# QuickPCIe Expert for Xilinx

## Getting Started

Version 1.5.6 February 2016

# QuickPCIe Expert

## Getting Started

## Document Change History

| Date | Version Number | Changes |
|---|---|---|
| February 2016 | 1.5.6 | • Added support for Virtex UltraScale. |
| March 2015 | 1.5.5 | • Added support for Kintex UltraScale. |
| December 2014 | 1.5.4 | • Updated Vivado tool requirements.<br>• Added support for Gen1, and x1 and x2 lanes. |
| November 2014 | 1.5.3 | • Updated EDA tool requirements.<br>• Removed $mode argument from simulation.<br>• Updated $target argument values. |
| April 2014 | 1.5.2 | • Updated EDA tool requirements.<br>• Updated simulation file structure and targeted boards. |
| November 2013 | 1.5.1 | • Updated EDA tool requirements.<br>• Updated package names. |
| August 2013 | 1.5.0 | • Added support for x4 link width. |
| June 2013 | 1.4.6 | • Added support for 8.0 Gbps link speed (Gen3). |
| May 2013 | 1.4.5 | • Updated EDA tool requirements.<br>• Updated package description.<br>• Changed product name. |
| December 2012 | 1.4.3 | • Updated EDA tool requirements.<br>• Updated package description. |
| August 2012 | 1.4.1 | • Updated Reference Design registers.<br>• Updated Wizard information. |
| June 2012 | 1.4.0 | • First release |

## Proprietary Notice

# Table of Contents

# List of Figures

# Preface

## About this Document

### Intended Audience

This document has been written for design managers, system engineers, and designers who are evaluating or using the PLDA QuickPCIe Expert.

### Typographical Conventions

| | |
|---|---|
| *italic* | Highlights important notes or publications |
| **bold** | **Highlights interface elements.** |
| COURIER NEW | DENOTES TEXT USED IN A CODE EXAMPLE OR A SIGNAL. |

## Additional Reading

This section lists additional resources from PLDA and third-parties.

PLDA periodically updates its documentation. Please contact PLDA Technical Support or check the Web site at http://www.plda.com for current versions.

### PLDA Publications

Please refer to the following documents for further information:

- *QuickPCIe Expert Reference Manual for Xilinx*: The *Reference Manual* provides the complete functional description of the PLDA QuickPCIe Expert Core.
- *Bus Functional Model Reference Manual*: The *BFM 3 Reference Manual* provides the complete functional description of the PLDA PCI Express Testbench for PCI Express 3.0.

### Other Publications

Please refer to the following documents for information on specification standards:

- *PCI Express™ Base Specification Revision 3.0*
- *PHY Interface for the PCI Express, SATA and USB 3.0 Architectures, Version 4.0*

# Feedback and Contact Information

## Feedback about this Document

PLDA welcomes comments and suggestions about this documentation. Please contact PLDA Technical Support and provide the following information:

- the title of the document
- the page number to which your comments refer
- a description of your comments

## Contact information

**Corporate Headquarters**

PLDA
805, avenue J.R.G.G. de la Lauzière
13290 Aix-en-Provence
France

Tel: USA +1 408 273 4528 - International +33 442 393 600
Fax: +33 442 394 902

**Sales**

For sales questions, please contact sales@plda.com.

**Technical Support**

For technical support questions, please contact PLDA Support at https://www.plda.com/support using the Support Center if you have a PLDA online account.

If you don’t have a PLDA account, contact https://www.plda.com/user/register.

# Chapter 1 Before you Start...

This Getting Started guide is intended to enable you to integrate the PLDA QuickPCIe Expert into your design flow as quickly and easily as possible.

This chapter describes how to install the Core.

Chapter 2 describes the Reference Design that is provided in the Core package. You can use this Reference Design as an example to see how the Core works and to test that it is working correctly.

Chapter 3 describes how to use the Core Wizard to modify the parameters for the Core instance included in the Reference Design or to create a new Core instance using your own custom values and inputs.

Chapter 4 describes how to simulate your Core instance.

## 1.1 System Requirements

To install the QuickPCIe Expert package, you need:

- **Operating System**: Windows 2000/XP/Vista/7 or any Unix/Linux platform supporting Java
- **Hard Disk**: 1 GB for Core installation and component design

## 1.2 EDA Tools Requirements

One of the following EDA tools are required for simulation:

- ModelSim SE 10.4b or above
- NCSim 14.20-s024

**Note:** Aldec 2015.10 is supported on demand. Contact PLDA Technical support for more information.

For synthesis, you will also require:

- Vivado Design Suite 2015.3
- Xilinx Platform USB cable

## 1.3 Available Packages

The QuickPCIe Expert Core is available in any of the following packages:

- **Xilinx Source**, which contains full source code for the Core:
  quickpcie-expert_genX_vXXXbuildXXX_xilinx_source.tar.gz
- **Xilinx Full**: enables you to generate programming files for any supported device:
  quickpcie-expert_genX_vXXXbuildXXX_xilinx_full.tar.gz
- **Xilinx Board/Eval**, which is used for testing hardware: For the Board package, there is no time limit when connecting protocore signals with PLDA boards.
  quickpcie-expert_genX_vXXXbuildXXX_xilinx_board_eval.tar.gz

In the above packages, ‘genX’ can be Gen3, Gen2, or Gen1, while vXXX and buildXXX indicate the current version and build number.

**Note:** With the Eval package, the design will only work for a limited time.

## 1.4 Installing the Package

1. Unzip the tar.gz file using unzip software such as WinRAR or 7-Zip.
2. Open a shell and set the working directory to the directory where the Core package has been downloaded from the PLDA web site.
3. Create an empty directory.
4. At the prompt, type:

```
tar -xzf <full package name> -C <your directory path>
```

## 1.5 Package Features

The following table shows the directory structure and contents of your package:

| | |
|---|---|
| **Core** | **Simulation**<br>• **Modelsim**<br>  • **Verilog**: pre-compiled Verilog library<br>• **NCSim**<br>  • **Verilog**: Verilog-protected, UNIX/Linux |
| | **Source** (in the Full and Source packages only)<br>• **Verilog**: Original clear-text source code, Xilinx Hard IP and QuickPCIe Expert include files |
| | **Synthesis**<br>• **Xilinx**<br>  • **Verilog**: Single Xilinx clear-text (Source) or encrypted (Full, Board/Eval) file |
| **Documentation** | • quickpcie_expert_build_history_xx.pdf<br>• quickpcie_expert_revision_history_xx.pdf<br>• quickpcie_expert_getting_started_xx.pdf<br>• quickpcie_expert_reference_manual_xx.pdf<br>• pcie_bfm_3_reference_manual.pdf |
| **Software** | • API and test application for QuickPCIe Expert. |
| **Reference Design** | • **Simulation**<br>  • Endpoint<br>• **Source**<br>  • Endpoint<br>• **Synthesis**<br>  • Endpoint |
| **PLDA Testbench** | The PLDA Testbench for PCI Express 3.0, exclusively compatible with the Core, includes the following:<br>• Unlimited duration for purchased products<br>• BFM Instance: Root Port, Endpoint<br>• Number of Transactions: 1,000<br>• Monitor<br>Refer to the *PLDA BFM 3 Reference Manual* for more information. |
| **Wizard** | The QuickPCIe Expert wizard is used to create a top-level instance of the Core. See Chapter 3: Creating a Core Instance for more information. |

# Chapter 2 Reference Design

## 2.1 Introduction

The Reference Design demonstrates an implementation example of the QuickPCIe Expert Core. It enables you to perform DMA testing (Host to Card and Card to Host) through Stream and AXI Master interfaces. It also enables testing of the AXI4-Lite Master and Slave interfaces.

The complete Register Transfer Level (RTL) source code of the design is provided so that you can:

- Perform hardware testing using a prototyping board.
- Modify the Reference Design in order to integrate the Core with your own design.

## 2.2 Architecture of the Reference Design

### 2.2.1 Top-Level Blocks

The Reference Design is illustrated below:

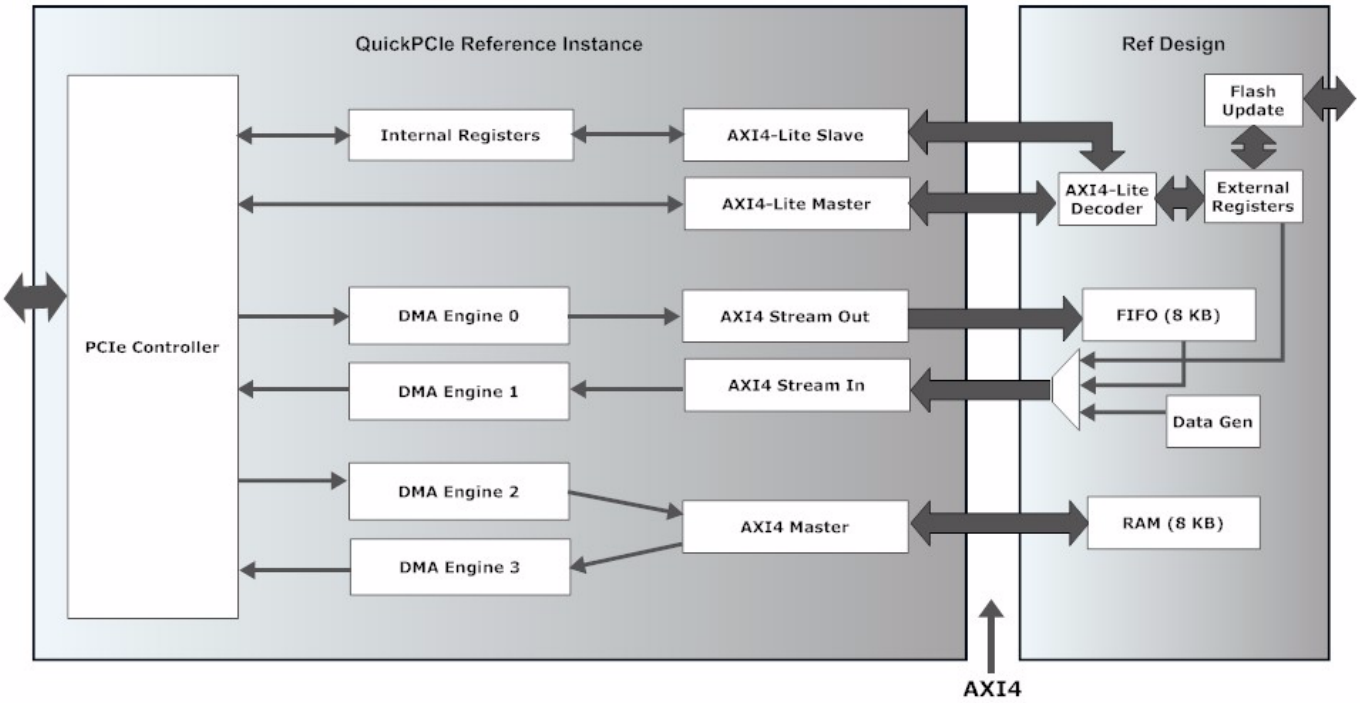


**Figure 1: Top-level blocks of the Reference Design**

- **AXI4-Lite Slave**: Loops back to the AXI4-Lite Master through the AXI4-Lite Decoder Module.
- **AXI4-Lite Master**: Accesses the Reference Design external registers. It also accesses the QuickPCIe Expert internal registers through the AXI4-Lite Slave interface. See Section 2.2.3.
- **AXI4 Stream Out**: Connected to DMA Engine 0 and loops back to AXI4 Stream In through an 8 KB FIFO.
- **AXI4 Stream In**: Connected to DMA Engine 1 and loops back to AXI4 Stream Out through an 8 KB FIFO or a data generator.
- **AXI4 Master**: Connected to DMA Engines 2 and 3 and to an 8 KB embedded SRAM.

## 2.2.2 Reference Design Files and File Structure

The following figure provides an overview of the files implicated in the Reference Design.

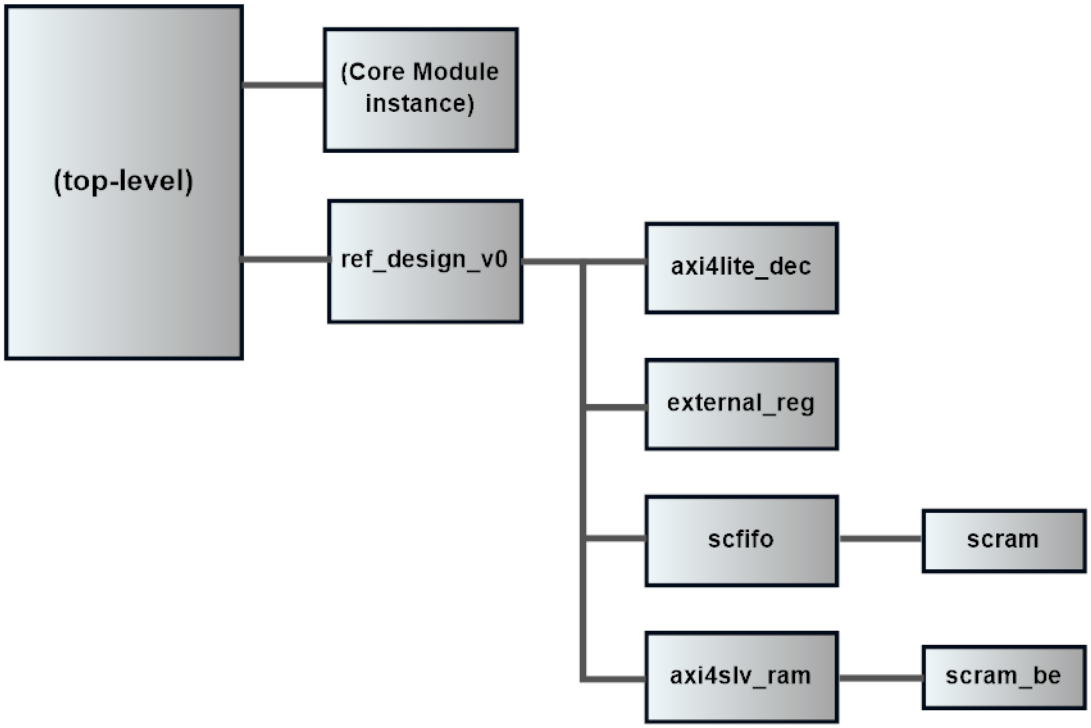


**Figure 2: Architecture of the simulation environment**

The following table describes each file used by the Reference Design.

| Module / File | Description |
|---|---|
| **(core_module_instance)** | QuickPCIe Expert Core module instance, as shown in Figure 1. |
| **(top_level)** | Reference Design top level, as shown in Figure 1. |
| **ref_design_v0** | Reference design wrapper that connects the peripheral modules. |
| **axi4lite_dec** | Sample interconnection module between one AXI4-Lite Slave interface and two AXI4-Lite Master interfaces. |
| **external_reg** | Sample of AXI4-Lite Slave access to external registers. |
| **scfifo** | Generic single-clock FIFO model. |
| **scram** | Generic SCRAM model. |
| **axi4slv_ram** | Example of AXI4 Slave access to SCRAM, which instantiates the scram_be module. |
| **scram_be** | Generic SCRAM with Byte Enable model. |

### 2.2.3 Registers

The following table describes the registers used by the Reference Design:

| Byte Address | R/W | Description |
|---|---|---|
| **External Registers** | | |
| 0x0000 - 0x0003 | RW | **REFD_MODE**: Reference Design Mode:<br>• Bit [0]:<br>  • when 0b, Stream Out is looped to Stream In through a FIFO<br>  • when 1b, Stream In data is automatically generated<br>• Bit [15:3]: Stream In Packet Size: indicates the size of Stream In Packets in QWords. A value of 13'd0 specifies a size of 2^13 = 8192 QW = 64KBytes.<br>• Bit [16]:<br>  • when 0b, Stream In Data are generated infinitely<br>  • when 1b, Stream In Data generation is stopped once Stream In Packet numbers have been sent.<br>• Bit [31:24]: Stream In Packet Number, from 1 (8'h01) to 255 (8'hff). |
| 0x0004 - 0x0007 | RW | **LED_CONTROL**: LED States for the Control Board:<br>• Bit [0]: when 1b, LED0 is on; when 0b, LED0 is off<br>• Bit [1]: when 1b, LED1 is on; when 0b, LED1 is off<br>• Bit [2]: when 1b, LED2 is on; when 0b, LED2 is off<br>• Bits [31:3]: Mailbox Registers |
| 0x0008 - 0x000B | RW | **INTERRUPT_CONTROL**: Interrupt Requests towards the PCIe Host:<br>• Bit [0]: when 1b, qpcie_interrupt_in[6] local interrupt signal is asserted<br>• Bit [1]: when 1b, qpcie_interrupt_in[7] local interrupt signal is asserted<br>• Bits [31:2]: Mailbox Registers |
| 0x000C - 0x001F | RW | **Mailbox Registers**: General purpose 32-bit read/write registers |
| 0x0020 - 0x0023 | R0 | **STR_FIFOCNT**: Stream FIFO Count (in words of 128-bits):<br>• Bits[9:0]: FIFO count from 0 to 512 words<br>• Bits [31:10]: reserved |
| 0x0024 - 0x0027 | RO | **USER_SWITCH**: User Switch states for the Control Board:<br>• Bit [0]: when 1b, Switch0 is on; when 0b, Switch0 is off<br>• Bit [1]: when 1b, Switch1 is on; when 0b, Switch1 is off<br>• Bit [2]: when 1b, Switch2 is on; when 0b, Switch2 is off<br>• Bits [31:3]: reserved |
| 0x0028 - 0x002B | -- | Reserved |
| 0x002C - 0x002F | -- | **REFD_PERF**: Reference Design Performance:<br>• Bit [15:0]: reserved<br>• Bit [25:16]: Reference Design Clock Frequency in MHz. 10'd250 indicates a frequency of 250 MHz, for example.<br>• Bit [31:26]: Reference Design Data Path in Bytes. For example, 6'd16 means 16 Bytes.<br>The maximal throughput that can reach an interface in MBytes/second is equal to the Reference Design Clock Frequency in MHz multiplied by the Data Path in Bytes. |
| 0x0030 - 0x0033 | RO | **STREAM_PERF**: Percentage of bandwidth used in last 134-ms period:<br>• Bit [15:0]: Percentage of Stream In interface bandwidth from 0h to FFFFh<br>• Bit [31:16]: Percentage of Stream Out interface bandwidth from 0h to FFFFh |

| Byte Address | R/W | Description |
|---|---|---|
| 0x0034 - 0x0037 | RO | **MASTER_PERF**: Percentage of bandwidth used in last 134-ms period:<br>• Bit [15:0]: Percentage of AXI Master Read I/F bandwidth from 0h to FFFFh<br>• Bit [31:16]: Percentage of AXI Master Write I/F bandwidth from 0h to FFFFh |
| 0x0038 - 0x003F | RW/RO | **FLASH_UPDATE**: Flash Update registers |
| **Internal Registers** | | |
| 0x1000 - 0x1FFF | RO/RW | **QuickPCIe Expert Internal Registers** |

# 2.3 Simulation of the Reference Design

## 2.3.1 Overview

The following figure offers an overview of the simulation environment:

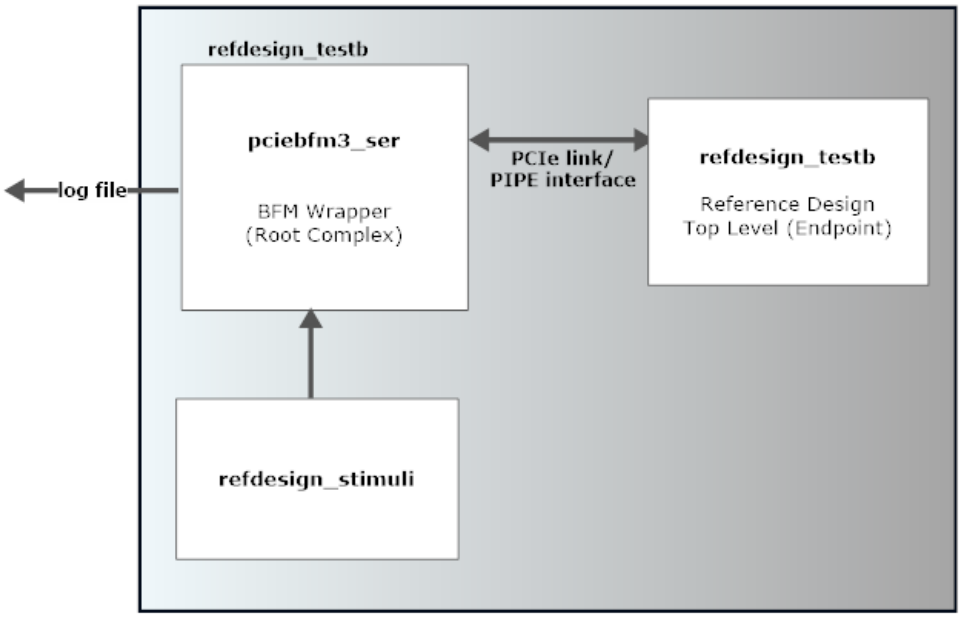


**Figure 3: Overview of the simulation environment**

The following table describes each file used in the Simulation environment:

| Module/File | Description |
|---|---|
| refdesign_stimuli | Verilog model of QuickPCIe Expert Test Application that lists different test cases. |
| pciebfm3_ser | PLDA BFM with serial link interface. |
| refdesign_testb | Testbench design that connects the BFM and the Reference Design. This file is the top-level file for simulation, described in Section 2.2.2. It also contains a stimuli module that sends PCI Express transactions to configure and test the design. |

## 2.3.2 Launching the Simulation

A **.do** script file is provided for simulating the project with ModelSim.

To run a simulation:

1. Change Directory to ./ref_design/simulation/endpoint.
2. Type the following command line to run the script file:

**do simulate.tcl $simulator $target $lane $payload $script $gen**

The following table describes each argument:

| Argument | Argument Name | Supported Values |
|---|---|---|
| $simulator | Simulator | • model (for ModelSim)<br>• ncsim<br>• aldec |
| $target | Device targeted | • vus<br>• kus<br>• v7gth<br>• v7gtx<br>• z7<br>• k7<br>• a7 |
| $lane | PCIe Link Width | • x1<br>• x2<br>• x4<br>• x8 |
| $payload | PCIe Maximum Payload Size | • 128B<br>• 256B |
| $script | Script | • m0 |
| $gen | Link speed - 2.5, 5.0 or 8.0 Gbps | • gen1<br>• gen2<br>• gen3 |

The simulation then performs each of the following steps:

- Initializes the PCIe BFM.
- Initializes the Reference Design PCIe Configuration.
- Reads and Writes to the QuickPCIe Expert Internal Registers.
- Programs a Direct DMA Transfer from the 64-bit PCIe link to the AXI Stream Interface.
- Programs a Direct DMA Transfer from the AXI Stream Interface to the 32-bit PCIe link.
- Programs two simultaneous Direct DMA Transfers between the PCIe link and the AXI Stream Interface.
- Programs a Direct DMA Transfer from the 64-bit PCIe link to the AXI Master Interface.
- Programs a Direct DMA Transfer from the AXI Master Interface to the 32-bit PCIe link.
- Programs a SG-DMA Transfer from the 64-bit PCIe link to the AXI Master Interface.
- Programs a SG-DMA Transfer from AXI Master Interface to the 64-bit PCIe link.
- Programs two simultaneous SG-DMA Transfers between the PCIe link and the AXI Stream Interface.
- Executes a Direct Write and Read between the 64-bit PCIe link and the AXI Master Interface.
- Closes the simulation.

# Chapter 3 Creating a Core Instance

The QuickPCIe Expert Wizard generates a Verilog wrapper that instantiates the Core with custom values, input ports, and output ports.

The following sections describe how to instantiate the Core using the Wizard.

WARNING:

- The file created by the Wizard *should not* be modified with a text editor. Open the Wizard and select an existing .v file in order to modify an existing instance of the Core.
- If you have already created an instance of the Core and you create a second instance of the Core with the same name, the Wizard will overwrite your original file without warning.

Follow the directions below to launch a Wizard and create an instance of the Core:

1. Launch the Wizard:
   - **Linux**: Browse to the /wizard/linux.gtk.x86 or /wizard/linux.gtk.x86_64 directory and run pldaWizard.
   - **Windows**: Browse to the /wizard/win32.win32.x86 or /wizard/win32.win32.x86_64 directory and run pldaWizard.exe.

   The following window appears:

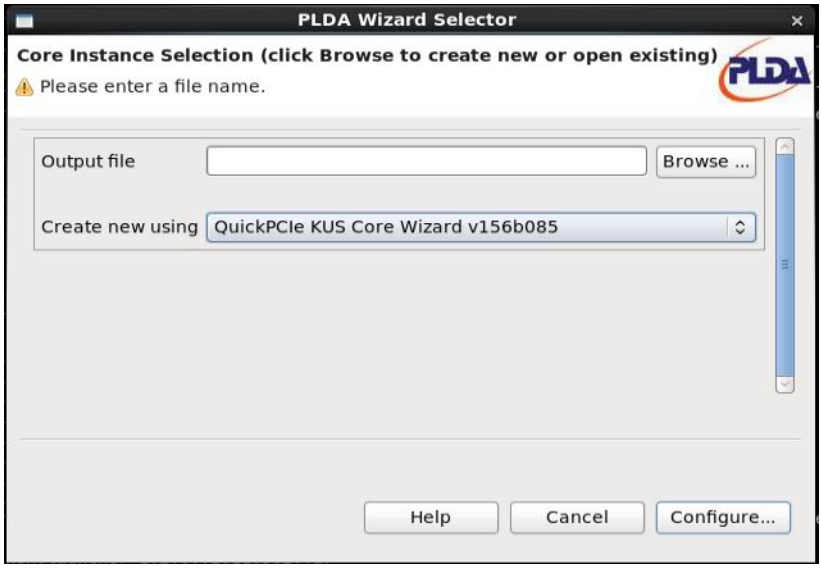


**Figure 4: The choose wrapper window**

To modify an existing wrapper, browse your Hard Drive, select the desired wrapper, and click **Open**.

To create a new wrapper, type a name in the **Name** text box and click **Open**. The Wizard will then guide you through the modification or creation of your Core instance.

You can click the **Help** button on any page in the Wizard for more information about the settings on that page.

# Chapter 4 Simulating your Design

This chapter provides background information for running a functional simulation of the Core. The simulation environment includes the Core, the Core wrapper, the PCI Express BFM3, and all or part of your design (application layer logic).

## 4.1 Simulation Environment

The figure below illustrates a typical simulation environment using a serial or parallel interface

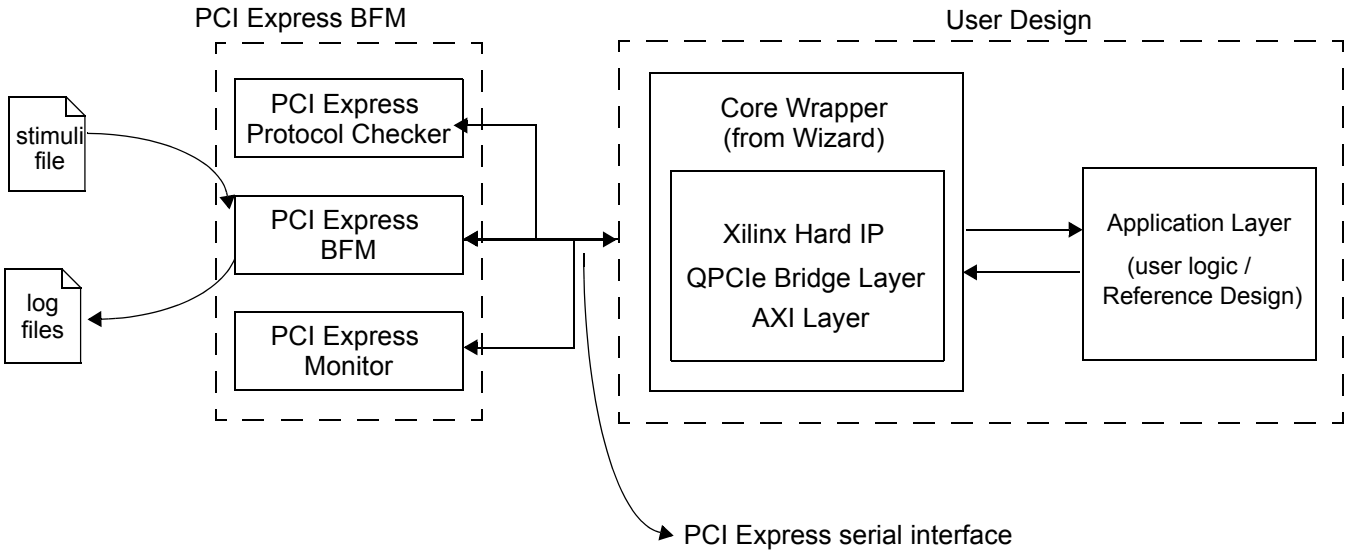


**Figure 5: Simulation environment**

The simulation architecture includes the following elements:

- **PCI Express BFM3**: The Core package includes evaluation versions of the PLDA PCI Express 3.0 BFM. The BFM is provided as compiled libraries or as protected code for various simulators. It includes a link monitor, and can be instantiated at the serial, parallel, and PIPE interface levels. Refer to the *BFM3 Reference Manual* in the documentation directory for additional information.
- **User Design**: Application layer logic implemented by the designer that connects to the Core's local Application Layer interface. For a tryout simulation, a fully functional or complete design is not required, however, all of the Core's Application Layer inputs should be set to appropriate values. The interface between the Core and the Application Layer is described in the *Reference Manual*.
- **QuickPCIe Expert Core wrapper**: This is the customized wrapper you create with the Core Wizard, as described in the previous section.
- **Xilinx Hard IP**: This is a customized IP instance of the Xilinx PCI Express Hard IP dedicated to the QuickPCIe Expert Core.
- **QPCIe Layer**: This layer consists of the Internal Registers, DMA Engine Modules, Address Translator Modules and an Interconnect Module.
- **AXI Layer**: This layer consists of the AXI4 and AXI4-Lite Slave and Master interfaces, the AXI4 Master Descriptor interface, and the AXI4 Stream In/out interfaces.

# 4.2 Simulating with the PLDA PCIe BFM3

## 4.2.1 ModelSim

The BFM is delivered as a simulation library. To use the BFM, map the library **pciebfm_lib**. When compiling with Verilog, you must include the **.h** file located in the **include** directory. When launching vsim, add the BFM library with the following command: **vsim -L pciebfm_lib**

## 4.2.2 NCSim

The BFM is delivered in the form of encrypted files. To use the BFM, compile the encrypted library **pciebfm_lib**.

When compiling with Verilog, you must include the **.h** file located in the **include** directory.

## 4.2.3 Simulating with ModelSim

The following directions describe how to create a project and compile the necessary libraries in order to run a simulation with the Core. Note that this section is offered as background information only. The **simulate.tcl** script provided with the Core package automatically performs these tasks and starts a simulation for you.

1. Launch ModelSim.
2. Select the working directory:
   - Choose Change Directory from the File menu. The **Choose a Directory** window appears.
   - Browse your hard drive and select or create a directory in which to compile your project.
3. Create a work library:
4. Map the required libraries:
   1. Create and map the PLDA BFM (PCI-Express Testbench) library. At the ModelSim prompt, type:
      **vlib libs/pciebfm_lib**
      **vmap pciebfm_lib «libs/pciebfm_lib»**
      **vlog -quiet -work pciebfm_lib \ +incdir+$pciebfm_path/source/vlog/include **
      **$pciebfm_path/simulation/$model/vlog/pciebfm_lib.vp**

   2. Map the QuickPCIe Expert library. At the ModelSim prompt, type:
      **vlib libs/qpcie_lib**
      **vmap q_lib «libs/qpcie_lib»**
      **vlog -quiet -work $core_lib_name \ +incdir+$core_config_path \ +incdir+$core_include_path **
      **$core_path/simulation/$model/vlog/qpcie_lib.vp **
      where "$core path" is replaced by the directory in which you installed your project.
5. Compile your core instance file(s) previously generated with the QuickPCIe Expert Wizard, by typing:
   - **vlog -work work <absolute path>/<filename>.v**
6. Compile all your design files.
7. Compile the stimuli file used to simulate your design by generating traffic with the Core. An example file, ref_design_stimuli, is provided with the Reference Design.
8. Compile your top level that connects your design with the Core, the BFM, and the stimuli instance. Example files (ref_design_testb) are provided with the Reference Design.
9. Simulate the project, by typing:
   **vsim -t ps -L pciebfm_lib -L qpcie_lib work.<design_top_level>**

### 4.2.4 Additional Xilinx Considerations

In order to simulate your design, Xilinx smart models must be installed and configured and Xilinx simulation libraries must be compiled. Please refer to the Vivado installation guide for details.

Use the memory model furnished with the Xilinx Library of Parameterized Modules (UNISIM). Note that the DCRAM located in the Xilinx primitive directory uses this Xilinx library for compilation.