Fully Compliant PCI Interface in an XC3164A-2 FPGA

January 1995 Application Note

Summary

This application note describes an XC3164A-2 design for a PCI-compliant interface. This implementation uses conservative design practices to guarantee the critical timing paths. The design was created and simulated using Verilog.

Xilinx Family

XC3100A

Demonstrates

Verilog-based design entry

High-performance design techniques

Table of Contents

Introduction

The high intrinsic speed of the XC3100A-2 family makes it possible to implement a fully compliant Peripheral Component Interconnect (PCI) solution.

PCI has very stringent timing requirements. This application note uses proven, conservative FPGA design principles to guarantee real-word performance. Beware of other application notes that only implement a PCI paper design.

This application note describes a PCI target interface to a bank of SRAMs shared with a local microprocessor. Critical portions of the design are analyzed in detail to illustrate the development process. The PCI target design was coded in Verilog HDL and verified on the PC using the Simucad Silos/Verilog HDL simulator. The design was synthesized to an XC3164A-2PQ160 using Exemplar Logic's CORE logic synthesis package. The Xilinx XACT® 5.0 place and route and static timing analysis back-end tools were used to route the design and to verify that PCI timing constraints had been met.

Implementing high-performance designs in an FPGA generally requires a higher-than-normal degree of pipelining to achieve the highest performance. In the PCI

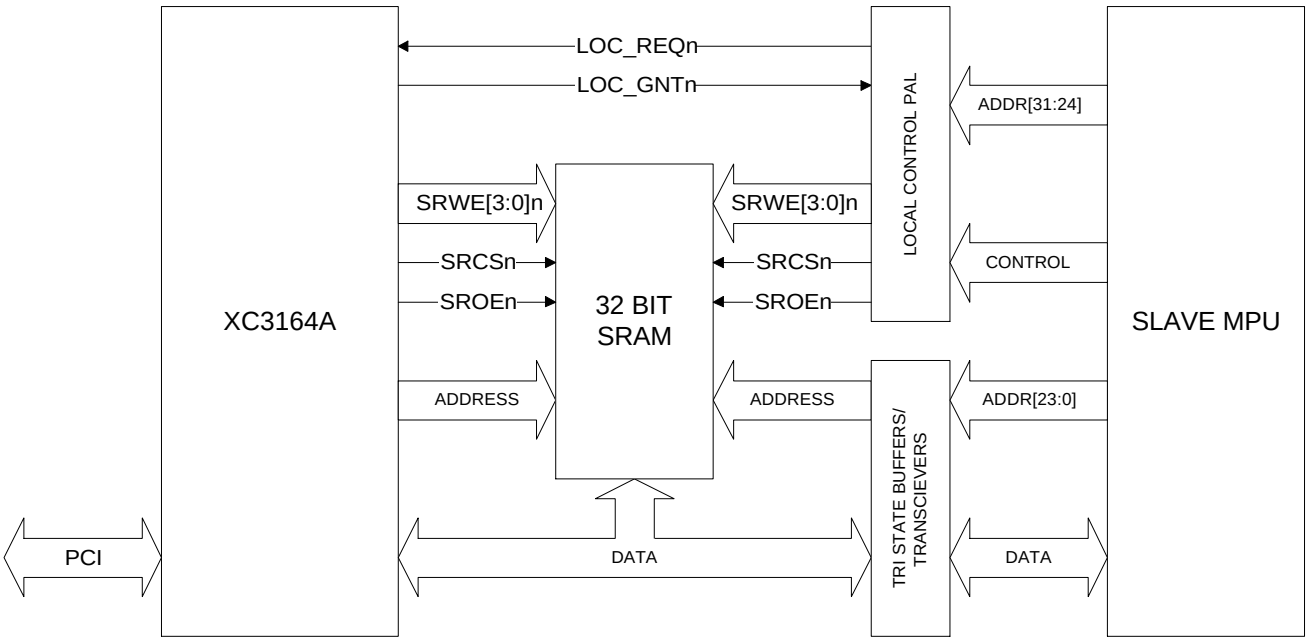

Figure 1. System block diagram.

target application, the pipelining increases the initial access latency but produces a design that performs burst transfers at 33 MHz.

System Description

The example design is an interface to a bank of static RAM shared with an on-board host microprocessor, as shown in Figure 1. This is typical for an intelligent peripheral controller where the SRAM is used to communicate with a PCI-based host.

The static RAM eases the FPGA design requirements because the required PCI Configuration Space Registers (CSR) are mapped onto the lowest 256 bytes of the SRAM. In operation, the on-board microprocessor loads the CSR register block as part of its boot process. While a separate local control EPLD is shown, it could also be incorporated into the FPGA, which has eleven I/Os and 60% of the Configuration Logic Blocks (CLBs) unused. In addition, some of the high-order SRAM address pins (SR_ADDR) can be eliminated if more pins are needed for additional functionality.

The PCI target interface supports single and burst memory and configuration space transfers into the SRAM at a 33-MHz rate. Timing on the SRAM side of the interface is not as critical as that for PCI. SRAMs with an access time of 15 ns are required in order to operate at a 30 ns cycle time. The balance of the cycle is consumed by IOB output propagation delay and input setup time. The Static RAM Chip Select (SRCSn) output pulse is carefully placed to ensure that write cycle setup and hold requirements are met. Timing constraints assure that this critical signal is properly placed and routed. Relaxing the burst transfer rate to two clocks per transfer eases the SRAM timing requirements, permitting lower cost SRAMs.

The design includes a two-way round-robin arbiter with a simple request/grant protocol. The microprocessor subsystem asserts LOC_REQn when it requires access to the SRAM, and is allowed to enable its address, data, and control signal drivers as soon as it receives LOC_GNTn asserted. To provide for buffer turnaround, the arbiter always goes through a dead state when granting access to the SRAM. Because the arbiter is round-robin, neither side can exclude the other, provided the local request is de-asserted at the end of each transfer. The internal PCI_REQ (for SRAM) signal meets this requirement. At power up, the local subsystem keeps its request asserted, and thus excludes PCI masters from accessing the SRAM until the CSR register addresses are initialized.

Block Diagram Description

Figure 2 shows the PCI target interface block diagram. Major functional sub-modules include PARITY, PIPCON, CRITPATH, and ARBITER.

Registered input and output buffers are used on both the PCI and SRAM ports of the PCI Target Controller. The resulting pipelined data flow eases timing requirements. For the write burst shown in Figure 3, PCI_AD signals are latched by registered input buffers on one clock edge, and appear on the SRAM data port one clock later. A similarly pipelined data flow occurs in the reverse direction on read cycles, as shown in Figure 6.

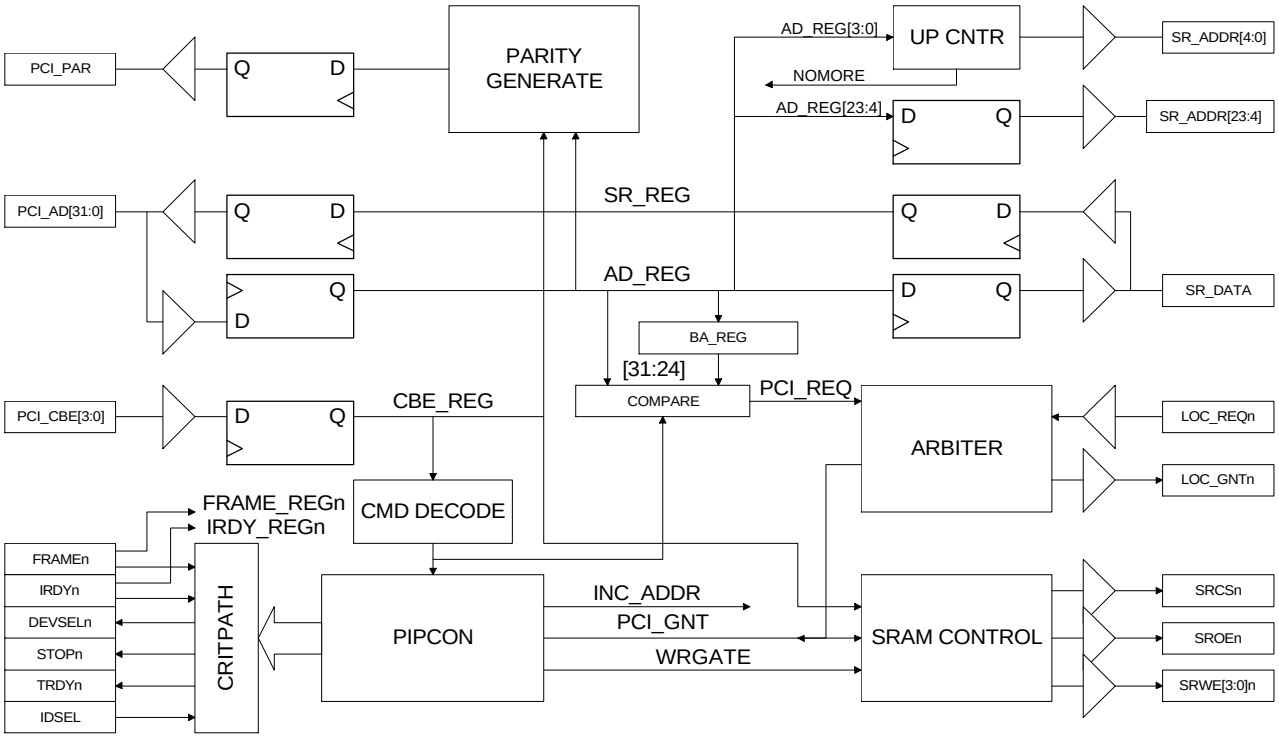

Figure 2. PCI target block diagram.

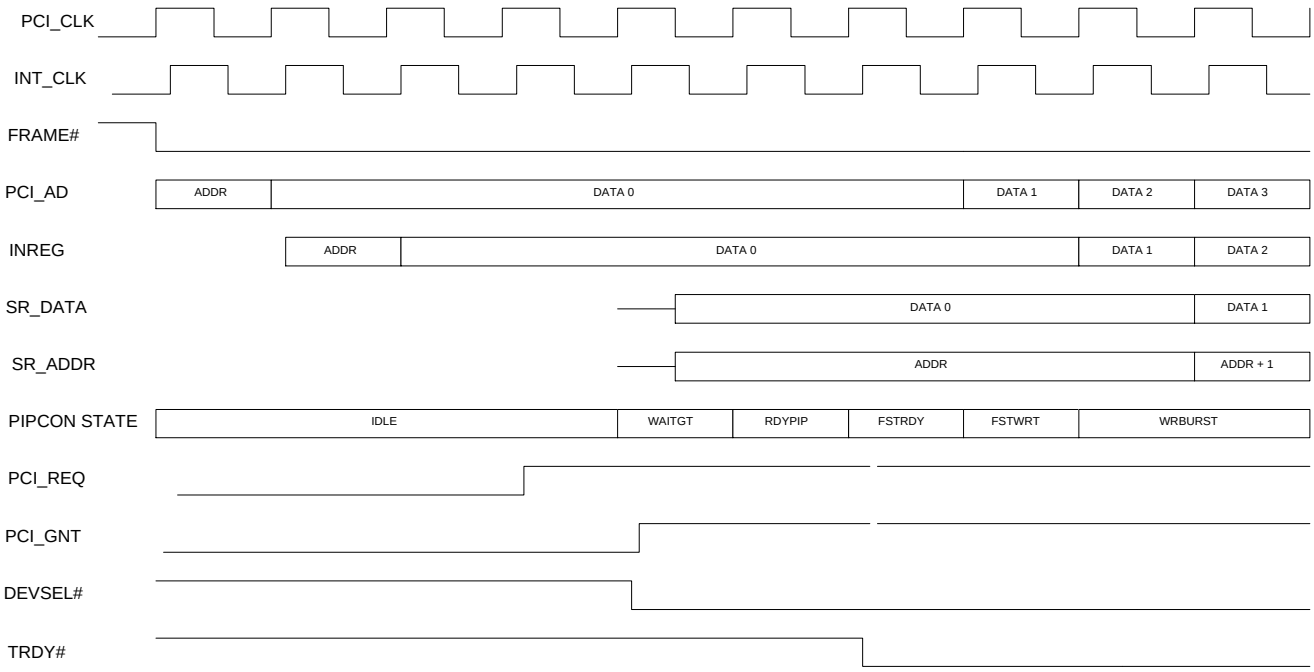

Figure 3. Write-cycle data flow.

AD_REG, the registered PCI_AD input bus, is connected to the Base Address Register and comparator and to the register-counter structure which supplies the SRAM address outputs. The address comparator output is registered as PCI_REQ.

PIPCON, the top-level state machine, sets up the pipeline for continuous data flow. Figure 3 shows its state for each cycle during the initiation of a write burst. PIPCON's outputs control the assertion of TRDYn, the incrementing of the memory address and the SRAM write timing. Its state diagram is shown in Figure 4.

While PIPCON controls the front end of the cycle, CRITPATH handles its termination and interruptions for initiator wait states. The two modules provide a layered solution to the control problem. PIPCON is part of a highly pipelined control structure that cannot react on the same clock edge to PCI control signal transitions. CRITPATH implements a single level of gating on PIPCON outputs, which provides the real-time response required for PCI protocol.

PIPCON state machine outputs are gated by CRITPATH before connection to the D inputs of the output registers for the PCI handshake signals. PIPCON also generates WRGATE and INC_ADDR, which control the pipeline and SRAM. CRITPATH must directly sense PCI signals IRDYn and FRAMEn in order to meet the handshake timing requirements.

CRITPATH includes a single function generator in front of each of the registered output buffers for TRDY#, STOP#, and DEVSEL#, and an internal flip-flop named ENDEDn. The primary inputs to these function generators are direct, unregistered PCI FRAME# and IRDY#. Additional inputs come from the ENDEDn flip-flop and PIPCON. ENDEDn asserts on the clock edge in which FRAME# is High and IRDY# and TRDY# are Low, and asserted until FRAME# is sampled Low to begin another PCI transfer. The same signals, plus ENDEDn, are sensed in each of the other CRITPATH function generators, and are used to de-assert the associated output signals at cycle end. ENDEDn is also used throughout the remainder of the logic to terminate operations at cycle end.

CRITPATH is so named because only 7 ns, including routing delays, are available for its operation. This is one of only two critical timing paths in the entire design. The other is the combinatorial SRCSn logic which controls the SRAM write operation. These turned out to be the only two blocks requiring manual placement and routing.

The PCI Target includes a parity generator circuit. The pipelined data path allows for two clock cycles to meet PCI requirements. In the first cycle, a single layer of CLBs reduces the 32 SR_DATA to seven intermediate registered partial-parity signals. In the next cycle these are XORed with each other and the XOR of the CBEs in two levels of function generators, as the corresponding data is driven onto PCI. On the next clock edge,, the PAR parity signal is driven.

A difficult timing problem for the low order SRAM address outputs was solved by using internal flip-flops to mirror the address output registers. During a burst transfer, the address counter must increment at 33 MHz. A short clock-to-output delay on the output registers is required to maximize the SRAM access time. If the SRAM outputs were fed back through chip input buffers, the closed-loop propagation delay would exceed the cycle time. If internal registers were routed to the output buffers, the clock-to-pad delay would suffer, thus requir-

ing the use of faster SRAMs. To meet both requirements, the address counter was coded as a multiplexer-flip-flop structure. The output of the multiplexer drives both internal CLB flip-flops and the registered outputs. The clock rate is maximized by using the internal flip-flops to generate the next counter state. The clock-to-pad delay is minimized by use of the output registers.

PCI Compatibility

Proof of PCI compatibility includes examining dynamic bus drive, setup and hold times for I/Os, and demonstrating the ability to implement the required functionality at 33 MHz. For example, initiator-induced wait states and burst-termination timing are critical paths in this PCI target interface.

The XC3100A-2 devices have CMOS outputs specified with 8 mA of static output current. PCI dynamic drive requirements are not covered by the original data sheet parameters in the data book. However, the XC3100A-2 family I/Os have been characterized to guarantee PCI compatibility in Plastic Quad-Flat Pack (PQFP) packages.

PCI output timing compliance is easily verified from data sheet parameters. The PCI specification requires that an output be valid in less than 11 ns from the rising CLK edge. XC3100A-2 registered three-state outputs configured for fast slew rate have a maximum propagation delay of 4.0 ns. When the 4.0 ns worst-case internal clock distribution delay is added to this, the specification is met with 3 ns to spare. Routing delays make it extremely difficult to meet this timing using non-registered outputs. However, registered outputs are a natural part of a pipelined data flow architecture, and are readily accommodated in the data path. Unfortunately, this complicates the control path and accounts for at least one clock period of latency.

Determining the PCI timing compatibility of FPGA inputs requires a careful examination of setup and hold time requirements and is not verifiable from standard XC3100A-2 data sheet parameters. PCI compliance requires a setup time of 7 ns and a hold time of 0 ns relative to PCI_CLK. Xilinx guarantees 7 ns setup and 0 ns hold time relative to the external clock for the input flip-flops of the XC3100A-2 family. An alternative to using registered inputs is to use a direct input to a CLB flip-flop located immediately adjacent to the pad. The CLB flip-flop has a setup time of only 1.4 ns to the internal clock. The routing delay to the CLB must compensate for the clock buffer delay. A maximum routing delay of 6.3 ns is permissible. This should be entered using XACT–Performance™ as a pad-to-setup timing constraint for the place and route tool. Both approaches are used in this design to implement registered inputs.

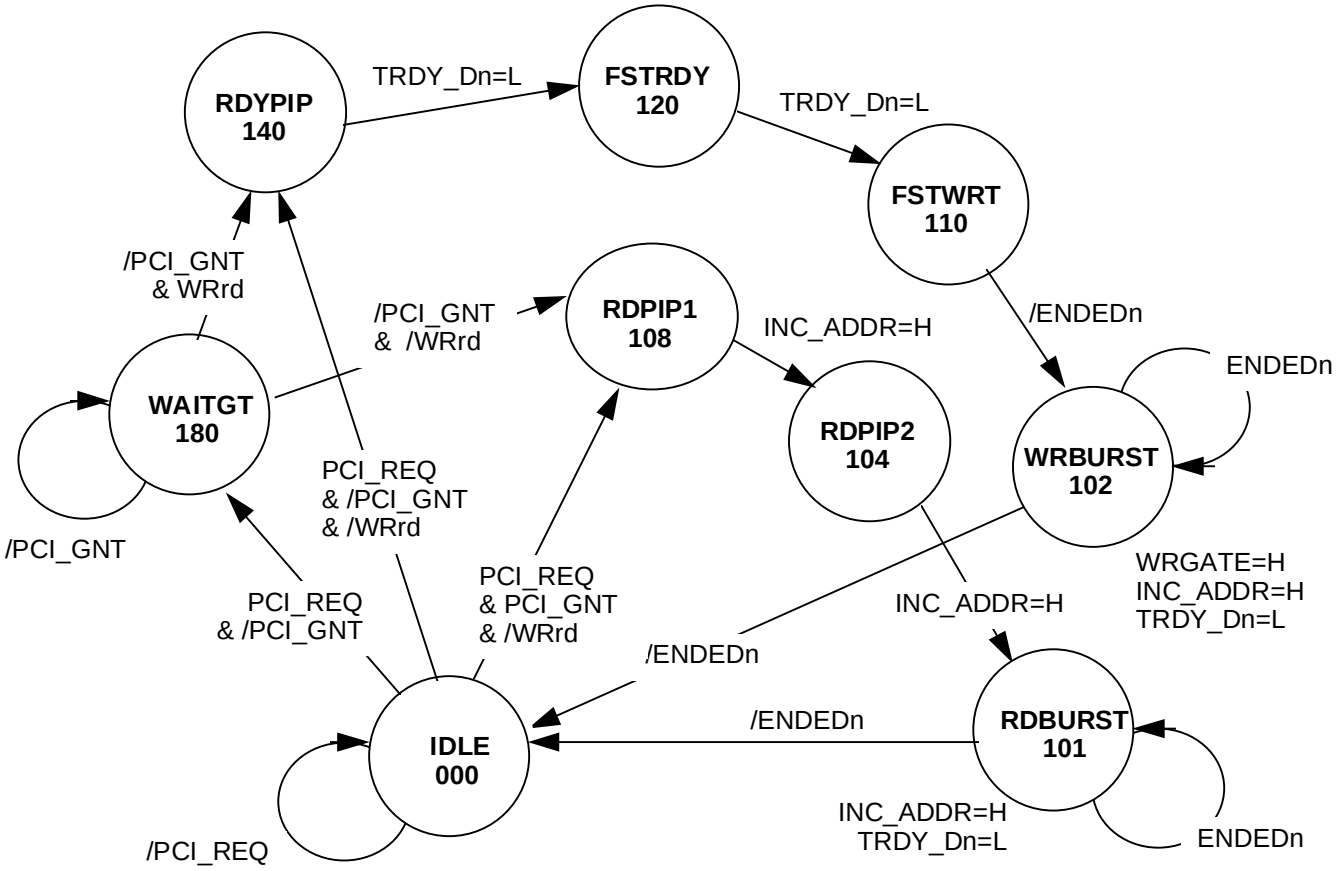

Figure 4. PIPCON State Diagram.

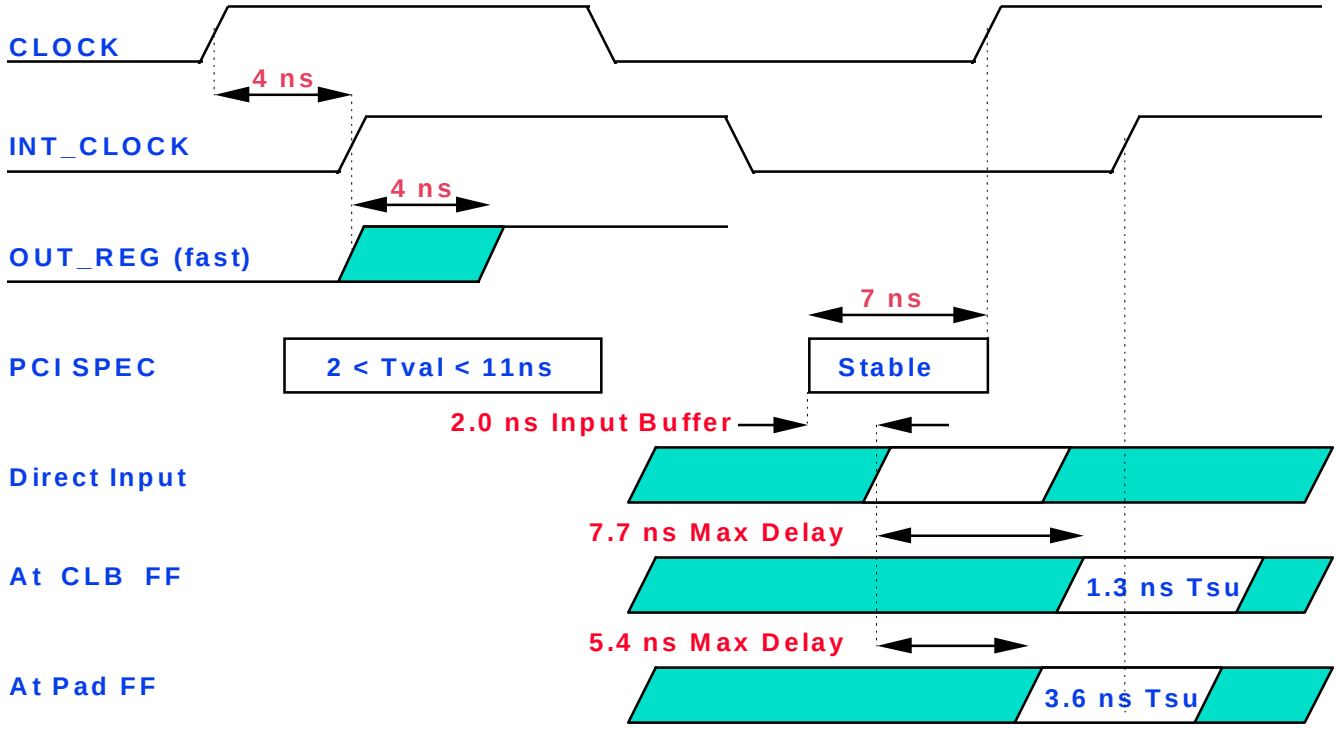

Figure 5. PCI and XC3100A-2 I/O timing.

Critical Paths and Corner Cases

Data Path

The pipelined data path provided by the registered I/Os on both ports of the chip removes all data path timing issues from the critical path.

Handshake

The PCI protocol allows pipelined design implementation, except for the sequence of events which starts on the last transfer of a burst, or when the initiator inserts wait states on a read cycle. To handle this case, PCI signals IRDY# and FRAME#1 must be sensed directly,

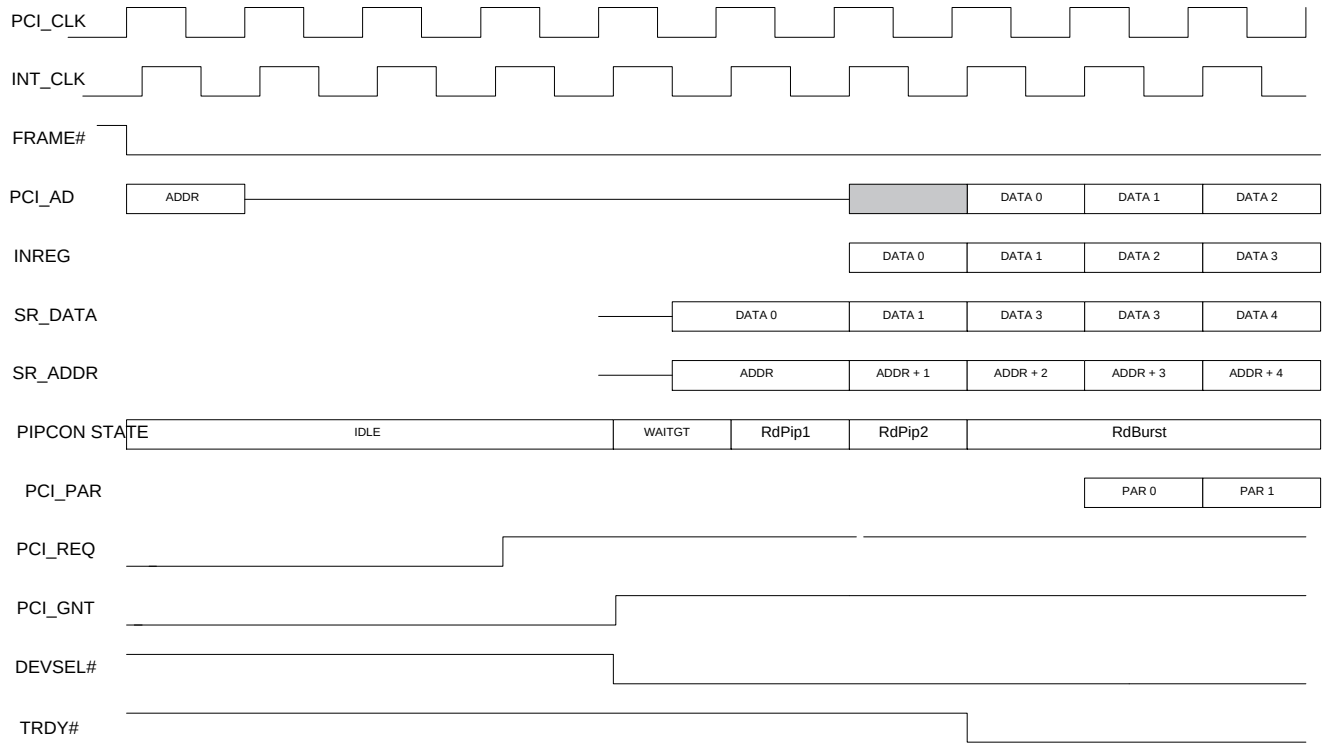

Figure 6. Read-cycle data flow.

passed through a CLB function generator and then setup before CLK at the registered IOBs TRDY# and DEVSEL# and the internal register, ENDEDn. In these CLBs, the direct PCI inputs gate the state machine outputs. This is the *only* possible way to meet this critical timing in the XC3100A-2.

Output Enable/Disable

Enable/disable times for XC3100A-2 outputs are significantly slower than the clock-to-output delay. Fortunately, the PCI specification provides 28 ns for output disable. In this design both PCI and SRAM outputs are enabled one clock early, to remove the output enable/disable times from the critical path.

Initiator Wait States

Initiator wait states on writes are easily handled in the pipelined design. The registered IRDY# input is passed down the pipeline along with the data. At the appropriate point, this signal inhibits the write pulse and address incrementing in the pipeline slots corresponding to initiator wait states.

Initiator wait states on read cycles disrupt the pipeline which, in the absence of initiator wait states, normally reads ahead two words in order to keep data flowing at full speed. Consequently, when the initiator inserts a wait state, the address counter is several clocks past that of the last data transferred before it can be halted or reversed. Rather than hold the bus in wait states while the pipeline is refilled, the PCI target first de-asserts TRDY#, then asserts STOP#, to signal a retry. The master then repeats the transfer when it is ready. The intention is to minimize fanout requirements on critical path signal IRDY# rather than to avoid complications in the address counter and control logic. The justification for this approach is that initiator read wait states should be exceedingly rare.

SRAM Write Timing

On write cycles, SRCSn is ANDed with CLK. Routing delays move this half-clock-period pulse to the middle of the clock period. The CLK input buffer and CLB are roughly equal in delay to the global CLK buffer. Therefore, if the routing delay for the CLK pin to SRCSn output is less than 15 ns, SRCSn will be High before the earliest possible address change at the end of the cycle. This requirement was met using XACT–Performance to constrain the default pad-to-pad timing to less than 19 ns, which was achieved without trouble.

Floorplanning and Routing

Early in the design, it was recognized that good placement was essential to meeting the stringent PCI timing constraints. The block diagram and a PCI outline diagram showing PCI connector pinout were used as inputs to the floorplanning process. IOB assignment was based on a compromise between board layout and Simultaneous Switching Output (SSO) considerations. Since most of the device pins were to be used, it was decided to distribute the two busses around the chip, placing them in groups on either side of the package ground pins to minimize switching noise. The busses were partitioned into 8 bit groups and placed around the package to minimize PCB trace length. Once IOBs were assigned, it was a simple matter of placing critical control logic and data path blocks into immediately adjacent CLBs. Timing constraints were added, based on identified critical signals. XACT–Performance timing was ignored on multiple-cycle paths such as the non-counting address outputs, output enables, and RST#. CRITPATH CLBs and pads were placed manually and the routing optimized by hand, after initial PPR placement and routing. The interface design is I/O-intensive. Since the design only utilizes 40% of the internal resources, a complete route took just a few minutes.

Both routing and timing are optimized when the PCI_AD pins are interleaved with SR_DATA pins. However, if parity checking is attempted, all PCI AD signals should be grouped on one side of the chip, to minimize parity check routing delays. Another consideration for parity check is that PERR# and SERR# would need to be placed on either side of the PAR pad.

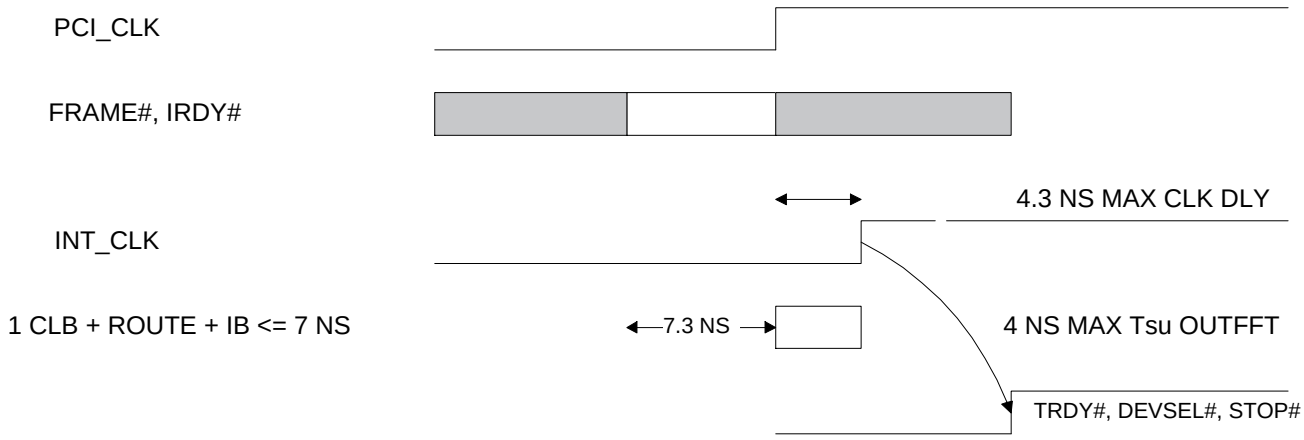

Figure 7. Handshake critical path.

Coding and Synthesis

The same block diagram used as an input to the floor planning process served as a model from which the data path portion of the design was coded. To implement the data path, the register stages shown in the block diagram in Figure 2 were instantiated. Several iterations were required to get the synthesis tool to produce registered I/Os. This problem was solved ultimately by directly instantiating them in a layer of hierarchy containing just the I/O buffers and the logic module. Next, the control logic was implemented. The design was done in an incremental fashion with verification, synthesis, and routing performed at intermediate points gradually approaching the required functionality and performance.

Design of the control logic started by diagramming the AD pipeline timing to determine control and handshake signal requirements. In recognition of the critical paths involved in initiator wait states and burst termination, several attempts were made to develop simple gating state machines which could shut off the outputs of larger protocol state machines reducing the fanout and depth of logic required on IRDYn and FRAMEn. Ultimately these were reduced to the single function generator per output implemented in the CRITPATH module. As the design progressed, additional control pipeline stages were added to keep the logic depth between pipeline stages to 2 (sometimes 3) CLBs maximum. This is manifested primarily in the address selection logic where DEVSEL# waits until clock period 5 and TRDY# until clock period 7 of a write cycle.

For good results, the HDL code must employ constructs which map well onto primitives available in the target technology, and must be written with due consideration given to timing requirements and routing delays. The most obvious example is PIPCON, the one-hot state machine. This design technique is well suited for high performance and compact implementation in a register-rich architecture. This particular implementation includes a register instantiation that implies D-type flip-flops with direct resets only, matching what is available in the technology. Less obvious, but equally important, is the structuring of logic equations in light of the available 5-input CLB. A first effort is to design the logic so that more than 5 inputs are never required. When this fails, intermediate logic equations should be written taking into account early and late arriving signals. When this is done explicitly, the synthesis tool does not destroy timing-optimized gate ordering in search of area optimization. As discussed earlier, the care taken in logic design must be matched with equal care during the placement and routing phase.

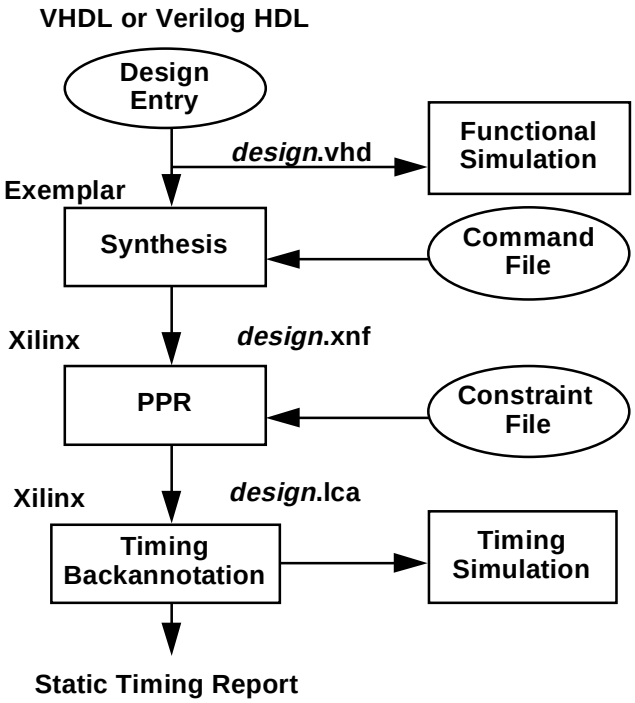

Figure 8. Exemplar/Xilinx design flow.

Table 1. PCI Target Interface CLB Requirements

Function	Module/Signals	CLBs
State Machine	PIPCON, CRITPATH, ARBITER	29
Address Decode	HIT, AVALID, PCI_REQ	4
Base Addr. Reg.	BA_AD_REG, BA_WR, BAREG_SELQ	4
Parity Generator	PARITY	9
Address Reg. & Counter	ADD_CNT, NOMORE, ADDR_REG	10
SRAM Control	SRWEn, SRCSn, SROEn	4
I/O Data Path	Registered I/Os only	0
I/O Control	Remainder	25
Total CLBs Required		85

Verification

A complete system simulation environment was implemented by writing both an SRAM model and tasks to generate PCI cycles. Tests of single and burst read and write cycles with and without initiator-inserted waits states (via IRDYn) and with and without arbitration latency at the SRAM are available as electronic files.

Using the Design Files

This design is available on diskette and may be requested through your local Xilinx sales office or by sending an E-mail with your name, company, mailing address, phone number, and FAX number to **pci@xilinx.com**.

This section also describes what software is required to run the design and the steps involved. Also, please read through the ***Limitations and Restrictions*** section.

The files are arranged in two directories—one for schematic-based design (**/VIEWDRAW**) and one for Verilog-based designs (**/VERILOG**).

Software Requirements

The following software is required to process this design:

- PKUNZIP 2.04e, or later, unarchiving program.
- Xilinx XACT 5.0 FPGA development system, including the PPR place and route program and the X-BLOX module generator.

Verilog Design

- The Exemplar CORE logic synthesis software.

VIEWdraw Schematic Design

- VIEWdraw® or VIEWdraw-LCA® schematic editor. This software is required in order to make modifications to the schematics.

Using the Design on Your System

1. Create a new directory called **PCI_FPGA** on your hard disk.
2. Copy the file called **PCI_FPGA.EXE** into the **PCI_FPGA** directory.
3. Type **PCI_FPGA.EXE** on the command line. This extracts a **README.TXT** file, and a hierarchical archive of the design files called **PCI_3164.ZIP**.
4. Invoke **PKUNZIP -D PCI_3164.ZIP** to extract the files, including their hierarchical path names, onto your disk.
5. If using the schematic design files, edit the **VIEWDRAW.INI** file. Make sure that the VIEWlogic® design library pointers are set appropriately for your machine. You will find the library pointers near the end of the file.

Limitations and Restrictions

WARNING: THIS IS AN UNTESTED DESIGN.

Xilinx, Inc. does not make any representation or warranty regarding this design or any item based on this design. Xilinx disclaims all express and implied warranties, including but not limited to the implied fitness of this design for a particular purpose and freedom from infringement. Without limiting the generality of the foregoing, Xilinx does not make any warranty of any kind that any item developed based on this design, or any portion of it, will not infringe any copyright, patent, trade secret or other intellectual property right of any person or entity in any country. It is the responsibility of the user to seek licenses for such intellectual property rights were applicable. Xilinx shall not be liable for any damages arising out of or in connection with the use of the design including liability for lost profit, business interruption, or any other damages whatsoever.

Design Support and Feedback

This application note may undergo future revisions and additions. If you would like to be updated with new versions of this application note, or if you have questions, comments, or suggestions please send an E-mail to

pci@xilinx.com

or a FAX addressed to "XC3164A PCI Application Note Developers" sent to

1+(408) 879-4442.

Conclusion

The Xilinx XC3164A-2 FPGA device provides a fully-compliant PCI interface. Conservative, pipelined design techniques guarantees 33 MHz worst-case performance, even through logic synthesis.

North America

Corporate Headquarters

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

Tel: 1+(408) 559-7778
FAX: 1+(408) 559-7114

Northern California

Xilinx, Inc.
3235 Kifer Road
Suite 320
Santa Clara, CA 95051

Tel: (408) 245-1361
FAX: (408) 245-0517

Southern California

Xilinx, Inc.
15615 Alton Parkway
Suite 280
Irvine, CA 92718

Tel: (714) 727-0780
FAX: (714) 727-3128

New Hampshire

Xilinx, Inc.
61 Spit Brook Road
Nashua, NH 03060

Tel: (603) 891-1096
FAX: (603) 891-0890

Pennsylvania

Xilinx, Inc.
905 Airport Rd.
Suite 200
West Chester, PA 19380

Tel: (610) 430-3300
FAX: (610) 430-0470

Texas

Xilinx, Inc.
4100 McEwen
Suite 237
Dallas, TX 75244

Tel: (214) 960-1043
FAX: (214) 960-0927

Illinois

Xilinx, Inc.
939 N. Plum Grove Road
Suite H
Schaumburg, IL 60173

Tel: (708) 605-1972
FAX: (708) 605-1976

North Carolina

Xilinx, Inc.
6080-C Six Forks Road
Raleigh, NC 27609

Tel: (919) 846-3922
FAX: (919) 846-8316

Europe

United Kingdom

Xilinx, Ltd.
Suite 1B, Cobb House
Oyster Lane, Byfleet
Surry KT14 7DU
UNITED KINGDOM

Tel: (44) 932-349401
FAX: (44) 932-349499

France

Xilinx Sarl
Espace Jouy Technology
21, rue Albert Calmette, Bt. C
78353 Jouy en Josas Cedex
FRANCE

Tel: (33) 1-34-63-01-01
FAX: (33) 1-34-63-01-09

Germany

Xilinx, GmbH
Dorfstr. 1
85609 Aschheim
München
GERMANY

Tel: (49) 89-904-5024
FAX: (49) 89-904-4748

Japan

Xilinx, K.K.
Daini-Nagaoka Bldg. 2F
2-8-5, Hatchobori Chuo-ku
Tokyo 104
JAPAN

Tel: (03) 3297-9191
FAX: (03) 3297-9189

Asia Pacific

Hong Kong

Xilinx Asia Pacific
Unit No. 2318-2309
Tower 1, Metroplaza
Hing Fong Road
Kwai Fong, N.T.
HONG KONG

Tel: (852) 2410-2717
FAX: (852) 2494-7159
E-mail: **hongkong@xilinx.com**

XILINX®

The Programmable Logic CompanySM

Printed in U.S.A. California Mailing P/N: **10395**