Libero® SoC v2022.3

SmartFusion®2, IGLOO®2, RTG4™, PolarFire®, and PolarFire SoC FPGA High Speed Serial Interface Simulation User Guide

Introduction

The High Speed Serial Interface (SERDES) in SmartFusion® 2, IGLOO® 2, RTG4™, PolarFire®, and PolarFire SoC supports multiple high-speed serial protocols. The SERDESIF macro includes a PMA block, which is a serializer and de-serializer (SERDESIF) analog block that supports multiple serial protocols on its physical lanes.

The data path from PMA varies with the protocol used:

- For the PCIe protocol, the data path from PMA includes the PCIe PCS, which is completely bypassed for all non-PCIe protocols.
- For the XAUI protocol, the data path includes an XAUI extender.
- For other serial protocols, the data path includes EPCS.

SmartFusion 2, IGLOO 2, RTG4, PolarFire, and PolarFire SoC support the following protocols:

- PCIe 1.0, 1.1, and 2.0, XAUI
- Any user-defined high-serial protocol implemented in SmartFusion 2 fabric accessing SERDESIF lanes through the EPCS interface

This user guide describes the simulation flows supported in the software for the SERDESIF block. The SERDESIF block communicates with the fabric and IO-PADs. The SERDESIF has the following interfaces towards the fabric.

Table 1. SERDESIF Interfaces

Interface	Description
AXI/AHB Master Interface	64/32-bit AXI master interface for data communication with the fabric in PCIe mode.
AXI/AHB Slave Interface	64/32-bit AXI slave interface for data communication with the fabric in PCIe mode.
APB Slave Interface	32-bit APB slave interface for configuration.
External PCS Interface	20-bit EPCS Interfaces available for 1, 2, 3, or 4 lane configurations at custom speed. Each 20-bit EPCS lane interface is independent.

Table of Contents

1. Operating Modes

The High Speed Serial Interface (SERDESIF) block operates in the following modes:

- PCIe
- XAUI
- EPCS

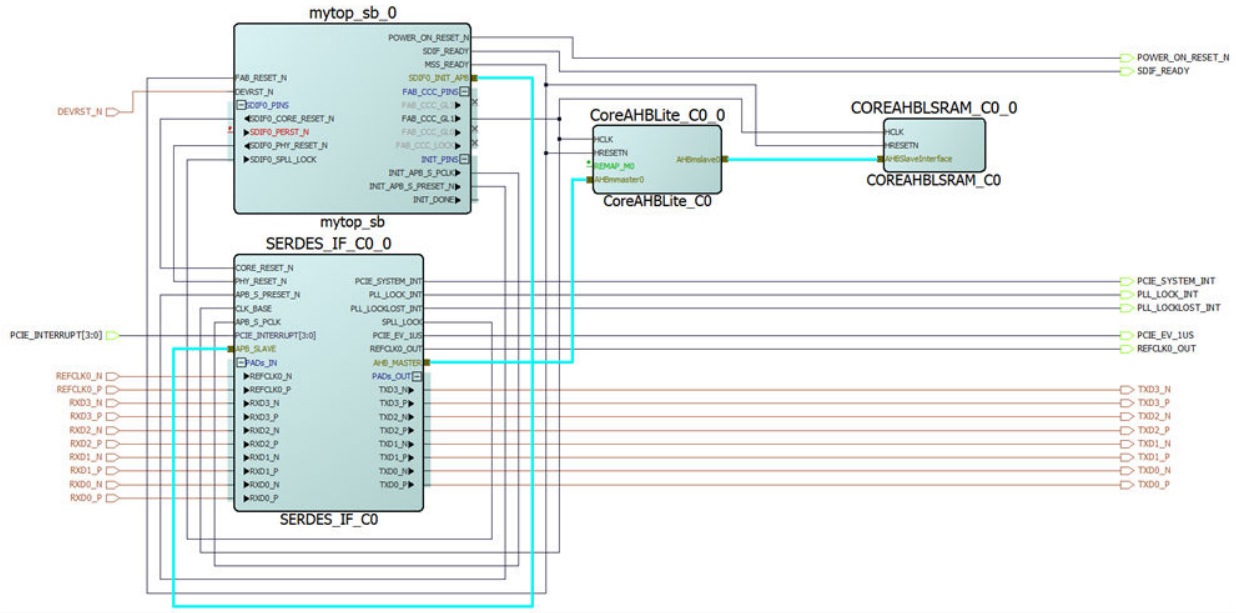
You can use the High Speed Serial Interface Configurator to select the operating mode and the number of lanes to be used for each protocol. For details, refer to the High Speed Serial Interface Configurator User Guide.

In the following figure:

- The SDIF0_INIT_APB interface on the System Builder component connected to the SERDES block (APB Slave) is configuring the SERDES registers.
- The SERDES block configured in PCIe mode (AHB Master) is writing into an AHBL SRAM (AHB Slave) using the AHBLite bus (CoreAHBLite).

Note: The MSS FIC_2_APB_MASTER register configuration interface (APB Master), along with the CoreConfigP and CoreResetP blocks inside the System Builder component, are responsible for configuring and initializing the SERDES blocks used with SmartFusion2.

Figure 1-1. SERDES Block with APB Configuration Path Accessing an AHBLite Fabric Slave



You can also select the simulation mode, depending on the SERDESIF operating mode, as shown in the following table. For details about supported protocols and configurations, refer to the SmartFusion2 and IGLOO2 High Speed Serial Interface Configuration User Guide and RTG4 High Speed Serial Interface (PCIe, EPCS, and XAUI) Configuration User Guide.

Table 1-1. Matching SERDESIF Modes with Available Simulation Modes

SERDESIF Mode	Available Simulation Modes
PCIe	BFM_CFG, BFM_PCIe, RTL
XAUI, EPCS	BFM_CFG, RTL

Table 1-2. Simulation Mode Descriptions

Simulation Mode	Description
BFM_CFG	In this mode, only the APB configuration bus of the SERDESIF is available for simulation. You can write/read the different configuration and status bits of the SERDESIF registers through its APB slave interface. Refer to "BFM_CFG Simulation Mode". **Note:** In this mode, the physical interface of the SERDES IP block is inactive. In addition, the values of the status bits do not change in response to SERDESIF transactions. They are maintained at their reset values while you are allowed to write to the configuration bits. This simulation option is available for all the SERDESIF operating modes.
BFM_PCIe	In this mode, you can access the APB bus similar to BFM_CFG mode. You can also communicate with the SERDESIF block through the master and slave AXI or AHB bus interfaces. This mode allows you to validate your fabric interfaces to the SERDESIF block and is available for the PCIe operating mode only. **Note:** In this mode, the physical interface of the SERDES IP block is inactive.
RTL	This mode allows you to fully simulate the SERDESIF block from the fabric interface to the serial I/O interface. This simulation mode is available for all the SERDESIF operating modes. The simulation runtime for this mode is longer than the runtime for the other SERDESIF simulation modes.

1.1 BFM_CFG Simulation Mode

BFM_CFG simulation mode allows you to interact with the APB slave interface of the SERDESIF block. The SERDESIF has three addressable spaces from the APB bus:

- SERDESIF System Registers
- PCIe Bridge/Core Registers
- SERDES Macro Registers

The SERDESIF is configured using the PCIe Bridge/Core Registers and SERDES Macro registers (see the following figures).

Figure 1-2. Memory Map of PCIe_0 (Single PCIe Mode)

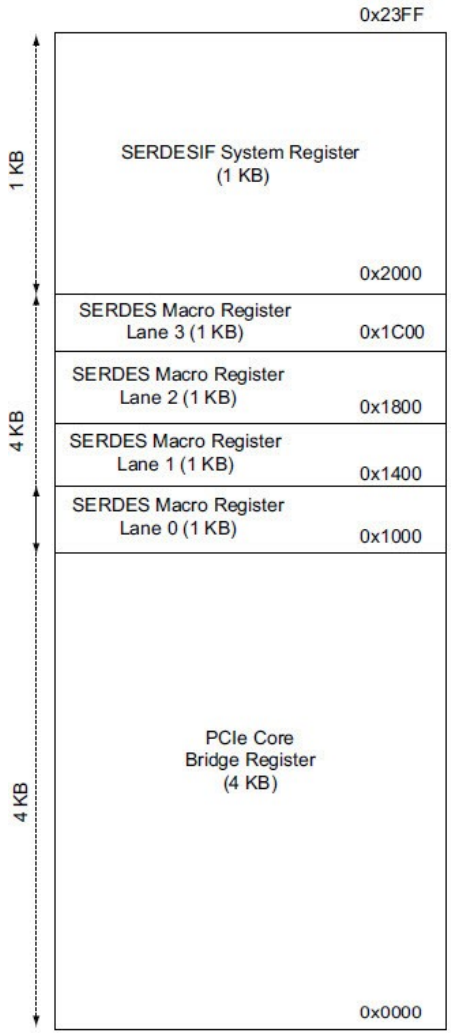
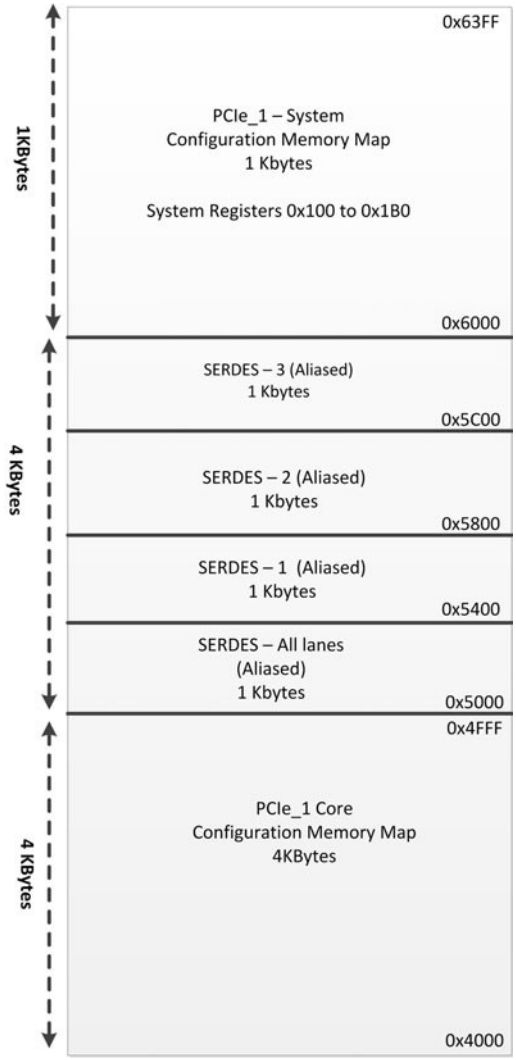


Figure 1-3. Memory Map of PCIe_1 (second PCIe controller in Dual PCIe mode) of SERDESIF2 and SERDESIF3 Core (Available on M2S090T/TS devices only)



In BFM_CFG simulation mode, only one of these address spaces can be accessed. The accessible address space for PCIe_0 is from address offset 0x2000 to 0x23FF of the SERDESIF System Register memory map. However, address space 0x2100 through 0x3FFF is reserved for future use. Do not read and write into this reserved area because it may result in PSLVERR responses. For PCIe_1 (second SERDES controller in Dual PCIe Mode for M2S090T/TS devices), the accessible address space is 0x6000 through 0x63FF. However, address space 0x6100 through 0x7FFF is reserved for future use. Do not read and write into this reserved area because it may result in PSLVERR error.

You can write into all the write-allowed configuration registers of the SERDESIF blocks to configure their operation according to your needs using the APB interface. As a result, you must have an APB bus master to send the commands to the SERDESIF block, such as the MSS configuration APB FIC2 interface or another APB master in the fabric.

From the simulation log, you can read back the configurations written to verify that the writes went through correctly. You can also read back the contents of the status registers that are accessible from the APB interface. However, the value of the status bits does not reflect the true state of the SERDESIF block. Only the reset values of these registers are read back.

In this mode, the application layer of the SERDESIF block is not used. The application layer is only used for configuring/setting up the SERDESIF, SERDES and PCIe APB register. As a result, the fabric datapath interface of

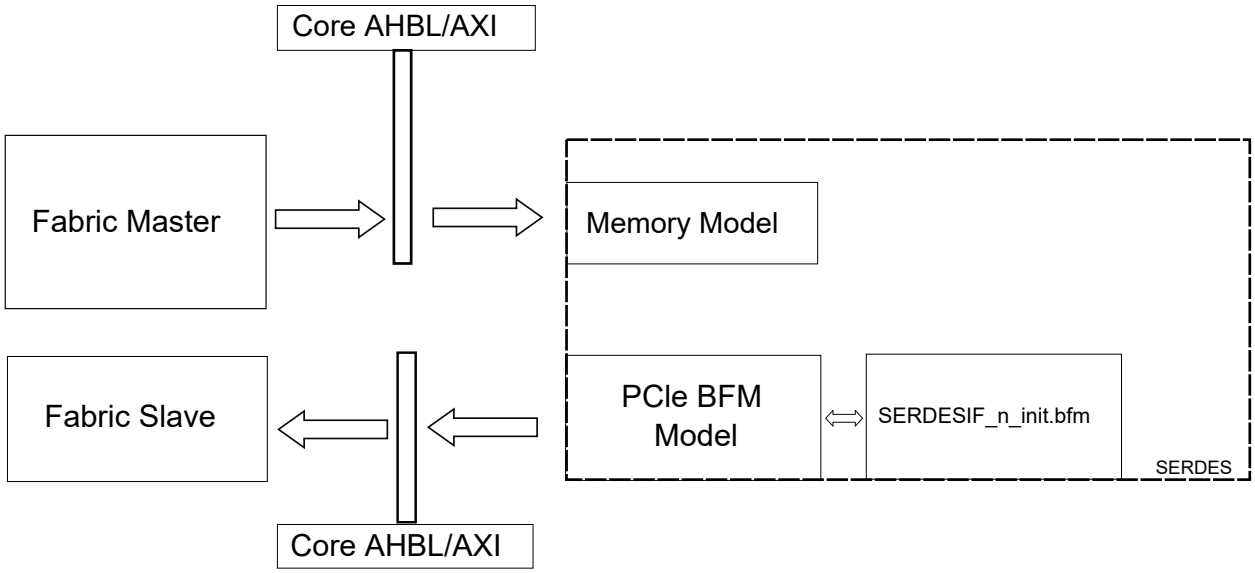
the SERDESIF does not respond to external stimulations. You can use this simulation mode with any operational mode of the SERDESIF block.

1.2 BFM_PCIe Simulation Mode

BFM_PCIe simulation mode is available only for PCIe mode for Smartfusion2, IGLOO2, and RTG4. In BFM_PCIe simulation mode (see the following figure), you can transmit/receive data from/to the fabric using the AXI/AHBLite interface of the SERDESIF. Using an APB Master, you can also read and write to the APB register interface of the SERDESIF to access the configuration registers.

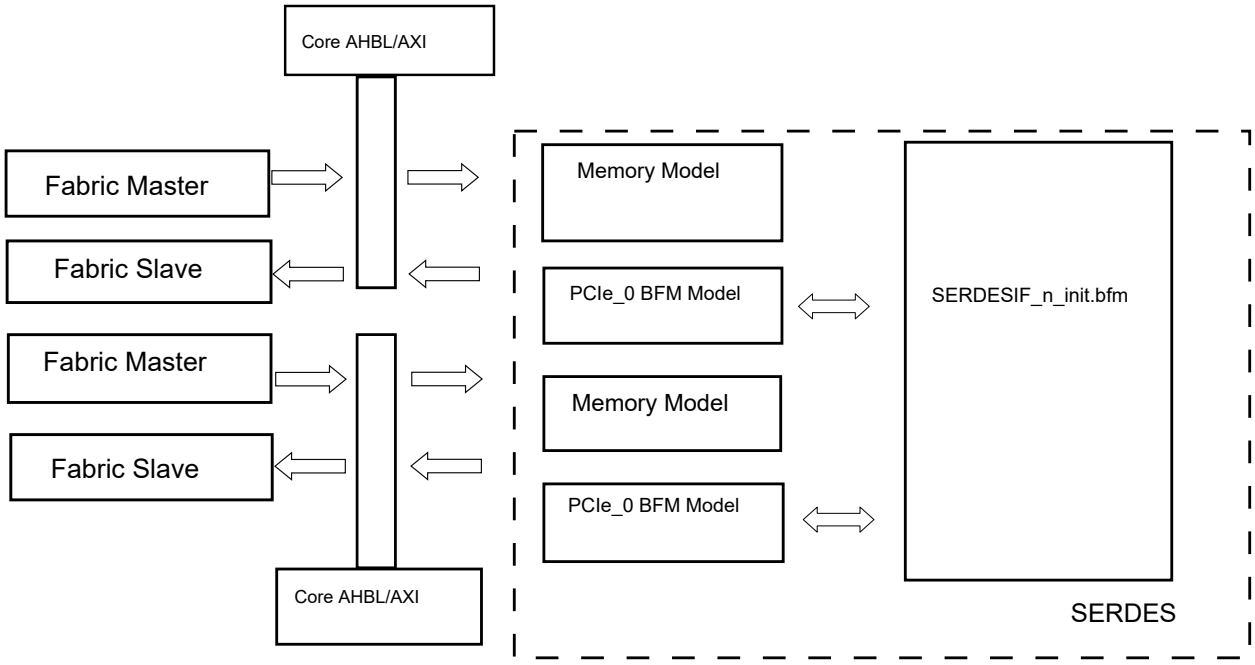
The PCIe BFM Model in the following figure is an AHBLite or AXI Master that accesses the AMBA High-Performance (AHB)-Lite Slaves over the AHBLite bus. The Memory Model is an AHBLite Slave that provides a simple read and write memory. The SERDESIF_n.init.bfm contains BFM write commands to initialize the SERDES prior to simulation.

Figure 1-4. BFM_PCIe Simulation Mode Diagram



The following figure shows the simulation mode diagram for SmartFusion2 M2S090T/TS devices that support PCIe for Protocol 1 and Protocol 2.

Figure 1-5. BFM_PCIe Simulation Mode Diagram for Dual PCIe Mode (M2S/M2GL090T/TS)



The BFM_PCIe simulation mode uses BFM commands to emulate data being transferred through the SERDESIF block across the AXI/AHB bus interface to the fabric. The physical layer of the PCIe protocol is not implemented in this simulation mode and you cannot witness data transfer on the serial interface of the SERDESIF block.

The BFM_PCIe simulation mode provides you with an AXI/AHB bus master and an AXI/AHB bus slave based on the SERDESIF PCIe bus configuration.

1.3 SERDES AXI/AHB Bus Slave Interface

The AXI and AHB bus slave interfaces available in the BFM_PCIe simulation mode provide a 64-bit or 32-bit slave interface for fabric communication. The slave acts as a memory that you can interact with by initiating write and read bus transactions using the appropriate bus master. Because the slave acts as a memory model, whatever is written into the slave can be read back from the same address.

Figure 1-6. AXI Slave Simulation Mode

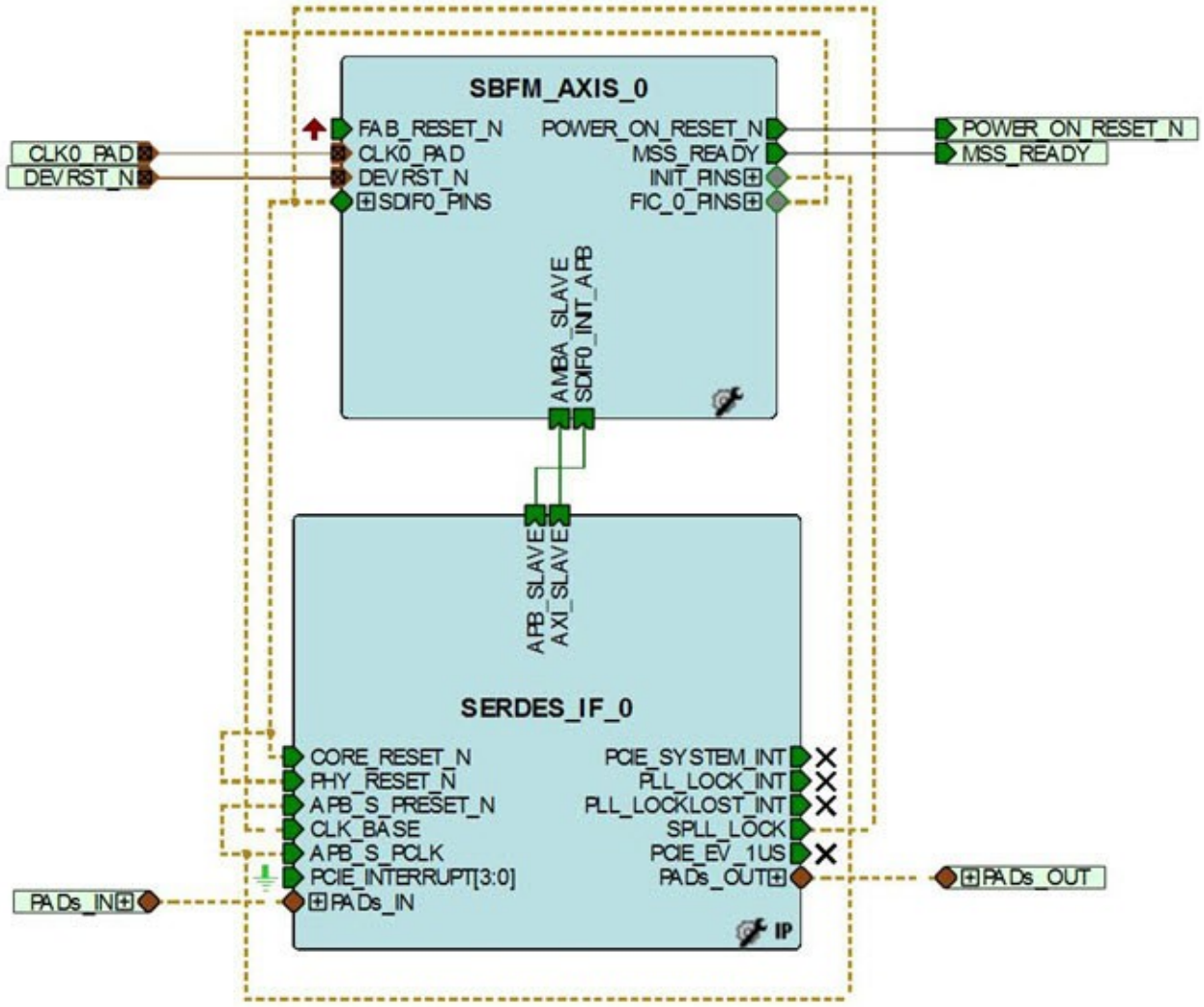
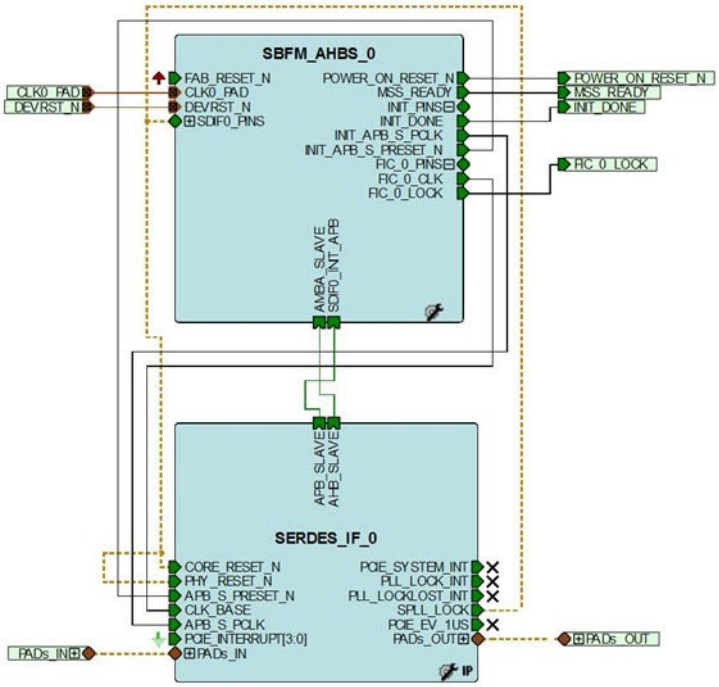


Figure 1-7. AHB Slave Simulation Mode



1.4 SERDES AXI/AHB Bus Master Interface

The AXI and AHB bus master interfaces in the BFM_PCIe simulation mode allow you to emulate a 64-bit AXI or a 32-bit AHB master, depending on the bus configuration. In the bus master cases, you must write BFM bus commands to instruct the model to start bus transactions to the fabric similar to the MSS. For more information, refer to the SmartFusion2 FPGA Microcontroller Subsystem BFM Simulation Guide.

Figure 1-8. AXI Master Simulation Mode

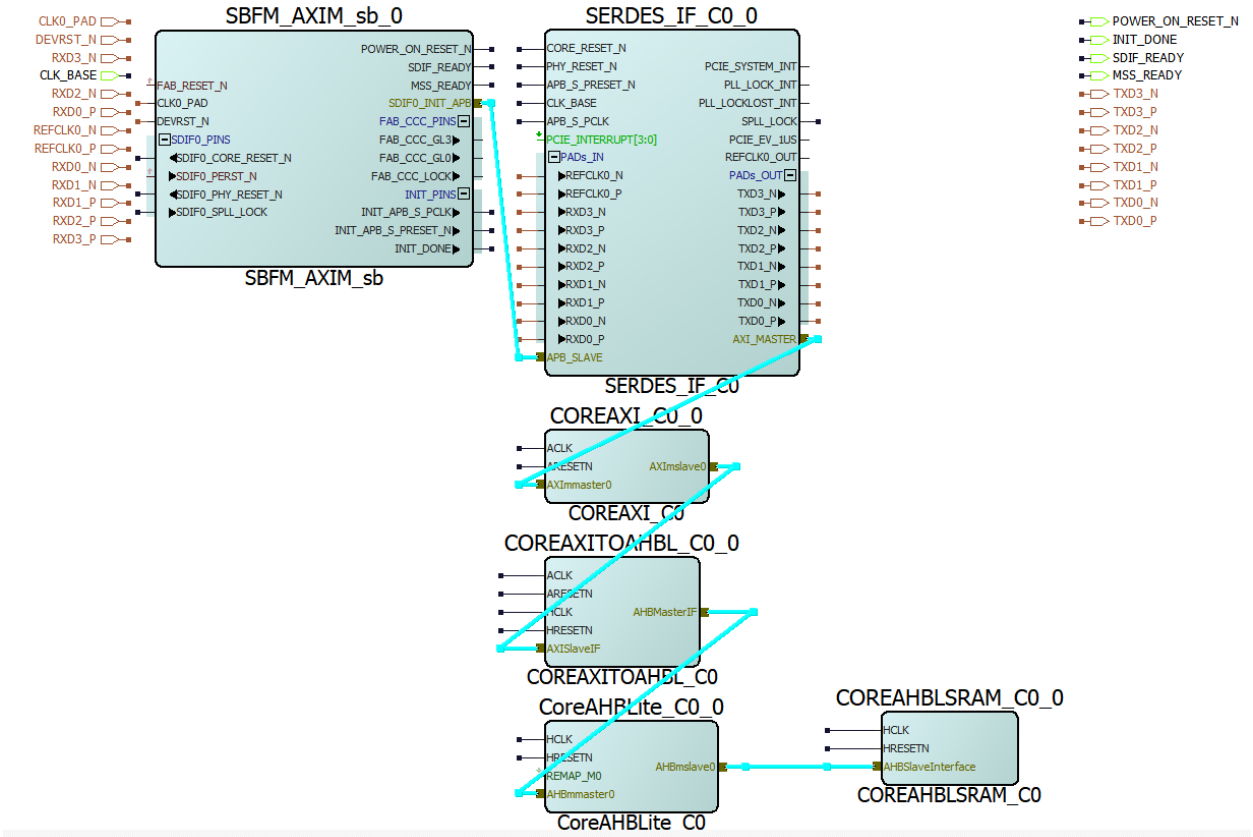
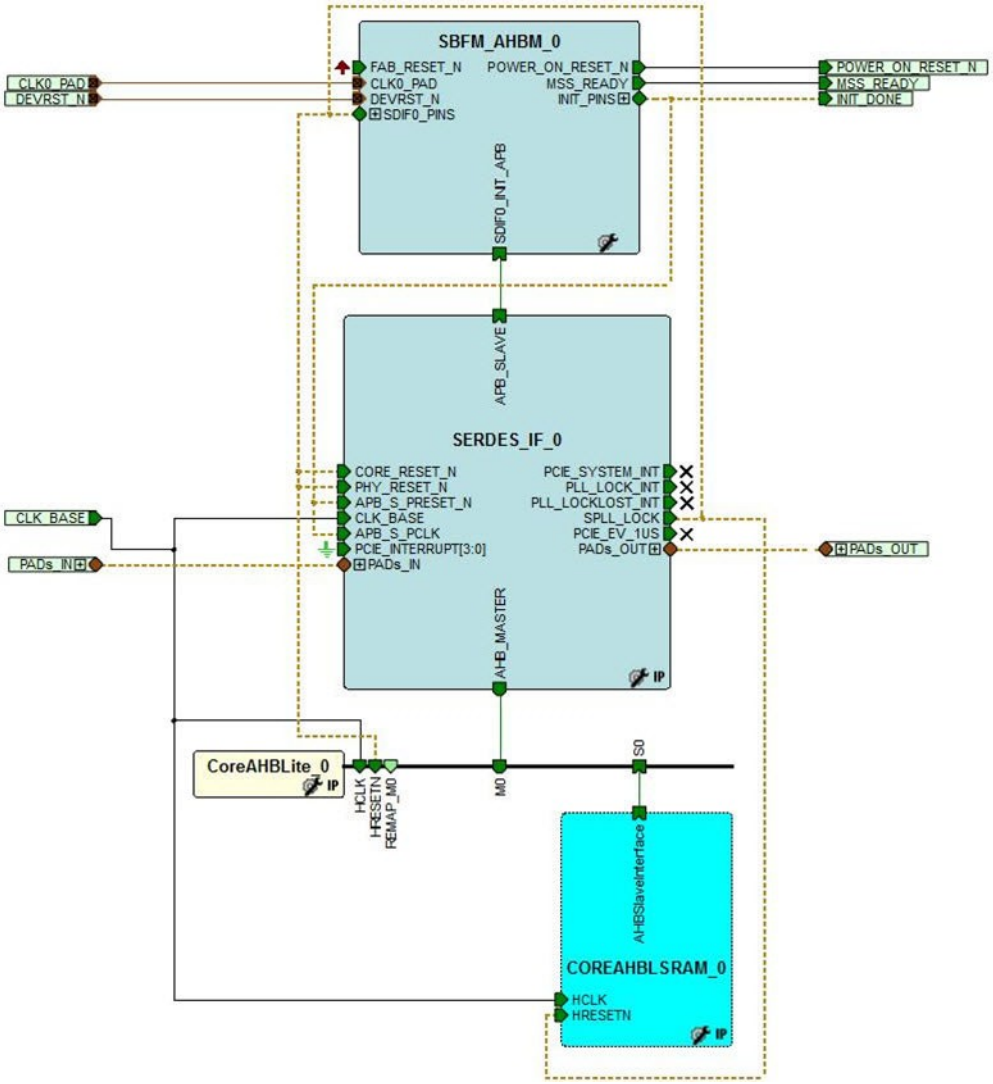


Figure 1-9. AHB Master Simulation Mode

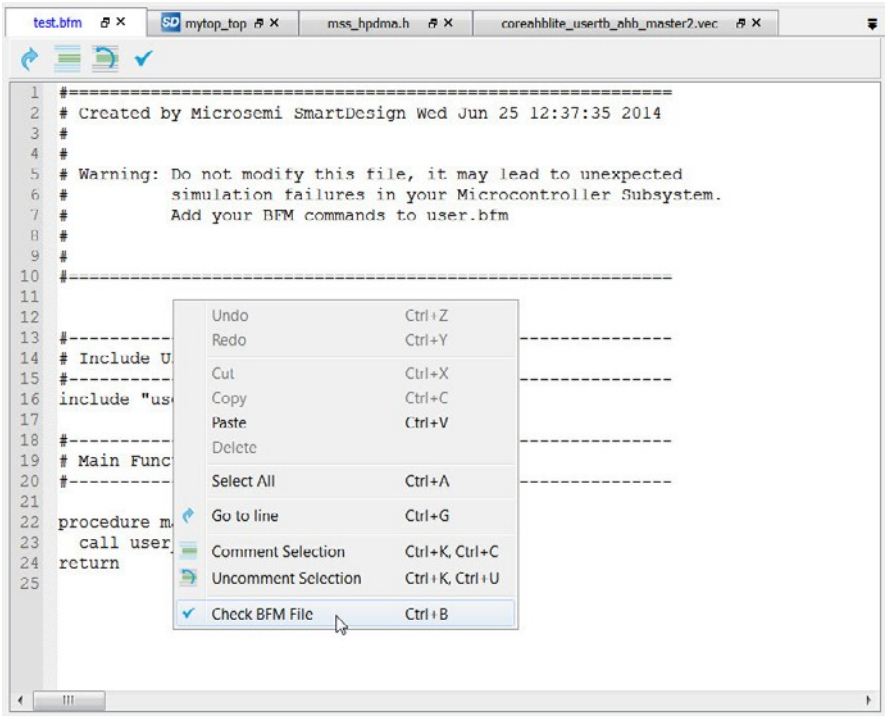


You must include your BFM instructions in the `<project>/simulation/SERDESIF_<n>_user.bfm` file, where n refers to the SERDESIF instance you are trying to simulate (e.g., SERDES_IF_0/1/2/3). The built-in SERDES BFM model interprets these instructions and initiates AHB/AXI transactions in sequence.

You can check your BFM commands before invoking the simulator. Libero® performs a syntax check on your BFM commands and displays the results in the log tab. To check your BFM commands:

1. Open the BFM file `SERDESIF_<n>_user.bfm` in the Libero Text Editor.
2. Right-click and select**Check BFM File** (see the following figure) or click the Check File icon.

Figure 1-10. Check BFM File



3. Select the **Log** tab to view the result.

The BFM commands used in the SERDES BFM files are similar to the BFM commands used by the MSS bus masters. Refer to CoreAMBA BFM User's Guide for a complete list of available BFM commands. There are additional BFM commands you can use only for PCIe AXI BFM simulation to emulate 64-bit AXI transactions.

- `write64 w <base_address> <base_address_offset> <32-bit MSB> <32-bit LSB>`
  This command makes the SERDES bus master start a 64-bit transaction on the external bus for a slave whose address is given by the <base_address> and <base_address_offset> using the data given by <32-bit MSB> and <32-bit LSB>. For example:

  ```
  write64 w 0x00000000 0x0 0xA0A1A2A3 0xB0B1B2B3;
  ```

- `readcheck64 w <base_address> <base_address_offset> <32-bit MSB> <32-bit LSB>`
  This command makes the SERDES bus master start a 64-bit read transaction for the address given by <base_address> and <base_address_offset>. It compares the 64-bit read data to the data given by <32-bit MSB> and <32-bit LSB>. If the data matches, the command will succeed; otherwise, it will error-out. For example:

  ```
  readcheck64 w 0x00000000 0x0 0xA0A1A2A3 0xB0B1B2B3;
  ```

- `read64 w <base_address> <base_address_offset>`
  This command makes the SERDES bus master start a 64-bit read transaction for the address given by <base_address> and <base_address_offset>. For example:

  ```
  read64 w 0x00000000 0x0 0xA0A1A2A3 0xB0B1B2B3;
  ```

- `writemult64 w <base_address> <base_address_offset> <32-bit MSB> <32-bit LSB>…`
  This command makes the SERDES bus master start a 64-bit transaction in burst mode on the external bus for a slave with address given by the <base_address> and <base_address_offset> using the data given by <32-bit MSB> and <32-bit LSB>. This command performs as many writes as <32-bit MSB>

and <32-bit LSB> pairs as specified by you. Write operations are done to consecutive address locations starting from the base offset specified. For example:

```
writemult64 w 0x00000000 0x0 0xA1 0xB1 0xA2 0xB2 0xA3 0xB3 0xA4 0xB4;
```

- `readmult64 w <base_address> <base_address_offset> <n>`
  This command makes the SERDES bus master start a 64-bit burst read transaction for consecutive address locations starting from <base_address> and <base_address_offset> for <n> number of times. For example:

  ```
  readmult64 w 0x00000000 0x0 5;
  ```

- `readmultchk64 w <base_address> <base_address_offset> <32-bit MSB> <32-bit LSB>`
  This command makes the SERDES bus master start a 64-bit burst read transaction for the address given by <base_address> and <base_address_offset>. It compares the 64-bit read data to the data given by <32-bit MSB> and <32-bit LSB>. The number of reads and corresponding compares depend on the number of <32-bit MSB> and <32-bit LSB> pairs specified in this command. If the data matches, the command will succeed; otherwise, it errors out. For example:

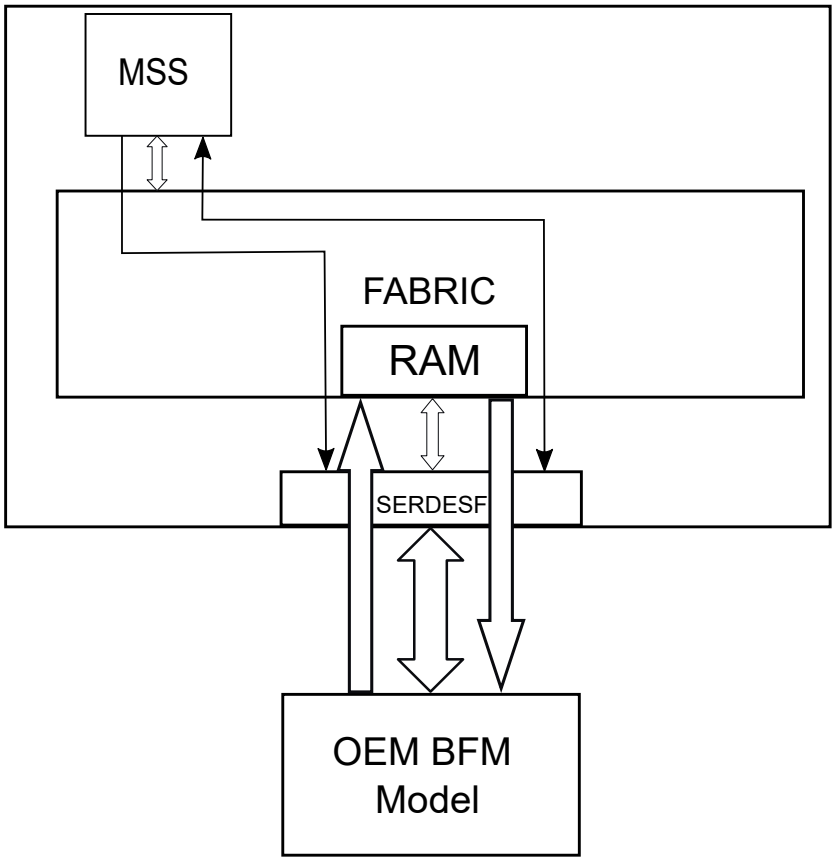
  ```
  readmultchk64 w 0x00000000 0x0 0xA1 0xB1 0xA2 0xB2 0xA3 0xB3 0xA4 0xB4;
  ```

1.5 RTL Simulation Mode

RTL simulation mode is available for all the SERDESIF modes. This simulation mode allows you to simulate PCIe, XAUI, and EPCS. The RTL simulation mode supports all of the protocol communication layers, including the physical layer, and provides accurate cycle simulation for your design. However, using RTL simulation incurs some run-time penalties.

You are responsible for having your own model for an off-chip IP that can communicate with the SERDESIF block in the same protocol used by the SERDESIF block when using this mode. For example, if you are using a PCIe configured SERDESIF block, you must have your own PCIe IP off-chip block that can communicate with the SERDESIF block using the PCIe protocol, as shown in the following figure. The same is true for all the other SERDESIF block modes of operation. Because the IP user block is off-chip, it must connect to your design in the top-level test bench.

Figure 1-11. RTL Simulation Model



IGLOO 2 and SmartFusion 2 simulation has been enhanced to support Tri-state 'z' behavior on the PCIe Tx lines. Choose PCIe block to drive between the following states:

- Common voltage signal typically '0' (by default)
- Tri state ('z')

The Tri-state on Tx lines is achieved by passing a value 1'b1 to parameter `PCIE_TX_ELECIDLE_Z` through v_{sim} command as shown.

Modelsim:

```
-gtestbench/top_0/SERDES_IF2_C0_0/SERDES_IF2_C0_0/SERDESIF_INST/PCIE_TX_ELECIDLE_Z=1'b1
```

Where, the hierarchy for the parameter is:

- `testbench` is the name of the design testbench.
- `top_0` is the Libero SmartDesign instance in the testbench.
- `SERDES_IF2_C0_0/SERDES_IF2_C0_0` (2 level of hierarchy) is the instance name in configurator generated netlist.
- `SERDESIF_INST` is the instance name of this macro, where `PCIE_TX_ELECIDLE_Z` is declared with default value of 1'b0'.

The O50T device uses only `SYSTEM_SERDESIF_SOFT_RESET` register (address offset 0x08) for EPCS and SERDES lane register reset, whereas all other devices use `SYSTEM_SERDESIF_SOFT_RESET` register (address offset 0x08) for EPCS reset and `SYSTEM_EPCS_RSTN_SEL` (address offset 0x6C) for SERDES lane register reset. You must provide the simulation parameter to achieve the functionality of O50T device as shown in the following example.

Modelsim: -g SIM050T=1

1.6 BFM Files for SERDES Simulation

Libero generates a list of BFM files in the simulation folder for SERDES simulation (see the following table). You should not edit some BFM files because it can lead to unexpected simulation failures.

Table 1-3. Generated BFM Files

BFM File Name	Function	Remarks
Test.bfm	Top-level BFM. Contains the main function.	Libero-generated. Do not edit.
User.bfm	Contains user BFM commands. Calls subsystem_init function.	Edit this file to add user BFM commands.
Subsystem.bfm	Contains subsystem memory map. Defines the name and base address of each subsystem resource. Calls the init function to initialize subsystem.	Do not edit.
Peripherals_init.bfm	Contains the Memory Map of all peripherals including SERDES. Calls the SERDES_<0/ 1/2/3>_init.bfm to initialize SERDES.	Do not edit.
SERDES_<0/1/2/3>_init.bfm	Writes to SERDES registers to initialize SERDES	Do not edit.
SERDESIF_<0/1/2/3>_PCIE_<0/1>_user.bfm (Dual PCIe Mode Operation) SERDESIF_<0/1/2/3>_user.bfm (Single PCIe Mode Operation)	Contains user BFM commands.	Edit this file to add your BFM commands for SERDES simulation.

1.7 PolarFire Transceiver Simulation

The PolarFire FPGA and PolarFire SoC FPGA families include multiple embedded low-power, performance-optimized transceivers. Each transceiver has both the Physical Medium Attachment (PMA), protocol physical coding sub-layer (PCS) logic, and interfaces to the FPGA fabric. The transceiver has a multi-lane architecture with each lane natively supporting serial data transmission rates from 500 Mb/s (250 Mb/s with interpolation) to 12.7 Gb/s.
The transceiver is organized into four distinct transmit (Tx) and receive (Rx) blocks:

- PMA
- PCS interface block, including a dedicated PCIe PCS
- Transmit PLL (Tx PLL)
- Reference clock inputs

The high-speed PMA blocks connect to the FPGA fabric through the PCS block. The PMA generates the required clocks and converts the transmit data from parallel to serial, and receive data from serial to parallel. Each PMA block includes a connection to a PCS block and an associated interface to the FPGA fabric making up a transceiver lane. The PCS interface block provides several industry-standard interfaces for use in protocol-specific designs.

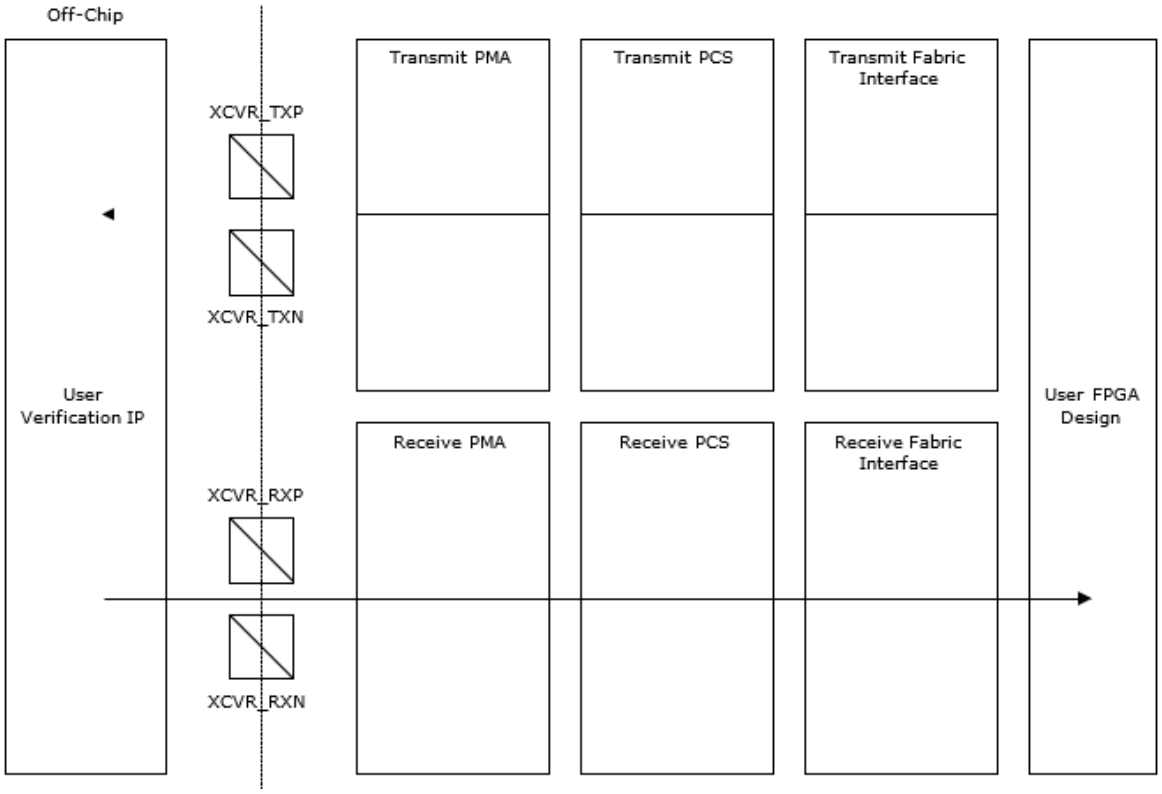
A group of four transceiver lanes is called a quad. Each quad has a local transmit PLL used exclusively within the four transceiver lanes. Additional transmit PLLs are shared between quads.

In addition to the 8b10b, 64b6xb, PIPE, and PMA blocks, two PCIe PCS logic blocks are included in each device. These blocks include hard embedded logic that provides full-featured PCIe endpoint/root port subsystem. These PCIe sub-systems (PCIESS) have hard connections to multiple transceiver lanes, providing flexibility for ×1, ×2, and

×4 width links. For more information about PCIe, see the PolarFire FPGA and PolarFire SoC FPGA PCI Express User Guide.

TL simulation mode is available for all of the transceiver modes. This simulation mode enables the simulation of all the protocol communication layers (including the PMA, PCS, and fabric interfaces) and provides accurate cycle simulation for the design. However, using RTL simulation incurs some run-time penalties. Microchip provides a specific PCIe BFM model for enhanced simulation of PCIe designs using the embedded PCIe controllers, see the PolarFire FPGA and PolarFire SoC FPGA PCI Express User Guide.

Figure 1-12. RTL Simulation Block Diagram



RTL simulation mode simulates the XCVR block from the fabric interface to the serial I/O interface. RTL simulation mode is available for all of the XCVR modes. This mode supports all of the protocol communication layers, including the physical layer, and provides accurate cycle simulation for the design. Using RTL simulation, however, experiences some run-time penalties. As the IP user block is off-chip, it must be connected to the user design in the top-level test bench. It is your responsibility to provide the model for the off-chip IP that can communicate with the XCVR block in the same protocol used by the XCVR block when using this mode. Post-synthesis simulation is available with PF_XCVR_ERM designs when configured with both ERM OFF and ON.

To minimize simulation time, certain peripherals in the transceiver do not have full behavioral models. These models are replaced with memory models that output a message indicating when the memory locations inside the peripheral are accessed. The memory models are created by using register information that is generated by Libero. The XCVR register data is found at `<LiberoProject>\component\work\<top_level>\<xcvr_component_name>`.

The peripheral signals do not toggle based on any writes to registers, or react to any signal inputs on the protocol pins.

Using RTL simulation mode, the FPGA designer can have an off-chip verification IP model that communicates with the transceiver. For example, if the design uses a 8b10b XCVR block, the FPGA designer must have a 8b10b verification IP off-chip block to communicate with the XCVR block using the required protocol. When the IP user block is off-chip, it must be connected to the design in the top-level test bench.

1.8 PolarFire PCI Express Simulation

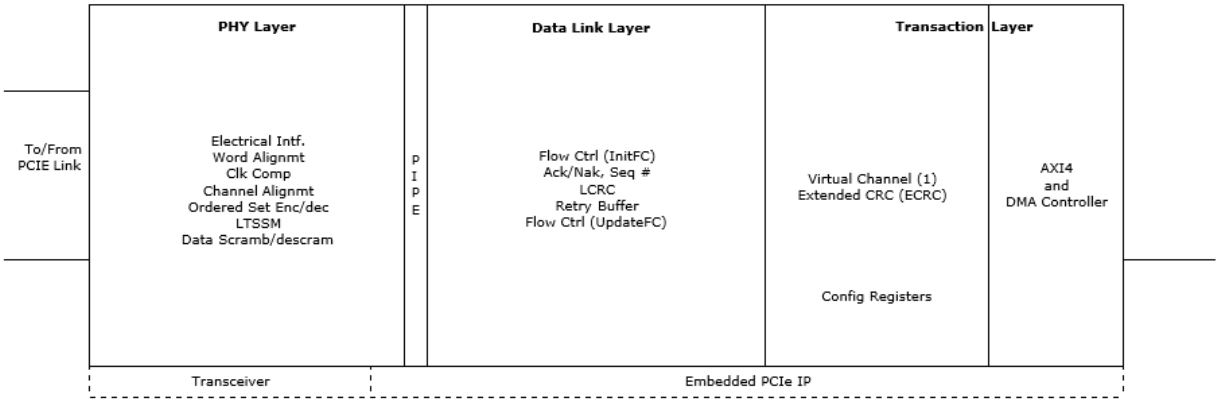
PCI Express (PCIe) is a scalable, high-bandwidth serial interconnect technology that maintains compatibility with existing PCI systems. Microchip's PolarFire SoC FPGAs and PolarFire FPGAs contain fully integrated PCIe endpoint and root port subsystems with optimized embedded controller blocks that use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

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

PCIe supports the following two simulation modes:

- Bus Functional Model (BFM)
- Full Register-Transfer Level (RTL) model

Figure 1-13. PCIe Functional Layers of PCIESS



1.8.1 Bus Functional Model (BFM)

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

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

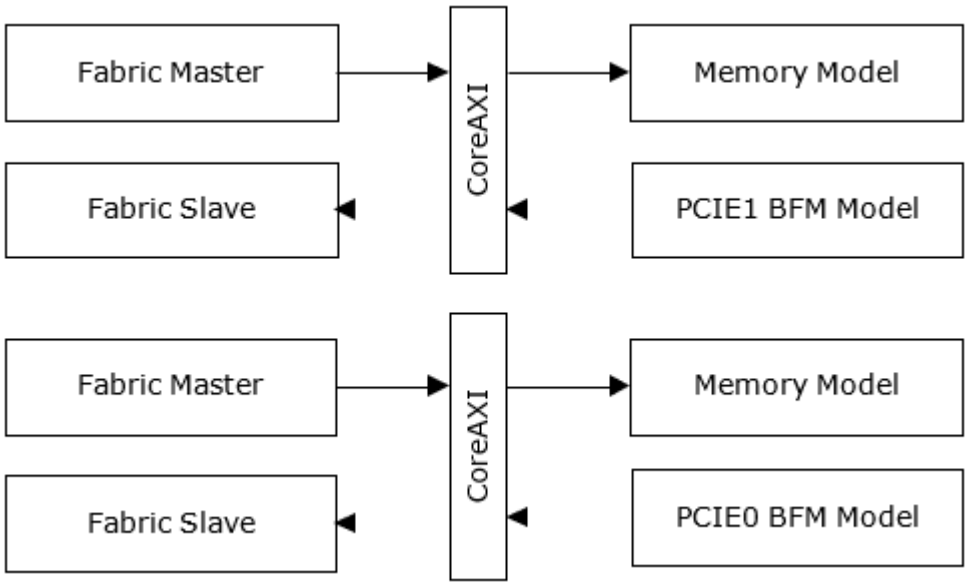
The AXI4 bus master in the PCIe BFM simulation mode enables emulating 64-bit AXI master transactions. Libero SoC generates user-customizable BFM files that instruct the model to start transactions to the fabric. The BFM allows you to use a text file to issue the transactions from the PCIe AXI master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

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

The BFM commands used in the PCIESS BFM files are similar to the BFM commands used by the bus masters.
Note: Simulation of APB interface is not supported in the BFM mode

The following figure shows the PCIESS BFM structure.

Figure 1-14. PCIESSBFM Structure



1.8.2 Full Register-Transfer Level (RTL) Model

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

When using VIP models, observe the following guidelines:

- Verification IP must be configured properly.
  - BFM type — Type of BFM (0 = root port and 1 = end point).
  - Number of lanes — Number of connected lanes.
  - I/O Size — Size of internal I/O space (12 to 24).
  - MEM32_SIZE — Size of internal 32-bit addressing memory space (12 to 24).
  - MEM64_SIZE — Size of internal 64-bit addressing memory space (12 to 24).
  - PCLK — PIPE clock frequency depends on the signaling rate and PIPE interface width configuration.
- The receiver pin for XCVR must not be in an unused state. The following example code snippet is used in the test bench to prevent the transmitter pin from going into an unused state.

```
rxp[i]<=(tx_1b[i]===1'bX || tx_1b[i]===1'bZ) ? 1'b0 : tx_1b[i];
rxn[i]<=(tx_1b[i]===1'bX || tx_1b[i]===1'bZ) ? 1'b0 : ~tx_1b[i];
```

The receiver pin for VIP model must not be in an unused state. The following example code snippet is used in the test bench to prevent the transmitter pin from entering an unused state.

```
rx_1b[i]<=(txp[i]==txn[i] || txp[i]===1'bX) ? 1'bZ : txp[i];
```

where:

- `i` — Number of BFM lanes.
- `txp` and `txn` — Transmitter pins from XCVR.
- `rxp` and `rxn` — Receiver pins from XCVR.
- `tx_1b` — Transmitter pin from VIP model.
- `rx_1b` — Receiver pin from VIP model.

2. PCIe BAR Address Translation Support

In a PCIE controller, the master PCIE controller will write into a certain BAR address and the PCIE slave will convert that address into the fabric address using the address translation registers.

The following commands convert a PCIE address to a fabric address.

2.1 Increased Max Burst Length

For burst mode, the maximum supported burst length has been increased to 32. As a result, instructions for `writemult64`, `readmult64`, and `readmultchk64` can have data 32 times to accommodate the AXI4 protocol.

2.2 writepcie64

Command syntax:

```
writepcie64 x <window_index> <bar_index> <address_increment> <dataH_check> <dataL_check>
```

This command makes the SERDES bus master start a 64-bit transaction on the external bus for a slave whose address is given by `<window_index> <bar_index> <address_increment>` using the data given by the <32-bit MSB> and <32-bit LSB>.

Example:

```
writepcie64 x 0x00 0x0 0x1 0x8;
```

2.3 readpcie64

Command syntax:

```
readpcie64 x <window_index><bar_index> <address_increment>
```

This command makes the SERDES bus master start a 64-bit read transaction for the address given by <window_index><bar_index> and <address_increment>.

Example:

```
readpcie64 x 0x01 0x0;
```

2.4 readpciecheck64

Command syntax:

```
readpciecheck64 x <window_index><bar_index> <address_increment> <dataH_check> <dataL_check>
```

This command makes the SERDES bus master start a 64-bit read transaction for the address given by <window_index><bar_index> and <address_increment>. It compares the 64-bit read data to the data given by <32-bit MSB> and <32-bit LSB>. If the data matches, the command succeeds; otherwise, it errors-out.

Example:

```
readpciecheck64 x 0x02 0x0 0x1 0x8;
```

2.5 writepciemult64

Command syntax:

```
writepciemult64 x <window_index><bar_index> <address_increment> <dataH_check> <dataL_check> …
<dataH_check> <dataL_check>
```

This command makes the SERDES bus master start a 64-bit transaction in burst mode on the external bus for a slave with address given by the <window_index><bar_index> and <address_increment> using the data given by <32-bit MSB> and <32-bit LSB>. This command performs as many writes as <32-bit MSB> and <32-bit LSB> pairs are specified by you. Write operations are done to consecutive address locations starting from the address increment specified.

Example:

```
writepciemult64 x 0x03 0x0 0xA1 0xA2 0xB1 0xB2 0xC1 0xC2 0xD1 0xD2;
```

2.6 readpciemult64

Command syntax:

```
readpciemult64 x <window_index><bar_index> <address_increment> <number_of_bursts>
```

This command makes the SERDES bus master start a 64-bit burst read transaction for consecutive address locations starting from <window_index><bar_index> and <address_increment> for <n> number of times.

Example:

```
readpciemult64 x 0x04 0x0 4;
```

2.7 readpciecheckmult64

Command syntax:

```
readpciemultchk64 x <window_index><bar_index> <address_increment> <dataH_check> <dataL_check>
… <dataH_check> <dataL_check>
```

This command makes the SERDES bus master start a 64-bit burst read transaction for the address given by <window_index><bar_index> and <address_increment>. It compares the 64-bit read data to the data given by <32-bit MSB> and <32-bit LSB>. Number of reads and corresponding compares depend on the number of <32-bit MSB> and <32-bit LSB> pairs specified in this command. If the data matches, the command succeeds; otherwise, it errors-out.

Example:

```
readpciemultchk64 x 0x05 0x0 0xA1 0xA2 0xB1 0xB2 0xC1 0xC2 0xD1 0xD2 0xE1 0xE2;
```

3. New AXI BFM Commands

3.1 readstore64

Command syntax:

```
readstore64  x  <address> <address_increment> <integer variable for MSB> <integer variable
for LSB>
```

This command performs a read cycle and stores the 64-bit data in the specified variable.

Example:

```
readstore64 x 0x10000000 0x0 dataH dataL;
```

3.2 readmask64

Command syntax:

```
readmask64 x <address> <address_increment> <dataH_check> <dataL_check> <maskH> <maskL>
```

This command performs a read cycle and checks the read data. The data is masked as follows:

read_data & mask = data & mask.

Example:

```
readmask64 x 0x10000000 0x0 0xa 0xb 0x0 0xf
```

4. DMA Support

The PCIe controller has an internal DMA controller that achieves fast data transfer rates between PCIe and AXI domains. To support fast data transfer rates, a small set of configuration registers provide support for initiating DMA transfers by interpreting the DRI transactions intended for the DMA interface. In addition, commands have been added for configuring the DMA registers.

To set the DMA command:

1. Store the DMA data required for DMA transfer in the file name `DMADATA.vec`. The file available in the path is `../simulation/DMADATA.vec`.
2. Set BFM timeout value to 4096, which supports a DMA transfer of 4KB. For example: timeout 4096.
3. Set the DMA data using the following command:
   setup `0xa <DMA_data>`.
   Example:

```
setup 0xa 0x1
set command[2] to 0: When the user wants the DMA_data to increment by 1 starting with
0x1.
set command[2] to 1: When the user wants the DMA_data to be random data.
set command[2] to 2: When the user wants the DMA_data to be taken from vector file.
```

4. Set the DMA source address and destination address using the following command:
   `setup 0x8 <source_address> <destination_address>`.
   Example:

```
setup 0x8 0x0 0x1000000
```

5. Set the DMA length in bytes and the DMA control register using the following command:
   setup 0x9 <dma_length> <control_register>
   Example:

```
setup 0x9 0x100 0x3
DMA control register details,
     DMA control register bit[0]: set this bit value to 1 for DMA go
     DMA control register bit[1]: set this bit value to 1 for DMA transfer from PCIe
domain to fabric domain.
     DMA control register bit[2]: set this bit value to 1 for DMA transfer from Fabric
domain to PCIe domain.
```

6. Refer to the following example of DMA setup commands:

```
procedure main;
# providing a timeout value to overwrite the default value of 512
timeout 4096;
# setup command when the user wants the DMA_data to be random data.
setup 0xa 0x1;
# setup command to initiate a DMA transfer from PCIe domain to fabric
 setup 0x8 0x0 0x10000000;
 setup 0x9 0x101 0x3;
# setup command to initiate a DMA transfer from fabric to PCIe domain
 setup 0x8 0x10000000 0x0;
 setup 0x9 0x101 0x5;
return
```

5. Revision History

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

Revision	Date	Description
D	12/2022	The following change is made in this revision: • Updated the 1.5. RTL Simulation Mode section.
C	12/2021	Added the following new sections: • 1.7. PolarFire Transceiver Simulation • Section 1.8. PolarFire PCI Express Simulation • Section 1.8.1. Bus Functional Model (BFM) • Section 1.8.2. Full Register-Transfer Level (RTL) Model
B	04/2021	Released with Libero SoC Design Suite v2021.1 without changes from v12.6.
A	11/2020	Document converted to Microchip template. Initial Revision.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

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, **650.318.8044**

Microchip Information

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet- Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, Clockstudio, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, GridTime, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, IntelliMOS, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, KoD, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-

ISBN: 978-1-6683-1640-5

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820