SmartFusion2 and IGLOO2

High Speed Serial Interface Configuration

Microsemi

Table of Contents

Introduction

The High Speed Serial Interface block in the SmartFusion2 and IGLOO2 families (Figure 2) provides multiple high speed serial protocols, such as PCIe end-point and XAUI. In addition, it enables the FPGA fabric to connect with the External Physical Coding Sublayer (EPCS) Interface and implement any user-defined protocol in the fabric. The device may contain one or more High Speed Serial Interface blocks depending on its size. Refer to the SmartFusion2 or IGLOO2 Datasheet.

As you make selections in the core configurator, it automatically narrows down the subsequent choices and defaults. Only the relevant ports appear in the generated macro.

In this document, we describe how you can configure a High Speed Serial Interface instance and define how the signals are connected. For more details about the High Speed Serial Interface, refer to the SmartFusion2 or IGLOO2 High Speed Serial Interfaces User's Guide.

To access the High Speed Serial Interface Configurator:

1. Instantiate the High Speed Serial Interface (SERDES) core from the Catalog into the SmartDesign Canvas, as shown in Figure 1.

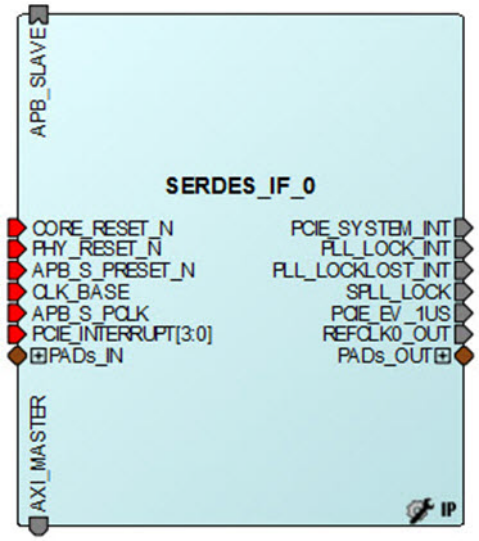

***Figure 1* • SERDES Block Instantiation on the SmartDesign Canvas**

2. Double-click on each SERDES block on the Canvas to open the Configurator.

   By default, SERDESIF_0 is checked when you open the Configurator. If the instance name of the SERDES you have instantiated in the SmartDesign canvas is not SERDES_IF_0, uncheck

SERDESIF_0 and check the correct SERDES identification box to match the correct SERDES instance name.

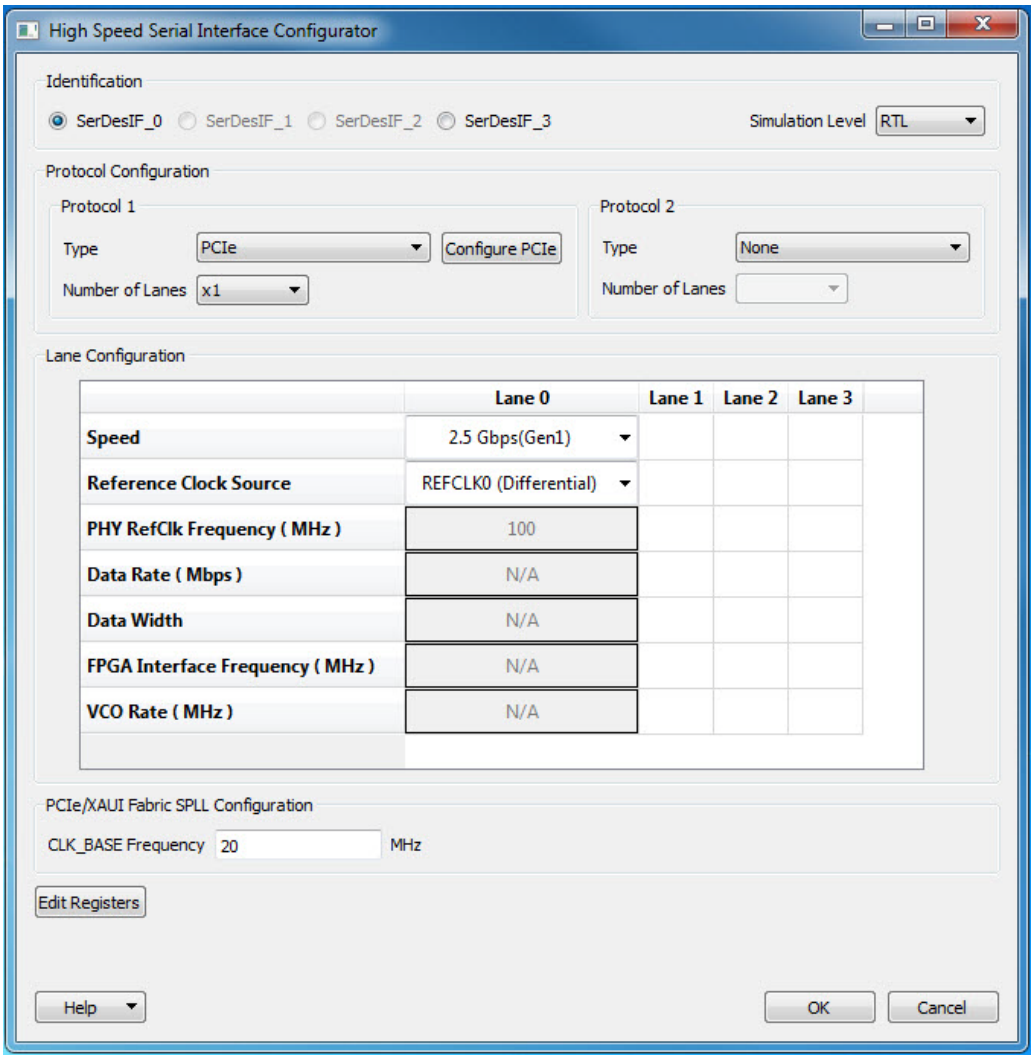

Figure 2 • **High Speed Serial Interface Configurator**

1 – Functionality

Identification

SmartFusion2 and IGLOO2 devices may contain one or more High Speed Serial Interface blocks. The first row of checkboxes lets you identify which High Speed Serial Interface block (SERDESIF_0, SERDESIF_1, SERDESIF_2, SERDESIF_3) is being configured.

Note: Some devices have only one High Speed Serial Interface block. If so, you must select SERDESIF_0. Refer to your device datasheet (SmartFusion2 Datasheet; IGLOO2 Datasheet) for a list of resources available on a device.

Protocol Configuration

IGLOO2 and SmartFusion2 have one SERDES block that supports two protocols: Protocol 1 and Protocol 2. For each Protocol, you must configure the Type and Number of Lanes.

Protocol 1

Select your Protocol type from the drop-down menu:

- PCIe
- PCIe (Reverse)
- XAUI
- EPCS

Protocol 2

Select your Protocol type from the drop-down menu:

- EPCS (available only when PCIe or PCIe Reverse is selected for Protocol 1)

When you select the Protocol PCI or PCIe Reverse, you must click the Configure PCIe button to configure additional options for the PCI mode of SERDES. See "Configure PCIe" on page 7 for details.

Note: You must Configure Protocol 1 before configuring Protocol 2.

Protocol 2 Types are context sensitive; they depend on the options you have selected in Protocol 1.

Protocol 2 Type selection is disabled when you select XAUI in Protocol 1. It is activated only when PCIe, PCIe Reverse or EPCS is selected for Protocol 1 and you use less than four lanes. Refer to Table 1-1 for the Protocol 1 and Protocol 2 configuration combinations.

Number of Lanes

Select the number of lanes you wish to configure for Protocol 1 from the drop-down menu:

- X1 - Configure for 1 lane
- X2 - Configure for 2 lanes
- X4 - Configure for all 4 lanes

Note: Items in the drop-down list are context sensitive and depend on the Protocol Type.

If Protocol Type is XAUI, all four lanes are selected by default.

Protocol 1 and 2: Type, Number of Lanes, Speed

Table 1-1 shows the protocol combinations that are feasible within a single High Speed Serial Interface block.

Table 1-1 • Available Protocols

Protocol Type	Protocol #	Lane Width	Lane Assignment	Description	Speed Choices
PCIe	Protocol 1	x1	Lane 0		Gen1 (2.5 Gbps), Gen2 (5.0 Gbps)
		x2	Lane 0, Lane 1		
		x4	Lane 0, Lane 1, Lane 2, Lane 3		
PCIe Reverse	Protocol 1	x2	Lane 2, Lane 3		Gen1 (2.5 Gbps), Gen2 (5.0 Gbps)
		x4	Lane 0, Lane 1, Lane 2, Lane 3		
XAUI	Protocol 1	x4	Lane 0, Lane 1, Lane 2, Lane 3		3.125 Gpbs
EPCS	Protocol 1	x1	Users can select Lane 0, 1, 2 or 3		Custom Speed
		X2	Lane 0, Lane 1		
		x4	Lane 0, Lane 1, Lane 2, Lane 3		
	Protocol 2	x1	Users can select Lane 2 or 3	Available only when Protocol 1 is PCIe, PCIe Reverse or EPCS	
		x2	Lane 2, Lane 3		

Configure PCIe

The Configure PCIe button appears only when you select the PCIe or PCI Reverse Protocol.

Configuration

The Configuration tab sets your Identification Registers, Fabric Interface, Bus Address Registers and Options (Figure 1-1).

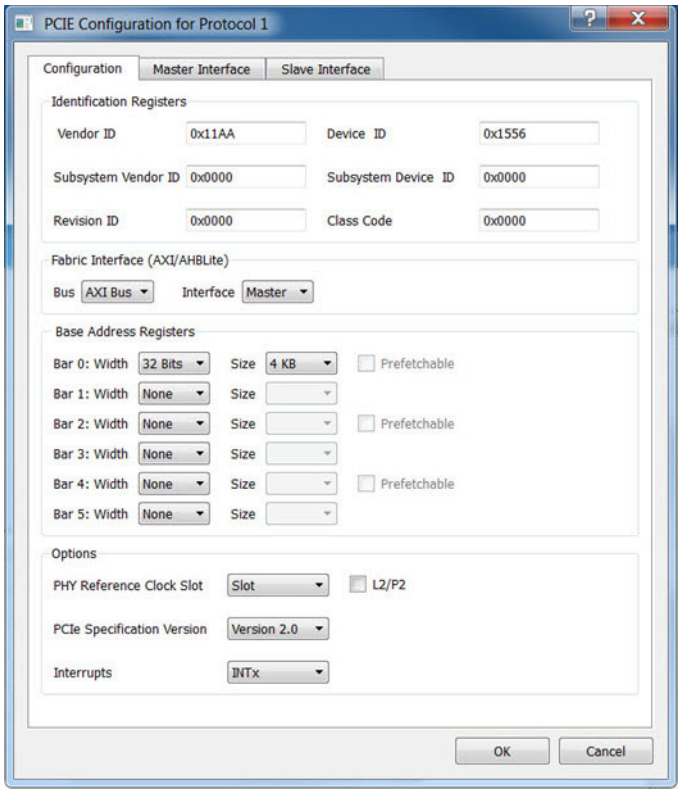

Figure 1-1 • PCIe Configuration - Configuration Tab

Identification Registers

You can assign 16-bit hexadecimal signatures to the following six identification registers for PCIe:

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

Fabric Interface

Use this field to configure the Bus Standard (AXI or AHBLite) and the Interface (Master Only, Slave Only, or both) for the PCIe protocol.

In PCIe mode, the SERDES block can act as an AXI or AHBLite Master.

You must instantiate a COREAXI or CoreAHBLite Bus into the SmartDesign Canvas and then connect the Master and/or Slave Bus Interface (BIF) of the SERDES to the Master and/or Slave BIF of the COREAXI bus or COREAHBLite bus.

Base Address Registers

The individual fields of the six Base Address Registers (Bar 0 through Bar 5) can be configured as follows:

- **Width** - The width on even registers can be 32 bit or 64 bit. If an even register is set to 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 KB to 2 GB. Some devices may go up to 1 GB. Refer to your device datasheet (SmartFusion2 Datasheet; IGLOO2 Datasheet) for more information.
- **Prefetchable** - Enabled only on even registers with 64-bit width.

Options

Options enables you to configure the following:

- **PHY Reference Clock Slot** - Sets your reference clock to Slot or Independent.
- **L2/P2** - Click the checkbox to add PCIE_WAKE_N, PCIE_WAKE_REQ and PCIE_PERST_N ports.
- **PCIe Specification Version** - Sets the Specification Version to 1.0, 1.1 or 2.0.
- **Interrupt** - Sets the Interrupt to:
  - MSI 1
  - MSI 2
  - MSI 4
  - MSI 8
  - MSI 16
  - MSI 32
  - INTx

Your Interrupt selection sets bit 16 of the PCIE_MSI_CTRL_STATUS register and bits [19:17] of the PCIE_MSI_CTRL_STATUS register as shown in Table 1-2 below.

Table 1-2 • MSI and Register Settings

Interrupt Selected	Setting for Bit 16 of PCIE_MSI_CTRL_STATUS Register	Setting for Bits [19:17] of PCIE_MSI_CTRL_STATUS Register
MSI 1	1	000
MSI 2	1	001
MSI 4	1	010
MSI 8	1	011
MSI 16	1	100
MSI 32	1	101
INTx	0 (Disable MSI)	000

Master Interface

PCIe/PCIe Reverse Protocol 1 or Protocol 2 enables you to use the Master Interface tab to configure up to four PCI windows (Window 0 through Window 3) with the following parameters (as shown in Figure 1-2):

- Size
- PCIe BAR (Base Address Register)
- Local Address

- PCIe Address

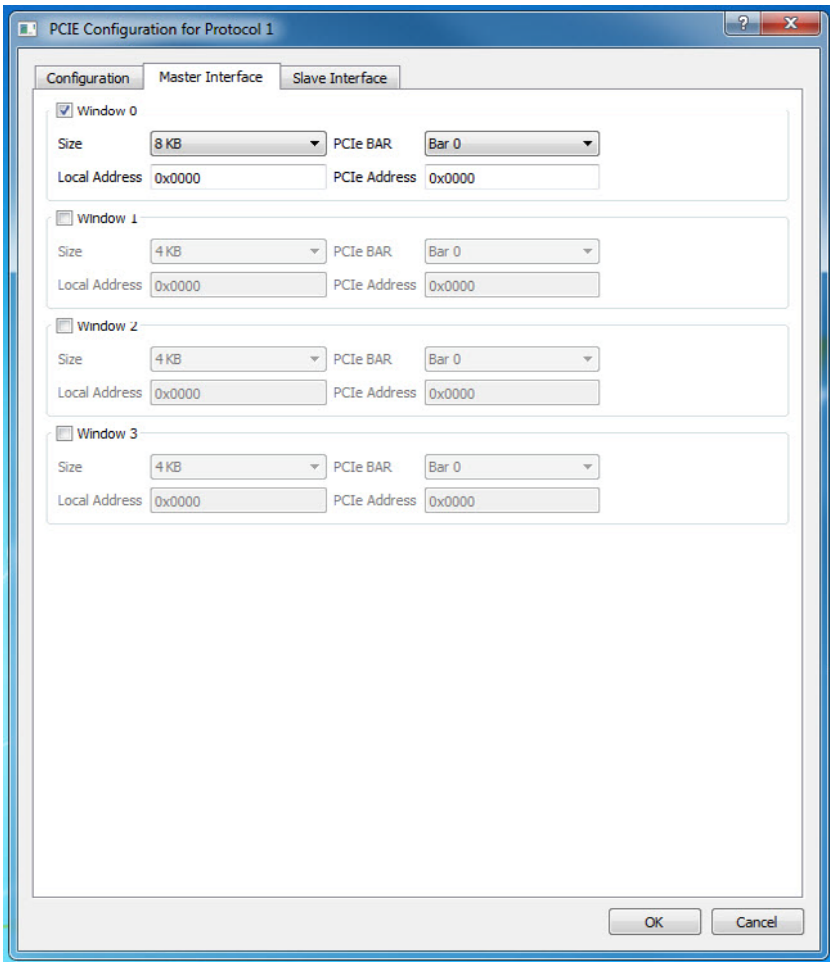

Figure 1-2 • PCIe Configuration - Master Interface Tab

Size

For each of the windows 0 through 3, select one of the available window sizes: 4KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB, 512KB, 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB, 256MB, 512MB, 1GB and 2GB.

The default selection is 4KB. The size selected is mapped to bits [31:12] of WindowsX_1 where X can be 0, 1, 2, or 3.

PCIe BAR

Select one of the following BAR (Base Address Register):

- BAR0
- BAR1
- BAR2
- BAR3
- BAR4
- BAR5
- BAR0/1
- BAR2/3

- BAR4/5

Bar Size is mapped to bits [5:0] of WindowsX_2, where X can be 0, 1, 2 or 3, as shown in Table 1-3.

Table 1-3 • BAR Size and Corresponding Bit Settings

BAR Size	Bit Settings
BAR0, BAR0/1	0x01
BAR 1	0x02
BAR 2, BAR2/3	0x04
BAR 3	0x08
BAR 4, BAR4/5	0x10
BAR5	0x20

Local Address

Local Address is 20 bits wide and is mapped to bits [31:12] of Base address AXI Master WindowsX_0. The LSB bits [11:0] of AXI Master WindowsX_0 are reserved and the configurator will account for these bits. Do not include these reserved bits when you specify the local address.

PCIe Address

PCIe Address is mapped to bits [31:12] (LSB of Base address AXI Master WindowsX_2) and bits [31:0] (MSB of Base address AXI Master WindowsX_3).

Slave Interface

If you select PCIe or PCIe Reverse Protocol the Slave Interface tab enables you to configure up to four PCI windows, Window 0 through Window 3, with the following parameters (as shown in Figure 1-3 on page 11):

- Size
- Local Address
- PCIe Address
- Traffic Class: Selects the PCIe Traffic Class in the PCIe packet header.
- Relaxed Ordering: Enables you to generate the PCIe TLP using a selectable relaxed ordering bit.

- No Snoop: Enables you to generate the PCIe TLP using a selectable no snoop bit.

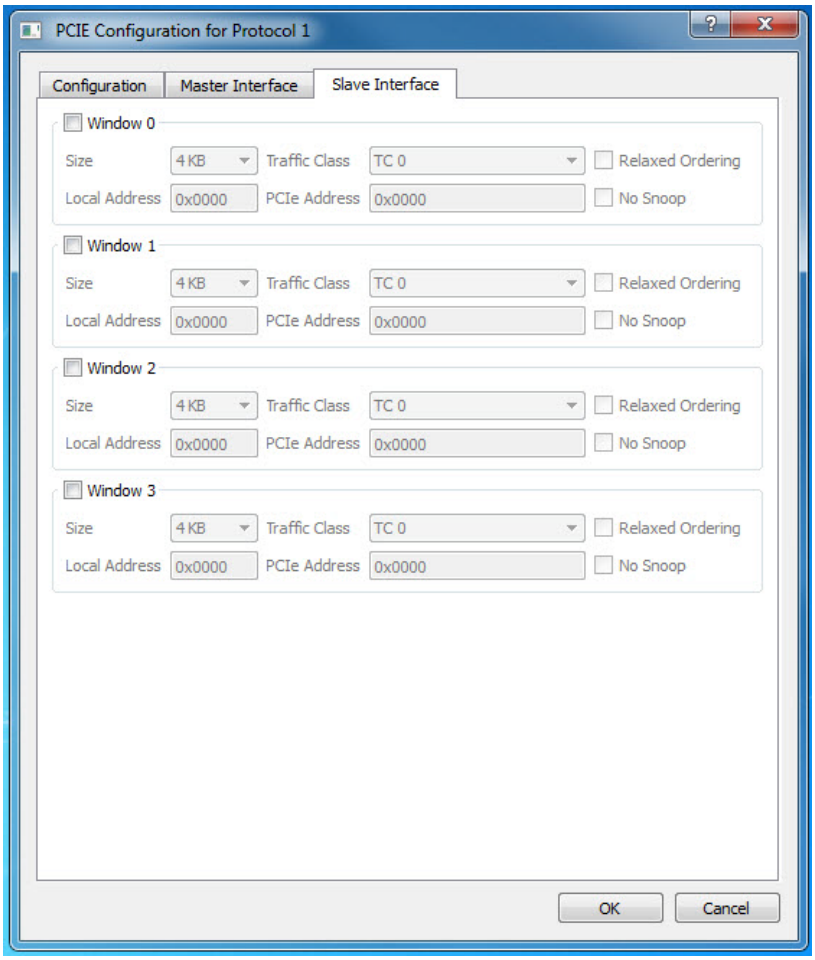

Figure 1-3 • PCIe Configuration - Slave Interface Tab

The Size, Local Address and PCIe Address options are the same as those for the Master Interface. Refer to "Master Interface" on page 8 for more information.

Traffic Class

Traffic Class enables you to set your Traffic Class and corresponding register bits, as shown in Table 1-4. The traffic class is required by the PCI Express packet header. Its value determines the relative priority of a given transaction as it traverses the PCIe link. You can use the Traffic Class value to create a priority scheme for different packets.

- TC 0 (Default)
- TC 1
- TC 2
- TC 3
- TC 4
- TC 5
- TC 6
- TC 7

Table 1-4 • Traffic Class and Corresponding Bit Setting

Traffic Class	Bit Setting for WindowX_2[4:2]
TC 0	000
TC 1	001
TC 2	010
TC 3	011
TC 4	100
TC 5	101
TC 6	110
TC 7	111

Lane Configuration

Use Lane Configuration to configure up to four lanes for your SERDES. The SERDES can be configured to run in dual-protocol mode. Refer to Table 1-1 on page 6 for lane configuration for dual mode operation.

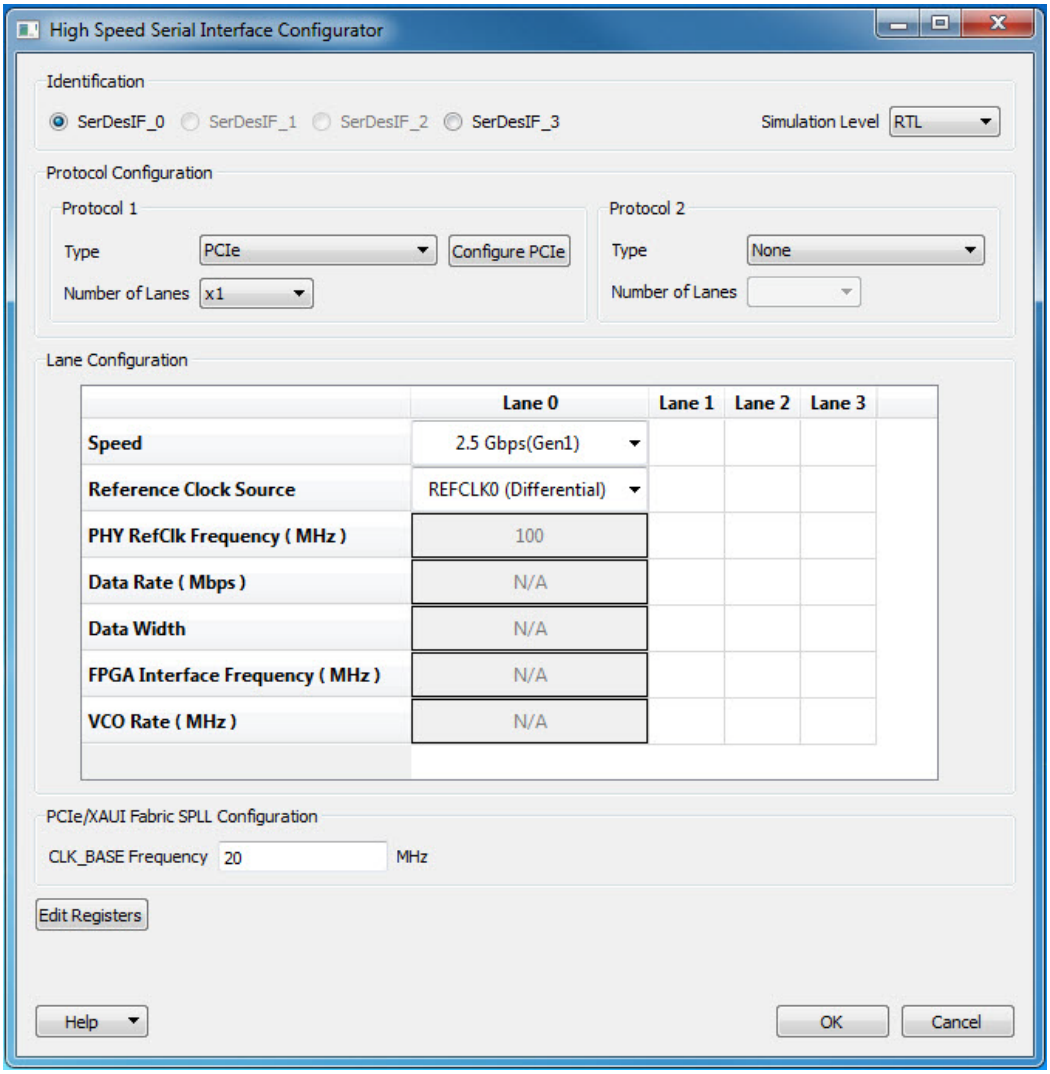

Figure 1-4 • High Speed Serial Interface Configurator

Speed - Available selections depend on your selected Protocol. Refer to Table 1-1 on page 6 for the valid Speeds.

Reference Clock Source - Two clock sources are available: REFCLK0 and REFCLK1. Each can be differential or single-ended. You can select one of the following options for Protocol 1 and Protocol 2:

- REFCLK0 (Differential),
- REFCLK1 (Differential),
- REFCLK0 (Single-Ended),
- REFCLK1 (Single-Ended)
- Fabric (Available only for EPCS Protocol)

Note: Lane 0 and Lane 1 share the same Reference Clock and Lane 2 and Lane 3 share the same Reference Clock. The selected Reference Clock is always available as REFCLK0_OUT or REFCLK1_OUT and can be used as clock source for logic inside Fabric.

PHY RefClk Frequency (MHz) - This is a fixed value for all protocols except EPCS Custom Speed, in which case you can enter values between 100 and 160 MHz.

Data Rate (Mbps) - Read-only fixed value for all protocols except EPCS Custom Speed, in which case you can select the data rate from the drop-down menu. Data Rates are computed based on the PHY RefClk Frequency.

Data Width - Read-only fixed value for all protocols except EPCS Custom Speed. The displayed value is computed and updated based on your selected PHY RefClk Frequency and Data Rate.

FPGA Interface Frequency (MHz) - Read-only fixed value for all protocols except EPCS Custom Speed. The displayed value is computed and updated based on the PHY RefClk Frequency and Data Rate you select.

VCO Rate (MHz) - Read-only fixed value for all protocols except EPCS Custom Speed. The displayed value is computed and updated based on the PHY RefClk Frequency and Data Rate you select.

PCIe/XAUI Fabric SPLL Configuration

The SPLL configuration fields are relevant only for PCIe and XAUI protocols (Figure 1-4). For the PCIe protocol, enter a valid value between 20 and 200 MHz for the CLK_BASE Frequency.

For the XAUI protocol, the CLK_BASE Frequency is read-only and fixed at 156.25 MHz.

EPCS Protocol and Bus Slicing

The Configurator sets the width of EPCS_TX_DATA and EPCS_RX_DATA ports to be always 20-bit wide. If your effective EPCS data width is less than 20 bits, you must slice the TX_DATA and RX_DATA bus and connect them to the rest of your design. The correct slices are: EPCS_<n>_RX_DATA [19:19-<width> + 1] and EPCS_<n>_TX_DATA [<width> -1:0] where <n> can be 0, 1, 2, or 3 depending on the configured lanes and <width> is the effective EPCS data width. You may tie the extra data bits off.

High Speed Serial Interface Control Registers

The High Speed Serial Interface has a set of registers that can be configured at runtime. The configuration values for these registers represent different parameters, for example, AXI BAR Window. For details about these registers, refer to the Microsemi SmartFusion2 or IGLOO2 High Speed Serial Interfaces User's Guide.

High Speed Serial Interface Registers Configuration

To enter the High Speed Serial Interface configuration values, specify the register values when you are configuring the High Speed Serial Interface. Click **Edit Registers** in the High Speed Serial Interface Configurator (Figure 2 on page 4) to open the Registers Configuration dialog box (Figure 1-5). Data entered in this configurator is written at power up in the High Speed Serial Interface registers, as

described in the SmartFusion2 DDR Controller and Serial High Speed Controller Initialization Methodology document.

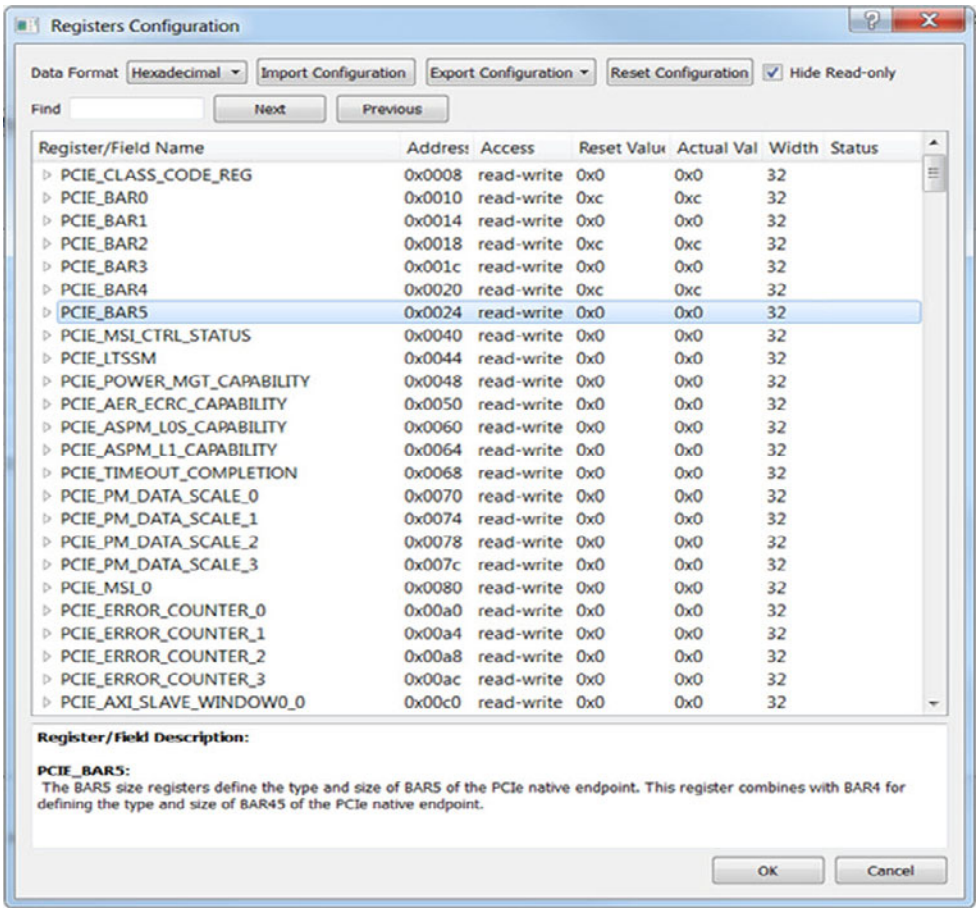

Figure 1-5 • High Speed Serial Interface Registers Configuration Dialog Box

Alternatively, you can click the Import Configuration button and import an existing configuration text file to configure the Registers.

The Registers Configuration dialog box enables you to enter High Speed Serial Interface register values using a graphical interface. The dialog box has the following features:

- **Registers Table** - Enter register values one-by-one using the Registers Table. To enter a register value, expand the register data tree (using the arrow or + sign), and click the **Actual Value** column to edit.
- **Import Configuration** - Import complete register configurations from text files. Register configuration syntax is shown below; Microsemi recommends using this method.
- **Export Configuration** - You can export the current register configuration data into a text file. The syntax of the exported file is the same as that of importable register configuration text files. For example:

```
PCIE_AXI_MASTER_WINDOW0_0              0x00000000
PCIE_AXI_MASTER_WINDOW0_1              0xfffff001
PCIE_AXI_MASTER_WINDOW0_2              0x00000000
PCIE_AXI_MASTER_WINDOW0_3              0x00000000
PCIE_AXI_MASTER_WINDOW1_0              0x00001000
PCIE_AXI_MASTER_WINDOW1_1              0xfffff001
PCIE_AXI_MASTER_WINDOW1_2              0x4
```

- **Reset Configuration** - Click Reset Configuration to undo any changes you have made to the register configuration. This deletes all register configuration data and you must either re-import or reenter this data. The data is reset to the hardware reset values.
- **Hide Read-Only Registers** - Enables you to show or hide the read-only registers in the Register Table. These registers are mostly status registers and do not contribute to the configuration.

When you generate your FPGA, the configuration register data entered in this configurator is used to initialize the High Speed Serial Interface simulation model when performing a BFM simulation.

Firmware (SmartFusion2 Only)

When you generate the SmartDesign, the following files are generated in the <project dir>/firmware/drivers_config/sys_config directory. These files are required for the CMSIS firmware core to compile properly and contain information regarding your current design, including peripheral configuration data and clock configuration information for the MSS. Do not edit these files manually; they are recreated every time your root design is regenerated.

- sys_config.c
- sys_config.h
- sys_config_SERDESIF_<0-3>.h - High Speed Serial Interface configuration data
- sys_config_SERDESIF_<0-3>.c - High Speed Serial Interface configuration data

Simulation Level

There are three levels of ModelSim simulation supported for the High Speed Serial Interface block depending on the selected protocol. See the SERDESIF Simulation User Guide for more information.

BFM_CFG - This level provides a Bus Functional Model of only the APB configuration bus of the High Speed Serial Interface block. You will be able to write and read the different configuration and status bits from the High Speed Serial Interface block through its APB slave interface. The status bits value will not change based on the APB state; they are kept at their reset values. This simulation level is available for all protocols.

BFM_PCIe - This simulation level provides the BFM_CFG level plus the ability to communicate with the High Speed Serial Interface block through the master and slave AXI or AHB bus interfaces. Although no serial communication actually goes through the High Speed Serial Interface block, this scenario enables you to validate the fabric interface connections. This simulation level is only available for the PCIe protocol.

RTL - This simulation level enables you to fully simulate the High Speed Serial Interface block from the fabric interface to the serial data interface. This results in a longer simulation runtime. This simulation level is available for all protocols.

Simulation Files - SmartFusion2

When you generate the SmartDesign associated with your MSS, the following simulation files are generated in the <project dir>/simulation directory:

- **test.bfm** - Top-level BFM file, first executed during any simulation that exercises the SmartFusion2 MSS' Cortex-M3 processor. It executes peripheral_init.bfm and user.bfm, in that order.
- **peripheral_init.bfm** - Contains the BFM procedure that emulates the CMSIS::SystemInit() function run on the Cortex-M3 before you enter the main() procedure. It copies the configuration data for any peripheral used in the design to the correct peripheral configuration registers and then waits for all the peripherals to be ready before asserting that the user can use these peripherals.
- **SERDESIF_<0-3>_init.bfm** - Contains BFM write commands that simulate writes of the High Speed Serial Interface configuration register data you entered (using the Edit Registers dialog box) into the High Speed Serial Interface registers.

- **SERDESIF_<0-3>_user.bfm** - Intended for user commands that simulate transactions being initiated off-chip (via the SERDES interface). You can simulate the datapath by adding your own commands in this file. Commands in this file will be "executed" after peripheral_init.bfm has completed.
- **user.bfm** - Intended for user commands. You can simulate the datapath by adding your own commands in this file. Commands in this file will be "executed" after peripheral_init.bfm has completed.

Using the files above, the configuration path is simulated automatically. You only need to edit the user.bfm file to simulate the datapath. Do not edit the test.bfm, peripheral_init.bfm, or SERDESIF_<0-3>_init.bfm files as these files are recreated every time your root design is regenerated.

Simulation Files - IGLOO2

When you generate the SmartDesign associated with your HPMS, the following simulation files are generated in the <project dir>/simulation directory:

- **ENVM_init.mem** - In IGLOO2 designs the configuration data for any peripheral used in the design is stored in the ENVM_init.mem file. The IGLOO2 simulation library uses this file. Libero SoC creates this file for the simulation just prior to the simulation run
- **SERDESIF_<0-3>_init.bfm** - Contains BFM write commands that simulate writes of the High Speed Serial Interface configuration register data you entered (using the Edit Registers dialog box) into the High Speed Serial Interface registers.
- **SERDESIF_<0-3>_user.bfm** - Intended for user commands that simulate transactions being initiated off-chip (via the SERDES interface). You can simulate the datapath by adding your own commands in this file. Commands in this file will be "executed" after peripheral_init.bfm has completed.

High Speed Serial Interface Configuration Path - SmartFusion2

The configuration register data is used by the CMSIS::SystemInit() function compiled with your firmware application code. The SystemInit() function is run before the user main() function in your application. The Peripheral Initialization solution requires that, in addition to specifying High Speed Serial Interface configuration register values, you configure the APB configuration data path in the MSS (FIC_2). The

SystemInit() function writes the data to the High Speed Serial Interface configuration registers via the FIC_2 APB interface.

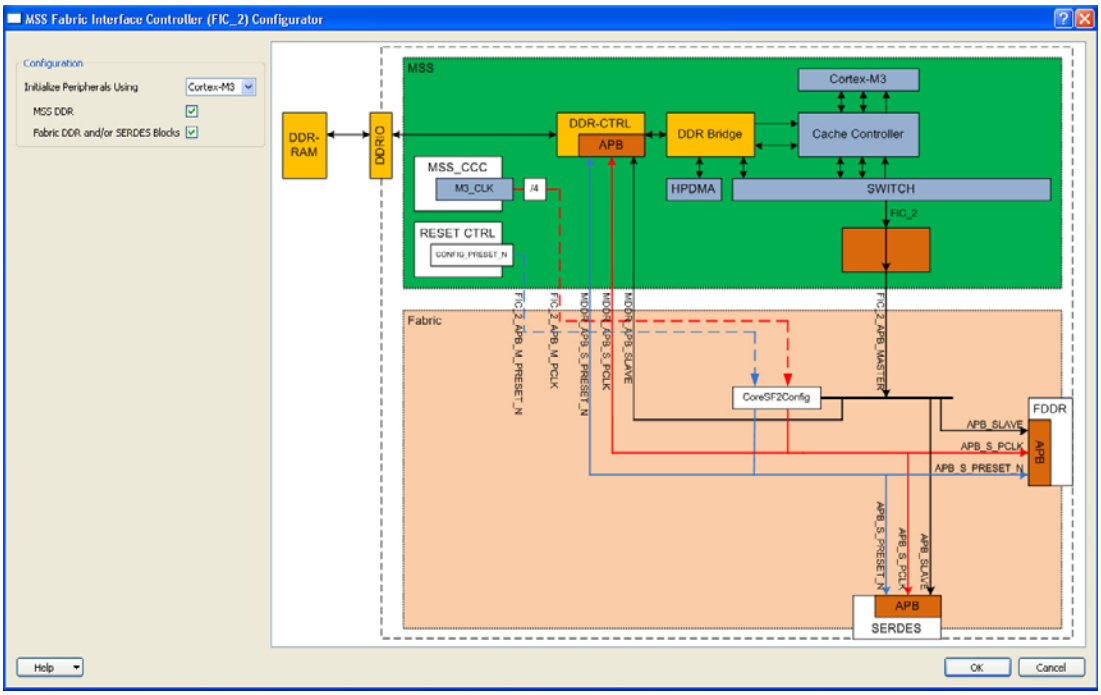

Figure 1-6 • FIC_2 Configurator Overview

To configure the FIC_2 interface:

1. Open the FIC_2 configurator dialog box (Figure 1-6) from the MSS configurator.
2. Select **Initialize peripherals using Cortex-M3**. Make sure that you have clicked the checkbox to enable **Fabric DDR and/or SERDES blocks** and the MSS DDR option (if you are using it).
3. Click **OK** to save your settings. This exposes the FIC_2 configuration ports (Clock, Reset, and APB bus interfaces), as shown in Figure 1-7.
4. Generate the MSS. The FIC_2 ports (FIC_2_APB_MASTER, FIC_2_APB_M_PCLK and FIC_2_APB_M_RESET_N) are now exposed at the MSS interface and can be connected to the CoreSF2Config and CoreSF2Reset as per the Peripheral Initialization solution specification.

For details on configuring and connecting the CoreSF2Config and CoreSF2Reset cores, refer to the SmartFusion2 DDR Controller and Serial High Speed Controller Initialization Methodology document.

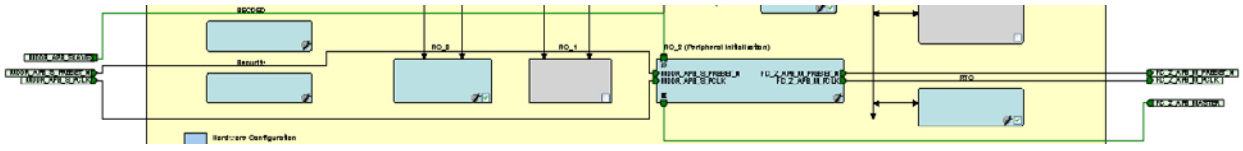

Figure 1-7 • FIC_2 Ports

High Speed Serial Interface Configuration Path - IGLOO2 Initialization

You must use System Builder for IGLOO2 designs that use the SERDES block. System Builder generates an APB bus to handle the internals of the High Speed Serial Interface Configuration Path used for the initial SERDES block configuration and initialization.

The SERDES block is always configured as an APB Slave to the System Builder Block; it cannot be configured as a master. You must configure a fabric master to use the SERDES block as a slave.

After generating your System Builder Block you must connect the SERDES configuration path signals to the System Builder Block:

- Connect the APB_SLAVE BIF of the SERDES block to the corresponding Slave BIF of the System Builder block.
- Connect the APB_S_PCLK port of the SERDES block to the INIT_APB_S_PCLK port of the System Builder block.
- Connect the APB_S_PRESET_N port of the SERDES block to the INIT_APB_S_PRESET_N port of the System Builder block.

SERDES initialization data will now be automatically loaded into the SERDES initialization registers at device bootup.

2 – Port Description

Table 2-1 **• APB Ports**

Port	Direction	Port Group
APB_S_PRDATA[31:0]	OUT	APB_SLAVE
APB_S_PREADY	OUT	
APB_S_PSLVERR	OUT	
APB_S_PADDR[13:2]	IN	
APB_S_PENABLE	IN	
APB_S_PSEL	IN	
APB_S_PWDATA[31:0]	IN	
APB_S_PWRITE	IN	
APB_S_PCLK	IN	
APB_S_PRESET_N	IN	

***Table 2-2* • PCIe Ports**

Port	Direction
CORE_RESET_N	IN
PHY_RESET_N	IN
CLK_BASE	IN
PCIE_INTERRUPT[3:0]	IN
PCIE_SYSTEM_INT	OUT
SPLL_LOCK	OUT
PLL_LOCK_INT	OUT
PLL_LOCKLOST_INT	OUT
PCIE_EV_1US	OUT
PCIE_WAKE_N	OUT
PCIE_WAKE_REQ	IN
PCIE_PERST_N	IN
REFCLK<x>_OUT where x can be 0 or 1 depending on whether REFCLK0 or REFCLK1 is selected as the Reference Clock Source.	OUT

***Table 2-3* • PCIe AXI Master Ports**

Port	Direction	Port Group
AXI_M_AWID[3:0]	OUT	AXI_MASTER
AXI_M_AWADDR[31:0]	OUT	
AXI_M_AWLEN[3:0]	OUT	
AXI_M_AWSIZE[1:0]	OUT	
AXI_M_AWBURST[1:0]	OUT	
AXI_M_AWVALID	OUT	
AXI_M_AWREADY	IN	
AXI_M_WID[3:0]	OUT	
AXI_M_WSTRB[7:0]	OUT	
AXI_M_WLAST	OUT	
AXI_M_WVALID	OUT	
AXI_M_WDATA[63:0]	OUT	
AXI_M_WREADY	IN	
AXI_M_BID[3:0]	IN	
AXI_M_BRESP[1:0]	IN	
AXI_M_BVALID	IN	
AXI_M_BREADY	OUT	
AXI_M_ARID[3:0]	OUT	
AXI_M_ARADDR[31:0]	OUT	
AXI_M_ARLEN[3:0]	OUT	
AXI_M_ARSIZE[1:0]	OUT	
AXI_M_ARBURST[1:0]	OUT	
AXI_M_ARVALID	OUT	
AXI_M_ARREADY	IN	
AXI_M_RID[3:0]	IN	
AXI_M_RDATA[63:0]	IN	
AXI_M_RRESP[1:0]	IN	
AXI_M_RLAST	IN	
AXI_M_RVALID	IN	
AXI_M_RREADY	OUT	

***Table 2-4* • PCIe AXI Slave Ports**

Port	Direction	Port Group
AXI_S_AWID[3:0]	IN	AXI_SLAVE
AXI_S_AWADDR[31:0]	IN	
AXI_S_AWLEN[3:0]	IN	
AXI_S_AWSIZE[1:0]	IN	
AXI_S_AWBURST[1:0]	IN	
AXI_S_AWVALID	IN	
AXI_S_AWREADY	OUT	
AXI_S_AWLOCK[1:0]	IN	
AXI_S_WID[3:0]	IN	
AXI_S_WSTRB[7:0]	IN	
AXI_S_WLAST	IN	
AXI_S_WVALID	IN	
AXI_S_WDATA [63:0]	IN	
AXI_S_WREADY	OUT	
AXI_S_BID[3:0]	OUT	
AXI_S_BRESP[1:0]	OUT	
AXI_S_BVALID	OUT	
AXI_S_BREADY	IN	
AXI_S_ARID[3:0]	IN	
AXI_S_ARADDR[31:0]	IN	
AXI_S_ARLEN[3:0]	IN	
AXI_S_ARSIZE[1:0]	IN	
AXI_S_ARBURST[1:0]	IN	
AXI_S_ARVALID	IN	
AXI_S_ARLOCK[1:0]	IN	
AXI_S_ARREADY	OUT	
AXI_S_RID[3:0]	OUT	
AXI_S_RDATA[63:0]	OUT	
AXI_S_RRESP[1:0]	OUT	
AXI_S_RLAST	OUT	
AXI_S_RVALID	OUT	
AXI_S_RREADY	IN	

***Table 2-5* • PCIe AHBLite Master Ports**

Port	Direction	Ports Group
AHB_M_HADDR[31:0]	OUT	AHB_MASTER
AHB_M_HBURST[1:0]	OUT	
AHB_M_HSIZE[1:0]	OUT	
AHB_M_HTRANS[1:0]	OUT	
AHB_M_HWRITE	OUT	
AHB_M_HWDATA[31:0]	OUT	
AHB_M_HREADY	IN	
AHB_M_HRESP	IN	
AHB_M_HRDATA[31:0]	IN	

***Table 2-6* • PCIe AHBLite Slave Ports**

Port	Direction	Ports Group
AHB_S_HSEL	IN	AHB_SLAVE
AHB_S_HADDR[31:0]	IN	
AHB_S_HBURST[1:0]	IN	
AHB_S_HSIZE[1:0]	IN	
AHB_S_HTRANS[1:0]	IN	
AHB_S_HWRITE	IN	
AHB_S_HWDATA[31:0]	IN	
AHB_S_HREADYOUT	OUT	
AHB_S_HRESP	OUT	
AHB_S_HREADY	IN	
AHB_S_HRDATA[31:0]	OUT	

Table 2-7 • **XAUI Ports**

Port	Direction
XAUI_RXD[63:0]	OUT
XAUI_RXC[7:0]	OUT
XAUI_RX_CLK	OUT
XAUI_VNDRESLO[31:0]	OUT
XAUI_VNDRESHI[31:0]	OUT
XAUI_MMD_MDC	IN
XAUI_MMD_MDI	IN
XAUI_MMD_MDI_EXT	IN
XAUI_MMD_MDOE_IN	IN
XAUI_MMD_PRTAD[4:0]	IN
XAUI_MMD_DEVID[4:0]	IN
XAUI_LOOPBACK_IN	IN
XAUI_MDC_RESET	IN
XAUI_TX_RESET	IN
XAUI_RX_RESET	IN
XAUI_TXD[63:0]	IN
XAUI_TXC[7:0]	IN
XAUI_MMD_MDO	OUT
XAUI_MMD_MDOE	OUT
XAUI_LOWPOWER	OUT
XAUI_LOOPBACK_OUT	OUT
XAUI_MDC_RESET_OUT	OUT
XAUI_TX_RESET_OUT	OUT
XAUI_RX_RESET_OUT[3:0]	OUT
CORE_RESET_N	IN
PHY_RESET_N	IN
SPLL_LOCK	OUT
PLL_LOCK_INT	OUT
PLL_LOCKLOST_INT	OUT
XAUI_OUT_CLK	OUT
XAUI_PMA_READY_N	OUT
REFCLK<x>_OUT where x can be 0 or 1 depending on whether REFCLK0 or REFCLK1 is selected as the Reference Clock Source.	OUT

Table 2-8 • **EPCS Ports per Lane**

Port	Direction	Ports Group
EPCS_<n>_PWRDN	IN	EPCS_<n>_IN Where n can be 0, 1, 2 or 3 depending on the number of configured lanes.
EPCS_<n>_TX_VAL	IN	
EPCS_<n>_TX_OOB	IN	
EPCS_<n>_RX_ERR	IN	
EPCS_<n>_RESET_N	IN	
EPCS_<n>_TX_DATA[<wd>:0]	IN	
EPCS_FAB_REF_CLK When Fabric is selected as the Reference Clock Source in the Configurator	IN	
EPCS_<n>_READY	OUT	EPCS_<n>_OUT Where n can be 0, 1, 2 or 3 depending on the number of configured lanes.
EPCS_<n>_TX_CLK_STABLE	OUT	
EPCS_<n>_TX_CLK	OUT	
EPCS_<n>_RX_CLK	OUT	
EPCS_<n>_RX_VAL	OUT	
EPCS_<n>_RX_IDLE	OUT	
EPCS_<n>_TX_RESET_N	OUT	
EPCS_<n>_RX_RESET_N	OUT	
EPCS_<n>_RX_DATA[19:0]	OUT	
REFCLK<x>_OUT Where x can be 0 or 1 depending on whether REFCLK0 or REFCLK1 is selected as the Reference Clock	OUT	

Note: <n> indicates the lane on which EPCS is configured.

<wd> can have a maximum value of 19 (that is 20 bits) depending on the configurations.

In the EPCS custom speed mode, regardless of the data rate chosen in the configurator, the configurator exposes all 20 bits for EPCS_<n>_TX_DATA and EPCS_<n>_RX_DATA ports. If the chosen data rate in the configurator is less than 20 bits, you must slice the EPCS_<n>_TX_DATA bus and the EPCS_<n>_RX_DATA bus and connect them to the rest of your design. The correct slices are: EPCS_<n>_RX_DATA [19:19 -<width> + 1] and EPCS_<n>_TX_DATA [<width> -1:0] where <n> can be 0, 1, 2, or 3 depending on the configured lanes and width is the effective EPCS data width. For the extra data bits, you may tie them off.

***Table 2-9* • PAD Ports**

Ports	Direction	Ports Group	Description
REFCLK<x>_SE	IN	PADs_IN	Where x can be 0 or 1, depending on whether REFCLK0 (Single-ended) or REFCLK1 (Single-ended) is selected as the Reference Clock Source.
RXD0_P, RXD0_N	IN		Differential input pair for lane 0 (Rx data)
RXD1_P, RXD1_N	IN		Differential input pair for lane 1 (Rx data)
RXD2_P, RXD2_N	IN		Differential input pair for lane 2 (Rx data)
RXD3_P, RXD3_N	IN		Differential input pair for lane 3 (Rx data)
REFCLK<x>_P, REFCLK<x>_N	IN		Differential input reference clock pair. These port names can be REFCLK0 or REFCLK1, depending on your selection (refer to Figure 1 on page 3).
TXD0_P, TXD0_N	OUT	PADs_OUT	Differential output pair for lane 0 (Tx data)
TXD1_P, TXD1_N	OUT		Differential output pair for lane 1 (Tx data)
TXD2_P, TXD2_N	OUT		Differential output pair for lane 2 (Tx data)
TXD3_P, TXD3_N	OUT		Differential output pair for lane 3 (Tx data)

A – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

5-02-00367-6/09.14