Using pre-implemented LogiCORE PCI Interfaces with VHDL and Verilog

March 1997 (Version 1.2ed) Application Note

Summary

This application note details the steps required to implement and simulate LogiCORE PCI Interfaces with VHDL and Verilog.

Xilinx LogiCORE Required

PCI Interface v 1.1
Xilinx CORE Generator
[http://www.xilinx.com/products/logicore/logicore.htm]

Development Tools Required

XACTstep 6.0.1
VIEWlogic Workview Office or Powerview

Table of Contents

Introduction

To accelerate your FPGA-based PCI design, Xilinx has developed a flow and a set of files that will allow you to instantiate the LogiCORE PCI Interface design in your VHDL or Verilog code. This combines the performance and predictability of schematic entry, with the flexibility of HDL. The advantages of this approach are:

- PCI timing compliance is assured through use of a guide file and placement constraints.
- The LogiCORE PCI Interface can be quickly modified in the CORE Generator.
- This approach saves time and effort compared with attempting to meet timing in generic HDL cores.

As a result, you can focus your time and effort on the system level design. This will cut your development time by several months.

PCI CORE Generator

Xilinx has developed a unique and innovative methodology for modifying and using LogiCORE PCI Interfaces. The new CORE Generator for PCI allows the user to easily and quickly parameterize the PCI macro independently of EDA tools. The CORE Generator is available at the Xilinx LogiCORE VIP Lounge:

`http://www.xilinx.com/products/logicore/logicore.htm`

The current Viewlogic based version of the LogiCORE PCI Interface will continue to be available, including source schematics. The previous version of this application note (version 1.1) covered the use of these schematics and the wrapper files. The LogiCORE PCI interface used for the previous version is identical to the one generated by the CORE Generator. As a result, if you started with the Viewlogic/pci_hdl version of the files, you do not have to change your VHDL/Verilog designs to use the new PCI CORE Generator.

This application note is for users of the Xilinx CORE Generator, and can not be used with the version 1.0 LogiCORE PCI Interface module.

Installing the files

Before starting this installation, you should have the PCI Master or Slave already installed in the pcim or pcis directories as detailed in the *LogiCORE PCI Master and Slave Interface User's Guide.*

Go to the PCI VIP Lounge on the Xilinx Web Site [http://www.xilinx.com/products/logicore/logicore.htm]. Log in, then create a customized version of the LogiCORE PCI Interface with the Xilinx CORE Generator. Select and download the appropriate archive file for your computer (xpci.zip for PCs; xpci.tar.gz for Workstations). Copy this file to your pcim or pcis directory. These files will be extracted into a directory called xpci. This installation will include the .xnf files from the LogiCORE PCI Interface. You will need to copy the xpci/xnf/ directory into the desired working directory, either xpci/vhdl or xpci/verilog.

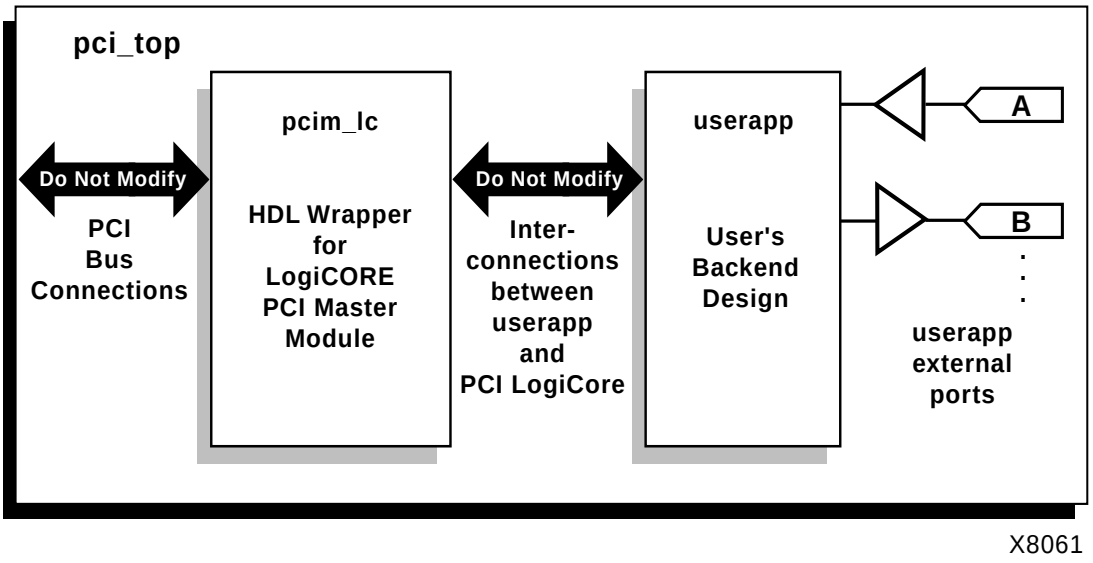

Figure 1: PCI HDL Design Overview

HDL Implementation Methodology

The general methodology for including the LogiCORE PCI Interface in a VHDL or Verilog design is to instantiate the PCI macro wrapper file in your top-level HDL design. The PCI macro wrapper is an interface between the your top level design and the current version 1.1 PCI LogiCORE design. This wrapper, comes in two versions; pcim_lc, which has the initiator (pci_lc_i) instantiated; and pcis_lc, which has the target (pci_lc_t) instantiated. For simplicity, this wrapper will be generically referred to as pcim_lc; if you are doing a target design simply use pcis_lc. During synthesis, a black-box model of the PCI macro wrapper design is made by the synthesis tool, and a “dont_touch” attribute is applied to it. After synthesis, the black-box model is replaced by an .sxnf netlist called pcim_lc.sxnf. This netlist is read by syn2xnf, and used for placing and routing the design. A constraint file, i13p208h.cst, and a guide file, i13p208h.lca, are used during place and route to maintain predictability of the performance of the design.

Simulation follows current existing VHDL and Verilog flows; details on this will be given later in this application note.

The design pci_top is included as an example top-level HDL design. Both VHDL and Verilog versions are included. A sample Synopsys synthesis script is also included, as synopsys.dc. A black-box model of the pcim_lc design is included for both VHDL and Verilog.

In figure 1, an overview of the design is shown. The PCI LogiCORE Master is already instantiated in pcim_lc. Do not change or delete any of the connections marked “Do Not Modify”; these are required for the guide file to work correctly and insure PCI compliant timing. In your script do not “set_port_is_pad” or “insert_pads” on any of the PCI Bus Connections; the PCI LogiCORE Design already has the required I/O structures. Doing so will lead to errors in the XACTstep tools. The ports marked “userapp external ports” should be included in your set_port_is_pad {A, B,...} statement.

Files from the CORE Generator

The following files and directories are included in the xpci directory:

```
4013e.spd
readme.tx
makebits.mbo
config.txt
cst_file/
  i13p208h.cst
  t13p208h.cst
docs/
  pci_hdl.pdf
guide/
  i13p208h.lca
  t13p208h.lca
wrapper/
  pcim_lc.sxnf
  pcis_lc.sxnf
```

```
verilog/
  pci_top.v
  pcim_lc.v
  pcis_lc.v
  userapp.v
  synopsys.dc
vhdl/
  pci_top.vhd
  top_cfg.vhd
  pcim_lc.vhd
  pcis_lc.vhd
  userapp.vhd
  synopsys.dc
xnf/
```

File Details

config.txt
This contains the settings you entered into the CORE Generator.

4013e.spd
Speed file containing the latest speeds for the 4013e device.

makebits.mbo
This contains the suggested settings for the makebits tool.

cst_file/
These constraint files allow the use of the standard LogiCORE PCI Interface timing and placement constraints in this HDL flow.

docs/
The pci_hdl.pdf file is this application note.

guide/
These guide files used by ppr to assure proper placement and routing of critical signals such as FRAME- so that they meet timing.

wrapper/
The .sxnf file is the wrapper that ties this flow together. It eliminates the necessity of determining how to interface each signal to design. Use the pcim_lc.sxnf for a master design; use the pcis_lc.sxnf for a target.

verilog/
This directory contains files needed by Verilog users:
pci_top.v - Example top-level HDL design module
pcim_lc.v - Black-box PCI Interface module (Master)
pcis_lc.v - Black-box PCI Interface module (Slave)
userapp.v - Template for back-end module
synopsys.dc - Synopsys synthesis script

vhdl/
This directory contains files needed by VHDL users:
pci_top.vhd - Example top-level HDL design module
top_cfg.vhd - Example top-level HDL design configuration
pcim_lc.vhd - Black-box PCI Interface (Master design)
pcis_lc.vhd - Black-box PCI Interface (Slave design)
userapp.vhd - Template for back-end module
synopsys.dc - Synopsys synthesis script

xnf/
This directory contains the xnf files created by the CORE Generator.

HDL Simulation Methodology

VHDL Simulation with Synopsys

In the current VHDL Xilinx Synopsys Interface (XSI) flow, there are two types of VHDL simulation available to customers: functional and timing. Timing simulation is a fully developed flow in XSI with Synopsys' VSS tool.

Functional simulation, will be supported in the April release of the CORE Generator. It will generate a functional model that you can use to functionally simulate the LogiCORE PCI module.

Verilog Simulation with Verilog-XL

In the Xilinx Synopsys/Verilog XL flow, both timing and functional simulation will be fully supported in the April release of the CORE Generator.

Using the LogiCORE PCI Interface with VHDL

Implementation

To use the LogiCORE PCI Interface in a VHDL design flow follow these steps. If using the target design, use pci_lc_t instead of pci_lc_i. The

- Analyze all your VHDL files. Run these commands from the pcim/pci_hdl/vhdl directory:
  `analyze -f vhdl userapp.vhd`
  `analyze -f vhdl top.vhd`
- `elaborate top`
- `set_dont_touch pcim_lc`
- `set_port_is_pad {A, B,...userapp external ports}`
- `insert_pads`
- (user's timespecs here)
- `compile`
- `replace_fpga`
- `write -f xnf -h -o pci_top.sxnf`
- Synopsys writes out a file called pcim_lc.sxnf. Delete this file, then copy pcim_lc.sxnf from the wrapper/ directory into your current working directory.
- `syn2xnf -f -d xnf pci_top.sxnf`
- Implement the design in the XACTstep tools, using the included constraint and guide files as directed in the *LogiCORE PCI Master and Slave Interface User's*

Guide.

Functional Simulation

Functional simulation will be supported through the VHDL model that will be generated at the same time the CORE Generator creates your xnf file. This model can be imported into the VHDL SImulator of your choice. This and the steps to do this will be available in the April release of the CORE Generator.

Timing Simulation

After implementation, the timing simulation flow for XSI is as follows:

- `xdelay -dw pci_top.lca`
- `lca2xnf pci_top.lca`
- `xnf2vss pci_top.xnf`, which will produce a pci_top_vss.vhd file and pci_top.sdf file.
- `vhdlan top.cfg.vhd` which links the architecture inside pci_top_vss.vhd to the entity in the pci_top.vhd file.
- `vhdlan pci_top_vss.vhd`
- vhdlan on your testbench VHDL file

Using the LogiCORE PCI Interface with Verilog

Implementation

To use the LogiCORE PCI Interface in a Verilog design flow, follow these steps. If using the target design, use pci_lc_t instead of pci_lc_i.

- Analyze all your Verilog files. Run these commands from the pcim/pci_hdl/verilog directory:
  `read -f verilog userapp.v`
  `read -f verilog top.v`
- `set_port_is_pad {A, B,...userapp external ports}`
- `insert_pads`
- (user's timespecs here)
- `compile`
- `replace_fpga`
- `write -f xnf -h -o pci_top.sxnf`
- Synopsys writes out a file called pcim_lc.sxnf. Delete this file, then copy pcim_lc.sxnf from the wrapper/ directory into your current working directory.
- `syn2xnf -f -d xnf pci_top.sxnf`
- Implement the design in the XACTstep tools, using the included constraint and guide files, as directed in the *LogiCORE PCI Master and Slave Interface User's Guide*.

Functional Simulation

Functional simulation will be supported through the Verilog model that will be generated at the same time the CORE Generator creates your xnf file. This model can be imported into the Verilog SImulator of your choice. This and the steps to do this will be available in the April release of the CORE Generator.

Timing Simulation

The timing simulation flow for Verilog XL is as follows:

- `xmake pci_top.sxnf`
- `xdelay -dw pci_top.lca`
- `timenet -x pci_top.lca`
  or
- `timenetx -x pci_top.lca`
- `verilog +neg_tchk +splitsuh pci_top.stim pci_top.v`

"pci_top.stim" is the testbench file. "pci_top.v" is the structural verilog file created by timenet/timenetx.

Note that it is critical to specify the -x option in timenet/timenetx so that xnfba does not restore the original SCHNM properties.

Additional Information

If you have questions or need more information about PCI or the Xilinx CORE program, please visit WebLINX, our web pages at:

`http://www.xilinx.com/products/logicore/logicore.htm`

or e-mail the Xilinx technical support hotline:

`mailto:hotline@xilinx.com`

You can also reach our hotline at 1-800-255-7778.

XILINX® The Programmable Logic Company℠

Headquarters

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
U.S.A.

Tel: 1 (800) 255-7778
or 1 (408) 559-7778
Fax: 1 (800) 559-7114

Net: hotline@xilinx.com
Web: http://www.xilinx.com

North America

Irvine, California
(714) 727-0780

Englewood, Colorado
(303)220-7541

Sunnyvale, California
(408) 245-9850

Schaumburg, Illinois
(847) 605-1972

Nashua, New Hampshire
(603) 891-1098

Raleigh, North Carolina
(919) 846-3922

West Chester, Pennsylvania
(610) 430-3300

Dallas, Texas
(214) 960-1043

Europe

Xilinx Sarl
Jouy en Josas, France
Tel: (33) 1-34-63-01-01
Net: frhelp@xilinx.com

Xilinx GmbH
Aschheim, Germany
Tel: (49) 89-99-1549-01
Net: dlhelp@xilinx.com

Xilinx, Ltd.
Byfleet, United Kingdom
Tel: (44) 1-932-349401
Net: ukhelp@xilinx.com

Japan

Xilinx, K.K.
Tokyo, Japan
Tel: (03) 3297-9191

Asia Pacific

Xilinx Asia Pacific
Hong Kong
Tel: (852) 2424-5200
Net: hongkong@xilinx.com

© 1996 Xilinx, Inc. All rights reserved. The Xilinx name and the Xilinx logo are registered trademarks, all XC-designated products are trademarks, and the Programmable Logic Company is a service mark of Xilinx, Inc. All other trademarks and registered trademarks are the property of their respective owners.

Xilinx, Inc. does not assume any liability arising out of the application or use of any product described herein; nor does it convey any license under its patent, copyright or maskwork rights or any rights of others. Xilinx, Inc. reserves the right to make changes, at any time, in order to improve reliability, function or design and to supply the best product possible. Xilinx, Inc. cannot assume responsibility for the use of any circuitry described other than circuitry entirely embodied in its products. Products are manufactured under one or more of the following U.S. Patents: (4,847,612; 5,012,135; 4,967,107; 5,023,606; 4,940,909; 5,028,821; 4,870,302; 4,706,216; 4,758,985; 4,642,487; 4,695,740; 4,713,557; 4,750,155; 4,821,233; 4,746,822; 4,820,937; 4,783,607; 4,855,669; 5,047,710; 5,068,603; 4,855,619; 4,835,418; and 4,902,910. Xilinx, Inc. cannot assume responsibility for any circuits shown nor represent that they are free from patent infringement or of any other third party right. Xilinx, Inc. assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction if such be made.