see Table 191, page 164).
SYSTEM_SERDESIF(SERDESIF2)_INTR_STATUS	0x5C	R/O	SECDED interrupt status for PCIe memories (see Table 192, page 165).
Reserved	0x60	–	–
SYSTEM_REFCLK_SEL	0x64	R/W	Reference clock selection for the four lanes of PMA (see Table 194, page 165).
SYSTEM_PCLK_SEL	0x68	R/W	SERDESIF clock selection (see Table 195, page 165).
SYSTEM_EPCS_RSTN_SEL	0x6C	R/W	EPCS reset signal selection from fabric (see Table 196, page 166).
SYSTEM_CHIP_ENABLES	0x70	R/O	GEN2 enable for PCIe (see Table 197, page 166).
SYSTEM_SERDES_TEST_OUT	0x74	R/O	Status Test out output of PCIe PHY (see Table 198, page 166).
Reserved	0x78	R/W	Reserved (see Table 199, page 167).
SYSTEM_RC_OSC_SPLL_REFCLK_SEL	0x7C	R/W	Reference clock selection for SPLL (see Table 200, page 167).
SYSTEM_SPREAD_SPECTRUM_CLK	0x80	R/W	Spread spectrum clocking configuration (see Table 201, page 167).
SYSTEM_CONF(CONF2)_AXI_MSTR_WNDW_0	0x84	R/W	PCIe AXI3-master window0 configuration register – 0 (see Table 202, page 167).
SYSTEM_CONF(CONF2)_AXI_MSTR_WNDW_1	0x88	R/W	PCIe AXI3-master window0 configuration register – 2 (see Table 203, page 167).
SYSTEM_CONF(CONF2)_AXI_MSTR_WNDW_2	0x8C	R/W	PCIe AXI3-master window0 configuration register – 2 (see Table 204, page 167).
SYSTEM_CONF(CONF2)_AXI_MSTR_WNDW_3	0x90	R/W	PCIe AXI3-master window0 configuration register – 3 (see Table 205, page 168).
SYSTEM_CONF(CONF2)_AXI_SLV_WNDW_0	0x94	R/W	PCIe AXI3-slave window0 configuration register – 0 (see Table 206, page 168).
SYSTEM_CONF(CONF2)_AXI_SLV_WNDW_1	0x98	R/W	PCIe AXI3-slave window0 configuration register – 1 (see Table 207, page 168).
SYSTEM_CONF(CONF2)_AXI_SLV_WNDW_2	0x9C	R/W	PCIe AXI3-slave window0 configuration register – 2 (see Table 208, page 168).
SYSTEM_CONF(CONF2)_AXI_SLV_WNDW_3	0xA0	R/W	PCIe AXI3-slave window0 configuration register – 4 (see Table 209, page 168).
SYSTEM_DESKEW_CONFIG	0xA4	R/W	PLL REF clock DESKEW register (see Table 210, page 168).
SYSTEM_ DEBUG_MODE_KEY	0xA8	R/W	Enables/Disables APB bus to monitor PCIE test pins (See Appendix C and Table 211, page 169).
SYSTEM_ADVCONFIG(ADVCONFIG2)	0xB4	R/W	see Table 212, page 169
SYSTEM_ ADVSTATUS(ADVSTATUS2)	0xB8	R/O	Indicates the reset phase of the PCIE controller (see Table 213, page 170).

Table 168 • **SERDESIF System Registers** ***(continued)***

Register Name	Address Offset	Register Type	Description
SYSTEM_ENHANCEMENT	0xC8	R/W	M2S/M2GL060 and M2S/M2GL090 devices only (see Table 214, page 170).

Note: Refer to the individual register description for the reset value.

Note: R/W: Read and write allowed R/O: 0 Read only

Note: Refer to Register Lock Bits Configuration, page 151 for lock capabilities of the SERDESIF System Registers.

Note: Physical Address offsets are referenced (PCIE0)(PCIE1).

Note: PCIE1 system registers only available in M2S/M2GL060 and M2S/M2GL090 devices.

Table 169 • **SYSTEM_SER_PLL_CONFIG_LOW**

Bit Number	Name	Reset Value	Description
[18:16]	PLL_OUTPUT_DIVISOR	0x1	These bits set SPLL output divider value: 000: ÷1 001: ÷2 010: ÷4 011: ÷8
[15:6]	PLL_FEEDBACK_DIVISOR	0x2	These bits set SPLL feedback divider value (SSE = 0) (binary value + 1) 0000000000: ÷1 0000000001: ÷2 0000000010: ÷3 … 1111111111: ÷1,025
[5:0]	PLL_REF_DIVISOR	0x2	These bits set SPLL reference divider value (binary value+1): 000000: ÷1 000001: ÷2 000010: ÷3 … 111111: ÷65 Both REFCK and post-divide REFCK must be within the range specified in the PLL datasheet.

Table 170 • **SYSTEM_SER_PLL_CONFIG_HIGH**

Bit Number	Name	Reset Value	Description
16	PLL_PD	0x0	A power-down (PD) signal is provided for lowest quiescent current. When PD is asserted, the PLL powers down and outputs are low. PD has precedence over all other functions.
15	PLL_FSE	0x0	This signal selects the external input paths: 0: Feedback (FB) pin input 1: Reserved

Table 170 • **SYSTEM_SER_PLL_CONFIG_HIGH** *(continued)*

Bit Number	Name	Reset Value	Description
14	PLL_MODE_3V3	0x1	Analog voltage selection 1: 3.3 V 0: 2.5 V Selects between 2.5 V and 3.3 V analog voltage operation mode (wrong selection may cause PLL not to function, but will not damage the PLL).
13	PLL_MODE_1V2	0x1	Core voltage selection 1: 1.2 V 0: Reserved
12	PLL_BYPASS	0x1	A bypass signal is provided which both powers down the PLL core and bypasses it as that PLLOUT tracks REFCK. Bypass has precedence over reset. Microsemi recommends that either Bypass or reset are asserted until all configuration controls are set in the desired working value; the power supply and reference clocks are stable within operating range, and the feedback path is functional. Either bypass or reset may be used for power-down IDDQ testing.
11	PLL_RESET	0x1	PLL reset signal (asserted high).
[10:7]	PLL_LOCKCNT	0xF	These bits contain lock counter value (2^ (binary value + 5)): 0000: 32 0001: 64 … 1111: 1048576 The above mentioned lock counter values represent the number of reference cycles present before the lock is asserted or detected.
[6:4]	PLL_LOCKWIN	0x0	These bits contain phase error window for lock assertion as a fraction of divided reference period: 000: 500ppm 100: 8000ppm 001: 1000ppm 101: 16000ppm 010: 2000ppm 110: 32000ppm 011: 4000ppm 111: 64000ppm Values are at typical process, voltage, and temperature (PVT) only and are not PVT compensated.
[3:0]	PLL_FILTER_RANGE	0x9	These bits contain PLL filter range: 0000: BYPASS 0111: 18–29 MHz 0001: 1–1.6 MHz 1000: 29–46 MHz 0010: 1.6–2.6 MHz 1001: 46–75 MHz 0011: 2.6–4.2 MHz 1010: 75–120 MHz 0100: 4.2–6.8 MHz 1011: 120–200 MHz 0101: 6.8–11 MHz 0110: 11–18 MHz

Note: All the registers are 32-bit. Bits, which are not shown in the table, are reserved.

Table 171 • **SYSTEM_SERDESIF_SOFT_RESET**

Bit Number	Name	Reset Value	Description
10	AXI2_SOFTRESET	0x0	AXI Interface soft reset (Active High). Holds the AXI interface logic part of the second PCIe controller in soft reset. Used to prevent AXI interface responding to AXI requests prior the SERDES PLL(SPLL) locking.
9	AXI_SOFTRESET	0x0	AXI Interface soft reset (Active High). Holds the AXI interface logic part of the main PCIe controller in soft reset. Used to prevent AXI interface responding to AXI requests prior the SERDES PLL (SPLL) locking.
8	PCIE2_CTRL_CFGRESET	0x0	Second PCIe controller configuration space reset, emulates hot reset function (Active high).
7	PCIE_CTRL_CFGRESET	0x0	PCIe controller configuration space reset, emulates hot reset function (Active high).
6	PCIE2_CTLR_SOFTRESET	0x1	Second PCIe controller soft RESET (Active low).
5	SERDES_LANE3_SOFTRESET	0x1	SERDES lane3 soft reset
4	SERDES_LANE2_SOFTRESET	0x1	SERDES lane2 soft reset
3	SERDES_LANE1_SOFTRESET	0x1	SERDES lane1 soft reset
2	SERDES_LANE0_SOFTRESET	0x1	SERDES lane0 soft reset
1	XAUI_CTLR_SOFTRESET	0x1	XAUI controller soft reset
0	PCIE_CTLR_SOFTRESET	0x1	PCIe controller soft reset

Note: All the registers are 32-bit. Bits not shown in the table are reserved.

Note: Bits 10:6 are only available for M2S060/090 and M2GL060/090 devices. Registers are reserved for other devices.

Table 172 • **SYSTEM_SER_INTERRUPT_ENABLE**

Bit Number	Name	Reset Value	Description
3	FPLL_LOCKLOST_INT_ENABLE	0x0	This bit sets FPLL lock lost interrupt output enable.
2	FPLL_LOCK_INT_ENABLE	0x0	This bit sets FPLL lock interrupt output enable.
1	SPLL_LOCKLOST_INT_ENABLE	0x0	This bit sets SERDES PLL lock lost interrupt output enable.
0	SPLL_LOCK_INT_ENABLE	0x0	This bit sets SERDES PLL lock interrupt output enable.

Table 173 • **SYSTEM_CONFIG(CONFIG2)_AXI_AHB_BRIDGE**

Bit Number	Name	Reset Value	Description
1	CFGR_AXI_AHB_MASTER(PCIE0) CFGR2_AXI_AHB_MASTER(PCIE1)	0x1	Defines whether AXI3/AHB slave interface is implemented on the master interface to fabric. 0: AHB, 32-bit AHB slave implemented in fabric (not supported) 1: AXI3, 64-bit AXI3 slave implemented in fabric

Table 173 • **SYSTEM_CONFIG(CONFIG2)_AXI_AHB_BRIDGE** ***(continued)***

Bit Number	Name	Reset Value	Description
0	CFGR_AXI_AHB_SLAVE(PCIE0) CFGR2_AXI_AHB_SLAVE(PCIE1)	0x1	Defines whether AXI3/AHB master interface is implemented on the slave interface to fabric. 0: AHB, 32-bit AHB master implemented in fabric (not supported) 1: AXI3, 64-bit AXI3 master implemented in fabric

Note: PCIE1 is available in M2S/M2GL060 and M2S/M2GL090 devices.

Table 174 • **SYSTEM_CONFIG(CONFIG2)_ECC_INTR_ENABLE**

Bit Number	Name	Reset Value	Description
[7:4]	CFGR_PCIE_ECC_INTR_EN(PCIE0) CFGR2_PCIE_ECC_INTR_EN(PCIE1)	0x7	This bit sets the ECC interrupt enable for PCIe Tx, Rx, and Rp memories. Bit 0 1: Rp - ECC interrupt enabled 0: ECC interrupt disabled Bit 1 1: Rx - ECC interrupt enabled 0: ECC interrupt disabled Bit 2 1: Tx - ECC interrupt enabled 0: ECC interrupt disabled
[3:0]	CFGR_PCIE_ECC_EN(PCIE0) CFGR2_PCIE_ECC_EN(PCIE1)	0x7	This bit sets the ECC enable for PCIe Tx, Rx, and Rp memories. Bit 0 1 - Rp - ECC enabled- 1'b0: ECC – disabled Bit-1: 1'b1 - Rx - ECC enabled- 1'b0: ECC – disabled Bit-2: 1'b1 - Tx - ECC enabled- 1'b0: ECC – disabled

Note: PCIE1 is available in M2S/M2GL060 and M2S/M2GL090 devices.

Table 175 • **Reserved Register**

Bit Number	Name	Reset Value	Description
–	Reserved	0x0	–

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 176 • **Reserved Register**

Bit Number	Name	Reset Value	Description
–	Reserved	0x0	–

Table 177 • **SYSTEM_CONFIG(CONFIG2)_PCIE_PM**

Bit Number	Name	Reset Value	Description
3	CFGR_TX_SWING(PCIE0) CFGR2_TX_SWING(PCIE1)	0x0	Transmit swing: This signal is a per-link signal, which is generated by each link PCIe. The PCS logic performs the internal mapping of link to lanes. **Note**: This signal is only for PCIe Gen2 controller, not for PCIe GEN1 controller. This field is set to 1 when the Transmit Swing is checked in configurator.
2	CFGR_L2_P2_ENABLE(PCIE0) CFGR2_L2_P2_ENABLE(PCIE1)	0x0	L2/P2 enable. 1'b1: Enable L2/P2 (Default-L2P2-Enabled) 1'b0: Disable L2/P2 If L2/P2 is enabled, cfgr_pm_auxpwr should also be enabled.
1	CFGR_PM_AUX_PWR(PCIE0) CFGR_PM_AUX_PWR(PCIE1)	0x0	Slot auxiliary power: This signal specifies whether the device uses the slot auxiliary power source. This signal is used only used if the core supports D3 cold. 1'b1: Auxiliary power source available. Default L2P2-Enabled. 1'b0: Auxiliary power source unavailable.
0	CFGR_SLOT_CONFIG(PCIE0) CFGR2_PM_AUX_PWR(PCIE1)	0x0	Slot clock configuration: This signal is used only to inform the configuration space, if the reference clock of the PHY is same as that of the slot. 0: Independent clock 1: Slot clock This signal is synchronous to CLK.

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Note: PCIE1 is available in M2S/M2GL060 and M2S/M2GL090 devices.

Table 178 • **SYSTEM_CONFIG_PHY_MODE_0**

Bit Number	Name	Reset Value	Description
[15:0]	CONFIG_PHY_MODE	0x0	For each lane, this signal selects the protocol default settings of the PHY, which sets the reset value of the registers space. For instance, the following mapping is associated to a four lane PHY: phy_mode[3:0]: Mode associated to lane0 phy_mode[7:4]: Mode associated to lane1 phy_mode[11:8]: Mode associated to lane2 phy_mode[15:12]: Mode associated to lane3 PHY_MODE settings: 4'b0000 - PCIE mode 4'b0001 - XAUI mode 4'b0010 - Reserved 4'b0011 - Reserved 4'b0100 - Reserved 4'b0101 - EPCS mode 4'b1111 - SERDES PHY lane is off

Table 179 • SYSTEM_CONFIG_PHY_MODE_1

Bit Number	Name	Reset Value	Description
[11:8]	CONFIG_REG_LANE_SEL	0xF	Lane select: This signal defines which lanes are accessed and must be one-hot encoded for read transaction. For write transaction, one or several lanes can be written in the same time when several bits are asserted.
[7:4]	CONFIG_LINKK2LANE	0xF	This signal is used in PCIe mode in order to select the association of lane to link and must be one-hot encoded (each lane can be associated only to one link). For example, a four lane PHY, which can be configured in 1 or 2 link might have pipe_lk2ln[3:0]: lane associated to link 0 pipe_lk2ln[7:4]: lane associated to link 1 It is mandatory that this signal is static at power-up or stable before reset de-assertion.
[3:0]	CONFIG_EPCS_SEL	0x0	For each lane, one bit of this signal defines whether the external PCS interface is used or the PCIe PCS is enabled: 0b: PCIe mode 1b: External PCS mode For instance, the mapping associated to a four lane PHY is: epcs_sel[0]: External PCS selection associated to lane0 epcs_sel[1]: External PCS selection associated to lane1 epcs_sel[2]: External PCS selection associated to lane2 epcs_sel[3]: External PCS selection associated to lane3

Table 180 • SYSTEM_CONFIG_PHY_MODE_2

Bit Number	Name	Reset Value	Description
[7:0]	CONFIG_REXT_SEL	0x0	For each lane, 2 bits of this signal select whether the Tx, Rx, and Rx equalization calibration is performed by the PMA control logic of the lane or use the calibration result of adjacent lane (upper or lower lanes): 00b: perform calibration using the lane calibration algorithm, which also requires that the Rext resistor is present on board 01b: use calibration result of lower lane 10b: use calibration result of upper lane 11b: reserved

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 181 • SYSTEM_CONFIG(CONFIG2)_PCIE_0

Bit Number	Name	Reset Value	Description
[31:16]	PCIE_DEVICE_ID	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe device ID.
[15:0]	PCIE_VENDOR_ID	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe vendor ID.

Table 182 • **SYSTEM_CONFIG(CONFIG2)_PCIE_1**

Bit Number	Name	Reset Value	Description
[31:16]	PCIE_SUB_DEVICE_ID	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe subsystem device ID.
[15:0]	PCIE_SUB_VENDOR_ID	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe subsystem vendor ID.

Table 183 • **SYSTEM_CONFIG(CONFIG2)_PCIE_2**

Bit Number	Name	Reset Value	Description
[31:16]	PCIE_CLASS_CODE	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe class code.
[15:0]	PCIE_REV_ID	0x0	Specifies hardwired settings for PCIe identification registers: Defines PCIe revision ID.

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 184 • **SYSTEM_CONFIG(CONFIG2)_PCIE_3**

Bit Number	Name	Reset Value	Description
[5:2]	K_BRIDGE_SPEC_REV	0x0	Sets PCI Express specification version capability: 0000: Core is compliant with PCIe Specification 1.0a 0001: Core is compliant with PCIe Specification 1.1 0010: Core is compliant with PCIe Specification 2.0
1	K_BRIDGE_EMPH	0x0	This bit selects the level of de-emphasis for an upstream component. When the link is operating at 5.0 Gbps speed support: 1 indicates de-emphasis of 3.5 dB 0 indicates de-emphasis of 6 dB
0	K_BRIDGE_SPEED	0x0	PCIe Link speed supports: 0: 2.5 Gbps Gen1 1: 5.0 Gbps Gen2

Table 185 • **SYSTEM_CONFIG(CONFIG2)_BAR_SIZE_0_1**

Bit Number	Name	Reset Value	Description
[17:13]	CONFIG_BAR_SIZE_1	0x0	These bits set the size of the BAR1 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_1 - 5'd22 translates to BAR0 - (2 MB) "1111_1111_1110_0000_0000_0000_0000_CONFIG_BAR_CONTROL_1"

Table 185 • **SYSTEM_CONFIG(CONFIG2)_BAR_SIZE_0_1**

Bit Number	Name	Reset Value	Description
[12:9]	CONFIG_BAR_CONTROL_1	0x0	LSB bits of BAR1 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00-32-bit memory, 10 - 64-bit memory Bit3: Prefetchable/non-prefetchable memory
[8:4]	CONFIG_BAR_SIZE_0	0x0	These bits set the size of the BAR0 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_0 - 5'd21 translates to BAR0 - (1 MB) "1111_1111_1111_0000_0000_0000_0000_CONFIG_BAR_CONTROL_0"
[3:0]	CONFIG_BAR_CONTROL_0	0x0	LSB bits of BAR 0 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00-32-bit memory, 10 - 64-bit memory Bit3: Prefetchable/non-prefetchable memory

Note: All registers are 32-bit. Bits not shown in the table are reserved.All registers are 32-bit. Bits not shown in the table are reserved.

Table 186 • **SYSTEM_CONFIG(CONFIG2)_BAR_SIZE_2_3**

Bit Number	Name	Reset Value	Description
[17:13]	CONFIG_BAR_SIZE_3	0x0	These bits set the size of the BAR3 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_3 - 5'd24 translates to BAR3 - (8 MB) "1111_1111_1000_0000_0000_0000_0000_CONFIG_BAR_CONTROL_3"
[12:9]	CONFIG_BAR_CONTROL_3	0x0	[3:0] LSB bits of BAR 3 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00-32-bit memory, 10-64-bit memory Bit3: Prefetchable/non-prefetchable memory
[8:4]	CONFIG_BAR_SIZE_2	0x0	These bits set the size of the BAR2 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_2 - 5'd23 translates to BAR0 - (4 MB) "1111_1111_1100_0000_0000_0000_0000_CONFIG_BAR_CONTROL_2"
[3:0]	CONFIG_BAR_CONTROL_2	0x0	[3:0] LSB bits of BAR 2 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00-32-bit memory, 10 - 64-bit memory Bit3: Prefetchable/non-prefetchable memory

Table 187 • **SYSTEM_CONFIG(CONFIG2)_BAR_SIZE_4_5**

Bit Number	Name	Reset Value	Description
[17:13]	CONFIG_BAR_SIZE_5	0x0	These bits set the size of the BAR5 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_5 - 5'd26 translates to BAR5 - (32 MB) "1111_1110_0000_0000_0000_0000_0000_CONFIG_BAR_CONTROL_5"

Table 187 • **SYSTEM_CONFIG(CONFIG2)_BAR_SIZE_4_5** ***(continued)***

Bit Number	Name	Reset Value	Description
[12:9]	CONFIG_BAR_CONTROL_5	0x0	[3:0] LSB bits of BAR 5 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00 - 32-bit memory, 10 - 64-bit memory Bit3: Prefetchable/non-prefetchable memory
[8:4]	CONFIG_BAR_SIZE_4	0x0	These bits set the size of the BAR4 memory. For example, 32-bit BAR: CONFIG_BAR_SIZE_4 - 5'd25 translates to BAR4 - (16 MB) "1111_1111_0000_0000_0000_0000_0000_CONFIG_BAR_CONTROL_4"
[3:0]	CONFIG_BAR_CONTROL_4	0x0	[3:0] LSB bits of BAR 4 register in PCIe core register map Bit0: Memory/IO type indicator Bit[2:1]: Size of memory, 00-32 bit memory, 10 - 64-bit memory Bit3: Prefetchable/non-prefetchable memory

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 188 • **SYSTEM_SER_CLK_STATUS**

Bit Number	Name	Reset Value	Description
1	FAB_PLL_LOCK	0x0	Fabric PLL lock information, CLK_BASE, 1: LOCKED
0	PLL_LOCK	0x0	SPLL lock information, 1: LOCKED

Table 189 • **Reserved Register**

Bit Number	Name	Reset Value	Description
–	Reserved	0x0	–

Table 190 • **Reserved Register**

Bit Number	Name	Reset Value	Description
–	Reserved	0x0	–

Table 191 • **SYSTEM_SER_INTERRUPT**

Bit Number	Name	Reset Value	Description
0	PLL_LOCK_INT	0x0	SPLL/FPLL lock interrupt output
1	PLL_LOCKLOST_INT	0x0	SPLL/FPLL lock lost interrupt output

Table 192 • SYSTEM_SERDESIF(SERDESIF2)_INTR_STATUS

Bit Number	Name	Reset Value	Description
[2:0]	SERDESIF_INTR_STATUS	0x0	ECC interrupt status for PCIe memories

Table 193 • Reserved Register

Bit Number	Name	Reset Value	Description
–	Reserved	0x0	–

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 194 • SYSTEM_REFCLK_SEL

Bit Number	Name	Reset Value	Description
[3:2]	LANE23_REFCLK_SEL	0x0	Reference clock selection for lane2 and lane3 of PMA: 00: Selects refclk_io0 clock for lane2 and lane3 as reference clock 01: Selects refclk_io1 clock for lane2 and lane3 as reference clock 10: Reserved 11: Selects fab_ref_clk clock for lane2 and lane3 as reference clock
[1:0]	LANE01_REFCLK_SEL	0x0	Reference clock selection for lane0 and lane1 of PMA: 00: Selects refclk_io0 clock for lane0 and lane1 as reference clock 01: Selects refclk_io1 clock for lane0 and lane1 as reference clock 10: Reserved 11: Selects fab_ref_clk clock for lane0 and lane1 as reference clock

Table 195 • SYSTEM_PCLK_SEL

Bit Number	Name	Reset Value	Description
[5:4]	PIPE_PCLKIN_LANE23_SEL	0x0	PIPE clock input selection for lane2 and lane3, can be selected from one of pipeclk_out[3:0]: 00: Selects pipeclk_out[0] clock as pipeclk_in for lane2 and lane3. 01: Selects pipeclk_out[1] clock as pipeclk_in for lane2 and lane3. 10: Selects pipeclk_out[2] clock as pipeclk_in for lane2 and lane3. 11: Selects pipeclk_out[3] clock as pipeclk_in for lane2 and lane3.

Table 195 • **SYSTEM_PCLK_SEL** *(continued)*

Bit Number	Name	Reset Value	Description
[3:2]	PIPE_PCLKIN_LANE01_SEL	0x0	PIPE clock input selection for lane0 and lane1, can be selected from one of pipeclk_out[3:0]: 00: Selects pipeclk_out[0] clock as pipeclk_in for lane0 and lane1. 01: Selects pipeclk_out[1] clock as pipeclk_in for lane0 and lane1. 10: Selects pipeclk_out[2] clock as pipeclk_in for lane0 and lane1. 11: Selects pipeclk_out[3] clock as pipeclk_in for lane0 and lane1.
[1:0]	PCIE_CORECLK_SEL	0x0	PCIe core clock selection. PCIe core clock can be selected from one of pipeclk_out[3:0]: 00: Selects pipeclk_out[0] clock as PCIe core clock. 01: Selects pipeclk_out[1] clock as PCIe core clock. 10: Selects pipeclk_out[2] clock as PCIe core clock. 11: Selects pipeclk_out[3] clock as PCIe core clock.

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Note: SYSTEM_PCLK_SEL register is used with EPCS mode and PCIE.

Table 196 • **SYSTEM_EPCS_RSTN_SEL**

Bit Number	Name	Reset Value	Description
[3:0]	FABRIC_EPCS_RSTN_SEL	0x0	EPCS reset signal selection from FABRIC

Table 197 • **SYSTEM_CHIP_ENABLES**

Bit Number	Name	Reset Value	Description
0	GEN2_SUPPORTED	0x1	GEN2 enable for PCIe: 1: GEN2 enabled 0: GEN2 disabled

Table 198 • **SYSTEM_SERDES_TEST_OUT**

Bit Number	Name	Reset Value	Description
[31:0]	SERDES_TEST_OUT	0x0	Status TESTOUT output of PCIe PHY. SERDES_TEST_OUT[31:24] - Debug signal for lane3 SERDES_TEST_OUT[23:16] - Debug signal for lane2 SERDES_TEST_OUT[15:80] - Debug signal for lane1 SERDES_TEST_OUT[7:0] - Debug signal for lane0 Bit[0]: Tx PLL reset - Active low signals Bit[1]: Rx PLL reset - Active low signals Bit[2]: Activity detected Bit[3]: CDR PLL locked on data Bit[4]: Tx PLL locked Bit[5]: Rx PLL locked Bit[7:6]: reserved

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 199 • **Reserved**

Bit Number	Name	Reset Value	Description
0	Reserved	0x1	–

Table 200 • **SYSTEM_RC_OSC_SPLL_REFCLK_SEL**

Bit Number	Name	Reset Value	Description
0	RC_OSC_REFCLK_SEL	0x1	This bit sets RC OSC as reference clock selection for SPLL.

Table 201 • **SYSTEM_SPREAD_SPECTRUM_CLK**

Bit Number	Name	Reset Value	Description
[7:3]	PLL_SERDESIF_SSMF	0x0	Spread spectrum clocking configuration register for feedback divider.
[2:1]	PLL_SERDESIF_SSMD	0x0	Spread spectrum clocking configuration register for reference divider.
0	PLL_SERDESIF_SSE	0x0	Spread spectrum clocking configuration register.

Table 202 • **SYSTEM_CONF(CONF2)_AXI_MSTR_WNDW_0**

Bit Number	Name	Reset Value	Description
[31:0]	CONF_AXI_MSTR_WNDW_0	0x0	PCIe AXI3-master Window0 configuration register – 0

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 203 • **SYSTEM_CONF(CONF2)_AXI_MSTR_WNDW_1**

Bit Number	Name	Reset Value	Description
[31:0]	CONF_AXI_MSTR_WNDW_1	0x0	PCIe AXI3-master Window0 configuration register – 1

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 204 • **SYSTEM_CONF(CONF2)_AXI_MSTR_WNDW_2**

Bit Number	Name	Reset Value	Description
[31:0]	CONF_AXI_MSTR_WNDW_2	0x0	PCIe AXI3-master Window0 configuration register – 2

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 205 • **SYSTEM_CONF(CONF2)_AXI_MSTR_WNDW_3**

Bit Number	Name	Reset Value	Description
[3:0]	CONF_AXI_MSTR_WNDW_3	0x0	PCIe AXI3-master Window0 configuration register – 3

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 206 • **SYSTEM_CONF(CONF2)_AXI_SLV_WNDW_0**

Bit Number	Name	Reset Value	Description
[31:0]	CONF_AXI_SLV_WNDW_0	0x0	PCIe AXI3-slave Window0 configuration register – 0

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 207 • **SYSTEM_CONF(CONF2)_AXI_SLV_WNDW_1**

Bit Number	Name	Reset Value	Description
[31:0]	CONF_AXI_SLV_WNDW_1	0x0	PCIe AXI3-slave Window0 configuration register – 1

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 208 • **SYSTEM_CONF(CONF2)_AXI_SLV_WNDW_2**

Bit Number	Name	Reset Value	Description
[31:0]	CONF_AXI_SLV_WNDW_2	0x0	PCIe AXI3-slave Window0 configuration register – 2

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 209 • **SYSTEM_CONF(CONF2)_AXI_SLV_WNDW_3**

Bit Number	Name	Reset Value	Description
[3:0]	CONF_AXI_SLV_WNDW_3	0x0	PCIe AXI3-slave Window0 configuration register – 3

Note: All registers are 32-bit. Bits not shown in the table are reserved.

Table 210 • **SYSTEM_DESKEW_CONFIG**

Bit Number	Name	Reset Value	Description
[3:2]	DESKEW_PLL_FDB_CLK	0x0	These bits set the PLL FEEDBACK clock DESKEW register. Delay cells addition in the path of FEEDBACK clock to PLL. 00: Bypass delay cells 01: Add 1-cells 10: Add 2-cells 11: Add 3-cells

Table 210 • **SYSTEM_DESKEW_CONFIG** ***(continued)***

Bit Number	Name	Reset Value	Description
[1:0]	DESKEW_PLL_REF_CLK	0x0	These bits set the PLL REF clock DESKEW register. Delay cells addition in the path of REFERENCE clock to PLL. 00: Bypass delay cells 01: Add 1-cells 10: Add 2-cells 11: Add 3-cells

Table 211 • **SYSTEM_ DEBUG_MODE_KEY**

Bit Number	Name	Reset Value	Description
8	DEBUG_MODE_KEY (M2S/M2GL060 and M2S/M2GL090 only)	0x0	Debug Mode PCIe controller select 1'b0: Controller 0 1'b1: Controller 1
[7:0]	DEBUG_MODE_KEY	0x0	0xA5 is the debug key. Once is correctly written into this REGISTER, APB READ-BUS is multiplexed with PCIe DEBUG data. PCIe DEBUG data is available only when APB-READ is not taking place.

Table 212 • **SYSTEM_ADVCONFIG(ADVCONFIG2)**

Bit Number	Name	Reset Value	Description
6	PCIE_CONFIG_NOSTALL	0x0	PCIE Enabled. 1 - if PCIe configuration space is read before the PMA clock (PCLK) is stable, an APB SLVERR will be generated, otherwise the APB cycle is stalled until the PCLK is stable. PCI Disabled. 1 - An APB PSLVERR is generated if the PCIe configuration space is accessed, otherwise the APB cycle is allowed to complete with simple assertion of PREADY.
5	ENABLE_PERSTN_SUPPORT	0x0	1 enables the ability for the internal PERST monitor to enter reset when PERSTN is asserted. 0 PERSTN is only used to detect the exit from L2P2 and initiate the reset sequence.
4	DISABLE_PIPE_RESET	0x0	1 disables the ability for the internal PERST monitor and L2P2 reset generator to perform an automatic reset sequence of the PIPE logic and PMA block.
3	DISABLE_PCIE_RESET	0x0	1 disables the ability for the internal DLUP/HOTRST and L2P2 reset generator to perform an automatic reset sequence of the main PCIe core logic.
2	K_INFER_ELEC_IDLE	0x0	Enables the inferred electrical idle function in the PCIE core.
1	RESERVED	0x0	–
0	RESERVED	0x0	–

Table 213 • SYSTEM_ ADVSTATUS(ADVSTATUS2)

Bit Number	Name	Reset Value	Description
3	PCIE_ROOT_PORT_IRQ	0x0	Indicates that the root port interrupt is active.
2	PCIE_LANE_REVERSAL_STATUS	0x0	Indicates that PCIE controller is operating in reversed mode.
[1:0]	PCIE_RESET_PHASE	0x0	Indicates the reset phase of the PCIE controller. 00: Reset phase B 01: Reset phase C 11: Operational The PCIe configuration space must not be accessed until phase C is reached. If accesses are attempted in phase B, the APB PREADY will be deasserted until phase C is reached, or if PCIE_CONFIG_NOSTALL = 1 an APB SLVERR generated.

Table 214 • SYSTEM_ENHANCEMENT

Bit Number	Name	Reset Value	Description
11	RESERVED	0x0	Reserved
10	EPCS_RXSKIP_ENABLE	0x0	1 enables the EPCS RXSKIP inputs to the PMA cores. Functions for XAUI and EPCS modes.
9	XGXS_INTERNAL_RESET	0x0	1 - the resets outputs on the XGXS block are connected directly to the reset inputs
[8:7]	CONFIG_DUAL_LINK2LANE3[1:0]	0x0	Specifies which PCIe link the PMA lane is allocated to in DUAL PCI mode. 2'b00: Lane is not used by any PCIe link 2'b01: Lane is connected to PCIe link 0 2'b10: Lane is connected to PCIe link 1 2'b11: Reserved
[6:5]	CONFIG_DUAL_LINK2LANE2[1:0]	0x0	Specifies which PCIe link the PMA lane is allocated to in DUAL PCI mode. 2'b00: Lane is not used by any PCIe link 2'b01: Lane is connected to PCIe link 0 2'b10: Lane is connected to PCIe link 1 2'b11: Reserved
[4:3]	CONFIG_DUAL_LINK2LANE1[1:0]	0x0	Specifies which PCIe link the PMA lane is allocated to in DUAL PCI mode. 2'b00: Lane is not used by any PCIe link 2'b01: Lane is connected to PCIe link 0 2'b10: Lane is connected to PCIe link 1 2'b11: Reserved
[2:1]	CONFIG_DUAL_LINK2LANE0[1:0]	0x0	Specifies which PCIe link the PMA lane is allocated to in DUAL PCI mode. 2'b00: Lane is not used by any PCIe link 2'b01: Lane is connected to PCIe link 0 2'b10: Lane is connected to PCIe link 1 2'b11: Reserved

Table 214 • SYSTEM_ENHANCEMENT

Bit Number	Name	Reset Value	Description
0	ENABLE_DUAL_PCI	0x0	0: SERDESIF operates with single PCIE controller as other devices in the family. 1: SERDESIF operates with dual PCIE controllers/links. PMA lanes are allocated to PCI links as defined in CONFIG_DUAL_LINK2LANE. CONFIG_LINK2LANE is not used.

7.4 SERDES Macro Register

The SERDES macro register map contains control and status information for the SERDES block and lanes. Each block uses 256 register bytes. However, these 256 bytes are mapped to 1 KB to make 32-bit APB output. The APB-to-SERDES programming interface bridge is implemented to convert the system 32-bit APB bus transactions into appropriate 8 bits. Since the SmartFusion2 and IGLOO2 devices map the 4 SERDES lanes into 1KB blocks., the overall register map size is 4 KB. The physical offset location of the SERDES macro registers from the SERDESIF system memory map as shown in Figure 84, page 150.

The 1 Kbyte register space can be divided between the protocol-specific read/write register and generic purpose register.

- Configuration PHY registers (offset 0x000 to 0x03C): These 16 registers are protocol-specific, with a reset value depending on the selected protocol, according to CONFIG_PHY_MODE register settings. For example, PLL_F_PCLK_RATIO register may have different reset values for PCIe and XAUI mode. PCIe Gen1 features are configured using these 16 8-bit registers.
- PCIe 5 Gbps PHY registers (offset 0x040 to 0X0BC): These 32 registers are specific to the PCIe protocol when running at 5 Gbps.
- SERDES Electrical Parameter registers (offset 0X0C0 to 0X18C): These 48 registers are internally reported values of parameters programmed inside the SERDES block. These register descriptions are reserved for factory testing only.
- SERDES Testing registers (offset 0X190 to 0x1FC): These registers are used for testing the SERDES block. These register descriptions are reserved for factory testing only.
- SERDES Recompute register (offset 0x200): This register is a command register that requires PMA control logic to recompute the SERDES parameter based on the new set of register values programmed.
- SERDES PRBS Error Counter registers (offset 0x204 to 0x400): There are 14 registers that are used for bit error rate testing. These registers are for lane0, lane1, lane2, or lane3 and the only difference between lane0, lane1, lane2, and lane3 is the base address specifying which lane it is for. The rest of the register spaces are unused.

The following table lists the SERDES Macro register mapping including reset values. Unused lanes will default to 0x00 values.

Table 215 • SERDES Macro Lane Registers

Register Name	Address Offset (Hex)	Reset Value	Type	Description
CR0	0x000	0x80	RW	Lane Control register 0 (see Table 216, page 177)
ERRCNT_DEC	0x004	0x20	RW	Clock count for error counter decrement (see Table 217, page 178)
RXIDLE_MAX_ERRCNT_THR	0x008	0x48 or 0xF8	RW	Error counter threshold – RX0 idle detect maximum latency (see Table 218, page 178) Reset value for PCIe mode: 0x48 Reset value for other mode: 0xF8

Table 215 • **SERDES Macro Lane Registers *(continued)***

Register Name	Address Offset (Hex)	Reset Value	Type	Description
IMPED_RATIO	0x00C	0x6D	RW	TX impedance ratio (see Table 219, page 178)
PLL_F_PCLK_RATIO	0x010	0x24, 0x34, or 0x00	RW	PLL F settings and PCLK ratio Reset value for PCIe mode (see Table 220, page 178): 0x24: 16-bit pipe interface and 250 MHz PCLK 0x34: 16-bit pipe interface and other PCLK 0x24: 8-bit pipe interface Reset value for other modes: 0x00
PLL_M_N	0x014	0x04, 0x13, or 0x69	RW	PLL M and N settings (see Table 221, page 179) Reset value for PCIe mode: 0x04 Reset value for XAUI mode: 0x13 Reset value for EPCS mode: 0x69
CNT250NS_MAX	0x018	0x7C, 0x27, or 0x1F	RW	250 ns timer base count (see Table 222, page 179) Reset value for PCIe mode: 0x7C Reset value for XAUI mode: 0x27 Reset value for EPCS mode: 0x1F
RE_AMP_RATIO	0x01C	0x00	RW	RX equalization amplitude ratio (see Table 223, page 180)
RE_CUT_RATIO	0x020	0x00	RW	RX equalization cut frequency (see Table 224, page 180)
TX_AMP_RATIO	0x024	0x6D	RW	TX amplitude ratio (Gen 1 PCIe and lower data rates, see Table 225, page 180)
TX_PST_RATIO	0x028	0x15 or 0x00	RW	TX post-cursor ratio (see Table 226, page 180) Reset value for PCIe mode: 0x15 Reset value for XAUI mode: 0x15 Reset value for EPCS mode: 0x15
TX_PRE_RATIO	0x02C	0x00	RW	TX pre-cursor ratio (see Table 227, page 181)
ENDCALIB_MAX	0x030	0x10	RW	End of calibration counter (see Table 228, page 181)
CALIB_STABILITY_COUNT	0x034	0x38	RW	Calibration stability counter (see Table 229, page 181)
POWERDOWN	0x038	0x00	RW	Power-down feature (see Table 230, page 182)
RX_OFFSET_COUNT	0x03C	0x70	RW	RX offset counter (see Table 231, page 183)
PLL_F_PCLK_RATIO_5GBPS (PCIe Gen2 Protocol Only)	0x040	0x24 or 0x04	RW	PLL F settings and PCLK ratio (in PCIe 5 Gbps speed) (see Table 232, page 183) 0x24: 16-bit pipe 0x04: 8-bit pipe
PLL_M_N_5GBPS (PCIe Gen2 Protocol Only)	0x044	0x09	RW	PLL M and N settings (in PCIe 5 Gbps speed) (see Table 233, page 184)
CNT250NS_MAX_5GBPS	0x048	0x7C	RW	250 ns timer base count (in PCIe 5 Gbps speed, see Table 234, page 184)

Table 215 • **SERDES Macro Lane Registers *(continued)***

Register Name	Address Offset (Hex)	Reset Value	Type	Description
TX_PST_RATIO_DEEMP0_FULL	0x050	0x15	RW	TX Post-Cursor ratio with TXDeemp = 0, Full swing, Gen2 speeds (see Table 235, page 184)
TX_PRE_RATIO_DEEMP0_FULL	0x054	0x00	RW	TX pre-cursor ratio TXDeemp = 0, full swing, Gen2 speeds (see Table 236, page 184)
TX_PST_RATIO_DEEMP1_FULL	0x058	0x20	RW	TX post-cursor ratio with TXDeemp = 1, Full swing, Gen2 speeds (see Table 237, page 184)
TX_PRE_RATIO_DEEMP1_FULL	0x05C	0x00	RW	TX pre-cursor ratio TXDeemp = 1, full swing, Gen2 speeds (see Table 238, page 185)
TX_AMP_RATIO_MARGIN0_FULL	0x060	0x80	RW	TX amplitude ratio TXMargin = 0, full swing, Gen2 speeds (see Table 239, page 185)
TX_AMP_RATIO_MARGIN1_FULL	0x064	0x78	RW	TX amplitude ratio TXMargin = 1, full swing, Gen2 speeds (see Table 240, page 185)
TX_AMP_RATIO_MARGIN2_FULL	0x068	0x68	RW	TX amplitude ratio TXMargin = 2, full swing, Gen2 speeds (see Table 241, page 185)
TX_AMP_RATIO_MARGIN3_FULL	0x06C	0x60	RW	TX amplitude ratio TXMargin = 3, full swing, Gen2 speeds (see Table 242, page 185)
TX_AMP_RATIO_MARGIN4_FULL	0x070	0x58	RW	TX amplitude ratio TXMargin = 4, full swing, Gen2 speeds (see Table 243, page 186)
TX_AMP_RATIO_MARGIN5_FULL	0x074	0x50	RW	TX amplitude ratio TXMargin = 5, full swing, Gen2 speeds (see Table 244, page 186)
TX_AMP_RATIO_MARGIN6_FULL	0x078	0x48	RW	TX amplitude ratio TXMargin = 6, full swing, Gen2 speeds (see Table 245, page 186)
TX_AMP_RATIO_MARGIN7_FULL	0x07C	0x40	RW	TX amplitude ratio TXMargin = 7, full swing, Gen2 speeds (see Table 246, page 186)
RE_AMP_RATIO_DEEMP0	0x080	0x00	RW	RX equalization amplitude ratio TXDeemp = 0 (see Table 259, page 189)
RE_CUT_RATIO_DEEMP0	0x084	0x00	RW	RX equalization cut frequency TXDeemp = 0 (see Table 260, page 189)
RE_AMP_RATIO_DEEMP1	0x088	0x00	RW	RX equalization amplitude ratio TXDeemp = 1 (see Table 261, page 189)
RE_CUT_RATIO_DEEMP1	0x08C	0x00	RW	RX equalization cut frequency TXDeemp = 1 (see Table 262, page 190)
TX_PST_RATIO_DEEMP0_HALF	0x090	0x15	RW	TX post-cursor ratio with TXDeemp = 0, half swing (see Table 247, page 186)
TX_PRE_RATIO_DEEMP0_HALF	0x094	0x00	RW	TX pre-cursor ratio TXDeemp = 0, half swing (see Table 248, page 187)
TX_PST_RATIO_DEEMP1_HALF	0x098	0x20	RW	TX post-cursor ratio with TXDeemp = 1, half swing (see Table 249, page 187)
TX_PRE_RATIO_DEEMP1_HALF	0x09C	0x00	RW	TX pre-cursor ratio TXDeemp = 1, half swing (see Table 250, page 187)
TX_AMP_RATIO_MARGIN0_HALF	0x0A0	0x50	RW	TX amplitude ratio TXMargin = 0, half swing, Gen2 speeds (see Table 251, page 187)

Table 215 • **SERDES Macro Lane Registers** ***(continued)***

Register Name	Address Offset (Hex)	Reset Value	Type	Description
TX_AMP_RATIO_MARGIN1_HALF	0x0A4	0x58	RW	TX amplitude ratio TXMargin = 1, half swing, Gen2 speeds (see Table 252, page 187)
TX_AMP_RATIO_MARGIN2_HALF	0x0A8	0x48	RW	TX amplitude ratio TXMargin = 2, half swing, Gen2 speeds (see Table 253, page 188)
TX_AMP_RATIO_MARGIN3_HALF	0x0AC	0x40	RW	TX amplitude ratio TXMargin = 3, half swing, Gen2 speeds (see Table 254, page 188)
TX_AMP_RATIO_MARGIN4_HALF	0x0B0	0x38	RW	TX amplitude ratio TXMargin = 4, half swing, Gen2 speeds (see Table 255, page 188)
TX_AMP_RATIO_MARGIN5_HALF	0x0B4	0x30	RW	TX Amplitude ratio TXMargin = 5, half swing, Gen2 speeds (see Table 256, page 188)
TX_AMP_RATIO_MARGIN6_HALF	0x0B8	0x28	RW	TX amplitude ratio TXMargin = 6, half swing, Gen2 speeds (see Table 257, page 188)
TX_AMP_RATIO_MARGIN7_HALF	0x0BC	0x20	RW	TX amplitude ratio TXMargin = 7, half swing, Gen2 speeds (see Table 258, page 189)
PMA_STATUS	0x0C0	0x80	RO	PMA status register- correct read back value = 0x80 (see Table 263, page 190)
PRBS_CTRL	0x190	0x00	RW	PRBS control register (see Table 264, page 190)
PRBS_ERRCNT	0x194	0x00	RO	PRBS error counter register (see Table 265, page 191)
PHY_RESET_OVERRIDE	0x198	0x00	RW	PHY reset override register (see Table 266, page 191)
PHY_POWER_OVERRIDE	0x19C	0x00	RW	PHY power override register (see Table 267, page 192)
CUSTOM_PATTERN_7_0	0x1A0	0x00	RW	Custom pattern byte 0 (see Table 268, page 192)
CUSTOM_PATTERN_15_8	0x1A4	0x00	RW	Custom pattern byte 1 (see Table 269, page 192)
CUSTOM_PATTERN_23_16	0x1A8	0x00	RW	Custom pattern byte 2 (see Table 270, page 193)
CUSTOM_PATTERN_31_24	0x1AC	0x00	RW	Custom pattern byte 3 (see Table 271, page 193)
Note: Registers 49-99 are factory reserved for testing purposes.				
CUSTOM_PATTERN_39_32	0x1B0	0x00	RW	Custom pattern byte 4 (see Table 272, page 193)
CUSTOM_PATTERN_47_40	0x1B4	0x00	RW	Custom pattern byte 5 (see Table 273, page 194)
CUSTOM_PATTERN_55_48	0x1B8	0x00	RW	Custom pattern byte 6 (see Table 274, page 194)
CUSTOM_PATTERN_63_56	0x1BC	0x00	RW	Custom pattern byte 7 (see Table 275, page 194)
CUSTOM_PATTERN_71_64	0x1C0	0x00	RW	Custom pattern byte 8 (see Table 276, page 195)

Table 215 • **SERDES Macro Lane Registers** ***(continued)***

Register Name	Address Offset (Hex)	Reset Value	Type	Description
CUSTOM_PATTERN_79_72	0x1C4	0x00	RW	Custom pattern byte 9 (see Table 277, page 195)
CUSTOM_PATTERN_CTRL	0x1C8	0x00	RW	Custom pattern control (see Table 278, page 195)
CUSTOM_PATTERN_STATUS	0x1CC	0x00	RO	Custom pattern status register (see Table 279, page 196)
PCS_LOOPBACK_CTRL	0x1D0	0x00	RW	PCS loopback control (see Table 280, page 196)
GEN1_TX_PLL_CCP	0x1D4	0x06	RW	Gen1 transmit PLL current charge pump (see Table 281, page 197)
GEN1_RX_PLL_CCP	0x1D8	0x66	RW	Gen1 receive PLL current charge pump (see Table 282, page 197)
GEN2_TX_PLL_CCP	0x1DC	0x06	RW	Gen2 receive PLL current charge pump (see Table 283, page 197)
GEN2_RX_PLL_CCP	0x1E0	0x66	RW	Gen2 receive PLL current charge pump (see Table 284, page 198)
CDR_PLL_MANUAL_CR	0x1E4	0x00	RW	CDR PLL manual control (see Table 285, page 198)
UPDATE_SETTINGS	0x200	0x00	WO	Update settings command register (see Table 286, page 199)
PRBS_ERR_CYC_FIRST_7_0	0x280	0x00	RO	PRBS first error cycle counter register bits[7:0] (see Table 287, page 199)
PRBS_ERR_CYC_FIRSTLAST_15_8	0x284	0x00	RO	PRBS first error cycle counter register bits[15:8] (see Table 288, page 199)
PRBS_ERR_CYC_FIRSTLAST_23_16	0x288	0x00	RO	PRBS first error cycle counter register bits[23:16] (see Table 289, page 199)
PRBS_ERR_CYC_FIRSTLAST_31_24	0x28C	0x00	RO	PRBS first error cycle counter register bits[31:24] (see Table 290, page 200)
PRBS_ERR_CYC_FIRSTLAST_39_32	0x290	0x00	RO	PRBS first error cycle counter register bits[39:32] (see Table 291, page 200)
PRBS_ERR_CYC_FIRSTLAST_47_40	0x294	0x00	RO	PRBS first error cycle counter register bits[47:40] (see Table 292, page 200)
PRBS_ERR_CYC_FIRSTLAST_49_48	0x298	0x00	RO	PRBS first error cycle counter register bits [49:48] (see Table 293, page 200)
Note: Registers 129 to 159 are not used.				
PRBS_ERR_CYC_LAST_7_0	0x2A0	0x00	RO	PRBS last error cycle counter register bits[7:0] (see Table 294, page 201)
PRBS_ERR_CYC_LAST_15_8	0x2A4	0x00	RO	PRBS last error cycle counter register bits[15:8] (see Table 295, page 201)
PRBS_ERR_CYC_LAST_23_16	0x2A8	0x00	RO	PRBS last error cycle counter register bits[23:16] (see Table 296, page 201)
PRBS_ERR_CYC_LAST_31_24	0x2AC	0x00	RO	PRBS last error cycle counter register bits[31:24] (see Table 297, page 202)

Table 215 • **SERDES Macro Lane Registers** ***(continued)***

Register Name	Address Offset (Hex)	Reset Value	Type	Description
PRBS_ERR_CYC_LAST_39_32	0x2B0	0x00	RO	PRBS last error cycle counter register bits[39:32] (see Table 298, page 202)
PRBS_ERR_CYC_LAST_47_40	0x2B4	0x00	RO	PRBS last error cycle counter register bits[47:40] (see Table 299, page 202)
PRBS_ERR_CYC_LAST_49_48	0x2B8	0x00	RO	PRBS last error cycle counter register bits[49:48] (see Table 300, page 202)

7.4.1 SERDES Block Register Bit Definitions

The following tables give bit definitions for the registers of the SERDES block registers.

The published register field syntax is prefixed by LANEn (where n is 0:3). The register bit is appended to the block register name.

Example: LANE0_CR0.AUTOSHIFT

Table 216 • **CR0**

Bit Number	Name	Reset Value	Description
7	AUTO_SHIFT	0x1	Defines whether the electrical idle 1 pattern is automatically shifted in the SERDES macro after loading the drive pattern. When set to 1, electrical idle I or Drive mode can be entered within a single aTXClkp clock cycle. When set to 0, 23 clock cycles are required to dynamically switch between electrical idle I and Drive mode. In general, this bit is always set to 1. *Unused lanes are set to 0.*
6	FORCE_RX_DETECT	0x0	Forces the result of PCIe receiver detect operation to be always detected. This register can be used on unreliable results of RX detect operations. When set to 1, the result of the PCIe receiver detect operation is always positive and thus makes the PHY non-compliant to PCIe.
[5:4]	CDR_REFERENCE	0x0	Defines the CDR reference PLL mode. By default, these two bits must be set to 00 when RefClk is used for the CDR reference clock.
3	PMA_DRIVEN_MODE	0x0	Puts the CDR PLL in PMA driven mode. When set to 0, the PCS driven mode is selected for locking the SERDES CDR circuitry and when set to 1, PMA driven mode is used. PMA driven mode not supported.
2	CDR_PLL_DELTA	0x0	Defines the frequency comparator threshold value to switch from fine grain locking to frequency lock and thus control the input signal of the PMA macro when CDR is configured in PMA driven mode, and the equivalent function when the PMA is configured in PCS driven mode. When set to 0, the RX clock and TX clock must be in a 0.4% difference range; 0.8% when set to 1.
1	SIGNAL_DETECT_THRESHOLD	0x0	Defines the Schmitt trigger signal detection threshold used to detect electrical idle on RX. When set to 0, threshold is 125 mV (±40%), and when set to 1, threshold is 180 mV (±33%).
0	TX_SELECT_RX_FEEDBACK	0x0	Must be set to 0 when RefClk is used for TX PLL. Set to 1 when the CDR PLL is used as TX PLL reference clock.

Note: This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready).

Table 217 • **ERRCNT_DEC**

Bit Number	Name	Reset Value	Description
[7:0]	ERRCNT_DEC	0x20	In PCS driven mode, the PMA control logic counts the number of errors detected by the PCS logic in order to decide to switch back to frequency lock mode of the CDR PLL. This counter is used to decrement the error counter every 16*errcnt_dec[7:0] aTXClk clock cycles.

Note: This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready).

Table 218 • **RXIDLE_MAX_ERRCNT_THR**

Bit Number	Name	Reset Value	Description
[7:4]	RXIDLE_MAX	0xF	Defines the number of clock cycles required before the activity detected output of the PMA macro and reports either electrical idle or valid input data. This register must be set to at least 3 because the activity detected signal is considered as metastable by the PCS logic.
[3:0]	ERRCNT_THR	0x8	In PCS driven mode, the PMA control logic counts the number of errors detected by the PCS logic in order to decide to switch back to frequency lock mode of the CDR PLL. This register defines the error counter threshold value after which the CDR PLL switches

Note: This register can be reprogrammed any time during operation.

Table 219 • **IMPED_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	IMPED_RATIO	0x8	Fine-tunes the impedance ratio of the PMA macro with a nominal value of 100 Ω, corresponding to a multiplication factor of 1, which is encoded 8'd128. A 150 Ω impedance corresponds to 2/3 ratio, encoded 8'd85.

Note: This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready).

Table 220 • **PLL_F_PCLK_RATIO**

Bit Number	Name	Reset Value	Description
[7:6]	Reserved	PCIe mode: 0x0 XAUI mode:0x0 EPCS mode:0x0	

Table 220 • **PLL_F_PCLK_RATIO**

Bit Number	Name	Reset Value	Description
[5:4]	DIV_MODE0	PCIe mode: 0x3 XAUI mode: 0x0 EPCS mode: 0x0	Defines the ratio between PCLK and aTXClk. PCLK is used by the PCIe PCS logic as well as by the majority of the PMA control logic and thus is also useful for other protocols in order to reduce the amount of logic requiring a high aTXClk frequency. In non-PCIe mode, this register is only useful if pipe_pclkout is used by any logic. A value of 00 is used for divide-by-1, 10 for divide by-2 and 11 for divide-by-4.
[3:0]	F	PCIe mode: 0x4 XAUI mode: 0x0 EPCS mode: 0x0	Defines the aRXF[3:0] and aTXF[3:0] settings of the PMA macro. The same F value is applied to both RX and TX PLL.

Note: This register must only be reprogrammed when PHY is under reset or when both RX PLL and TX PLL are under reset.

Table 221 • **PLL_M_N**

Bit Number	Name	Reset Value	Description
7	CNT250NS_MAX[8]	PCIe mode: 0x0 XAUI mode: 0x0 EPCS mode: 0x0	This bit is concatenated to the **Reg06** register as an MSB to define the 250 ns base time.
[6:5]	M[1:0]	PCIe mode: 0x0 XAUI mode: 0x0 EPCS mode: 0x1	Defines the TX PLL M values and CDR PLL M value settings of the PMA macro. For PCIe, it corresponds to the Gen1 settings. The same M value is applied to both RX and TX PLL.
[4:0]	N[4:0]	PCIe mode: 0x4 XAUI mode: 0x13 EPCS mode: 0x9	Defines the TX PLL N values and CDR PLL N value settings of the PMA macro. For PCIe, it corresponds to the Gen1 settings. The same N value is applied to both RX and TX PLL.

Note: This register must only be reprogrammed when PHY is under reset or when both RX PLL and TX PLL are under reset.

Table 222 • **CNT250NS_MAX**

Bit Number	Name	Reset Value	Description
[7:0]	CNT250NS_MAX	PCIe mode: 0x7C XAUI mode: 0x27 EPCS mode: 0x20	Defines the base count of a 250 ns event based on the aTXClk clock. This counter is used by the CDR PLL in PCS driven mode and also by the PMA control logic for operations such as receiver detect and electrical idle 2 and 3 states. In the case of a non-integer value, the base count should be rounded up. This register must be set correctly for all protocols.

Note: This register must only be reprogrammed when the PHY is under reset for proper operation. It impacts the PCS- driven CDR PLL mode as well as calibration and thus has no effect after calibration is completed (PMA is ready) or if the PHY CDR PLL is used in PMA driven mode.

Table 223 • **RE_AMP_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	RE_AMP_RATIO	0x00	Defines the RX equalization amplitude ratio where the maximum value of 8'd128 corresponds to 100%. If RX equalization is not used, this register can be set to zero.

Note: This register can be reprogrammed during normal operation but the effect will only appear when the parameters for the SERDES receiver are updated (at the end of calibration or when UPDATE_SETTINGS is programmed.

Table 224 • **RE_CUT_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	RE_CUT_RATIO	0x00	Defines the RX equalization cut frequency ratio, used in the computation of Rn[3:0] and Rd[3:0] equalization settings of the PMA macro. The encoding of this register is such that (Rn + Rd) = (RE_CUT_RATIO)/256*W_SETTING where W_SETTING is the result of RX equalization calibration.

Note: This register can be reprogrammed during normal operation but the effect will only appear when the parameters for the SERDES receiver are updated (at the end of calibration or when UPDATE_SETTINGS is programmed).

Table 225 • **TX_AMP_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO	0x80	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden. For PCIe, this register is used for Gen1 speed only.

Note: This register can be reprogrammed during normal operation but the effect will only appear when the parameters for the SERDES transmitter are updated (at the end of calibration, on entry or exit of TX electrical idle I or when UPDATE_SETTINGS is programmed).

Table 226 • **TX_PST_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	TX_PST_RATIO	0x15	Defines the TX post-cursor ratio for the Gen1 speed used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. A value of –3.5 dB corresponds to 8'd21 encoding. TX_PST_RATIO cannot be set to 0x00.

Note: This register can be reprogrammed during normal operation but the effect will only appear when the parameters for the SERDES transmitter are updated (at the end of calibration, on entry or exit of TX Electrical Idle I or when UPDATE_SETTINGS is programmed).

Table 227 • **TX_PRE_RATIO**

Bit Number	Name	Reset Value	Description
[7:0]	TX_PRE_RATIO	0x00	Defines the TX pre-cursor ratio for the Gen1 speed used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%.

Note: This register can be reprogrammed during normal operation but the effect appears only when the parameters for the SERDES transmitter are updated (at the end of calibration, on entry or exit of TX electrical idle I or when UPDATE_SETTINGS is programmed).

Table 228 • **ENDCALIB_MAX**

Bit Number	Name	Reset Value	Description
[7:0]	ENDCALIB_MAX	0x10	Defines the amount of time in microseconds required by the PMA to settle its electrical level after loading electrical idle 1 in the TX driver at the end of calibration. All operations are automatically performed by the PMA control logic but that the SERDES transmitter can start driving data on the link immediately after the end of calibration. By default (except if forbidden by protocol) a 10 µs delay between end of Calibration and Mission mode is set (but any value might work as well).

Note: This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready).

Table 229 • **CALIB_STABILITY_COUNT**

Bit Number	Name	Reset Value	Description
[7:5]	CALIB_SETTLE_MAX	0x1	Defines the amount of time in microseconds required by the PMA to settle its electrical level after loading electrical idle 1 in the TX driver at the end of calibration. Note that all operation is automatically performed by the PMA control logic but that the SERDES transmitter can start driving data on the link immediately after end of calibration. By default, except if forbidden by protocol, a 10 µs delay between end of calibration and mission mode is set (but any value might work as well).
[4:0]	CALIB_STABLE_MAX	0x18	This register defines the number of clock cycles before which the impedance calibrator results (aZCompOp = 1, impedance calibrator result is greater than nominal; and aZCompOp = 0, impedance calibrator result is less than nominal) signal can be checked for stability after impedance calibration control values (aZCalib) modification. aZCompOp = 1, when Impedance calibrator result > nominal 0, when Impedance calibrator result < nominal This is used for TX, RX, and RX equalization calibration.

Note: This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready).

Table 230 • **POWERDOWN**

Bit Number	Name	Reset Value	Description
[7:6]	RXIDLE_MSB	0x0	These bits are used as the most significant bits (MSBs) of the activity detector logic, to specify that no activity has been detected during up to 61 aTXClkp clock cycles. These bits are the two MSBs; the rxidle_max[3:0] field of Reg02 represents the least significant bit (LSB) part.
5	FORCE_SIGNAL	0x0	When this bit is set, the PHY disables the Idle detection circuitry and forces signal detection on the receiver. This bit is generally always set to disable (0) unless the activity detector logic must be bypassed. In that case, the PMA control logic always reports activity detected on the link (when set to 1). This bit can be used, for instance, if the activity detector of the SERDES PMA hard macro does not work for the selected protocol (as outside range of functionality).
4	FORCE_IDLE	0x0	When this bit is set, the PHY disables the Idle detection circuitry and forces electrical Idle detection on the receive side. By default, this bit is generally cleared and might be set only for very specific conditions or testing such as generating a fake loss of signal to the PCS or MAC layer, forcing a retraining of word aligner or any training state machine. As long as this bit is set, the activity detector logic of the PMA control logic reports that no signal is detected on the receive side. If CDR PLL PCS driven mode is selected, the CDR PLL will be directed in lock to the reference clock state, leading to potential wrong data received by the SERDES (because the CDR PLL is not locked to incoming data).
3	NO_FCMP	0x0	When set, this bit disables the frequency comparator logic of the PCS driven CDR PLL control logic. When not set, the frequency comparator logic is no longer part of the condition for going from fine-grain lock state to frequency acquisition. This mode locks to the refclk.
2	PMFF_ALL	0x0	Used with PCIe only, this register when set disables the function that waits for every active lane to have valid data to transmit before generating a global read enable. This bit is intended to be used in case of any issue with this function. When set, each lane might start transmitting data with one 500 MHz clock uncertainty (corresponding to 5 or 10 bits time, depending on the speed of the link). Even if violating the protocol requirement, the PCIe standard is strong enough to support this non-compliance.
1	CDR_ERR	0x0	When set, this register disables the error counter internally of the CDR PLL state machine, which switches back the CDR PLL to frequency mode acquisition when the number of errors counted is higher than the predefined error threshold. This bit is intended for disabling this function in the case of any issue with the PHY. This is available for all SERDES modes.

Table 230 • **POWERDOWN *(continued)***

Bit Number	Name	Reset Value	Description
0	CDR_P1	0x0	Defines the state of the CDR PLL when the PHY is in P1 Low power mode. When set to zero, the CDR PLL is put in reset and low power, enabling maximum power savings. When the opposite component sends the TS1 ordered set to drive the link in recovery, only the PIPE_RXELECIDLE signal is deasserted at the PIPE interface and the PHY waits for the controller to change the pipe_powerdown[1:0] signal back to P0 before retraining the CDR PLL (~6 µs) and sending received data to the controller. When set to 1, the CDR PLL is kept alive in Frequency lock mode in the P1 state, which enables a faster recovery time from the P1 state but which also consumes more power (all RX logic is kept alive and consumes power in the P1 state). This register must not be set for applications which remove the reference clock in P1 mode (generally associated with the CLKREQ# signal, express card application, and more generally power sensitive application).

Note: This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready), except for bit 2, which can only be modified under reset condition.

Table 231 • **RX_OFFSET_COUNT**

Bit Number	Name	Reset Value	Description
[7:5]	RXOFF_SETTLE_MAX	0x3	Defines the number of clock cycles before which the aRXDNullDat signal can be checked for stability after aRXDNull[3:0] modification. This is used also for aRXD, aRXT, and Schmitt trigger calibration. The value of this register expresses a number of (2*N+1) PCLK clock cycles.
[4:0]	RXOFF_STABLE_MAX	0x10	Defines the number of clock cycles where the aRXDNullDat signal is checked for stability.

Note: This register can be reprogrammed when the PHY is under reset or when calibration has completed (PMA is ready).

Table 232 • **PLL_F_PCLK_RATIO_5GBPS (PCIe Gen2 Protocol Only)**

Bit Number	Name	Reset Value	Description
[7:6]	Reserved	0x0	
[5:4]	DIV_MODE1	0x2	Defines the ratio between PCLK and aTXClk for the PCIe Gen2 protocol.
[3:0]	F	0x4	Defines the F setting for the TX PLL and CDR PLL of the PMA macro for the PCIe Gen2 protocol.

Note: This register must only be reprogrammed when PHY is under reset or when both RX PLL and TX PLL are under reset.

Table 233 • **PLL_M_N_5GBPS (PCIe Gen2 Protocol Only)**

Bit Number	Name	Reset Value	Description
7	CNT250NS_MAX_5G BPS[8]	0x0	Defines bit 7 and bit 6 of the cnt250ns_max counter mentioned in Reg18.
[6:5]	M	0x0	Defines the TX PLL M values and CDR PLL M value settings of the PMA macro for the PCIe Gen2 protocol.
[4:0]	N	0x9	Defines the TX PLL N values and CDR PLL N value settings of the PMA macro for the PCIe Gen2 protocol.

Note: This register must only be reprogrammed when PHY is under reset or when both RX PLL and TX PLL are under reset.

Table 234 • **CNT250NS_MAX_5GBPS**

Bit Number	Name	Reset Value	Description
[7:0]	CNT250NS_MAX_5GBPS[7:0]	0x7C EPCS-0x00	This register defines the base count of a 250 ns event based on the aTXClk clock. This counter is used by the CDR PLL in PCS driven mode.

Note: This register must only be reprogrammed when PHY is under reset for proper operation. It impacts the PCS-driven CDR PLL mode as well as calibration. Thus, it has no effect after calibration is completed (PMA is ready) or if the PHY CDR PLL is used in PMA driven mode.

The following registers in Table 235, page 184 through Table 258, page 189 can be reprogrammed during normal operation but the effect appears when the parameters for the SERDES transmitter are updated (on entry or exit of TX electrical idle I, when UPDATE_SETTINGS is programmed, or when any of the PIPE TXSwing, TXDeemp, or TXMargin signals is modified).

Table 235 • **TX_PST_RATIO_DEEMP0_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_PST_RATIO_DEEMP0_FULL	0x15 EPCS-0x00	Defines the TX post-cursor ratio used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. A value of –3.5 dB corresponds to 8’d21 encoding.

Table 236 • **TX_PRE_RATIO_DEEMP0_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_PRE_RATIO_DEEMP0_FULL	0x00	Defines the TX pre-cursor ratio used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%.

Table 237 • **TX_PST_RATIO_DEEMP1_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_PST_RATIO_DEEMP1_FULL	0x20	Defines the TX post-cursor ratio for the Gen2 speed used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. A value of –3.5 dB corresponds to 8’d21 encoding.

Table 238 • **TX_PRE_RATIO_DEEMP1_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_PRE_RATIO_DEEMP1_FULL	0x00	Defines the TX pre-cursor ratio for the Gen2 speed used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%.

Table 239 • **TX_AMP_RATIO_MARGIN0_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN0_FULL	0x80	This register implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 240 • **TX_AMP_RATIO_MARGIN1_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN1_FULL	0x78	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 241 • **TX_AMP_RATIO_MARGIN2_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN2_FULL	0x68	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 242 • **TX_AMP_RATIO_MARGIN3_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN3_FULL	0x60	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 243 • **TX_AMP_RATIO_MARGIN4_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN4_FULL	0x58	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 244 • **TX_AMP_RATIO_MARGIN5_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN5_FULL	0x50	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 245 • **TX_AMP_RATIO_MARGIN6_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN6_FULL	0x48	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 246 • **TX_AMP_RATIO_MARGIN7_FULL**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN7_FULL	0x40	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 247 • **TX_PST_RATIO_DEEMP0_HALF**

Bit Number	Name	Reset Value	Description
[7:0]	TX_PST_RATIO_DEEMP0_HALF	0x15	Defines the TX post-cursor ratio for the Gen2 speed used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. A value of –3.5 dB corresponds to 8’d21 encoding.

Table 248 • TX_PRE_RATIO_DEEMP0_HALF

Bit Number	Name	Reset Value	Description
[7:0]	TX_PRE_RATIO_DEEMP0_HALF	0x00	Defines the TX pre-cursor ratio for the Gen2 speed used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%.

Table 249 • TX_PST_RATIO_DEEMP1_HALF

Bit Number	Name	Reset Value	Description
[7:0]	TX_PST_RATIO_DEEMP1_HALF	0x20	Defines the TX post-cursor ratio for the Gen2 speed used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. A value of –3.5 dB corresponds to 8'd21 encoding.

Table 250 • TX_PRE_RATIO_DEEMP1_HALF

Bit Number	Name	Reset Value	Description
[7:0]	TX_PRE_RATIO_DEEMP1_HALF	0x00	Defines the TX pre-cursor ratio for the Gen2 speed used for selecting the de-emphasis of the switching bit versus non-switching bit. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%.

Table 251 • TX_AMP_RATIO_MARGIN0_HALF

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN0_HALF	0x50	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 252 • TX_AMP_RATIO_MARGIN1_HALF

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN1_HALF	0x58	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 253 • **TX_AMP_RATIO_MARGIN2_HALF**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN2_HALF	0x48	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 254 • **TX_AMP_RATIO_MARGIN3_HALF**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN3_HALF	0x40	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 255 • **TX_AMP_RATIO_MARGIN4_HALF**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN4_HALF	0x38	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 256 • **TX_AMP_RATIO_MARGIN5_HALF**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN5_HALF	0x30	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 257 • **TX_AMP_RATIO_MARGIN6_HALF**

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN6_HALF	0x28	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

Table 258 • TX_AMP_RATIO_MARGIN7_HALF

Bit Number	Name	Reset Value	Description
[7:0]	TX_AMP_RATIO_MARGIN7_HALF	0x20	Implements the TX amplitude ratio used by the TX driver. A value of 128 corresponds to 100% (full voltage); a value of 0 corresponds to 0%. Values higher than 128 are forbidden.

The following registers in Table 259, page 189 through Table 262, page 190 can be reprogrammed during normal operation but the effect appears when the parameters for the SERDES transmitter are updated (on entry or exit of TX electrical idle I, when UPDATE_REGISTER is programmed, or when any of the PIPE TXSwing, TXDeemp, or TXMargin signals is modified).

Table 259 • RE_AMP_RATIO_DEEMP0

Bit Number	Name	Reset Value	Description
[7:0]	RE_AMP_RATIO_DEEMP0	0x00	Defines the RX Equalization amplitude ratio where the maximum value is 8’d128, corresponding to 100%. If RX equalization is not used, this register can be set to zero.

Note: This register must only be reprogrammed when the PHY is under reset for proper operation.

Table 260 • RE_CUT_RATIO_DEEMP0

Bit Number	Name	Reset Value	Description
[7:0]	RE_CUT_RATIO_DEEMP0	0x00	Defines the RX equalization cut frequency ratio, used in the computation of Rn[3:0] and Rd[3:0] equalization settings of the PMA macro. The encoding of this register is such that (Rn + Rd) = (RE_CUT_RATIO)/256*W_SETTING, W_SETTING being the result of the RX equalization calibration.

Table 261 • RE_AMP_RATIO_DEEMP1

Bit Number	Name	Reset Value	Description
[7:0]	RE_AMP_RATIO_DEEMP1	0x00	Defines the RX equalization amplitude ratio where the maximum value is 8’d128, corresponding to 100%. If RX equalization is not used, this register can be set to zero.

Note: This register must only be reprogrammed when the PHY is under reset for proper operation.

Table 262 • **RE_CUT_RATIO_DEEMP1**

Bit Number	Name	Reset Value	Description
[7:0]	RE_CUT_RATIO_DEEMP1	0x00	Defines the RX equalization cut frequency ratio, used in the computation of Rn[3:0] and Rd[3:0] equalization settings of the PMA macro. The encoding of this register is such that (Rn+Rd) = (RE_CUT_RATIO)/256*W_SETTING, W_SETTING being the result of RX equalization calibration.

Table 263 • **PMA_STATUS**

Bit Number	Name	Reset Value	Description
7	PMA_RDY	0x01	This read-only register indicates that the PMA has completed it internal calibration sequence after power-up and PHY reset de-assertion.
[6:0]	–		Reserved

Table 264 • **PRBS_CTRL**

Bit Number	Name	Reset Value	Description
7	–		Unused
6	PRBS_CHK	0x0	When set, this signal starts the PRBS pattern checker. It can be set at the same time as the PRBS generator while the PRBS checker logic waits for 256 clock cycles and CDR being in lock state to enable the PRBS pattern comparison (allowing a total latency of 256 cycles to loop back the transmitted data).
[5:4]	Reserved		
[3:2]	PRBS_TYP[1:0]	0x0	Defines the type of PRBS pattern which is applied. PRBS7 when set to 00, PRBS11 when set to 01, PRBS23 when set to 10, PRBS31 when set to 11.
1	LPBK_EN	0x0	When set, the PMA is put in near-end loopback (serial loopback from TX back to RX). PRBS tests can be done using the near-end loopback of the PMA, some load board, or any far-end loopback implemented in the opposite component. When near-end loopback bit is set, the idle detector always reports valid data, enabling the PCS driven CDR PLL locking logic to lock on input data.
0	PRBS_GEN	0x0	When set, this signal starts the PRBS pattern transmission.

Note: This register can be programmed any time but has functional impact because it can configure the SERDES in loopback or generate the PRBS pattern.

Table 265 • **PRBS_ERRCNT**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERRCNT[7:0]	0x00	This test reports the number of PRBS errors detected when the PRBS test is applied. This register is automatically cleared when the PRBS_EN register is cleared (requiring testing the value of this register when the test is running). The PRBS error counter saturates at 254 errors, the 255 count value corresponding to an error code where the CDR PLL is not locked to incoming data. When such an error code is detected, the PRBS test must wait for a longer time for the CDR PLL to synchronize on input data before enabling the PRBS checker or simply timeout, reporting that no data has been received at all.

Note: The PRBS error counter logic also counts errors when the PRB Sin variant (all zero value) is obtained, considering input data as error data.

Table 266 • **PHY_RESET_OVERRIDE**

Bit Number	Name	Reset Value	Description
7	RXHF_CLKDN	0x0	When set, this signal disables the RX PLL VCO settings by applying a static zero to the PMA aRXHfClkDnb signal.
6	TXHF_CLKDN	0x0	When set, this signal disables the TX PLL VCO by applying a static zero to the PMA aTXHfClkDnb signal.
5	RXPLL_RST	0x0	When set, this signal resets the RX PLL settings by applying a static zero to the PMA aCdrPllRstb signal.
4	TXPLL_RST	0x0	When set, this signal initializes the TX PLL settings by applying a static zero to the PMA aTXPllRstb signal.
3	RXPLL_INIT	0x0	When set, the signal initializes the RX PLL settings by applying a static one to the PMA aRXPllDivInit signal.
2	TXPLL_INIT	0x0	When set, this signal initializes the TX PLL settings by applying a static one to the PMA aTXPllDivInit signal.
1	RX_HIZ	0x0	When set, this signal forces the RX driver to hiZ, applying a static one to the PMA aForceRXHiZ signal.
0	TX_HIZ	0x0	When set, this signal forces the TX driver to hiZ, applying a static one to the PMA aForceTXHiZ signal.

Note: This register can be programmed any time but has functional impact on the SERDES because it can put the PLL under reset or place part of the SERDES in Low power mode, bypassing the functional mode.

Table 267 • **PHY_POWER_OVERRIDE**

Bit Number	Name	Reset Value	Description
[7:1]			Unused
0	RX_PWRDN	0x0	When set, this register forces the RX PMA logic to be in Power-down mode.

Note: This register can be programmed any time but has functional impact on the SERDES because it can power-down the receiver part of the SERDES, bypassing the functional mode.

Table 268 • **CUSTOM_PATTERN_7_0**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN[7:0]	0x00	Enables bit 7 to bit 0 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the line. In the latter case, the PMA hard macro function is used to perform a word alignment function.

Note: This register can be programmed any time but has no functional impact as long as the custom pattern generation is not enabled and selected.

Table 269 • **CUSTOM_PATTERN_15_8**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [15:8]	0x00	Enables bit 15 to bit 8 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line, but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the line. In the latter case, the PMA hard macro function is used to perform a word alignment function.

Note: This register can be programmed any time but has no functional impact as long as the custom pattern generation is not enabled and selected.

Table 270 • **CUSTOM_PATTERN_23_16**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [23:16]	0x00	Enables bit 23 to bit 16 to program a custom pattern instead of the implemented PRBS generator/checker. PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the line. In the latter case, the PMA hard macro function is used to perform a word alignment function.

Note: This register can be programmed any time but has no functional impact as long as the custom pattern generation is not enabled and selected.

Table 271 • **CUSTOM_PATTERN_31_24**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [31:24]	0x00	This register enables bit 31 to bit 24 to program a custom pattern instead of the implemented PRBS generator/checker. PRBS mode must still be selected to transmit this custom pattern on the transmit line, but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the line. In the latter case, the PMA hard macro function is used to perform word alignment function.

Note: This register can be programmed any time but has no functional impact as long as the custom pattern generation is not enabled and selected.

Table 272 • **CUSTOM_PATTERN_39_32**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [39:32]	0x00	Enables bit 39 to bit 32 to program a custom pattern instead of the implemented PRBS generator/checker. PRBS mode must still be selected to transmit this custom pattern on the transmit line, but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the line. In the latter case, the PMA hard macro function is used to perform a word alignment function.

Note: This register can be programmed any time but has no functional impact as long as the custom pattern generation is not enabled and selected.

Table 273 • **CUSTOM_PATTERN_47_40**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [47:40]	0x00	Enables bit 47 to bit 40 to program a custom pattern instead of the implemented PRBS generator/checker. PRBS mode must still be selected to transmit this custom pattern on the transmit line, but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for purpose of eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the line. In the latter case, the PMA hard macro function is used to perform a word alignment function.

Note: This register can be programmed any time but has no functional impact as long as the custom pattern generation is not enabled and selected.

Table 274 • **CUSTOM_PATTERN_55_48**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [55:48]	0x00	Enables bit 55 to bit 48 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the line. In the latter case, the PMA hard macro function is used to perform a word alignment function.

Note: This register can be programmed any time but has no functional impact as long as the custom pattern generation is not enabled and selected.

Table 275 • **CUSTOM_PATTERN_63_56**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [63:56]	0x00	Enables bit 63 to bit 56 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the line. In the latter case, the PMA hard macro function is used to perform a word alignment function.

Note: This register can be programmed any time but has no functional impact as long as the custom pattern generation is not enabled and selected.

Table 276 • **CUSTOM_PATTERN_71_64**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [71:64]	0x00	Enables bit 71 to bit 64 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the line. In the latter case, the PMA hard macro function is used to perform a word alignment function.

Note: This register can be programmed any time but has no functional impact as long as the custom pattern generation is not enabled and selected.

Table 277 • **CUSTOM_PATTERN_79_72**

Bit Number	Name	Reset Value	Description
[7:0]	CUSTOM_PATTERN [79:72]	0x00	Enables bit 79 to bit 72 to program a custom pattern instead of the implemented PRBS generator/checker. The PRBS mode must still be selected to transmit this custom pattern on the transmit line but this mode enables the generation of any repeated sequence of data. It can be used, for instance, for single lane PCIe compliance pattern generation (for the purpose of an eye diagram compliance check) or can even be looped back to the receiver to check if any error is detected on the line. In the latter case, the PMA hard macro function is used to perform a word alignment function.

Note: This register can be programmed any time but has no functional impact as long as the custom pattern generation is not enabled and selected.

Table 278 • **CUSTOM_PATTERN_CTRL**

Bit Number	Name	Reset Value	Description
7	RSVD		
6	CUST_AUTO	0x0	When this register is set, the word alignment is performed automatically by a state machine that checks whether the received pattern is word-aligned with the transmitted pattern and automatically use the PMA CDR PLL skip bit function to find the alignment. Once the word alignment is detected, the custom pattern checker is now word-aligned and the custom pattern checker can be enabled for detecting and counting any error over time.

Table 278 • **CUSTOM_PATTERN_CTRL *(continued)***

Bit Number	Name	Reset Value	Description
5	CUST_SKIP	0x0	This register is used in RX word alignment manual mode. The custom pattern requires word alignment in order to be checked by the receiver (as opposed to a PRBS pattern, which does not require this word alignment function). In Manual mode, read the CUST_SYNC register in order to check whether the word is aligned. If not in Manual mode, a one bit skip is to the CDR PLL must be done by writing one then zero in this register (and repeat this sequence until the receiver is aligned).
4	CUST_CHK	0x0	Enables the error counter. When clear, it also resets the error counter. Thus, the error counter must be checked before clearing this register.
[3:1]	CUST_TYP	0x0	Defines whether the custom pattern generated on the link is generated by the custom pattern register or by one of the hard-coded patterns: 000: Custom pattern register 100: All-zero pattern (0000…00) 101: All-one pattern (1111…11) 110: Alternated pattern (1010…10) 111: Dual alternated pattern (1100…1100)
0	CUST_SEL	0x0	When set, this signal replaces the PRBS data transmitted on the link by the custom pattern. The PRBS_SEL register must also be set for transmitting the custom pattern on the link.

Note: This register can be programmed any time but has functional impact on the SERDES because it can directly activate some part of the SERDES (aRXSkipBit), changing the current bitstream reception (thus creating alignment errors).

Table 279 • **CUSTOM_PATTERN_STATUS**

Bit Number	Name	Reset Value	Description
[7:5]	CUST_STATE	0x0	Reports the current state of the custom pattern word alignment state machine. It can be useful for debug purposes (Refer verilog code).
4	CUST_SYNC	0x0	Reports that the custom pattern is word-aligned.
[3:0]	CUST_ERROR[3:0]	0x0	When the custom pattern checker is enabled, this status register reports the number of errors detected by the logic when the custom word aligner is in synchronization (it starts counting only after a first matching pattern has been detected. The word alignment status can be checked through (cust_state==3'b101) or CUST_SYNC register asserted.

Table 280 • **PCS_LOOPBACK_CTRL**

Bit Number	Name	Reset Value	Description
[7:4]	–	–	Unused
3	MESO_SYNC	0x0	Is read-only and reports whether the mesochronous clock alignment state machine has completed its process, having thus aligned the CDR receive clock to the transmit clock.

Table 280 • **PCS_LOOPBACK_CTRL *(continued)***

Bit Number	Name	Reset Value	Description
2	MESO_LPBK	0x0	When set, this register enables Mesochronous loopback mode, which forces PMA received data to be re-transmitted on the PMA TX interface. This mode requires that no PPM exists between RX data and TX data (thus, both sides of the link use the same reference clock) and also performs alignment of the CDR clock to the transmit clock using the PMA CDR PLL skip bit functionality. This alignment is automatically performed by a state machine when this loopback register is set.
1	Reserved	0x0	–
0	Reserved	0x0	–

Table 281 • **GEN1_TX_PLL_CCP**

Bit Number	Name	Reset Value	Description
[7:3]	–	–	Reserved
[2:0]	ATXICP_RATE0[2:0]	0x0	Defines the TX PLL charge pump current when the PMA is running in PCIe Gen1 speed or in any other protocol. This register is R/W in order to enable changing the default value by register programming, which is expected to be performed before reset deassertion.

Note: This register can be programmed when the PHY is under reset.

Table 282 • **GEN1_RX_PLL_CCP**

Bit Number	Name	Reset Value	Description
7	Reserved	0x0	Reserved
[6:4]	ARXCDRICP_RATE0[2:0]	0x0	Defines the RX PLL charge pump current when the PMA is frequency locked and running in PCIe Gen1 speed or in any other protocol. This register is R/W in order to enable changing the default value by register programming, which is expected to be performed before reset deassertion.
3	Reserved	0x0	Reserved
[2:0]	ARXICP_RATE0[2:0]	0x0	Defines the RX PLL charge pump current when the PMA is CDR locked and running in PCIe Gen1 speed or in any other protocol. This register is R/W in order to enable changing the default value by register programming, which is expected to be performed before reset deassertion.

Note: This register can be programmed when the PHY is under reset.

Table 283 • **GEN2_TX_PLL_CCP**

Bit Number	Name	Reset Value	Description
7	Reserved	0x0	Reserved

Table 283 • **GEN2_TX_PLL_CCP**

Bit Number	Name	Reset Value	Description
[6:4]	ATXICP_RATE1[2:0]	0x0	Defines the TX PLL charge pump current when the PMA is running in PCIe Gen2 speed. This register is R/W in order to enable changing the default value by register programming, which is expected to be performed before reset deassertion.

Note: This register can be programmed when the PHY is under reset and is implemented only if PCIe Gen2 is supported by the PHY.

Table 284 • **GEN2_RX_PLL_CCP**

Bit Number	Name	Reset Value	Description
7	Reserved	0x0	Reserved
[6:4]	ARXCDRICP_RATE1[2:0]	0x0	Defines the RX PLL charge pump current when the PMA is frequency locked and running in PCIe Gen2 speed. This register is R/W in order to enable changing the default value by register programming, which is expected to be performed before reset deassertion.
3	Reserved	0x0	Reserved
[2:0]	ARXICP_RATE1[2:0]	0x0	Defines the RX PLL charge pump current when the PMA is CDR locked and running in PCIe Gen2 speed. This register is R/W in order to enable changing the default value by register programming, which is expected to be performed before reset deassertion

Note: This register can be programmed when the PHY is under reset and is implemented only if PCIe Gen2 is supported by the PHY.

Table 285 • **CDR_PLL_MANUAL_CR**

Bit Number	Name	Reset Value	Description
[7:3]	Reserved	0x0	Reserved
2	FINE_GRAIN	0x0	In PCS-driven mode when this register is set, it enables forcing the CDR PLL state machine in fine grain state. In this state, the CDR PLL locks on receive data, making RX data and RX CLOCKP valid on the PMA interface.
1	COARSE_GRAIN	0x0	When set, this register enables forcing the CDR PLL state machine when used in PCS driven mode (see Reg00 bit 3 set to 0) in coarse grain state. In this state, the CDR PLL performs a coarse grain lock on receive data, enabling adjustment of its clock up to 5000 PPM.
0	FREQ_LOCK	0x0	When set, this register enables forcing the CDR PLL state machine when used in PCS-driven mode (see Reg00 bit 3 set to 0) in frequency lock state. In this state, the CDR PLL does not lock on receive data but on the reference clock.

Table 286 • **UPDATE_SETTINGS**

Bit Number	Name	Reset Value	Description
[7:0]	UPDATE_SETTINGS[7:0]	0x00	Is a transient register (read always reports 0) where writing a 1 in bit 0 will trigger a new computation of PMA settings based on the value written in register space registers. Note that for PCIe, Microsemi recommends not using this command register when the link is not transitioning to low power state or changing rate.

Note: This register can be programmed any time, except during calibration, and triggers the RX/TX shift load logic to load new programmed settings into the SERDES. Thus, it must be written to 0x01 only after a coherent set of register programming has been programmed.

Table 287 • **PRBS_ERR_CYC_FIRST_7_0**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_FIRST[7:0]	0x00	PRBS last error cycle counter register bits[7:0]. This read-only register reports on which clock cycle the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of bit error rate testing (BERT).

Note: The first error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 288 • **PRBS_ERR_CYC_LAST_15_8**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_FIRST[15:8]	0x00	PRBS last error cycle counter register bits[15:8]. This read-only register reports on which clock cycle the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of bit error rate testing (BERT).

Note: The first error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 289 • **PRBS_ERR_CYC_LAST_23_16**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_FIRST[23:16]	0x00	PRBS last error cycle counter register bits[23:16]. This read-only register reports on which clock cycle the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of BERT.

Note: The first error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 290 • **PRBS_ERR_CYC_LAST_31_24**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_FIRST[31:24]	0x00	PRBS last error cycle counter register bits[31:24]. This read-only register reports on which clock cycle the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of BERT.

Note: The first error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 291 • **PRBS_ERR_CYC_LAST_39_32**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_FIRST[39:32]	0x00	PRBS last error cycle counter register bits[39:32]. This read-only register reports on which clock cycle the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of BERT.

Note: The first error cycle counter information complementing the total number errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 292 • **PRBS_ERR_CYC_LAST_47_40**

Bit Number	Name	Reset Value	Description
[7:2]	Reserved	0x0	Reserved
[1:0]	PRBS_ERR_CYC_FIRST[47:40]	0x0	PRBS last error cycle counter register bits [47:40]. This read-only register reports on which clock cycle the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of bit error rate testing (BERT).

Note: The first error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 293 • **PRBS_ERR_CYC_LAST_49_48**

Bit Number	Name	Reset Value	Description
[7:2]	Reserved	0x0	Reserved

Table 293 • **PRBS_ERR_CYC_LAST_49_48** ***(continued)***

Bit Number	Name	Reset Value	Description
[1:0]	PRBS_ERR_CYC_FIRST[49:48]	0x0	PRBS last error cycle counter register bits[49:48]. This read-only register reports on which clock cycle the error counter has first been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of BERT.

Note: The first error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 294 • **PRBS_ERR_CYC_LAST_7_0**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[7:0]	0x0	PRBS last error cycle counter register bits [7:0]. This read-only register reports on which clock cycle the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of BERT.

Note: The last error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 295 • **PRBS_ERR_CYC_LAST_15_8**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[15:8]	0x00	PRBS last error cycle counter register bits[15:8]. This read-only register reports on which clock cycle the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of bit error rate testing (BERT).

Note: The last error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 296 • **PRBS_ERR_CYC_LAST_23_16**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[23:16]	0x00	PRBS last error cycle counter register bits [23:16]. This read-only register reports on which clock cycle the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of BERT.

Note: The last error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 297 • **PPRBS_ERR_CYC_LAST_31_24**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[31:24]	0x00	PRBS last error cycle counter register bits[31:24]. This read-only register reports on which clock cycle the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of BERT.

Note: The last error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 298 • **PRBS_ERR_CYC_LAST_39_32**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[39:32]	0x0	PRBS last error cycle counter register bits[39:32]. This read-only register reports on which clock cycle the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of bit error rate testing (BERT).

Note: The last error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 299 • **PRBS_ERR_CYC_LAST_47_40**

Bit Number	Name	Reset Value	Description
[7:0]	PRBS_ERR_CYC_LAST[47:40]	0x0	PRBS last error cycle counter register bits[47:40]. This read-only register reports on which clock cycle the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of bit error rate testing (BERT).

Note: The last error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.

Table 300 • **PRBS_ERR_CYC_LAST_49_48**

Bit Number	Name	Reset Value	Description
[7:2]	Reserved	0x0	Reserved
[1:0]	PRBS_ERR_CYC_LAST[49:48]	0x0	PRBS last error cycle counter register bits[49:48]. This read-only register reports on which clock cycle the error counter has last been incremented after the PRBS error counter is enabled. It is a 50-bit counter, enabling performance of BERT.

Note: The last error cycle counter information complementing the total number of errors detected might give information about bursts of errors or distributed errors, but statistics might also be required. The test can be rerun several times with different test periods.