LogiCORE™ IP Endpoint Block Plus v1.8 for PCI Express®

Getting Started Guide

UG343 June 27, 2008

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/23/06	1.1	Initial Xilinx release.
2/15/07	2.0	Update core to version 1.2; Xilinx tools 9.1i.
5/17/06	3.0	Update core to version 1.3; updated for PCI-SIG compliance.
8/8/07	4.0	Update core to version 1.4; Xilinx tools 9.2i, Cadence IUS v5.8.
10/10/07	5.0	Update core to version 1.5, Cadence IUS v6.1.
3/24/08	6.0	Update core to version 1.6; Xilinx tools 10.1.
4/25/08	7.0	Update core to version 1.7.
6/27/08	8.0	Update core to version 1.8.

Table of Contents

Appendix: Additional Design Considerations

About This Guide

The *Endpoint Block Plus for PCI Express® Getting Started Guide* provides information about generating an Endpoint Block Plus for PCI Express (PCIe®) core, customizing and simulating the core using the provided example design, and running the design files through implementation using the Xilinx tools.

Contents

This guide contains the following chapters:

- Preface, "About this Guide," introduces the organization and purpose of this guide and the conventions used in this document.
- Chapter 1, "Introduction," describes the core and related information, including system requirements, recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.
- Chapter 2, "Licensing the Core" provides instructions for selecting a license option for the core.
- Chapter 3, "Quickstart Example Design," provides instructions for quickly generating, simulating, and implementing the example design and the dual core example design using the demonstration test bench.
- Appendix, "Additional Design Considerations," defines additional considerations when implementing the example design.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands you enter in a syntactical statement	**`ngdbuild`** `design_name`

Convention	Meaning or Use	Example
Italic font	References to other manuals	See the *User Guide* for details.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
<text in brackets>	User-defined variable for directory names.	<component_name>
Dark Shading	Items that are not supported or reserved	Unsupported feature
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **bus[7:0]**, they are required.	**ngdbuild** [option_name] design_name
Braces { }	A list of items from which you must choose one or more	**lowpwr ={on|off}**
Vertical bar |	Separates items in a list of choices	**lowpwr ={on|off}**
Vertical ellipsis . . .	Repetitive material that has been omitted	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' . . .
Horizontal ellipsis ...	Omitted repetitive material	**allow block** block_name loc1 loc2 ... locn;
Notations	The prefix '0x' or the suffix 'h' indicate hexadecimal notation	A read of address 0x00112975 returned 45524943h.
	An '_n' means the signal is active low	usr_teof_n is active low.

Online Document

The following linking conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details. See "Title Formats" in Chapter 1 for details.
Blue, underlined text	Hyperlink to a website (URL)	Go to www.xilinx.com for the latest speed files.

XILINX®

Chapter 1

Introduction

The Endpoint Block Plus for PCI Express is a high-bandwidth, scalable, and reliable serial interconnect building block for use with Virtex™-5 FPGA devices. This core supports Verilog® and VHDL. The example design described in this guide is provided in Verilog and VHDL.

This chapter introduces the core and provides related information, including system requirements, recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.

About the Core

The Endpoint Block Plus for PCIe core is a Xilinx CORE Generator™ IP core, included in the latest IP Update on the Xilinx IP Center. For additional information about the core, see the Block Plus for PCIe product page. For information about obtaining a license for the core, see Chapter 2, "Licensing the Core."

System Requirements

Windows

- Windows XP® Professional 32-bit/64-bit
- Windows Vista® Business 32-bit/64-bit

Linux

- Red Hat® Enterprise Linux WS v4.0 32-bit/64-bit
- Red Hat® Enterprise Desktop v5.0 32-bit/64-bit (with Workstation Option)
- SUSE Linux Enterprise (SLE) v10.1 32-bit/64-bit

Software

- ISE™ 10.1

  Check the release notes for the required Service Pack; ISE Service Packs can be downloaded from www.xilinx.com/support/download/index.htm.

Recommended Design Experience

Although the Endpoint Block Plus for PCIe is a fully verified solution, the challenge associated with implementing a complete design varies depending on the configuration and functionality of the application. For best results, previous experience building high

performance, pipelined FPGA designs using Xilinx implementation software and User Constraints Files (UCF) is recommended.

Additional Core Resources

For detailed information and updates about the core, see the following documents, available from the Block Plus for PCIe product page unless otherwise noted.

- *LogiCORE IP Endpoint Block Plus for PCI Express Data Sheet*
- *LogiCORE IP Endpoint Block Plus for PCI Express User Guide*
- *LogiCORE IP Endpoint Block Plus for PCI Express Release Notes* (available from the core directory after generating the core)
- *Virtex-5 Integrated Endpoint Block for PCI Express Designs User Guide* (UG197)

Additional information and resources related to the PCI Express technology are available from the following web sites:

- PCI Express at PCI-SIG
- PCI Express Developer's Forum

Technical Support

For technical support, go to www.xilinx.com/support. Questions are routed to a team of engineers with expertise using the Endpoint Block Plus for PCI Express core.

Xilinx provides technical support for use of this product as described in the *LogiCORE IP Endpoint Block Plus for PCI Express User Guide* and the *LogiCORE IP Endpoint Block Plus for PCI Express Getting Started Guide*. Xilinx cannot guarantee timing, functionality, or support of this product for designs that do not follow these guidelines.

Feedback

Xilinx welcomes comments and suggestions about the core and the accompanying documentation.

Core

For comments or suggestions about the core, please submit a WebCase from www.xilinx.com/support. Be sure to include the following information:

- Product name
- Core version number
- Explanation of your comments

Document

For comments or suggestions about this document, please submit a WebCase from www.xilinx.com/support. Be sure to include the following information:

- Document title
- Document number
- Page number(s) to which your comments refer
- Explanation of your comments

Chapter 2

Licensing the Core

This chapter provided licensing options for the Endpoint Block Plus for PCI Express core, which you must do before using the core in your designs. The core is provided under the terms of the Xilinx LogiCORE Site License Agreement, which conforms to the terms of the SignOnce IP License standard defined by the Common License Consortium.

Before you Begin

This chapter assumes you have installed the core using either the CORE Generator IP Software Update installer or by performing a manual installation after downloading the core from the web. For additional information about installing the core, see the Block Plus for PCIe product page.

License Options

The Endpoint Block Plus for PCIe core provides two license options: a Simulation Only Evaluation license and a Full license. The Simulation Only Evaluation license is provided by default with the CORE Generator. The Full license is also free of charge, but you must register for it on the Block Plus for PCIe product page. See "Obtaining a Full License" below.

Simulation Only

The Simulation Only Evaluation license, provided by default with CORE Generator, lets you assess the core functionality with either the provided example design or alongside your own design, and demonstrates the various interfaces to the core in simulation. (Functional simulation is supported by a dynamically generated gate-level netlist.)

Full

The Full license provides full access to all core functionality both in simulation and in hardware, including:

- Gate-level functional simulation support
- Back annotated gate-level simulation support
- Full implementation support including place and route and bitstream generation
- Full functionality in the programmed device with no time outs

Obtaining Your License

Simulation Only Evaluation License

The Simulation Only Evaluation license is provided with the CORE Generator system and requires no license file.

Obtaining a Full License

To obtain a Full license, you must register for access to the *lounge*, a secured area of the Block Plus for PCIe product page.

- From the product page, click Register to register and request access to the lounge. Access to the lounge is automatic and granted immediately.
- After you receive confirmation of lounge access, click Access Lounge from the product page and log in.
- Click Access Lounge on the product lounge page and fill out the license request form linked from this location; then click Submit to automatically generate the license. An e-mail containing the license and installation instructions will be sent immediately to the email address you specified.

Installing Your License File

If you select the Full License option, you will receive an email that includes instructions for installing your license file. In addition, information about advanced licensing options and technical support is provided.

Chapter 3

Quickstart Example Design

This chapter provides an overview of the Endpoint Block Plus for PCI Express example design (both single and dual core) and instructions for generating the core. It also includes information about simulating and implementing the example design using the provided demonstration test bench.

Overview

The example simulation design consists of two discrete parts:

- The Downstream Port Model, a test bench that generates, consumes, and checks PCI Express bus traffic.
- The Programmed Input Output (PIO) example design, a completer application for PCI Express. The PIO example design responds to Read and Write requests to its memory space and can be synthesized for testing in hardware.

Simulation Design Overview

For the simulation design, transactions are sent from the Downstream Port Model to the Block Plus core and processed by the PIO example design. Figure 3-1 illustrates the simulation design provided with the Block Plus core. For more information about the Downstream Port Model, see Appendix B, "Downstream Port Model Test Bench," in the *LogiCORE IP Endpoint Block Plus for PCI Express User Guide*.

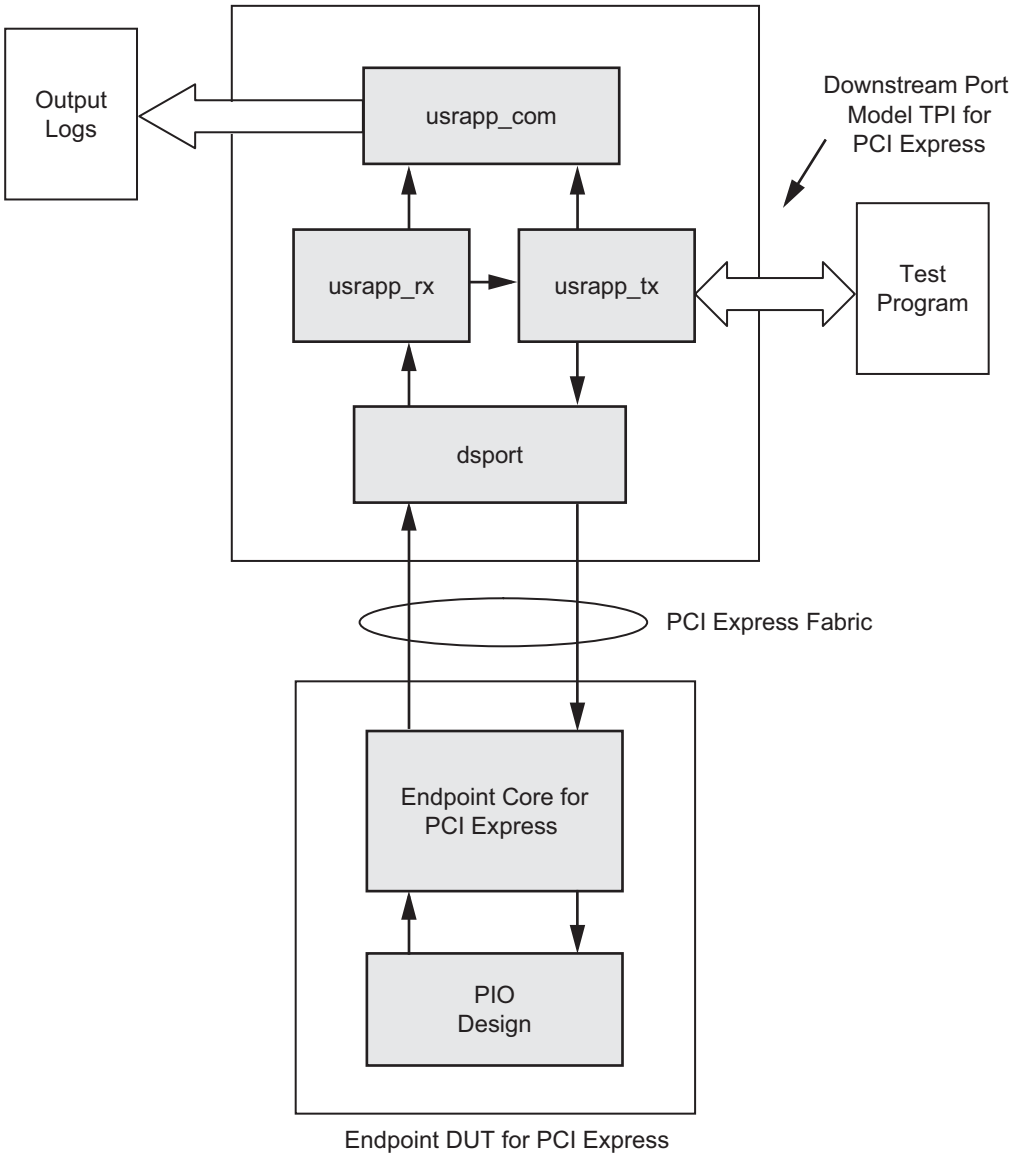

Figure 3-1: **Simulation Example Design Block Diagram**

Implementation Design Overview

The implementation design consists of a simple PIO example that can accept read and write transactions and respond to requests, as illustrated in Figure 3-2. Source code for the example is provided with the core. For more information about the PIO example design, see Appendix A, "Programmed Input Output Example Design," in the *LogiCORE IP Endpoint Block Plus for PCI Express User Guide* (UG341).

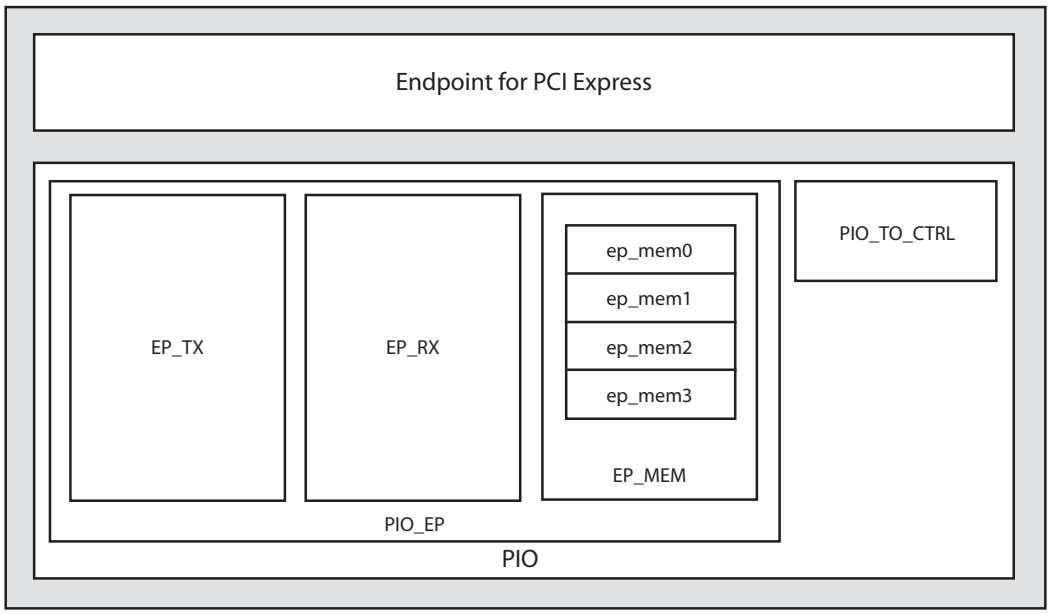

Figure 3-2: **Implementation Example Design Block Diagram**

Example Design Elements

The PIO example design elements include the following:

- Core netlists
- Core simulation models
- An example Verilog HDL or VHDL wrapper (instantiates the cores and example design)
- A customizable demonstration test bench to simulate the example design

The example design has been tested and verified with Xilinx ISE v10.1 and the following simulators:

- Cadence® IUS 6.1
- Synopsys® vcs_mxY-2006.06-SP1
- Mentor Graphics® ModelSim® v6.3c

Note: Currently, the VHDL demonstration test bench supports only ModelSim and IUS.

Generating the Core

To generate a core using the default values in the CORE Generator Graphical User Interface (GUI), do the following:

1. Start the CORE Generator.

   For help starting and using the CORE Generator, see the Xilinx CORE Generator Guide, available from the ISE documentation web page.

2. Choose File > New Project.

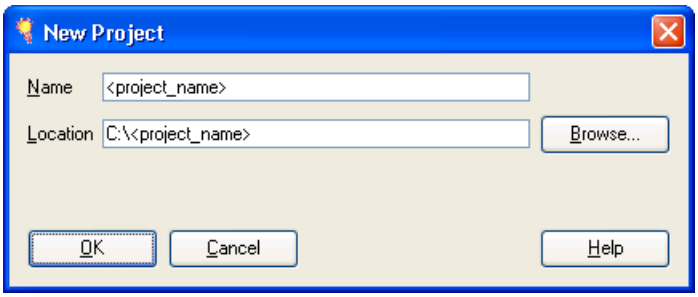

Figure 3-3: **New Project Dialog Box**

3. Enter a project name and location, then click OK. `<project_dir>` is used in this example. The Project Options dialog box appears.

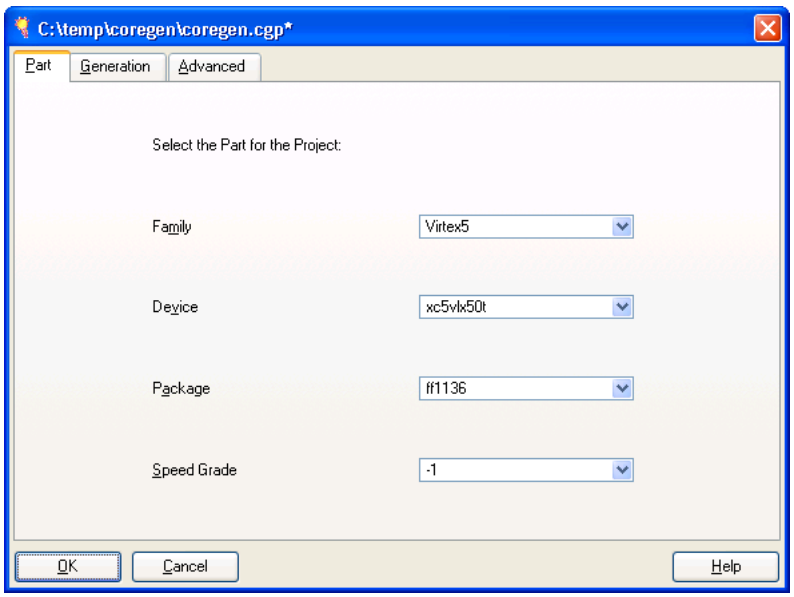

Figure 3-4: **Project Options**

4. Set the project options:

   From the Part tab, select the following options:

   - **Family**: Virtex5
   - **Device**: xc5vlx50t
   - **Package**: ff1136
   - **Speed Grade**: -1

   Note: If an unsupported silicon device is selected, the core is dimmed (unavailable) in the list of cores.

   From the Generation tab, select the following parameters, and then click OK.

   - **Design Entry**. Select either VHDL or Verilog.
   - **Vendor**. Select Synplicity® or ISE (for XST).

5. Locate the core in the selection tree under Standard Bus Interfaces/PCI Express; then double-click the core name to display the Block Plus main screen.

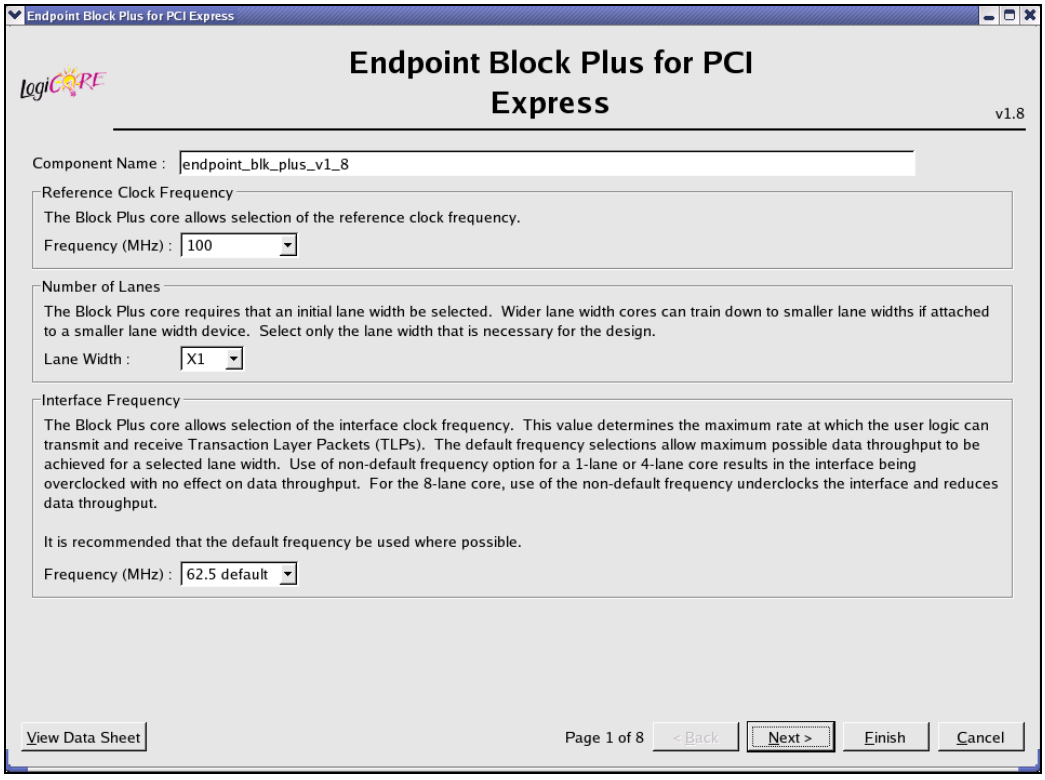

Figure 3-5: **Endpoint Block Plus Main Screen**

6. In the Component Name field, enter a name for the core. `<component_name>` is used in this example.
7. Click Finish to generate the core using the default parameters. The core and its supporting files, including the PIO example design and Downstream Port Model test bench, are generated in the project directory.

For detailed information about the example design files and directories see "Directory Structure and File Contents," page 20.

Simulating the Example Design

The example design provides a quick way to simulate and observe the behavior of the core. The simulation environment provided with the Block Plus core performs simple memory access tests on the PIO example design. Transactions are generated by the Downstream Port Model and responded to by the PIO example design.

- PCI Express Transaction Layer Packets (TLPs) are generated by the test bench transmit user application (`pci_exp_usrapp_tx`). As it transmits TLPs, it also generates a log file, `tx.dat`.
- PCI Express TLPs are received by the test bench receive user application (`pci_exp_usrapp_rx`). As the user application receives the TLPs, it generates a log file, `rx.dat`.

For more information about the test bench, see Appendix B, "Downstream Port Model Test Bench," in the *LogiCORE IP Endpoint Block Plus for PCI Express User Guide*.

Setting up for Simulation

To run the gate-level simulation you must have the Xilinx Simulation Libraries compiled for your system. See the Compiling Xilinx Simulation Libraries (COMPXLIB) in the *Xilinx ISE Synthesis and Verification Design Guide*, and the *Xilinx ISE Software Manuals and Help*. Documents can be downloaded from www.xilinx.com/support/software_manuals.htm.

Simulator Requirements

Virtex-5 device designs require either a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator or a SWIFT-compliant simulator.

- For a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator, ModelSim v6.3c is currently supported.
- For a SWIFT-compliant simulator, Cadence IUS v6.1 and Synopsys VCS 2006.06-SP1 are currently supported.

Note for Cadence IUS users: The work construct must be manually inserted into your CDS.LIB file as shown below.

```
DEFINE WORK WORK
```

Running the Simulation

For Cadence IUS

The simulation scripts provided with the example design support pre-implementation (RTL) simulation. The existing test bench can be used to simulate with a post-implementation version of the example design.

The pre-implementation simulation consists of the following components:

- Verilog or VHDL model of the test bench
- Verilog or VHDL RTL example design
- The Verilog or VHDL model of the Endpoint Block Plus for PCI Express

1. To run the simulation, go to the following directory:

   `<project_dir>/<component_name>/simulation/functional`

2. Run the script that corresponds to your simulation tool using one of the following:
   - **VCS**: `simulate_vcs.sh`
   - **Cadence IUS**: `simulate_ncsim.sh`
   - **ModelSim**: `vsim -do simulate_mti.do`

Implementing the Example Design

After generating the core, the netlists and the example design can be processed using the Xilinx implementation tools. The generated output files include scripts to assist you in running the Xilinx software.

To implement the example design:

Open a command prompt or terminal window and type the following:

Windows

```
ms-dos> cd <project_dir>\<component_name>\implement
ms-dos> implement.bat
```

Linux

```
% cd <project_dir>/<component_name>/implement
% ./implement.sh
```

These commands execute a script that synthesizes, builds, maps, and place-and-routes the example design, and then generates a post-par simulation model for use in timing simulation. The resulting files are placed in the `results` directory and execute the following processes:

1. Removes data files from the previous runs.
2. Synthesizes the example design using either Synplicity Synplify or XST.

   - The core is instanced as a black box within the example design.
3. `ngdbuild`. Builds a Xilinx design database for the example design.

   Inputs:

   Part-Package-Speed Grade selection:

   XC5VLX50T-FF1136-1

   Example design UCF:

   `xilinx_pci_exp_blk_plus_1_lane_ep-XC5VLX50T-FF1136-1.ucf`
4. `map`: Maps design to the selected FPGA using the constraints provided.
5. `par`: Places cells onto FPGA resources and routes connectivity.
6. `trce`: Performs static timing analysis on design using constraints specified.
7. `netgen`: Generates a logical Verilog HDL or VHDL representation of the design and an SDF file for post-layout verification.
8. `bitgen`: Generates a bitstream file for programming the FPGA.

The following FPGA implementation related files are generated in the results directory:

- `routed.bit`
  FPGA configuration information.
- `routed.v[hd]`
  Verilog or VHDL functional Model.

- `routed.sdf`
  Timing model Standard Delay File.
- `mapped.mrp`
  Xilinx map report.
- `routed.par`
  Xilinx place and route report.
- `routed.twr`
  Xilinx timing analysis report.

The script file starts from an EDIF/NGC file and results in a bitstream file. It is possible to use the Xilinx ISE GUI to implement the example design. However, the GUI flow is not presented in this document.

Directory Structure and File Contents

The Endpoint Block Plus for PCIe example design directories and their associated files are defined in the sections that follow. Click a directory name to go to the desired directory and its associated files.

Example Design

<project directory>
Top-level project directory; name is user-defined

<project directory>/<component name>
Core release notes readme file

<component name>/doc
Product documentation

<component name>/example_design
Verilog or VHDL design files

<component name>/implement
Implementation script files

implement/results
Results directory, created after implementation scripts are run, and contains implement script results

<component name>/simulation
Simulation scripts

simulation/dsport
Simulation files

simulation/functional
Functional simulation files

simulation/tests
Test command files

Note: For the dual core example design directory structure and file contents, see "Dual Core Directory Structure and File Contents," page 26.

<project directory>

The project directory contains all the CORE Generator project files.

Table 3-1: **Project Directory**

Name	Description
<project_dir>	
<component_name>.ngc	Top-level netlist.
<component_name>.v[hd]	Verilog or VHDL simulation model.
<component_name>.xco	CORE Generator project-specific option file; can be used as an input to the CORE Generator.
<component_name>_flist.txt	List of files delivered with core.
<component_name>.{veo|vho}	Verilog or VHDL instantiation template.

Back to Top

<project directory>/<component name>

The component name directory contains the release notes readme file provided with the core, which may includes tool requirements, last-minute changes, updates, and issue resolution.

Table 3-2: **Component Name Directory**

Name	Description
<project_dir>/<component_name>	
pcie_blk_plus_readme.txt	Release notes file.

Back to Top

<component name>/doc

The doc directory contains the PDF documentation provided with the core.

Table 3-3: **Doc Directory**

Name	Description
<project_dir>/<component_name>/doc	
pcie_blk_plus_ds551.pdf	*LogiCORE IP Endpoint Block Plus for PCI Express Data Sheet*
pcie_blk_plus_gsg343.pdf	*LogiCORE IP Endpoint Block Plus for PCI Express Getting Started Guide*
pcie_blk_plus_ug341.pdf	*LogiCORE IP Endpoint Block Plus for PCI Express User Guide*

Back to Top

<component name>/example_design

The example design directory contains the example design files provided with the core.

Table 3-4: **Example Design Directory**

Name	Description
<project_dir>/<component_name>/example_design	
pci_exp_8_lane_64b_ep.v pci_exp_4_lane_64b_ep.v pci_exp_1_lane_64b_ep.v	Verilog top-level port list, applicable to the 8-lane, 4-lane, and 1-lane endpoint design, respectively.
<filename>.ucf	Example design UCF. Filename varies by lane-width, part, and package selected.
xilinx_pci_exp_8_lane_ep_product.v xilinx_pci_exp_4_lane_ep_product.v xilinx_pci_exp_1_lane_ep_product.v	Enables Block Plus 8-lane, 4-lane, and 1-lane cores, respectively, in the test bench.
xilinx_pci_exp_8_lane_ep.v xilinx_pci_exp_4_lane_ep.v xilinx_pci_exp_1_lane_ep.v xilinx_pci_exp_ep.vhd	Verilog or VHDL top-level PIO example design files for 8-lane, 4-lane, and 1-lane cores.
pci_exp_64b_app.v[hd] EP_MEM.v[hd] PIO.v[hd] PIO_EP.v[hd] PIO_EP_MEM_ACCESS.v[hd] PIO_TO_CTRL.v[hd] PIO_64.v PIO_64_RX_ENGINE.v[hd] PIO_64_TX_ENGINE.v[hd]	PIO example design files.

Back to Top

<component name>/implement

The implement directory contains the core implementation script files.

Table 3-5: **Implement Directory**

Name	Description
<project_dir>/<component_name>/implement	
xst.scr	XST synthesis script.
implement.bat implement.sh	DOS and Linux implementation scripts.
synplify.prj	Synplify synthesis script.
xilinx_pci_exp_1_lane_ep_inc.xst xilinx_pci_exp_4_lane_ep_inc.xst xilinx_pci_exp_8_lane_ep_inc.xst	XST project file for 1-lane, 4-lane, and 8-lane example design, respectively.

Back to Top

implement/results

The results directory is created by the implement script, after which the implement script results are placed in the results directory.

Table 3-6: **Results Directory**

Name	Description
<project_dir>/<component_name>/implement/results	
Implement script result files.	

Back to Top

<component name>/simulation

The simulation directory contains the simulation source files provided with the core.

Table 3-7: **Simulation Directory**

Name	Description
<project_dir>/<component_name>/simulation	
board_common.v	Contains test bench definitions.
board.v[hd]	Top-level simulation module.
sys_clk_gen_ds.v[hd]	System differential clock source.
sys_clk_gen.v[hd]	System clock source.
xilinx_pci_exp_cor_ep.f	List of files comprising the design being tested.
xilinx_pci_exp_defines.v	PCI Express application macro definitions.

Back to Top

simulation/dsport

The dsport directory contains the data stream simulation scripts provided with the core.

Table 3-8: **dsport Directory**

Name	Description
<project_dir>/<component_name>/simulation/dsport	
dsport_cfg.v[hd] pci_exp_expect_tasks.v pci_exp_1_lane_64b_dsport.v[hd] pci_exp_4_lane_64b_dsport.v[hd] pci_exp_usrapp_cfg.v[hd] pci_exp_usrapp_com.v pci_exp_usrapp_rx.v[hd] pci_exp_usrapp_tx.v[hd] xilinx_pci_exp_downstream_port.v[hd] xilinx_pci_exp_dsport.v[hd] test_interface.vhd	Downstream port model files.

Back to Top

simulation/functional

The functional directory contains functional simulation scripts provided with the core.

Table 3-9: **Functional Directory**

Name	Description
<project_dir>/<component_name>/simulation/functional	
board_rtl_x01_v4fx.f board_rtl_x04_v4fx.f board_rtl_x08_v4fx.f board_rtl_x01_v4fx_ncv.f board_rtl_x04_v4fx_ncv.f board_rtl_x08_v4fx_ncv.f board_rtl.f	List of files for RTL simulations.
simulate_mti.do	Simulation script for ModelSim.
simulate_ncsim.sh	Simulation script for Cadence IUS.
simulate_vcs.sh	Simulation script for VCS.
xilinx_lib_vcs.f xilinx_lib_vnc.f	Points to the required SmartModel.

Back to Top

simulation/tests

The tests directory contains test definitions for the example test bench.

Table 3-10: **Tests Directory**

Name	Description
<project_dir>/<component_name>/simulation/tests	
pio_tests.v sample_tests1.v tests.v[hd]	Test definitions for example test bench.

Back to Top

Dual Core Example Design

The dual core example design can be used as a starting point for designs with multiple Virtex-5 FPGA PCI Express blocks. The dual core example design provides both Verilog and VHDL source files, simulation scripts, and implementation files necessary to simulate and implement a target design that uses two Virtex-5 PCI Express Blocks.

Although all Virtex-5 FXT FPGAs support multiple Endpoint Blocks, the dual core example design is only generated by the CORE Generator when the PCI Express Endpoint Block Plus for PCI Express is generated using the following device and package combination(s):

Virtex 5 FX70T-FF1136 (XC5VFX70T-FF1136)

Note: The dual core example design may be used as a starting point for Virtex-5 FXT FPGAs and package combinations not defined above; however, some FXT devices do not support multiple 8-lane configurations. See "Chapter 5, Core Constraints" in the *Endpoint for PCI Express Block Plus User Guide* for information about devices that support the 8-lane configuration.

Figure 3-6 illustrates the dual core example design structure. The dual core example design instantiates two Virtex-5 PCI Express Block cores with the same configuration (that is, lane width, BAR configuration, and so forth). Designers using Virtex-5 PCI Express Blocks with different configurations need to generate separate endpoint cores using the CORE Generator and configure each as desired.

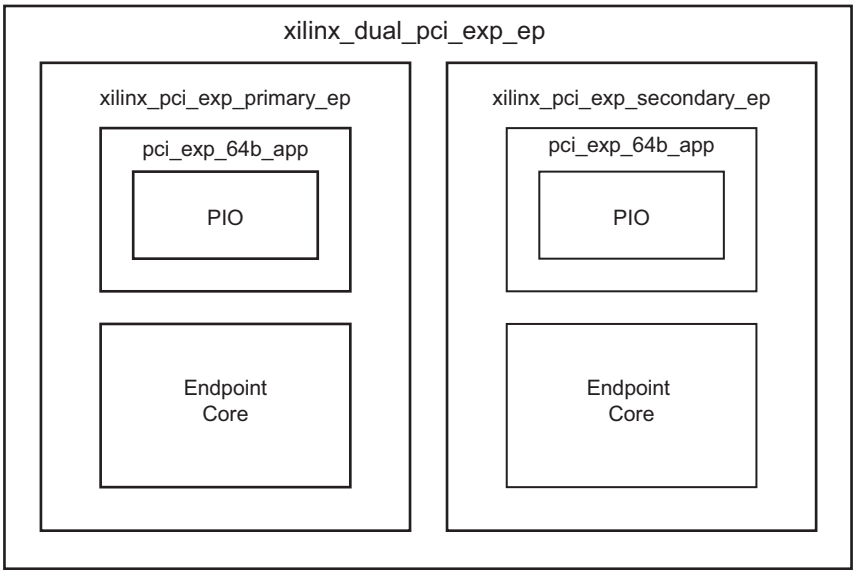

Figure 3-6: **Dual Core Design Block Diagram**

Dual Core Directory Structure and File Contents

When generating the Block Plus core with the Virtex-5 FX70T-FF1136 (XC5VFX70T-FF1136) FPGA, the PIO example design source files and scripts are generated using the directory structure specified in the "Directory Structure and File Contents" section.

For the dual core example design, in addition to the source files, simulation scripts, and implementation files generated in the <project name>/<component name> directory specified by the user, the following directories and associated files are used:

- <component name>/example_design
  Dual core Verilog or VHDL design files
  - example_design/dual_core
    Implementation script files
- <component name>/simulation
  Simulation Verilog or VHDL source files
  - simulation/functional
    Simulation scripts
- <component name>/implement
  Implementation scripts

<component name>/example_design

The example design directory includes the dual core example design ucf, which varies based on the device selected.

Table 3-11: **Example Design Directory**

Name	Description
xilinx_dual_*.ucf	Dual core example design ucf. Varies by lane-width, part, and package selected.

Back to Top

example_design/dual_core

The dual core directory contains the top-level and wrapper files for the dual core example design.

Table 3-12: **Dual Core Directory**

Name	Description
xilinx_dual_pci_exp_ep.v[hd]	Verilog or VHDL top-level dual core PIO example design file for 8-lane, 4-lane, and 1-lane dual cores.
xilinx_pci_exp_primary_ep.v[hd]	Verilog or VHDL wrapper for the PIO example design for 8-lane, 4-lane, and 1-lane primary core.
xilinx_pci_exp_secondary_ep.v[hd]	Verilog or VHDL wrapper for the PIO example design for 8-lane, 4-lane, and 1-lane secondary core.

Back to Top

<component name>/simulation

The simulation directory includes the dual core example design simulation files.

Table 3-13: **Simulation Directory**

Name	Description
board_dual.v[hd]	Top-level simulation module.
xilinx_dual_pci_exp_cor_ep.f	List of files comprising the design being tested.

Back to Top

simulation/functional

The functional directory contains the dual core example design simulation scripts.

Table 3-14: **Functional Directory**

Name	Description
simulate_dual_mti.do	ModelSim simulation script.
simulate_dual_ncsim.sh	Cadence IUS simulation script.
simulate_dual_vcs.sh	VCS simulation script.
board_dual_rtl_x0*.f	List of files for RTL simulations.
board_dual_rtl_x0*_ncv.f	List of files for RTL simulations.

Back to Top

<component name>/implement

The implement directory contains the dual core example design implementation script files.

Table 3-15: **Implement Directory**

Name	Description
implement_dual.sh	Linux implementation script.
xst_dual.scr	XST synthesis script.
xilinx_dual_pci_exp_*_lane_ep_inc.xst	XST project file for 1-lane, 4-lane, and 8-lane example design, respectively.

Back to Top

Simulation and implementation commands for the dual core example design are similar to the single core example design commands. To modify simulation and implementation commands for the dual core example design, see "Simulating the Example Design," page 18, and "Implementing the Example Design," page 19 and replace them with the respective dual core names.

The simulation test bench used with the dual core example design makes use of the downstream port TLP generator from the example design. Because the test bench has only one downstream port, only one of the two PCI Express Block cores can be sent TLPs per simulation run. The downstream port simulates the primary PCI Express Block core by default. To simulate the secondary PCI Express block, you can modify `board.v[hd]`.

Appendix

Additional Design Considerations

Package Constraints

This appendix describes design considerations specific to the Endpoint Block Plus for PCIe core. Table A-1 lists the smallest supported device and interface combinations for the Block Plus core.

Table A-1: **Supported Device and Interface Combinations**

Smallest Supported Device/Part Number	Data Bus Width/Speed	Wrapper File
XC5VLX20T FF323-1	Width: 64-bit Port Speed: 62.5 MHz	xilinx_pci_exp_1_lane_ep.v
XC5VLX20T FF323-1	Width: 64-bit Port Speed: 125 MHz	xilinx_pci_exp_4_lane_ep.v
XC5VLX30T FF665-1	Width: 64-bit Port Speed: 250 MHz	xilinx_pci_exp_8_lane_ep.v

User Constraints Files

The user constraints file (UCF) contains various constraints required for the Block Plus core. The user constraints file must always be used while processing a design and is specific to the target device. Based on the chosen lane width, part, and package, a suitable UCF is created by CORE Generator.

Wrapper File Usage

The wrapper contains an instance of the Block Plus core. When starting a new design, modify this wrapper to include all I/O elements and modules. One of the following files is generated by the CORE Generator based on the chosen lane width.

```
<project_dir>/<component_name>/example_design/xilinx_pci_exp_1_lane_ep.v
<project_dir>/<component_name>/example_design/xilinx_pci_exp_4_lane_ep.v
<project_dir>/<component_name>/example_design/xilinx_pci_exp_8_lane_ep.v
```