

Designing a PCI Target/Initiator in FPGAs



Xilinx Inc.



Brad Fawcett & Steve Knapp

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
E-mail: pci@xilinx.com

Current Activities:

Brad Fawcett is currently a Manager in the Applications Engineering group at Xilinx Inc. Among his many duties, Brad is the editor of the XCell Journal, the Xilinx user newsletter.

Steve Knapp holds the title of Vertical Applications Manager. Steve is one of the designers of the PCI LogiCore Interface

Authors' Backgrounds:

In his nine years with Xilinx Inc., Brad Fawcett has held a variety of applications engineering, technical marketing, and technical training positions. A veteran of 20 years in the industry, Brad holds Bachelor and Master of Science degrees in Electrical Engineering from the University of Illinois.

A veteran of ten years at Xilinx, Steve Knapp also has application engineering experience at Intel Corp., and holds a Bachelor of Science degree in Material Science from the Massachusetts Institute of Technology.

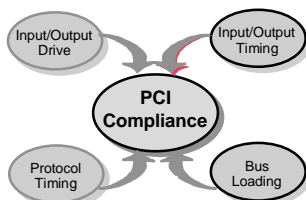
Designing a PCI Target/Initiator in FPGAs

The PCI LogiCore™ Interface

Agenda

- Introduction
- How was it done?
- How was it tested?
- How is it used?
- User benefits
- User experiences

PCI Technical Challenges



- 100% compliance required for:
 - Add-in boards to be plugged into any system
 - Systems that accept any PCI board
- “Follow the rules and it will work”

PCI Basics

- PCI = Peripheral Component Interconnect
- A high-performance bus interface
 - 33 MHz or 66 MHz
 - 32-bit or 64-bit
 - 5V or 3V
- Uses “reflective-wave” signaling

The PCI LogiCore Interface

The Challenge: Design a 32-bit, 33 MHz PCI Interface in an FPGA as a Re-Usable Core

- Why a PCI core?
 - Market demand
 - Difficult, time-consuming design
- Why an FPGA?
 - Time to market, no NREs, etc.
 - Flexibility for spec. changes & interpretations
 - Integration of bus interface with application-specific logic

FPGA Device Requirements

- PCI SIG electrical compliance checklist
- Sufficient capacity
 - Integrate PCI interface & “back-end” logic
 - Include FIFO buffers (on-chip memory)
- Sufficient performance
 - Support burst transfers
 - Meet set-up and hold time requirements
- Appropriate architectural features
 - Multiple 3-state output enables
 - On-chip bussing

Choices: XC4013E-2PQ208 (Target)
XC4013E-1PQ208 (Initiator)

Development System Requirements

- “Hard” macro for guaranteed performance
 - Schematic based (Viewdraw)
 - Instantiable in HDL
 - Floorplanner
 - Relationally-placed macros
 - Guide files
 - Timing-driven place & route
- } (XACTstep)
- Verification tools
 - Logic and timing simulation (Viewsim & VSS)
 - Static timing analysis (XDelay)
 - Protocol verification (VirtualChips)

LogiCore PCI Interface Module

- Fully-integrated, tested, and validated PCI interface
 - 32-bit interface compliant with version 2.1
 - Target/initiator and target-only versions
 - “Generic” back-end interface
 - Customizable
 - Verified with VirtualChips VHDL PCI simulation model
- Supports basic PCI functions
 - Type 0 configuration space
 - Memory reads and writes
 - I/O reads and writes

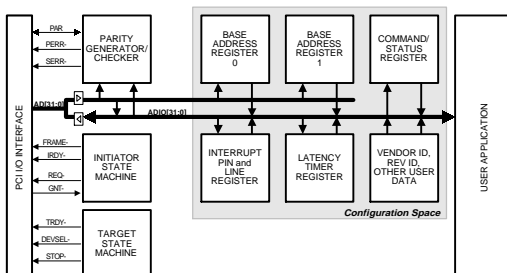
Designing a PCI Target/Initiator in FPGAs

The PCI LogiCore Interface

Agenda

- Introduction
- How was it done?
- How was it tested?
- How is it used?
- User benefits
- User experiences

LogiCore PCI Interface Block Diagram



Meeting Performance Goals

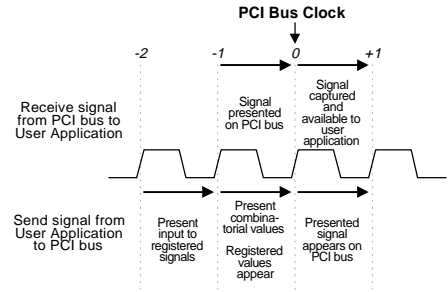
- HDL solution preferred by designers
- Current HDLs lack good placement, partitioning controls
- Solution: Specify design in VIEWlogic schematic
- Design (netlist) can be instantiated using HDLs
- XACTstep provides additional capabilities
 - “Guide” files to guarantee performance on critical paths
 - “XACT-Performance” timing-driving place and route
 - Floorplanner tool for pre-placing structured elements

Meeting I/O Performance Goals

7 ns Setup, 11 ns Clock-to-Output

- **Solution:** XC4000E input/output flip-flops provide guaranteed PCI-compliant pin-to-pin timing
- **Complication:** Input/output flip-flops add one cycle of latency (pipeline delay)
 - Some paths cannot tolerate the latency and still meet PCI compliance
 - Complicates target and initiator state machine design

Interface Pipelining



State Machine Design

- Based on PCI Spec., Appendix B equations
- “One-Hot” state encoding
 - Closely matches XC4000E architecture
 - Reduces fan-in, increases performance
- Match to application
 - Target with slow DEVSEL# speed
 - No address stepping
 - Support “user application”
- Predictive decoding

Problem: PCI-Compliant 100% Burst Transfers

- Receiving data at 100% burst is no problem
 - 7 ns setup time guaranteed using input flip-flops
- Sending data at 100% burst while maintaining 100% compliance is a problem
 - Follow PCI transaction rules (Appendix C, Rule 2c., 16)
 - Monitor IRDY# or TRDY# and switch 36 outputs with 7 ns setup and 11 clock-to-out
 - Darn near impossible with today’s programmable logic
 - Various “tricks” offer 100% burst but violate full compliance

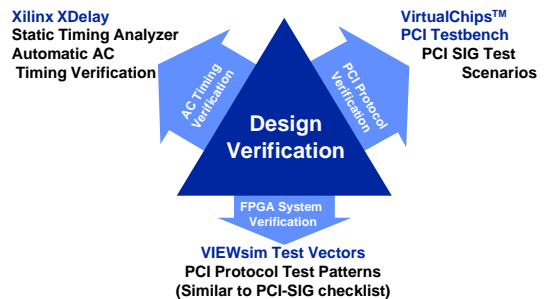
Designing a PCI Target/Initiator in FPGAs

The PCI LogiCore Interface

Agenda

- Introduction
- How was it done?
- **How was it tested?**
- How is it used?
- User benefits
- User experiences

Design Verification

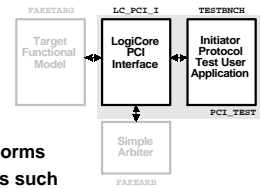


PCI Protocol Testing

- VirtualChips VHDL bus simulation model using Synopsys VSS simulator
- VIEWsim digital simulator with Xilinx-created testbench
 - Target functional model
 - VIEWsim command files that match PCI-SIG compliance test scenarios
 - Created extra Target test scenario to test target terminations (not covered in PCI-SIG checklist)
- Cross-checked results between VirtualChips and VIEWsim
- Results summarized in “*LogiCore PCI Protocol Checklist (v2.1)*”

VIEWsim PCI Protocol Testbench

- TESTBNCH.1 contains an example user application for performing Initiator protocol testing



- Target function model performs “unnatural acts” on the bus such as presenting incorrect parity, etc.
- Simple arbiter asserts GNT# after the LogiCore interface asserts REQ#. GNT# asserted for two cycles then de-asserted until next transaction

Designing a PCI Target/Initiator in FPGAs

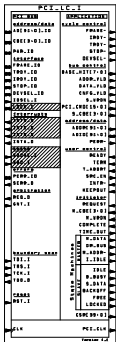
The PCI LogiCore Interface

Agenda

- Introduction
- How was it done?
- How was it tested?
- How is it used?
- User benefits
- User experiences

Customizing the LogiCore PCI Macro

- Addition of user application's unique back-end logic
- Options for customizing PCI bus interface
 - Implemented by changing schematic symbols
 - Main option: target-only or initiator/target
- Edit Configuration Space Header
 - Read/write registers implemented in CLB flip-flops
 - Read only registers implemented in LUTs
- Enable/disable latency timer & pipelining
 - Only used for burst transactions

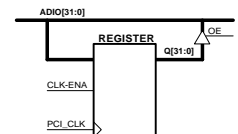


Back-End User Interface

- 32-bit multiplexed address/data bus
- 32-bit latched address
- PCI control/status signals
- User controls (e.g., Ready, Terminate, and Interrupt)
- State bits of bus control state machines

Back-End Logic

- Employ read/write registers for transferring data



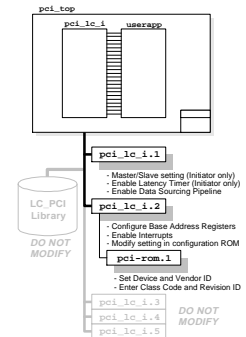
- All bus control signals & datapaths are pipelined
- Synchronous design techniques recommended
 - All signals should be registered
 - Increases performance and facilitates timing analysis

FIFO Buffers

- Needed to support burst transactions
 - Between LogiCore interface & back-end device
- Dual-port FIFOs easy to create using synchronous, dual-port distributed memory option in XC4000E/EX FPGAs
- LogiCore macro incl. a 16 x 32 read/write FIFO
 - Used in verification testing
- Separate dual-port FIFOs to support bus reads and writes recommended

LogiCore PCI Design Structure

- Most of the LogiCore PCI design in the LC_PCI library
- Only pci_lc_i.1 and .2 should be modified
- Designer should put user application in userapp "wrapper" schematic
- Keep all top-level names to make guide files work



Configuration Space Header

Device ID	Vendor ID	60h
Status	Command	64h
Class Code	Rev ID	68h
BIST	Header Type	6Ch
	Latency Time	6Ch
	Cache Line Size	6Ch
	Base Address Register 0 (BAR0)	10h
	Base Address Register 1 (BAR1)	14h
	Base Address Register 2 (BAR2)	18h
	Base Address Register 3 (BAR3)	1Ch
	Base Address Register 4 (BAR4)	20h
	Base Address Register 5 (BAR5)	24h
	Cardbus CSE Pointer	28h
Subsystem ID	Subsystem Vendor ID	2Ch
	Reserved	2Ch
	Expansion ROM Base Address	30h
	Reserved	30h
Max_Lat	Min_Gnt	34h
	Interrupt Pin	34h
	Interrupt Line	3Ch

- LogiCore module implements first 64 bytes of Type 0 CSH
- Symbols to customize Base Address Register size and modes
- Load hex table to specify read-only register contents (Vendor ID, etc..)

3 Steps to a Complete PCI Design

- Build the Target-Side Interface
 - Required in every PCI design
- Build the Initiator-Side Interface
 - Only required in Target/Initiator designs
- Build in Burst Support
 - Only required if supporting burst transfers
 - Possibly four separate design steps, depending on the application:
 - Target Read
 - Target Write
 - Initiator Read
 - Initiator Write

Step 1: Build a Target Interface

- Required in every target & every initiator design
- Configure the Base Address Registers (BARs)
- Edit the read-only values in Configuration Space Header (CSH) ROM
- Build interface to the read/write Target locations in the user application
- Decide how to force Target Termination conditions, if required

Target Termination Conditions

- Target can initiate various termination conditions:
 - **Normal Termination**: "Everything was perfect"
 - **Target Retry**: "Come back later. I can't respond just now." Can only be used on first transfer cycle.
 - **Target Disconnect with Data**: "I can't complete the transaction, but I'll give you what I have. Come back later for the rest."
 - **Target Disconnect without Data**: "I gave you everything that I can give you right now. You might try again later."
 - **Target Abort**: "Big trouble! You may have tried something illegal or I'm dead!"
- Uses READY and TERM signals sourced from back-end logic

Step 2: Build an Initiator Interface

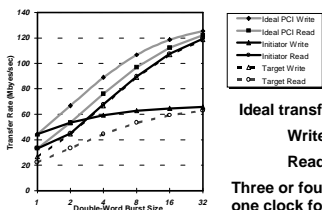
- Build the Initiator state machine
 - Drive the REQUEST and COMPLETE signals
 - Size and direction of data transfers?
 - How to handle any target termination conditions?
 - How to arbitrate between incoming Target accesses and pending Initiator transactions?
- Build the interface to drive the starting address on ADIO[31:0]
- Build interface to the read/write Initiator locations in the user application

Address Counters

- Only starting address is broadcast during a transaction
 - Broadcast during the address phase along with the PCI bus command on CBE[3:0]
- User application must keep track of current address during burst applications
 - Requires a loadable, binary counter large enough to cover desired address range.
- Example: Target has 4K address space. Needs 10-bit binary counter. Lower two bits are zero (32-bit transfers). Upper 20 bits can be register (not required if not used in the user application)



LogiCore PCI Interface Performance



Ideal transfer rates:
Write: 3-1-1-1
Read: 4-1-1-1

Three or four clocks to 1st data & one clock for each burst transfer

LogiCore module (~2 speed grade):

Initiator Read: 4-1-1-2

Initiator Write: 3-2-2-2

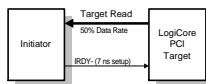
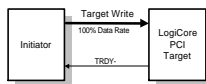
Target Read: 6-2-2-2

Target Write: 5-1-1-1

Step 3: Burst Transfer Support

- Can be up to four different interfaces
 - Read and write directions
 - Target and Initiator sides
- Build address counter and associated control logic
- Provide pipelined source data using SRC_EN signal
- Build FIFOs to support the specific application needs
- Build COMPLETE logic and transfer control

Automatic Wait-State Insertion



- LogiCore receives data at 100% burst and sends data at 50% burst
- IRDY- or TRDY- wait-states automatically inserted on burst transfers where the LogiCore interface is providing data
- LogiCore Initiator also inserts IRDY- wait-state before de-asserting FRAME- during a burst transfer
- Automatic wait-states never asserted on single data transfers

Designing a PCI Target/Initiator in FPGAs

The PCI LogiCore Interface

Agenda

- Introduction
- How was it done?
- How was it tested?
- How is it used?
- User benefits
- User experiences

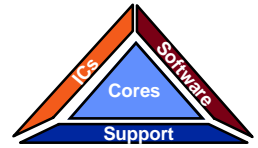
Benefits of Core-Based Design

- Decreased development time & effort
 - Allows focus on proprietary portion of design
 - Access “system expertise” of core’s designers
- Decreased development risk
 - Predictable functionality and performance
- Optimization for best device utilization

Core-Based Design Requirements

Silicon, Software, Service

- Predictable functionality and performance
- Optimized for best device utilization
- Accessible system expertise
- Supporting tools and collateral



But It's Not a Panacea ...

- PCI is difficult, even in ASICs
- Use of LogiCore PCI module requires
 - Understanding of PCI protocols
 - High-performance, pipelined design techniques
 - Use of guide files, TIMESPECS, floorplanner

Designing a PCI Target/Initiator in FPGAs

The PCI LogiCore Interface

Agenda

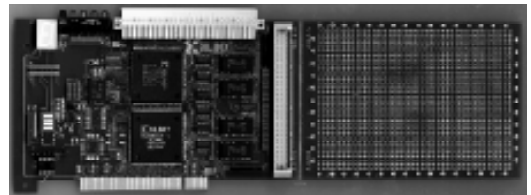
- Introduction
- How was it done?
- How was it tested?
- How is it used?
- User benefits
- User experiences

User Experiences

- LogiCore PCI interface already used by > 150 designers (as of 9/96)
- Examples:
 - XC6200 co-processor development board
 - Digital audio broadcast transceiver
 - High-performance graphics system
 - Networking sub-system interface

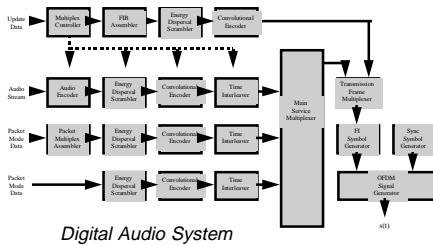
Xilinx XC6200 Co-Processor Development Board

- XC4013E as PCI Target interface to memory-mapped XC6200 co-processor



Digital Audio Transceiver

- XC4013E as PCI Target interface



Digital Audio Transceiver

“We chose Xilinx modules because they’re fully verified. As a result, the module worked the first time, saved us over six months in development time, and allowed us to focus our resources on our system design.”

Brian Warren, Senior Design Engineer,
Delco Electronics

High-Performance Graphics System

- XC4020E as PCI-PCI bridge
- Integrates bridge and support logic
- Unsupported use and unsupported device
- Implemented in Xilinx HardWire for high-volume production
- Helped to locate and repair flaw in another PCI ASIC

Networking Sub-System Interface

- XC4013E as PCI Initiator plus interface to network communication device
- Includes integrated data FIFO
- Embedded PCI application operating at 25 MHz

Interesting Things We Learned

- PCI specification open to interpretation in some areas
 - Gaps in PCI SIG compliance checklist
- Many available PCI interface designs for programmable logic are not fully compliant
 - Ignore wait states from receiving agent during burst
- PCI designs require a lot of technical support
 - Significant learning curve for designers

Summary

- PCI interface designs are difficult
- High-performance FPGA designs require floorplanning, timing & placement constraints, and re-entrant place & route tools
- Use of pre-designed cores accelerates design cycles and reduces design risk
 - Designer must understand application, FPGA tool use

Equipment Used

- **Platforms:** Sun workstations and PCs
- **Tools:**
 - Viewlogic schematic editor & simulator
 - XACTstep development system
 - FPGA implementation tools
 - XDelay static timing analyzer
 - Hardware debugger
 - Synopsys VSS simulator
 - VirtualChips PCI model

Recommended Resources Available Literature

- **PCI -SIG publications**
 - PCI Local Bus Specification
 - PCI Compliance Checklist
 - PCI System Design Guide
- **Xilinx publications**
 - LogiCore PCI Master and Slave User's Guide
 - LogiCore PCI Interface Protocol Compliance Checklist
 - XC4000 FPGA Series Product Specification
 - Implementing FIFOs in XC4000E Application Note

PCI-SIG
P.O. Box 14070
Portland, OR 97124
Tel: 1-800-433-5177

Recommended Resources Available Literature - Books

- **PCI System Architecture**
by Tom Shanley and Don Anderson
Mindshare Press
2202 Buttercup Dr.
Richardson, TX 75082
Tel: 214-231-2216
- **PCI Hardware and Software Architecture & Design**
by Edward Solari and George Willse
Annabooks
11848 Bernardo Center Dr., Suite 110
San Diego, CA
Tel: 800-462-1042



Recommended Resources Web Sites

- **Xilinx WebLINX PCI pages:**
LogiCore:
<http://www.xilinx.com/logicore/logicore.htm>
General PCI:
<http://www.xilinx.com/apps/pci.htm>
- **PCI-SIG**
<http://pcisig.com>

Recommended Resources

- **Design Consultants: HighGate Design Inc.**
12380 Saratoga/Sunnyvale Rd., Suite 8
Saratoga, CA 95070
Tel: 408-255-7160
E-mail: highgate@highgatedesign.com
- **PCI Bus Simulation Model: VirtualChips**
2107 N. First St., Suite 100
San Jose, CA 95131
Tel: 888-482-4477
E-mail: sales@vchips.com
Web: <http://www.vchips.com>