REAL

COMPLIANCE

FLEXIBILITY

PERFORMANCE

AVAILABILITY

The **Real-PCI™**

Xilinx PCI Data Book

XILINX®

Xilinx PCI

Data Book

Xilinx PCI Solutions (www)	www.xilinx.com/pci
Xilinx Home Page	www.xilinx.com
Application Service Centers	
North America Hotline:	+1 408-879-5199 (USA, Xilinx Headquarters)
	+1 800-255-7778
Fax:	+1 408-879-4442
Email:	hotline@xilinx.com
United Kingdom Hotline:	(44) 1932-820821
Fax:	(44) 1932-828522
Email:	ukhelp@xilinx.com
France Hotline:	(33) 1-3463-0100
Fax:	(33) 1-3463-0959
Email:	frhelp@xilinx.com
Germany Hotline:	(49) 89-93088-130
Fax:	(49) 89-93088-188
Email:	dlhelp@xilinx.com
Japan Hotline:	(81) 3-3297-9163
Fax:	(81) 3-3297-0067
Email:	jhotline@xilinx.com
Korea Hotline:	(82) 2-761-4277
Fax:	(82) 2-761-4278
Email:	korea@xilinx.com
Hong Kong Hotline:	(85) 2-2424-5200
Fax:	(85) 2-2424-7159
Email:	hongkong@xilinx.com

XILINX®
2100 Logic Drive
San Jose, CA 95124
United States of America
Telephone: +1 408-559-7778
Fax: +1 408-559-7114

Dear PCI customer,

On behalf of the PCI Team at Xilinx, and our CORE partners, welcome to our March 1999 PCI Data Book, and thank you for your interest in Xilinx PCI Solutions.

As the inventor and leading provider of Field Programmable Gate Array Technology, we want to pledge our continuing commitment to support your great ideas in logic design and PCI applications.

Since the last version of this databook we have added the Real 64/66 PCI™, the industry's first general-purpose 64-Bit, 66 MHz PCI Solution, and the PCI32 Spartan XL, a single-chip PCI solution at half the cost of standard PCI bridge chips.

Our mission is to provide you with a high-quality PCI solution that offers better flexibility, higher performance, and lower cost than any other available solution. Xilinx PCI allows you to integrate a PCI interface with your unique logic, into one flexible programmable device. Since the first PCI product introduction in February 1996, we have developed a complete solution for PCI including super-fast FPGAs, easy-to-use predictable LogiCORE modules with guaranteed timing, as well as PCI boards, drivers, and design examples. We believe you will find Xilinx PCI Solution interesting and we hope that you will consider us for future designs.

Together we can bring the great ideas to life!

Sincerely

Per Holmberg

LogiCORE Product Manager

CORE Solutions Group

Section Titles

Introduction

PCI Products

PCI64 Virtex Interface Version 3.0

PCI32 Virtex Version 3.0

PCI Compliance Checklists

Pinout and Configuration

Resources

Waveforms

Ordering Information and License Agreement

Sales Offices, Sales Representatives, and Distributors

Introduction

Introduction

PCI (Peripheral Component Interconnect) has become one of the most popular bus standards, not only for personal computers, but also for industrial computers, communication switches, routers, and instrumentation.

PCI is also a significant design challenge; the stringent electrical, functional, and timing specifications are difficult to meet in any technology and the standard keeps evolving to meet the dynamic needs of our industry.

This is why you need a flexible PCI solution that will meet both your current and future requirements, while guaranteeing full PCI compliance with no limitations on performance or functionality.

Using an FPGA for PCI

By integrating a fully-compliant PCI interface with an application-specific back-end design into one FPGA, you can achieve higher integration, higher performance and lower cost than other PCI solutions.

Further, the Xilinx PCI solution can be customized for a specific application and, as a result, the highest possible performance is achieved.

The flexibility of an FPGA makes it possible to update the PCI board, through software alone, in development or in the field. This significantly reduces your design risk and cuts development time.

Using Xilinx for PCI

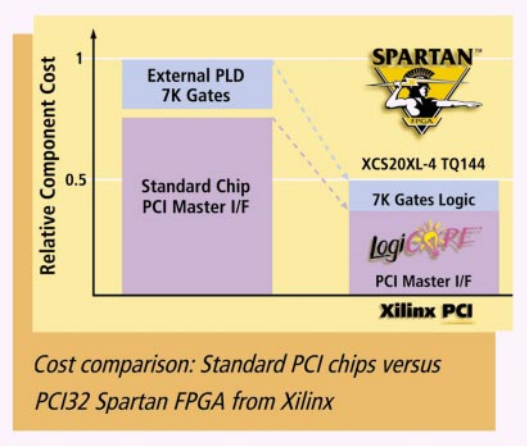
We provide the most cost-effective and high-performance PCI solution in the market by leveraging the flexibility of Xilinx Field Programmable Gate Arrays (FPGAs). We make PCI easy to design by providing a complete solution of proven cores, intuitive development tools, and world-class technical support.

Highest-Performance PCI

When we introduced the Real 64/66 PCI, we were first out with a fully compliant, general-purpose 64-bit 66MHz PCI solution. By using our LogiCORE PCI64 for Virtex, you can achieve the highest possible PCI performance, 528 MB/s.

Lowest-cost PCI

With our low-cost FPGA family, Spartan and SpartanXL, you can design your own unique PCI bridge with integrated FIFOs, DMA, and custom logic, at a cost only half of other available standard PCI bridge chips.



Cost comparison: Standard PCI chips versus PCI32 Spartan FPGA from Xilinx

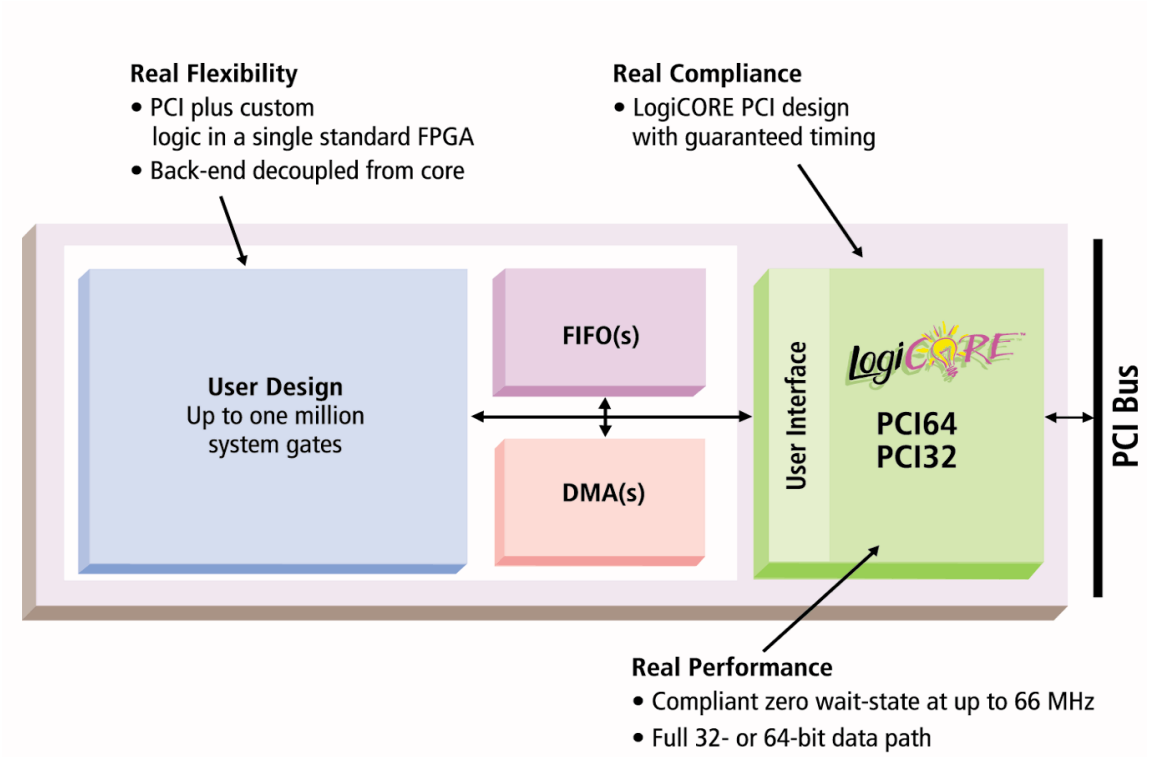
The Real-PCI from Xilinx

The Real-PCI from Xilinx is engineered to address all your requirements on a fully compliant PCI system. It provides you with

- Real Compliance
- Real Flexibility
- Real Performance
- Real Availability

Real Compliance

Our PCI cores have been used in over 1,000 customer designs. They are fully verified using our industry-proven testbench that simulates over six million unique PCI cycles. We also characterize our PCI cores together with our FPGAs to verify not only maximum timing, but also minimum and hold timing. Then, when we know that the timing constraints are met, we apply our unique Smart-IP technology to ensure that you achieve the same timing and functionality every time you implement the core. Thanks to our regular FPGA architecture with segmented routing, and because we use a modular core architecture where the FIFOs, DMA channels, and your unique back-end logic are de-coupled from the core, your own design will not affect the PCI interface timing.



Key Attributes of The Real PCI from Xilinx

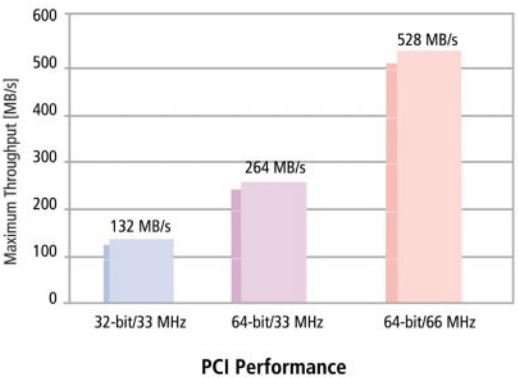
Real Flexibility

Our PCI cores are targeted to our standard off-the-shelf FPGAs, which were designed to be PCI compliant. This gives you a range of device sizes and packages to choose from, allowing you to integrate a fully-compliant Initiator/Target PCI interface, scalable dual-port FIFOs, customizable DMA channels, and 7,000 to 1,000,000 system gates of your own unique logic, all on a single device, for the lowest possible cost.

Because the FPGA is programmable, you can adapt to future needs and changes in the PCI standard by reconfiguring the FPGA device on your board.

Real Performance

All Xilinx PCI cores operate at maximum throughput, with 0 wait-state bursts. For example, the Xilinx Real-PCI 64/66 solution allows you to create 64-bit PCI systems that operate at up to 66MHz, delivering a sustained throughput of up to 528 Mbytes per second - the maximum performance you can get from PCI. Our PCI 32/33 cores supports up to 132 Mbytes per second.



	Bus Width [bits]	Max Frequency [MHz]	Signaling Environment [V]	CompactPCI Hot-Swap	ON-Chip Dual-Port FIFOs	MAX GATEs
Spartan	32	33	5	-	Yes	30K
SpartanXL	32	33	3.3 & 5	Yes*	Yes	30K
XC4000XLA	32	33	3.3 & 5	Yes*	Yes	100K
Virtex	32/64	66	3.3 & 5	Yes	Yes	1M

*Limited CompactPCI Hot Swap Support. See Xilinx App. Note *Implementing CompactPCI and Hot Swap CompactPCI with Xilinx PCI*

Real Availability

Real-PCI is here today. It includes a complete family of LogiCORE designs that are fully characterized for our XC4000XLA, Spartan, SpartanXL, and Virtex FPGAs. By using our standard, off-the-shelf manufacturing capability, leading edge silicon processes, excellent quality and testability, and lower manufacturing costs. Plus Real-PCI is not just cores and devices, it's also a complete system of development tools, support, services, and third-party Xilinx-authorized XPERTS to help you every step of the way.

Xilinx PCI Design Kits

To help you reduce your development time even further, Xilinx has teamed with Nallatech LTD and Virtual Computer Corp., both providing PCI prototyping boards, NuMega Software providing SW driver development tools, and Memec Design Services providing PCI expertise and design services. With our partners, we offer two complete PCI Design Kits, PCI64-DK for 64-bit 66MHz PCI, and PCI32-DK for 32-bit 33MHz PCI. The kits include prototyping boards, reference software drivers for Windows 95/98/NT, full-featured SW driver development tools, and synthesizable PCI bridge design examples in VHDL and Verilog.

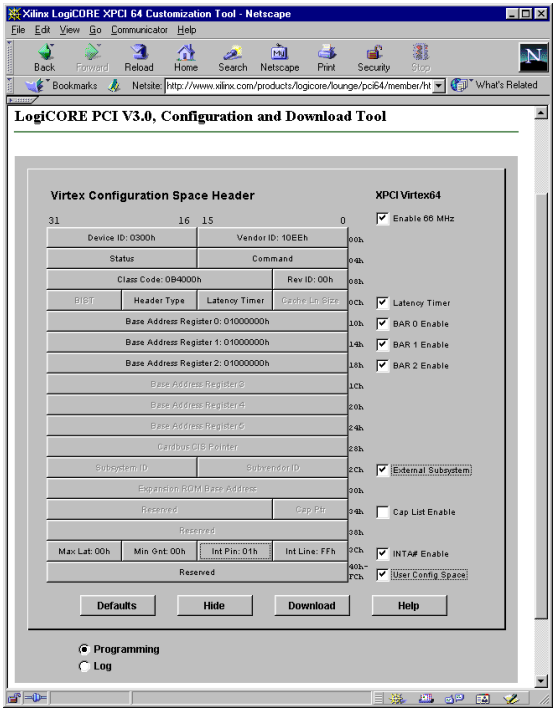
Xilinx PCI64 Design Kit

PCI over the Internet

As a part of our Silicon Xpresso initiative, we provide the LogiCORE products with all design files you need over the Internet. You will instantly have access to new versions of the core, new features, new application notes, and reference designs. As a Xilinx PCI customer, you select your own unique user name and password to access the core product. To help you configure the PCI core we have an intuitive graphical user interface (GUI) where you easily select the settings and features that you need. The process is simple:

1. Enter your configuration data into the GUI.
2. Click “Download,” and our Web tool builds your unique PCI interface with guaranteed timing, which you then download to your local computer.

The design files include the PCI design netlist, VHDL/Verilog simulation model and instantiation wrapper, and implementation constraints files.



About this Databook

The information in this databook is also available on the Xilinx web-site, WebLINX at

`www.xilinx.com/pci`

Xilinx will use the web as primary means of delivering and updating this information since it is so dynamic by nature. We strongly recommend that customers consult the web for the latest information on new product availability and datasheet revisions.

PCI64 Virtex Interface Version 3.0

May, 1999 Advanced Data Sheet

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-377-3259
E-mail: Techsupport: hotline@xilinx.com
Feedback: logicore@xilinx.com
URL: www.xilinx.com/pci

Introduction

With Xilinx LogiCORE PCI64 Virtex interface, a designer can build a customized, 64-bit, 0-66 MHz fully PCI compliant system with the highest possible sustained performance, 528 Mbytes/sec, and up to 1 Million System Gates in the Virtex family FPGA.

Features

- Fully 2.2 PCI compliant 64-bit, 0-66 MHz PCI Initiator/Target Interface
- Zero wait-state burst operation
- Programmable single-chip solution with customizable back-end functionality
- Pre-defined implementation for predictable timing in Xilinx Virtex Series FPGAs
- Incorporates Xilinx Smart-IP Technology
- 3.3 V Operation at 33-66 MHz
- 3.3 V and 5 V Operation at 0-33 MHz
- Master automatically handles 64-bit or 32-bit PCI transactions without knowing the bus width of the target
- Fully verified design tested with Xilinx testbench and hardware
- Configurable on-chip dual-port FIFOs can be added for maximum burst speed
- Supported Initiator functions (PCI Master only)
  - Memory Read, Memory Write, Memory Read Multiple (MRM), Memory Read Line (MRL) commands
  - I/O Read, I/O Write commands
  - Configuration Read, Configuration Write commands
  - Bus Parking
  - Special Cycles, Interrupt Acknowledge
  - Basic Host Bridging

LogiCORE™ Facts

Core Specifics	
Device Family	Virtex
Slices Used[1]	381-403
IOBs Used	88
System Clock f_{max}	0-66MHz
Device Features Used	Bi-directional data buses SelectIO Block SelectRAM+™ (optional user FIFO) Boundary scan (optional)

Supported Devices[2]/Percent Resources Used		
	I/O	**Slices**
XCV300-5/6 BG432[3]	28%	12%
XCV1000-5/6 FG680[3]	17%	3%

Provided with Core	
Documentation	*PCI Design Guide* *PCI Implementation Guide* *PCI Data Book*
Design File Formats	Verilog/VHDL Simulation Model Verilog/VHDL Instantiation Code NGO Netlist
Constraint Files	M1 User Constraint File (UCF) M1 Guide files
Verification Tools	Verilog/VHDL Testbench
Reference designs & application notes	Example designs: PING64 Reference Design Synthesizable PCI64 Bridge(SB07)

Design Tool Requirements	
Xilinx Core Tools	M1.5i SP2
Tested Entry/Verification Tools[4]	For CORE instantiation: Synopsys FPGA Express Synopsys FPGA Compiler Synplicity Synplify For CORE verification: Cadence Verilog XL MTI ModelSim PE/Plus V4.7g

Xilinx provides technical support for this LogiCORE™ product when used as described in the User's Guide and in the Application Notes. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices not listed above, or if customized beyond that referenced in the product documentation, or if any changes are made in sections of design marked as "DO NOT MODIFY".

1. The exact number of CLBs depends on user configuration of the core and level of resource sharing with adjacent logic. For example, a factor that can affect the size of the design are the number and size of the BARs.
2. Re-targeting the PCI core to an unlisted device or package will void the guarantee of timing. See “Smart-IP Technology - guaranteed timing” on page 3 for details.
3. Use -6 for 0-66 MHz operation and -5 for 0-33 MHz operation.
4. See Xilinx Web Site for update on tested design tools.

Features (cont.)

- Supported Target functions (PCI Master and Slave)
  - Type 0 Configuration Space Header
  - Up to 3 Base Address Registers (memory or I/O with adjustable block size from 16 Bytes to 2 GBytes, medium decode speed)
  - Parity Generation (PAR), Parity Error Detection (PERR# and SERR#)
  - Memory Read, Memory Write, Memory Read Multiple (MRM), Memory Read Line (MRL), Memory Write Invalidate (MWI) commands
  - I/O Read, I/O Write commands
  - Configuration Read, Configuration Write commands
  - 64-bit data transfers and 32-bit data transfers, burst transfers with linear address ordering
  - Target Abort, Target Retry, Target Disconnect
  - Full Command/Status Registers
- Available for configuration and download on the web
  - Web-based configuration tool
  - Generation of proven design files
  - Instant access to new releases

Applications

- Embedded applications within networking, telecommunication and industrial systems
- PCI add-in boards such as graphic cards, video adapters, LAN adapters and data acquisition boards
- Hot Swap CompactPCI boards
- Other applications that need PCI

General Description

The LogiCORE™ PCI64 Interfaces are pre-implemented and fully tested modules for the Xilinx Virtex Series FPGAs. The pinout for the device and the relative placement of the internal Configurable Logic Blocks (CLBs) are pre-defined. Critical paths are controlled by TimeSpecs and guide files to ensure predictable timing. This significantly reduces engineering time required to implement the PCI portion of your design. Resources can instead be focused on the unique back-end logic in the FPGA and on the system level design. As a result, LogiCORE™ PCI products can minimize your product development time.

Xilinx Virtex Series FPGAs enable designs of fully PCI-compliant systems. The devices meet all required electrical and timing parameters including AC output drive characteristics, input capacitance specifications (10pF), 3 ns setup and 0 ns hold to system clock, and 6 ns system clock to output. These devices meet all specifications for both 3.3 V (0-66 MHz) and 5 V PCI (0-33 MHz).

The PCI Compliance Checklist has detailed information about electrical compliance. Other features that enable efficient implementation of a complete PCI system in the Virtex Series includes:

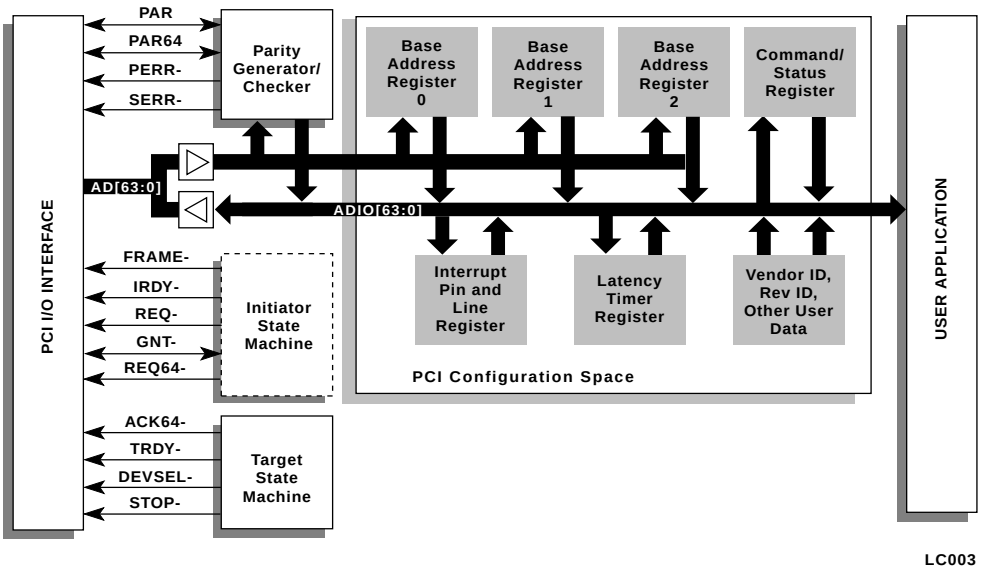


Figure 1: LogiCORE™ PCI64 Interface Block Diagram

- Block SelectRAM+™ memory: Blocks of on-chip ultra-fast RAM with synchronous write and dual-port RAM capabilities. Used in PCI Interfaces to implement FIFO
- Select-RAM™ memory: on-chip ultra-fast RAM with synchronous write option and dual-port RAM option. Used in PCI Interfaces to implement FIFO
- Individual output enable for each I/O
- Internal 3-state bus capability
- 8 global low-skew clock or signal distribution networks
- IEEE 1149.1-compatible boundary scan logic support

The Master and Slave Interface module is carefully optimized for best possible performance and utilization in the Virtex FPGA architecture. When implemented in a XCV300, 12% of the FPGA's slices are used.

Smart-IP Technology - guaranteed timing

Drawing on the architectural advantages of Xilinx FPGAs, new Xilinx Smart-IP technology ensures highest performance, predictability, repeatability, and flexibility in PCI designs. The Smart-IP technology is incorporated in every LogiCORE PCI Core.

Xilinx Smart-IP technology leverages the Xilinx architectural advantages, such as look-up tables (LUTs), distributed RAM, and segmented routing, as well as floorplanning information, such as logic mapping and relative location constraints. This technology provides the best physical layout, predictability, and performance. Additionally, these predetermined features allow for significantly reduced compile times over competing architectures.

PCI Cores made with Smart-IP technology are unique by maintaining their performance and predictability regardless of the device size.

To guarantee the critical setup, hold, and min. and max. clock-to-out timing, the PCI core is delivered with Smart-IP constraint files that are unique for a device and package combination. These constraint files guide the implementation tools so that the critical paths always are within PCI specification. Retargeting the PCI core to an unsupported device will void the guarantee of timing. Contact one of the Xilinx XPERTs partners for support of unlisted devices and packages. See the XPERTs section in chapter 7 of the Xilinx PCI Data Book for contact information.

Functional Description

The LogiCORE PCI64 Master and Slave Interface is partitioned into five major blocks and an user application as shown in Figure 1. Each block is described below.

PCI Configuration Space

This block provides the first 64 Bytes of Type 0, version 2.1 Configuration Space Header (CSH) (see Table 1) to support software-driven "Plug-and Play" initialization and configuration. This includes information for Command, Status, and three Base Address Registers (BARs). These BARs illustrate how to implement memory- or I/O-mapped address spaces.

Table 1: PCI Configuration Space Header

31	16	15	0	
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Rev ID	08h
BIST	Header Type	Latency Timer	*Cache Line Size*	0Ch
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 (BAR5)				20h
Base Address Register 5 (BAR5)				24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			CapPtr	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch
Reserved				40h-FFh

Note:
Italicized address areas are not implemented in the LogiCORE PCI64 Virtex Interface default configuration. These locations return zero during configuration read accesses.

Each BAR sets the base address for the interface and allows the system software to determine the addressable range required by the interface. Each BAR designated as a memory space can be made to represent a 32-bit or a 64-bit space.

Using a combination of Configurable Logic Block (CLB) flip-flops for the read/write registers and CLB look-up tables for the read-only registers results in optimized logic mapping and placement.

The capability for extending configuration space has been built into the backend interface. This capability, including the ability to implement a CapPtr in configuration space, allows the user to implement functions such as Advanced Configuration and Power Interface (ACPI) in the backend design.

PCI I/O Interface Block

The I/O interface block handles the physical connection to the PCI bus including all signaling, input and output synchronization, output three-state controls, and all request-grant handshaking for bus mastering.

Parity Generator/Checker

This block generates/checks even parity across the AD bus, the CBE lines, PAR and the PAR64 signal. It also reports data parity errors via PERR- and address parity errors via SERR-.

Target State Machine

This block controls the PCI interface for Target functions. The states implemented are a subset of equations defined in “Appendix B” of the *PCI Local Bus Specification*. The controller is a high-performance state machine using one-hot (state-per-bit) encoding for maximum performance. State-per-bit encoding of the next-state logic functions facilitates a high performance implementation in the Xilinx FPGA architecture.

Initiator State Machine

This block controls the PCI interface for Initiator functions. The states implemented are a subset of equations defined in “Appendix B” of the *PCI Local Bus Specification*. The Initiator Control Logic also uses state-per-bit encoding for maximum performance.

User Application with Optional Burst FIFOs

The LogiCORE PCI64 Interface provides a simple, general-purpose interface with a 64-bit data path and latched address for de-multiplexing the PCI address/data bus. The general-purpose user interface allows the rest of the device to be used in a wide range of 64-bit and 32-bit applications.

Typically, the user application contains burst FIFOs to increase PCI system performance. An on-chip read/write FIFO, built from the on-chip synchronous dual-port RAM (Block SelectRAM+™) available in Virtex Series FPGAs, supports data transfers in excess of 66 MHz.

Several synthesizable re-usable bridge designs including commonly used backend functions, such as doorbells and mailboxes, are provided with the core.

Interface Configuration

The LogiCORE PCI64 Interface can easily be configured to fit unique system requirements by using Xilinx web-based PCI configuration tool or by changing the Verilog, VHDL, or VIEW*logic* configuration file. The following customization options are supported by the LogiCORE product and described in product documentation.

- Initiator or target functionality (PCI Master only)
- Base Address Register configuration (1 - 3 Registers, size and mode)
- Configuration Space Header ROM
- Initiator and target state machine (e.g., termination conditions, transaction types and request/transaction arbitration)
- Burst functionality
- User Application including FIFO (back-end design)

Supported PCI Commands

Table 2 illustrates the PCI bus commands supported by the LogiCORE™ PCI64 Interface. The PCI Compliance Checklist has more details on supported and unsupported commands.

Table 2: PCI Bus Commands

CBE [3:0]	Command	PCI Master	PCI Slave
0000	Interrupt Acknowledge	No[1]	Yes
0001	Special Cycle	No[1]	Ignore
0010	I/O Read	Yes	Yes
0011	I/O Write	Yes	Yes
0100	Reserved	Ignore	Ignore
0101	Reserved	Ignore	Ignore
0110	Memory Read	Yes	Yes
0111	Memory Write	Yes	Yes
1000	Reserved	Ignore	Ignore
1001	Reserved	Ignore	Ignore
1010	Configuration Read	Yes	Yes
1011	Configuration Write	Yes	Yes
1100	Memory Read Multiple	Yes	Yes
1101	Dual Address Cycle	No[1]	Ignore
1110	Memory Read Line	Yes	Yes
1111	Memory Write Invalidate	No[1]	Yes

Note:
1. The Initiator can present these commands, however, they either require additional user-application logic to support them or have not been thoroughly tested.

Burst Transfer

The PCI bus derives its performance from its ability to support burst transfers. The performance of any PCI application depends largely on the size of the burst transfer. A FIFO to support PCI burst transfer can efficiently be implemented using the Virtex on-chip RAM features, both Distributed and Block SelectRAM+™.

Each Virtex CLB supports four 16x1 RAM blocks. This corresponds to 64 bits of single-ported RAM or 32 bits of dual-ported RAM, with simultaneous read/write capability. The Block SelectRAM+ can be used to create deep FIFOs.

Bandwidth

Xilinx LogiCORE PCI64 Interface supports fully compliant zero wait-state bust operations for both sourcing and

receiving data. This Interface supports a sustained bandwidth of up to 528 MBytes/sec. The design can be configured to take advantage of the ability of the LogiCORE PCI64 Interface to do very long bursts. Since the FIFO is not of fixed size, bursts can go on for as long as the chipset arbiter will allow. Furthermore, since the FIFOs and DMA are decoupled from the proven core, a designer can modify these functions without affecting the critical PCI timing.

The flexible Xilinx backend, combined with support for many different PCI features, gives users a solution that lends itself to being used in many high-performance applications. Xilinx is able to support different depths of FIFOs as well as dual port FIFOs, synchronous or asynchronous FIFOs and multiple FIFOs. The user is not locked into one DMA engine, hence, a DMA that fits a specific application can be designed.

The theoretical maximum bandwidth of a 64-bit, 66 MHz PCI bus is 528 MBytes/sec. Attaining this maximum bandwidth will depend on several factors, including the PCI design used, PCI chipset, the processor's ability to keep up with your data stream, the maximum capability of your PCI design, and other traffic on the PCI bus. Older chipsets and processors will tend to allow less bandwidth than newer ones.

No additional wait-states are inserted in response to a wait-state from another agent on the bus. Either IRDY or TRDY is kept asserted until the current data phase ends, as required by the V2.2 PCI Specification.

See Table 3 for PCI bus transfer rates for various operations.

Table 3: LogiCORE PCI64 Transfer Rates

Zero Wait-State Mode	
Operation	**Transfer Rate**
Initiator Write (PCI ← LogiCORE)	3-1-1-1
Initiator Read (PCI → LogiCORE)	4-1-1-1
Target Write (PCI→ LogiCORE)	5-1-1-1
Target Read (PCI ← LogiCORE)	6-1-1-1

***Note: Initiator Read and Target Write operations have effectively the same bandwidth for burst transfer.

Timing Specification

The Virtex Series FPGA devices, together with the LogiCORE PCI64 product enable design of fully compliant PCI systems. The maximum speed at which your back-end is capable of running can be affected by the size of the design as well as by the loading of the hot signals coming directly from the PCI bus. Table 4 shows the key timing parameters for the LogiCORE PCI64 Interface that must be met for full PCI compliance.

Table 4: 66 MHz Timing Parameters [ns]

Parameter	Ref.	PCI Spec.		LogiCORE PCI64 XCV300-06[4]	
		Min	Max	Min	Max
CLK Cycle Time	T_{cyc}	15	30	15[1]	30
CLK High Time	T_{high}	6		6	
CLK Low Time	T_{low}	6		6	
CLK to Bus Signals Valid[3]	T_{ICKOF}	2	6	2[2]	6
CLK to REQ# and GNT# Valid[3]	T_{ICKOF}	2	6	2[2]	6
Tri-state to Active	T_{on}	2		2[2]	
CLK to Tri-state	T_{off}		14		14[1]
Bus Signal Setup to CLK (IOB)	T_{PSD}		3		3
Bus Signal Setup to CLK (CLB)			5		5[1]
GNT# Setup to CLK	T_{PSD}		5		5
GNT# Setup to CLK (CLB)	T_{PSD}		5		5
Input Hold Time After CLK (IOB)	T_{PHD}		0		0
Input Hold Time After CLK (CLB)			0		0[2]
RST# to Tri-state	$T_{rst-off}$		40		40[2]

Notes:
1. Controlled by TIMESPECS, included in product
2. Verified by analysis and bench-testing
3. IOB configured for Fast slew rate
4. Virtex Timing will be included when silicon testing is finished.

Verification Methods

Xilinx has developed a system-level testbench that allows simulation of an open PCI environment in which a LogiCORE-PCI-based design may be tested by itself or with other simulatable PCI agents. Included in these agents are a behavioral host and target, and several "plug-in" modules, including a PCI signal recorder and a PCI protocol monitor. The Xilinx PCI testbench is a powerful verification tool that is also used as the basis for verification of the PCI LogiCORE. The PCI LogiCORE is also tested in hardware for electrical, functional, and timing compliance.

Ping Reference Design

The Xilinx "PING64" Application Example, delivered in Verilog and VHDL, has been developed to provide an easy-to-understand example which demonstrates many of the principles and techniques required to successfully use a LogiCORE PCI64 Interface in a System On A Chip solution. The PING64 design is also used as a test vehicle when verifying the PCI core.

Synthesizable PCI Bridge Design (SB07)

The synthesizable PCI bridge design, SB07, is an application bridge for use with the LogiCORE PCI64 Interface. It is delivered in Verilog and VHDL and has been fully tested with various devices. This example demonstrates how to interface to the PCI core and provide a modular foundation upon which to base other designs. The reference design can be easily modified to remove select portions of functionality.

This design is a general purpose data transfer engine to be used with the LogiCORE PCI64 Interface. Figure 3 presents a block diagram of the SB07 design. Typically, the user will customize the local interface to conform to a particular peripheral bus (ISA, VME, i960) or attach to a memory device. The design is modular so that unused portions may be removed. Other bridge applications can be designed using subsets of SB07. The *Synthesizable PCI Bridge Design Data Sheet* lists the set of features and specifics for the SB07 design.

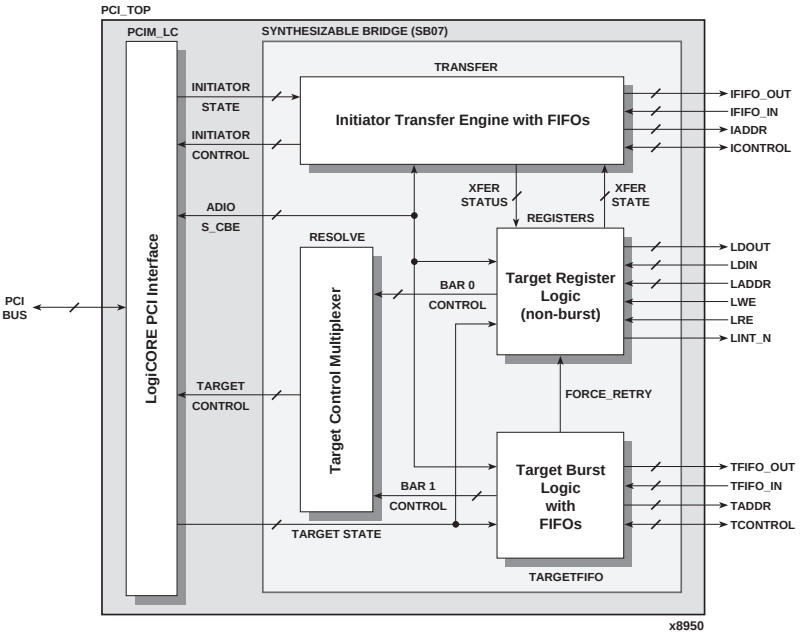


Figure 1: Block Diagram of Synthesizable Bridge Design for PCI64 LogiCORE Interface, SB07

Device Utilization

The Target/Initiator options require a variable amount of CLB resources for the PCI64 Interface.

Utilization of the device can vary slightly, depending on the configuration choices made by the designer. Factors that can affect the size of the core are:

- Number of Base Address Registers Used. Turning off any unused BARs will save resources.
- Size of the BARs. Setting the BAR to a smaller size requires more flip-flops. A smaller address space requires more flip-flops to decode.
- Latency timer. Disabling the latency timer will save resources. It must be enabled for bursting.

Recommended Design Experience

The LogiCORE PCI64 Interface is pre-implemented allowing engineering focus at the unique back-end functions of a PCI design. Regardless, PCI is a high-performance system that is challenging to implement in any technology, ASIC or FPGA. Therefore, previous experience with building high-performance, pipelined FPGA designs using Xilinx implementation software, TIMESPECs, and guide files is recommended. The challenge to implement a complete PCI design including back-end functions varies depending on configuration and functionality of your application. Contact your local Xilinx representative for a closer review and estimation for your specific requirements.

PCI32 Virtex Version 3.0

May, 1999 Advanced Data Sheet

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-377-3259
E-mail: Techsupport: hotline@xilinx.com
Feedback: logicore@xilinx.com
URL: www.xilinx.com/pci

Introduction

With Xilinx LogiCORE PCI32 Virtex Interface, a designer can build a customized, 32-bit, 33 MHz fully PCI compliant system with the highest possible sustained performance, 128 Mbytes/sec, and up to 300,000 System Gates in the Virtex family FPGA.

Features

- Fully 2.2 PCI compliant 32-bit, 33 MHz PCI Initiator/Target Interface
- Zero wait-state burst operation
- Programmable single-chip solution with customizable back-end functionality
- Pre-defined implementation for predictable timing in Xilinx Virtex Series FPGAs
- Incorporates Xilinx Smart-IP Technology
- 3.3 V and 5 V Operation
- Fully verified design tested with Xilinx testbench and hardware
- Configurable on-chip dual-port FIFOs can be added for maximum burst speed
- Supported Initiator functions
  - Memory Read, Memory Write, Memory Read Multiple (MRM), Memory Read Line (MRL) commands
  - I/O Read, I/O Write commands
  - Configuration Read, Configuration Write commands
  - Bus Parking
  - Special Cycles, Interrupt Acknowledge
  - Basic Host Bridging

LogiCORE™ Facts

Core Specifics	
Device Family	Virtex
Slices Used[1]	381-403
IOBs Used	53
System Clock f_{max}	0-33MHz
Device Features Used	Multi-standard SelectIO SelectMAP Configuration (optional) Block SelectRAM+™ (optional user FIFO) Boundary scan (optional)

Supported Devices[2]/Percent Resources Used		
	I/O	**Slices**
XCV300-5 BG432	17%	12%

Provided with Core	
Documentation	*PCI Design Guide* *PCI Implementation Guide* *PCI Data Book*
Design File Formats	Verilog/VHDL Simulation Model, Verilog/VHDL Instantiation Code, NGO Netlist,
Constraint Files	M1 User Constraint File (UCF) M1 Guide files
Verification Tools	Verilog/VHDL Testbench
Reference designs & application notes	Example designs PING Reference Design Synthesizable PCI32 Bridge(SB08)

Design Tool Requirements	
Xilinx Core Tools	M1.5i SP2
Tested Entry/Verification Tools[3]	For CORE instantiation: Synopsys FPGA Express Synopsys FPGA Compiler Synplicity Synplify For CORE verification: Cadence Verilog XL MTI ModelSim PE/Plus V4.7g

Xilinx provides technical support for this LogiCORE™ product when used as described in the Design and Implementation Guides and in the Application Notes. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices not listed above, or if customized beyond that referenced in the product documentation, or if any changes are made in sections of design marked as “DO NOT MODIFY”.

1. The exact number of Slices depends on user configuration of the core and level of resource sharing with adjacent logic. For example, a factor that can affect the size of the design are the number and size of the BARs.
2. Re-targeting the PCI core to an unlisted device or package will void the guarantee of timing. See “Smart-IP Technology - guaranteed timing” on page 11. for details.
3. See Xilinx Web Site for updates on tested design tools.

Features (cont.)

- Supported Target functions (PCI Master and Slave)
  - Type 0 Configuration Space Header
  - Up to 3 Base Address Registers (memory or I/O with adjustable block size from 16 Bytes to 2 GBytes, medium decode speed)
  - Parity Generation (PAR), Parity Error Detection (PERR# and SERR#)
  - Extended Capabilities Registers (backend module)
  - Memory Read, Memory Write, Memory Read Multiple (MRM), Memory Read Line (MRL), Memory Write Invalidate (MWI) commands
  - I/O Read, I/O Write commands
  - Configuration Read, Configuration Write commands
  - Interrupt Acknowledge
  - 32-bit data transfers, burst transfers with linear address ordering
  - Target Abort, Target Retry, Target Disconnect
  - Full Command/Status Registers
- Available for configuration and download on the web
  - Web-based configuration tool
  - Generation of proven design files
  - Instant access to new releases

Applications

- Embedded applications within telecommunication, networking, and industrial systems
- PCI add-in boards such as graphic cards, video adapters, LAN adapters and data acquisition boards
- Hot Swap CompactPCI boards
- Other applications that need PCI

General Description

The LogiCORE™ PCI32 Master and Slave Interface has pre-implemented and fully tested modules for the Xilinx Virtex Series FPGAs. The pinout for the device and the relative placement of the internal Configurable Logic Blocks (CLBs) are pre-defined. Critical paths are controlled by TimeSpecs and placement to ensure predictable timing. This significantly reduces engineering time required to implement the PCI portion of your design. Resources can instead be focused on the unique back-end logic in the FPGA and on the system level design. As a result, LogiCORE™ PCI products can minimize your product development time.

Xilinx Virtex Series FPGAs enable designs of fully PCI-compliant systems. The devices meet all required electrical and timing parameters including AC output drive characteristics, input capacitance specifications (10pF), 3 ns setup and 0 ns hold to system clock, and 11 ns system clock to output. These devices meet all specifications for both 3.3 V and 5 V PCI.

The PCI Compliance Checklist has detailed information about electrical compliance. Other features that enable efficient implementation of a complete PCI system in the Virtex Series includes:

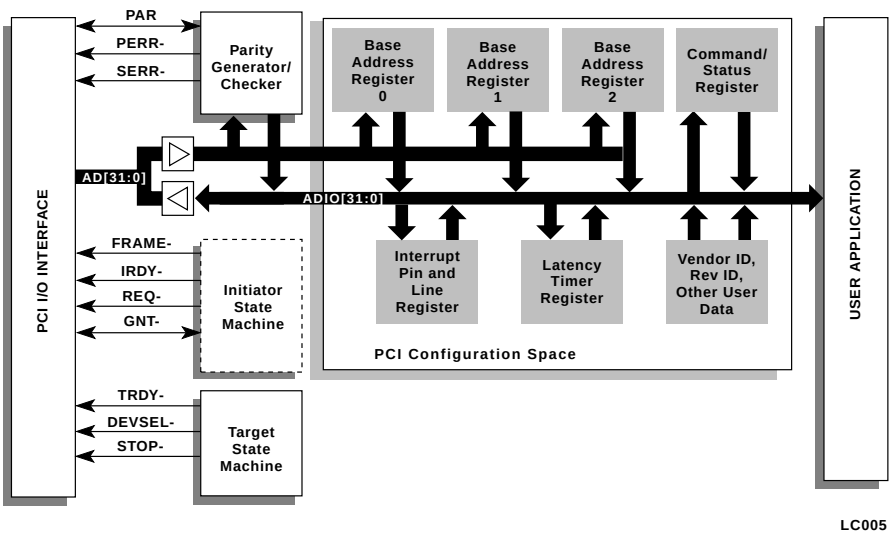


Figure 1: LogiCORE™ PCI32 Interface Block Diagram

- Block SelectRAM+™ memory: Blocks of on-chip ultra-fast RAM with synchronous write and dual-port RAM capabilities. Used in PCI Interfaces to implement FIFO
- Select-RAM™ memory: on-chip ultra-fast RAM with synchronous write option and dual-port RAM option. Used in PCI Interfaces to implement FIFO
- Individual output enable for each I/O
- Internal 3-state bus capability
- 4 global low-skew clock or signal distribution networks
- IEEE 1149.1-compatible boundary scan logic support

The Master and Slave Interface module is carefully optimized for best possible performance and utilization in the Virtex FPGA architecture. When implemented in a XCV300, 12% of the FPGA's slices are used.

Smart-IP Technology - guaranteed timing

Drawing on the architectural advantages of Xilinx FPGAs, new Xilinx Smart-IP technology ensures highest performance, predictability, repeatability, and flexibility in PCI designs. The Smart-IP technology is incorporated in every LogiCORE PCI Core.

Xilinx Smart-IP technology leverages the Xilinx architectural advantages, such as look-up tables (LUTs), distributed RAM, and segmented routing, as well as floorplanning information, such as logic mapping and relative location constraints. This technology provides the best physical layout, predictability, and performance. Additionally, these predetermined features allow for significantly reduced compile times over competing architectures.

PCI Cores made with Smart-IP technology are unique by maintaining their performance and predictability regardless of the device size.

To guarantee the critical setup, hold, and min. and max. clock-to-out timing, the PCI core is delivered with Smart-IP constraint files that are unique for a device and package combination. These constraint files guide the implementation tools so that the critical paths always are within PCI specification. Retargeting the PCI core to an unsupported device will void the guarantee of timing. Contact one of the Xilinx XPERTs partners for support of unlisted devices and packages. See the XPERTs section in chapter 7 of the Xilinx PCI Data Book for contact information.

Functional Description

The LogiCORE PCI32 Master and Slave Interface is partitioned into five major blocks and an user application as shown in Figure 1. Each block is described below.

PCI Configuration Space

This block provides the first 64 Bytes of Type 0, version 2.1 Configuration Space Header (CSH) (see Table 1) to support software-driven "Plug-and-Play" initialization and configuration. This includes information for Command, Status, and three Base Address Registers (BARs). These BARs illustrate how to implement memory or I/O mapped address spaces.

Table 1: PCI Configuration Space Header

31 16	15 0	
Device ID	Vendor ID	00h
Status	Command	04h
Class Code	Rev ID	08h
BIST / Header Type / Latency Timer	*Cache Line Size*	0Ch
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 (BAR5)		20h
Base Address Register 5 (BAR5)		24h
Cardbus CIS Pointer		28h
Subsystem ID	Subsystem Vendor ID	2Ch
Expansion ROM Base Address		30h
Reserved	CapPtr	34h
Reserved		38h
Max_Lat / Min_Gnt / Interrupt Pin	Interrupt Line	3Ch
Reserved		40h-FFh

Note:
Italicized address areas are not implemented in the LogiCORE PCI32 Virtex Interface default configuration. These locations return zero during configuration read accesses.

Each BAR sets the base address for the interface and allows the system software to determine the addressable range required by the interface. Each BAR designated as a memory space can be made to represent a 32-bit space.

Using a combination of Configurable Logic Block (CLB) flip-flops for the read/write registers and CLB look-up tables for the read-only registers results in optimized logic mapping and placement.

The LogiCORE PCI32 Interface includes the ability to add extended configuration capabilities as defined in the V2.2 PCI specification. This capability, including the ability to implement a CapPtr in configuration space, allows the user to implement extended functions such as Power Management, Hot Swap CSR, and Message Based Interrupts in the backend design.

PCI I/O Interface Block

The I/O interface block handles the physical connection to the PCI bus including all signaling, input and output synchronization, output three-state controls, and all request-grant handshaking for bus mastering.

Parity Generator/Checker

This block generates/checks even parity across the AD bus, the CBE lines, and the PAR signal. It also reports data parity errors via PERR- and address parity errors via SERR-.

Target State Machine

This block controls the PCI interface for Target functions. The states implemented are a subset of equations defined in “Appendix B” of the *PCI Local Bus Specification*. The controller is a high-performance state machine using one-hot (state-per-bit) encoding for maximum performance. State-per-bit encoding of the next-state logic functions facilitates a high performance implementation in the Xilinx FPGA architecture.

Initiator State Machine

This block controls the PCI interface for Initiator functions. The states implemented are a subset of equations defined in “Appendix B” of the *PCI Local Bus Specification*. The Initiator Control Logic also uses state-per-bit encoding for maximum performance.

User Application with Optional Burst FIFOs

The LogiCORE PCI32 Interface provides a simple, general-purpose interface with a 32-bit data path and latched address for de-multiplexing the PCI address/data bus. The general-purpose user interface allows the rest of the device to be used in a wide range of 32-bit applications.

Typically, the user application contains burst FIFOs to increase PCI system performance. An on-chip read/write FIFO, built from the on-chip synchronous dual-port RAM (Block SelectRAM+™) available in Virtex Series FPGAs, supports data transfers in excess of 66 MHz.

Several synthesizable re-usable bridge designs including commonly used backend functions, such as doorbells and mailboxes, are provided with the core.

Interface Configuration

The LogiCORE PCI32 Interface can easily be configured to fit unique system requirements by using Xilinx web-based PCI configuration tool or by changing the Verilog or VHDL configuration file. The following customization options are supported by the LogiCORE product and described in product documentation.

- Initiator or target functionality
- Base Address Register configuration (1 - 3 Registers, size and mode)
- Configuration Space Header ROM
- Initiator and target state machine (e.g., termination conditions, transaction types and request/transaction arbitration)
- Burst functionality
- User Application including FIFO (back-end design)

Supported PCI Commands

Table 2 illustrates the PCI bus commands supported by the LogiCORE™ PCI32 Interface. The PCI Compliance Checklist has more details on supported and unsupported commands.

Table 2: PCI Bus Commands

CBE [3:0]	Command	PCI Master	PCI Slave
0000	Interrupt Acknowledge	Yes	Yes
0001	Special Cycle	Yes	Ignore
0010	I/O Read	Yes	Yes
0011	I/O Write	Yes	Yes
0100	Reserved	No	Ignore
0101	Reserved	No	Ignore
0110	Memory Read	Yes	Yes
0111	Memory Write	Yes	Yes
1000	Reserved	No	Ignore
1001	Reserved	No	Ignore
1010	Configuration Read	Yes	Yes
1011	Configuration Write	Yes	Yes
1100	Memory Read Multiple	Yes	Yes
1101	Dual Address Cycle	No[1]	Ignore
1110	Memory Read Line	Yes	Yes
1111	Memory Write Invalidate	No[1]	Yes

Note:
1. The Initiator can present these commands, however, they either require additional user-application logic to support them or have not been thoroughly tested.

Burst Transfer

The PCI bus derives its performance from its ability to support burst transfers. The performance of any PCI application depends largely on the size of the burst transfer. A FIFO to support PCI burst transfer can efficiently be implemented using the Virtex on-chip RAM features, both Distributed SelectRAM and Block SelectRAM+™.

Each Virtex CLB supports four 16x1 RAM blocks. This corresponds to 64 bits of single-ported RAM or 32 bits of dual-ported RAM, with simultaneous read/write capability.

Each Virtex device has two columns of Block RAM. The V300 device has 65,536 bits of Block SelectRAM+™ that can be used to create deep, dual-ported FIFOs.

Bandwidth

Xilinx LogiCORE PCI32 Interface supports fully compliant zero wait-state burst operations for both sourcing and receiving data. This Interface supports a sustained bandwidth of up to 128 MBytes/sec. The design can be configured to take advantage of the ability of the LogiCORE PCI32 Interface to do very long bursts. Since the FIFO is not of fixed size, bursts can go on for as long as the chipset arbiter will allow. Furthermore, since the FIFOs and DMA are decoupled from the proven core, a designer can modify these functions without affecting the critical PCI timing.

The flexible Xilinx backend, combined with support for many different PCI features, gives users a solution that lends itself to being used in many high-performance applications. Xilinx is able to support different depths of FIFOs as well as dual port FIFOs, synchronous or asynchronous FIFOs and multiple FIFOs. The user is not locked into one DMA engine, hence, a DMA that fits a specific application can be designed.

The theoretical maximum bandwidth of a 32-bit, 33 MHz PCI bus is 128 MBytes/sec. Attaining this maximum bandwidth will depend on several factors, including the PCI design used, PCI chipset, the processor's ability to keep up with your data stream, the maximum capability of your PCI design, and other traffic on the PCI bus. Older chipsets and processors will tend to allow less bandwidth than newer ones.

No additional wait-states are inserted in response to a wait-state from another agent on the bus. Either IRDY or TRDY is kept asserted until the current data phase ends, as required by the V2.2 PCI Specification.

See Table 3 for PCI bus transfer rates for various operations.

Table 3: LogiCORE PCI32 Transfer Rates

Zero Wait-State Mode	
Operation	**Transfer Rate**
Initiator Write (PCI ← LogiCORE)	3-1-1-1
Initiator Read (PCI → LogiCORE)	4-1-1-1
Target Write (PCI→ LogiCORE)	5-1-1-1
Target Read (PCI ← LogiCORE)	6-1-1-1

Timing Specification

The Virtex Series FPGA devices, together with the LogiCORE PCI32 product enable design of fully compliant PCI systems. The maximum speed at which your back-end is capable of running can be affected by the size of the design as well as by the loading of the hot signals coming directly from the PCI bus. Table 4 shows the key timing parameters for the LogiCORE PCI32 Interface that must be met for full PCI compliance.

Table 4: 33 MHz Timing Parameters [ns]

Parameter	Ref.	PCI Spec.		LogiCORE PCI32 XCV300-5	
		Min	Max	Min	Max
CLK Cycle Time	T_{CYC}	30	∞	30[1]	∞
CLK High Time	T_{HIGH}	11		11	
CLK Low Time	T_{LOW}	11		11	
CLK to Bus Signals Valid[3]	T_{ICKOF}	2	11	2[2]	11[1]
CLK to REQ# Valid[3]	T_{ICKOF}	2	12	2[2]	12[1]
Tri-state to Active	T_{ON}	2		2[2]	
CLK to Tri-state	T_{OFF}		28		28[1]
Bus Signal Setup to CLK	T_{PSD}		7		7[1]
GNT# Setup to CLK	T_{PSD}		10		10[1]
Input Hold Time After CLK	T_{PHD}		0		0[2]
RST# to Tri-state	$T_{RST-OFF}$		40		40[1]

Notes:
1. Controlled by TIMESPECS, included in product
2. Verified by silicon characterization

Verification Methods

Xilinx has developed a system-level testbench that allows simulation of an open PCI environment in which a LogiCORE-PCI-based design may be tested by itself or with other simulatable PCI agents. Included in these agents are a behavioral host and target, and several "plug-in" modules, including a PCI signal recorder and a PCI protocol monitor. Using these tools, the PCI developers can write microcode-style test scripts that can be used to verify different bus-operation scenarios, including those in the PCI Compliance Checklist.

The Xilinx PCI testbench is a powerful verification tool that is also used as the basis for verification of the PCI LogiCORE. The PCI LogiCORE is also tested in hardware for electrical, functional, and timing compliance.

Ping Reference Design

The Xilinx "PING" Application Example, delivered in Verilog and VHDL, has been developed to provide an easy-to-understand example which demonstrates many of the principles and techniques required to successfully use a LogiCORE PCI32 Interface in a System On A Chip solution. The PING design is also used as a test vehicle when veri-

Synthesizable PCI Bridge Design (SB08)

The synthesizable PCI bridge design, SB08, is an application bridge for use with the LogiCORE PCI32 Interface. It is delivered in Verilog and VHDL and has been fully tested with various devices. This example demonstrates how to interface to the PCI core and provide a modular foundation upon which to base other designs. The reference design can be easily modified to remove select portions of functionality.

This design is a general purpose data transfer engine to be used with the LogiCORE PCI32 Interface. Figure 1 presents a block diagram of the synthesizable PCI bridge design. Typically, the user will customize the local interface to conform to a particular peripheral bus (ISA, VME, i960) or attach to a memory device. The design is modular so that unused portions may be removed. The *Synthesizable PCI Bridge Design Data Sheet* lists the set of features and specifics for the SB08 design.

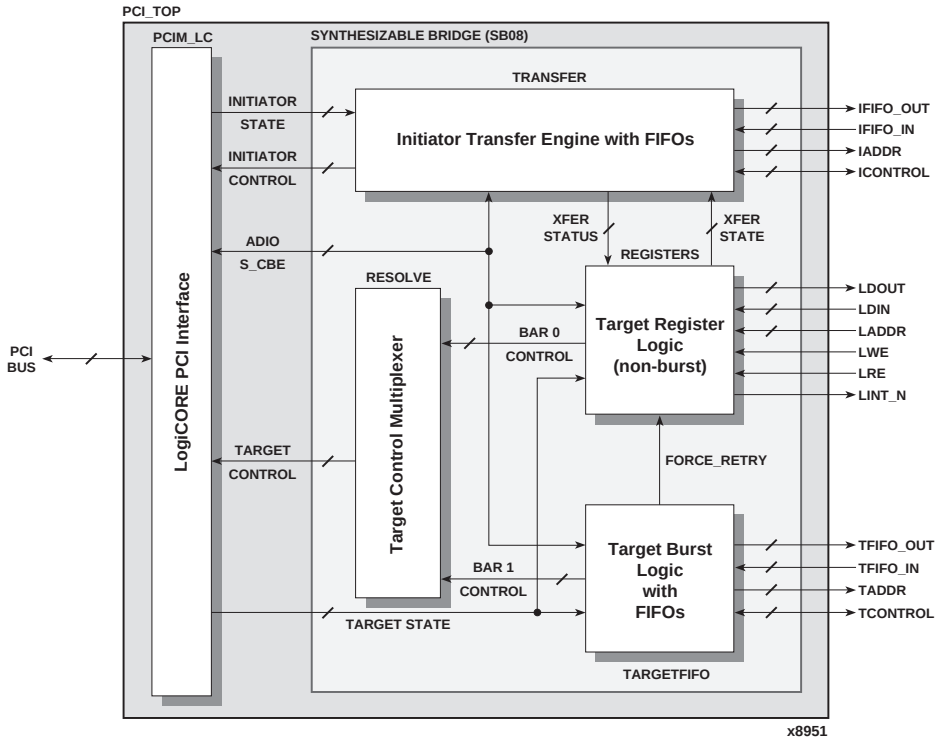


Figure 1: Block Diagram of Synthesizable Bridge Design for PCI32 LogiCORE Interface

Device Utilization

The Target/Initiator options require a variable amount of CLB resources for the PCI32 Interface.

Utilization of the device can vary slightly, depending on the configuration choices made by the designer. Factors that can affect the size of the core are:

- Number of Base Address Registers Used. Turning off any unused BARs will save resources.
- Size of the BARs. Setting the BAR to a smaller size requires more flip-flops. A smaller address space requires more flip-flops to decode.
- Latency timer. Disabling the latency timer will save resources. It must be enabled for bursting.

Recommended Design Experience

The LogiCORE PCI32 Interface is pre-implemented allowing engineering focus at the unique back-end functions of a PCI design. Regardless, PCI is a high-performance system that is challenging to implement in any technology, ASIC or FPGA. Therefore, previous experience with building high-performance, pipelined FPGA designs using Xilinx implementation software, TIMESPECs, and guide files is recommended. The challenge to implement a complete PCI design including back-end functions varies depending on configuration and functionality of your application. Contact your local Xilinx representative for a closer review and estimation for your specific requirements.

PCI32 4000 XLA Interface Version 3.0

May, 1999 Data Sheet

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-377-3259
E-mail: Techsupport:hotline@xilinx.com
Feedback: logicore@xilinx.com
URL: www.xilinx.com/pci

Introduction

With Xilinx LogiCORE PCI32 4000 XLA Interfaces Version 3.0, a designer can build a customized, 32-bit, 33 MHz fully PCI compliant system with the highest possible sustained performance, 132 Mbytes/sec, and up to 124,000 system gates in a XC4000XLA FPGA.

Features

- Fully 2.2 PCI compliant 32-bit, 33 MHz PCI Initiator/Target Interface
- Programmable single-chip solution with customizable back-end functionality
- Pre-defined implementation for predictable timing in Xilinx XC4000XLA FPGAs
- Incorporates Xilinx Smart-IP Technology
- 5 V and 3.3 V operation
- Zero wait-state burst operation
- Fully verified design
  - Tested with Xilinx internal testbench and in hardware (proven in FPGAs and HardWire devices)
- Configurable on-chip dual-port FIFOs can be added for maximum burst speed (see *Xilinx Documents* section)
- Supported Initiator functions
  - Memory Read, Memory Write, Memory Read Multiple (MRM), Memory Read Line (MRL) commands
  - I/O Read, I/O Write commands
  - Configuration Read, Configuration Write commands
  - Bus Parking
  - Special Cycles, Interrupt Acknowledge
  - Basic Host Bridging

LogiCORE™ Facts

Core Specifics	
Device Family	XC4000XLA
CLBs Used[1]	178 - 308
IOBs Used	53
System Clock f_{max}	0 - 33MHz
Device Features Used	Bi-directional data buses SelectRAM™ (optional user FIFO) Boundary scan (optional)

Supported Devices[2]/Resources Remaining		
	I/O	**CLB[1]**
XC4013XLA PQ208	101	268 - 398
XC4013XLA PQ240	135	268 - 398
XC4028XLA HQ240	135	716 - 846
XC4062XLA HQ240	135	1996 - 2126
XC4062XLA BG432	295	1996 - 2126

Provided with Core	
Documentation	*PCI Design Guide* *XLT to XLA Conversion Guide* *PCI Data Book*
Design File Formats	Verilog/VHDL Simulation Model Verilog/VHDL Instantiation Code NGO Netlist
Constraint Files	M1 User Constraint File (UCF) M1 Guide files
Verification Tools	VHDL\Verilog Testbench
Core Symbols	VHDL, Verilog
Reference designs & application notes	Ping Reference Design Synthesizable PCI Bridge Design

Design Tool Requirements	
Xilinx Core Tools	M1.5i
Tested Entry/Verification Tools[3]	For CORE instantiation: Synopsys FPGA Express, Compiler Synplicity Synplify For CORE verification: Cadence Verilog XL MTI ModelSim PE/Plus V4.7g

Support

Xilinx provides technical support for all LogiCORE products when used as described in product documentation. Xilinx cannot guarantee timing, functionality, or support if implemented in unspecified devices or customized beyond that referenced in product documentation, or if changes are made to “DO NOT MODIFY” sections of the design.

1. The exact number of CLBs depends on user configuration of the core and level of resource sharing with adjacent logic. For example, a factor that can affect the size of the design are the number and size of the BARs.
2. Re-targeting the PCI core to an unlisted device or package will void the guarantee of timing. See “Smart-IP Technology - guaranteed timing” on page 19 for details.
3. See Xilinx Web Site for update on tested design tools.

Features (cont.)

- Supported Target functions
  - Type 0 Configuration Space Header
  - Up to 3 Base Address Registers (memory or I/O with adjustable block size from 16 bytes to 2 GBytes, slow or medium decode speed)
  - Parity Generation (PAR), Parity Error Detection (PERR# and SERR#)
  - Memory Read, Memory Write, Memory Read Multiple (MRM), Memory Real Line (MRL), Memory Write, Invalidate (MWI) commands
  - I/O Read, I/O Write commands
  - Configuration Read, Configuration Write commands
  - 32-bit data transfers, burst transfers with linear address ordering
  - Target Abort, Target Retry, Target Disconnect
  - Full Command/Status Register
- Available for configuration and download on the Web
  - Web-based configuration with intuitive GUI
  - Generation of proven design files

Applications

- PCI add-in boards such as graphic cards, video adapters, LAN adapters and data acquisition boards
- Embedded applications within networking, telecommunication, and industrial systems
- CompactPCI boards
- Other applications that need PCI

General Description

The LogiCORE™ PCI32 4000 XLA Interfaces V3.0 are pre-implemented and fully tested modules for Xilinx XC4000XLA FPGAs (see *LogiCORE Facts* for list of supported devices). The pin-out and the relative placement of internal Configurable Logic Blocks (CLBs) are pre-defined. Critical paths are controlled by TimeSpecs and guide files to ensure that timing is always met. This significantly reduces engineering time required to implement the PCI portion of your design. Resources can instead be focused on unique back-end logic in FPGA and on system level design. Consequently, LogiCORE™ PCI products can minimize development time.

Xilinx XC4000XLA Series FPGAs enables designs of fully PCI-compliant systems. The devices meet all required electrical and timing parameters for 3.3V and 5V including AC output drive characteristics, input capacitance specifications (10pF), 7 ns setup and 0 ns hold to system clock, and 11 ns system clock to output.

The XC4000XLA devices have programmable clamp diodes as required by the PCI 3.3V electrical specification. For more details about this see the *XC4000XLA FPGA Data Sheet*.

The PCI Compliance Checklist (See the *Xilinx PCI Databook*) has additional details about electrical compliance. Other features that enable efficient implementation of a complete PCI system in the XC4000XLA include:

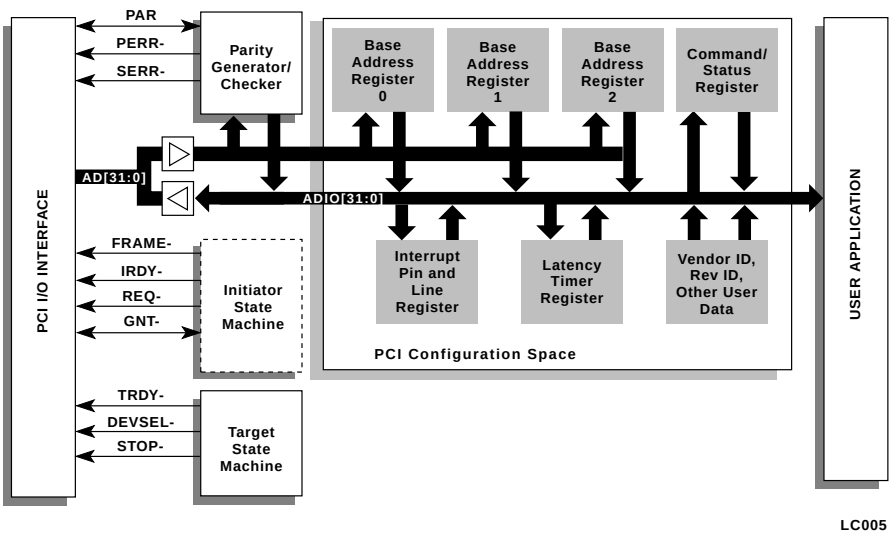


Figure 1: LogiCORE™ PCI32 Interface Block Diagram

- Select-RAM™ memory: on-chip ultra-fast RAM with synchronous write option and dual-port RAM option used in PCI Interfaces to implement the FIFO.
- Individual output enable for each I/O
- Internal 3-state bus capability
- 8 global low-skew clock or signal distribution networks
- IEEE 1149.1-compatible boundary scan logic support

The module is carefully optimized for best possible sustained performance and utilization in the XC4000XLA FPGA architecture. When implemented in a XC4013, more than 50% of the FPGA's resources remain for integrating a unique back-end interface and other system functions into a fully programmable one-chip solution. When implemented in a XC4062, 90% of the FPGA's resources remain.

Smart-IP Technology - guaranteed timing

Drawing on the architectural advantages of Xilinx FPGAs, new Xilinx Smart-IP technology ensures highest performance, predictability, repeatability, and flexibility in PCI designs. The Smart-IP technology is incorporated in every LogiCORE PCI Core.

Xilinx Smart-IP technology leverages the Xilinx architectural advantages, such as look-up tables (LUTs), distributed RAM, and segmented routing, as well as floorplanning information, such as logic mapping and relative location constraints. This technology provides the best physical layout, predictability, and performance. Additionally, these predetermined features allow for significantly reduced compile times over competing architectures.

PCI Cores made with Smart-IP technology are unique by maintaining their performance and predictability regardless of the device size.

To guarantee the critical setup, hold, and min. and max. clock-to-out timing, the PCI core is delivered with Smart-IP constraint files that are unique for a device and package combination. These constraint files guide the implementation tools so that the critical paths always are within PCI specification. Retargeting the PCI core to an unsupported device will void the guarantee of timing. Contact one of the Xilinx XPERTs partners for support of unlisted devices and packages. See the XPERTs section in chapter 7 of the Xilinx PCI Data Book for contact information.

Functional Description

The LogiCORE PCI32 4000 XLA Interfaces are partitioned into five major blocks, and the user application as shown in Figure 1. Each block is described below.

PCI I/O Interface Block

The I/O interface block handles physical connection to the PCI bus including all signaling, input and output synchronization, output three-state controls, and all request-grant handshaking for bus mastering.

Parity Generator/Checker

Generates/checks even parity across the AD bus, the CBE lines, and the PAR signal. Reports data parity errors via PERR- and address parity errors via SERR-.

Target State Machine

This block manages control over the PCI interface for Target functions. The states implemented are a subset of equations defined in "Appendix B" of the *PCI Local Bus Specification*. The controller is a high-performance state machine using state-per-bit (one-hot) encoding for maximum performance. State-per-bit encoding has narrower and shallower next-state logic functions that closely match the Xilinx FPGA architecture.

Initiator State Machine

This block manages control over the PCI interface for Initiator functions. The states implemented are a subset of equations defined in "Appendix B" of the *PCI Local Bus Specification*. The Initiator Control Logic also uses state-per-bit encoding for maximum performance.

PCI Configuration Space

This block provides the first 64 Bytes of Type 0, V 2.1, Configuration Space Header (CSH) (see Table 1) to support software-driven "Plug-and Play" initialization and configuration. This includes Command, Status, and three Base Address Registers (BARs). BAR 2 is not shown in figure 1. These BARs illustrate how to implement memory- or I/O-mapped address spaces. Each BAR sets base address for the interface and allows system software to determine addressable range required by the interface. Using a combination of Configurable Logic Block (CLB) flip-flops for the read/write registers and CLB look-up tables for the read-only registers results in optimized packing density and layout.

With this release, the hooks for extending configuration space has been built into the backend interface. These hooks, including the ability to implement a CapPtr in configuration space, allows the user to implement functions such as Advanced Configuration and Power Interface (ACPI) in the backend design.

Table 1: PCI Configuration Space Header

31	16	15	0	
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Rev ID	08h
BIST	Header Type	Latency Timer	*Cache Line Size*	0Ch
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 (BAR5)				20h
Base Address Register 5 (BAR5)				24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			CapPtr	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch
Reserved				40h-FFh

Note:
Italicized address areas are not implemented in LogiCORE PCI32 4000 XLA Interface default configuration. These locations return zero during configuration read accesses.

User Application with Optional Burst FIFOs

The LogiCORE PCI32 4000 XLA Interface is a general-purpose interface with a 32-bit data path and latched address for de-multiplexing the PCI address/data bus. The general-purpose user interface allows the rest of the device to be used in a wide range of applications.

Typically, the user application contains burst FIFOs to increase PCI system performance (An Application Note is available, please see the *Xilinx Documents* section). An on-chip read/write FIFO, built from the on-chip synchronous dual-port RAM (SelectRAM™) available in XC4000XLA devices, supports data transfers in excess of 33 MHz.

Interface Configuration

The LogiCORE PCI32 4000 XLA Interfaces can easily be configured to fit unique system requirements using Xilinx web-based graphical configuration tool or changing the VHDL or Verilog configuration file. The following customization is supported by the LogiCORE product and described in accompanying documentation.

- Initiator or target functionality (The core can be used as a target-only Interface)
- Base Address Register configuration (1 - 3 Registers, size and mode)
- Configuration Space Header ROM
- Initiator and target state machine (e.g., termination conditions, transaction types and request/transaction arbitration)
- Burst functionality
- User Application including FIFO (back-end design)

Supported PCI Commands

Table 2 illustrates the PCI bus commands supported by the LogiCORE™ PCI32 4000 XLA Interfaces. The PCI Compliance Checklist, found later in this data book, has more details on supported and unsupported commands.

Table 2: PCI Bus Commands

CBE [3:0]	Command	PCI Master	PCI Slave
0000	Interrupt Acknowledge	Yes	Yes
0001	Special Cycle	Yes	Ignore
0010	I/O Read	Yes	Yes
0011	I/O Write	Yes	Yes
0100	Reserved	Ignore	Ignore
0101	Reserved	Ignore	Ignore
0110	Memory Read	Yes	Yes
0111	Memory Write	Yes	Yes
1000	Reserved	Ignore	Ignore
1001	Reserved	Ignore	Ignore
1010	Configuration Read	Yes	Yes
1011	Configuration Write	Yes	Yes
1100	Memory Read Multiple	Yes	Yes
1101	Dual Address Cycle	No[1]	Ignore
1110	Memory Read Line	Yes	Yes
1111	Memory Write Invalidate	No[1]	Yes

Note:
1. The Initiator can present these commands, however, they either require additional user-application logic to support them or have not been thoroughly tested.

Burst Transfer

The PCI bus derives its performance from its ability to support burst transfers. The performance of any PCI application depends largely on the size of the burst transfer. A FIFO to support PCI burst transfer can efficiently be implemented using the XC4000XLA on-chip RAM feature, SelectRAM™. Each XC4000XLA CLB supports two 16x1 RAM blocks. This corresponds to 32 bits of single-ported RAM or 16 bits of dual-ported RAM, with simultaneous read/write capability.

Bandwidth

The Xilinx PCI32 4000 XLA Interfaces support a sustained bandwidth of up to 132 MBytes/sec (except in the XC4062XLA HQ240). The design can be configured to take advantage of the ability of the LogiCORE PCI32 Interface to do very long bursts. Since the FIFO isn't a fixed size, burst can go on as long as the chipset arbiter will allow. Furthermore, since the FIFOs and DMA are decoupled from the proven core, a designer can modify these functions without effecting the critical PCI timing.

The flexible Xilinx backend, combined with support for many different PCI features, gives users a solution that lends itself to being used in many high-performance applications. Xilinx is able to support different depths of FIFOs as well as dual port FIFOs, synchronous or asynchronous FIFOs and multiple FIFOs. The user is not locked into one DMA engine, hence, a DMA that fits a specific application can be designed.

The theoretical maximum bandwidth of a 32 bit, 33 MHz PCI bus is 132 MB/s. How close you get to this maximum will depend on several factors, including the PCI design used, PCI chipset, the processor's ability to keep up with your data stream, the maximum capability of your PCI design and other traffic on the PCI bus. Older chipsets and processors will tend to allow less bandwidth than newer ones.

In this version of the Interface, all devices are zero wait state except for the XC4062XLA HQ240, which is a one wait state design. The XC4013XLA-09, XC4028XLA-09 and XC4062XLA-09 support zero wait-state burst, equal to a sustained bandwidth of up to 132 MBytes/sec. Only the XC4062XLA HQ240 requires one wait-state while sourcing data. See Table 3 for a PCI bus transfer rates for various operations in either zero or one wait-state mode.

Table 3: LogiCORE PCI32 4000 XLA Transfer Rates

Zero Wait-State Mode	
Operation	**Transfer Rate**
Initiator Write (PCI ← LogiCORE)	3-1-1-1
Initiator Read (PCI → LogiCORE)	4-1-1-1
Target Write (PCI→ LogiCORE)	5-1-1-1
Target Read (PCI ← LogiCORE)	6-1-1-1
One Wait-State Mode (XC4062XLA HQ240 only)	
Operation	**Transfer Rate**
Initiator Write (PCI ← LogiCORE)	3-2-2-2
Initiator Read (PCI → LogiCORE)	4-1-1-1
Target Write (PCI→ LogiCORE)	5-1-1-1
Target Read (PCI ← LogiCORE)	6-2-2-2

Note: Initiator Read and Target Write operations have effectively the same bandwidth for burst transfer.

In the Zero wait-state mode, no wait-states are inserted either while sourcing data or receiving data. This allows a 100% burst transfer rate in both directions with full PCI compliance. No additional wait-states are inserted in response to a wait-state from another agent on the bus. Either IRDY or TRDY is kept asserted until the current data phase ends, as required by the V2.2 PCI Specification.

In one wait-state mode, the LogiCORE PCI32 4000 XLA Interface automatically inserts a wait-state when sourcing data (Initiator Write, Target Read) during a burst transfer. In this mode, the LogiCORE PCI32 4000 XLA Interface can accept data at 100% burst transfer rate and supply data at 50%.

Timing Specification

The XC4000XLA family, together with the LogiCORE PCI32 products enables design of fully compliant PCI systems. Backend design can affect the maximum speed your design is capable of. Factors in your back-end designs that can affect timing include loading of hot signals coming directly from the PCI bus, and gate count. Table 4 shows the key timing parameters for the LogiCORE PCI32 Interfaces that must be met for full PCI compliance.

Table 4: Timing Parameters [ns]

Parameter	Ref.	PCI Spec.		LogiCORE PCI32 4000 XLA XC4000XLA-1	
		Min	Max	Min	Max
CLK Cycle Time		30	∞	30[1]	∞
CLK High Time		11		11	
CLK Low Time		11		11	
CLK to Bus Signals Valid[3]	T_{ICKOF}	2	11	2[2]	8.5
CLK to REQ# and GNT# Valid[3]	T_{ICKOF}	2	12	2[2]	11
Tri-state to Active		2		2[2]	
CLK to Tri-state			28		28[1]
Bus Signal Setup to CLK (IOB)	T_{PSD}		7		7
Bus Signal Setup to CLK (CLB)			7		7[1]
GNT# Setup to CLK	T_{PSD}		10		7
GNT# Setup to CLK (CLB)	T_{PSD}		10		10
Input Hold Time After CLK (IOB)	T_{PHD}		0		0
Input Hold Time After CLK (CLB)			0		0[2]
RST# to Tri-state			40		40[2]

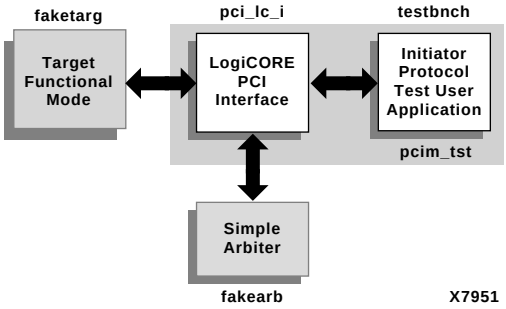
Notes:
1. Controlled by TIMESPECS, included in product
2. Verified by analysis and bench-testing
3. IOB configured for Fast slew rate

Verification Methods

Xilinx has developed a testbench with numerous vectors to test the Xilinx PCI design; this is included with the LogiCORE PCI32 4000 XLA Interfaces. A version of this testbench is also used internally by the Xilinx PCI team to verify the PCI32 Interfaces. Additionally, the PCI32 Interfaces have been tested in hardware for electrical, functional and timing compliance.

The testbench shipped with the interface verifies the PCI interface functions according to the test scenarios specified in the *PCI Local Bus Specification, V2.1*; see Figure 2. This testbench consists of 28 test scenarios, each designed to test a specific PCI bus operation. Refer to the checklists chapter in this databook for a complete list of scenarios.

Figure 2: PCI Protocol Testbench



Ping Reference Design

The Xilinx LogiCORE PCI “PING” Application Example, delivered in VHDL and Verilog, has been developed to provide an easy-to-understand example which demonstrates many of the principles and techniques required to successfully use a LogiCORE PCI32 4000 XLA Interface in a System On A Chip solution.

Synthesizable PCI Bridge Design Example

Synthesizable PCI bridge design examples, delivered in Verilog and VHDL, are available to demonstrate how to interface to the LogiCORE PCI32 4000 XLA V3.0 Interfaces and provides a modular foundation upon which to base other designs. See separate data sheet for details.

Device Utilization

Utilization can vary widely, depending on the configuration choices made by the designer. Options that can affect the size of the core are:

- Number of Base Address Registers Used. Turning off any unused BARs will save on resources. The core now includes a switch to force the entire deletion of unused Base Address Registers.
- Size of the BARs. Setting the BAR to a smaller size requires more flip-flops. A smaller address space requires more flip-flops to decode.
- Latency timer. Disabling the latency timer will save a few resources. It must be enabled for bursting.

Recommended Design Experience

The LogiCORE PCI32 4000 XLA Interfaces are pre-implemented allowing engineering focus at the unique back-end functions of a PCI design. Regardless, PCI is a high-performance system that is challenging to implement in any technology, ASIC or FPGA. Therefore, we recommend previous experience with building high-performance, pipelined FPGA designs using Xilinx implementation software, TIMESPECs, and guide files. The challenge to implement a complete PCI design including back-end functions varies depending on configuration and functionality of your application. Contact your local Xilinx representative for a closer review and estimation for your specific requirements.

PCI32 SpartanXL Interface Version 3.0

May, 1999 Data Sheet

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-377-3259
E-mail: Techsupport: hotline@xilinx.com
Feedback: logicore@xilinx.com
URL: www.xilinx.com/pci

Introduction

With Xilinx LogiCORE PCI32 SpartanXL Interface, a designer can build a cost-efficient, customizable, zero wait-state, 32-bit, 33MHz fully PCI compliant system in a SpartanXL family FPGA.

Features

- Fully 2.2 PCI compliant 32-bit, 33 MHz PCI Initiator/Target Interface
- Incorporates Xilinx Smart-IP Technology with pre-defined implementation for predictable timing in Xilinx SpartanXL FPGAs (see *LogiCORE Facts* for listing of supported devices)
- 3.3V and 5V operation with SpartanXL devices
- Zero wait-state burst operation
- Fully verified design
  - Tested with Xilinx internal testbench and in hardware (silicon proven)
- Configurable on-chip dual-port FIFOs can be added for maximum burst speed (see *Xilinx Documents* section)
- Programmable single-chip solution with customizable back-end functionality
- Supported Initiator functions
  - Memory Read, Memory Write, Memory Read Multiple (MRM), Memory Read Line (MRL) commands
  - I/O Read, I/O Write commands
  - Configuration Read, Configuration Write commands
  - Bus Parking
  - Special Cycles, Interrupt Acknowledge
  - Basic Host Bridging

LogiCORE™ Facts

Core Specifics	
Device Family	SpartanXL
CLBs Used[1]	152 - 268
IOBs Used	53
System Clock f_{max}	0 – 33MHz
Device Features Used	Bi-directional data buses SelectRAM™ (optional user FIFO) Boundary scan (optional)

Supported Devices[4]/Resources Remaining		
	I/O	CLB[1]
XCS20XL-4 TQ144	60	190 - 248[2]
XCS30XL-4 PQ208	107	308 - 424
XCS30XL-4 PQ240	141	308 - 424
XCS40XL-4 PQ208	107	516 - 632
XCS40XL-4 PQ240	141	516 - 632

Provided with Core	
Documentation	*PCI Design Guide* *SpartanXL Implementation Guide* *Spartan to SpartanXL Conversion Guide* *PCI Data Book*
Design File Formats	VHDL & Verilog Simulation Models NGO Netlist
Constraint Files	M1 User Constraint File (UCF) M1 Guide files
Verification Tools	VHDL and Verilog Testbench
Core Symbols	VHDL, Verilog
Reference designs	Synthesizable PCI Bridge Design

Design Tool Requirements	
Xilinx Core Tools	M1.5i
Tested Entry/Verification Tools[3]	For CORE instantiation: Synopsys FPGA Express, Compiler Synplicity Synplify For CORE verification: Cadence Verilog XL MTI ModelSim PE/Plus V4.7g

Support

Xilinx provides technical support for this LogiCORE™ product when used as described in the Design and Implementation Guides and in the Application Notes. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices not listed above, or if customized beyond that referenced in the product documentation, or if any changes are made in sections of design marked as “DO NOT MODIFY”.

Notes:
1. The exact number of CLBs depends on user configuration of the core and level of resource sharing with adjacent logic. Factors that can affect the size of the design are number and size of the BARs, and use of the latency timer.
2. The XCS20XL device supports up to one BAR.
3. See Xilinx Web Site for updates on tested design tools.
4. Re-targeting the PCI core to an unlisted device or package will void the guarantee of timing. See “Smart-IP Technology - guaranteed timing” on page 11. for details.

Features (cont.)

- Supported Target functions
  - Type 0 Configuration Space Header
  - Up to 2 Base Address Registers (memory or I/O with adjustable block size from 16 Bytes to 2 GBytes, slow decode speed)
  - Parity Generation (PAR), Parity Error Detection (PERR# and SERR#)
  - Extended Capabilities Registers (backend module)
  - Memory Read, Memory Write, Memory Read Multiple (MRM), Memory Real Line (MRL), and Memory Write & Invalidate (MWI) commands
  - I/O Read and I/O Write commands
  - Configuration Read and Configuration Write commands
  - 32-bit data transfers, burst transfers with linear address ordering
  - Target Abort, Target Retry, Target Disconnect
  - Full Command/Status Register
- Available for configuration and download on the web
  - Web-based configuration tool
  - Generation of proven design files
  - Instant access to new releases

Applications

- PCI add-in boards such as graphic cards, video adapters, LAN adapters, and data acquisition boards
- Embedded applications within networking, telecommunication, and industrial systems
- CompactPCI boards
- Other applications that need PCI

General Description

The LogiCORE™ PCI32 SpartanXL Interfaces are pre-implemented and fully tested modules for Xilinx SpartanXL FPGAs (see *LogiCORE Facts* for listing of supported devices). The pin-out and the relative placement of the internal Configurable Logic Blocks (CLBs) are pre-defined. Critical paths are controlled by TimeSpecs and guide files to ensure that timing is always met. This significantly reduces engineering time required to implement the PCI portion of your design. Resources can instead be focused on the unique back-end logic in the FPGA and the system level design. As a result, LogiCORE PCI products can minimize development time.

Xilinx SpartanXL family FPGAs enables the design of fully PCI compliant systems. These devices meet all specifications for 3.3 V and 5 V PCI and meet all required electrical and timing parameters including AC output drive characteristics, input capacitance specifications (10pF), 7 ns setup and 0 ns hold to system clock, and 11 ns system clock to output.

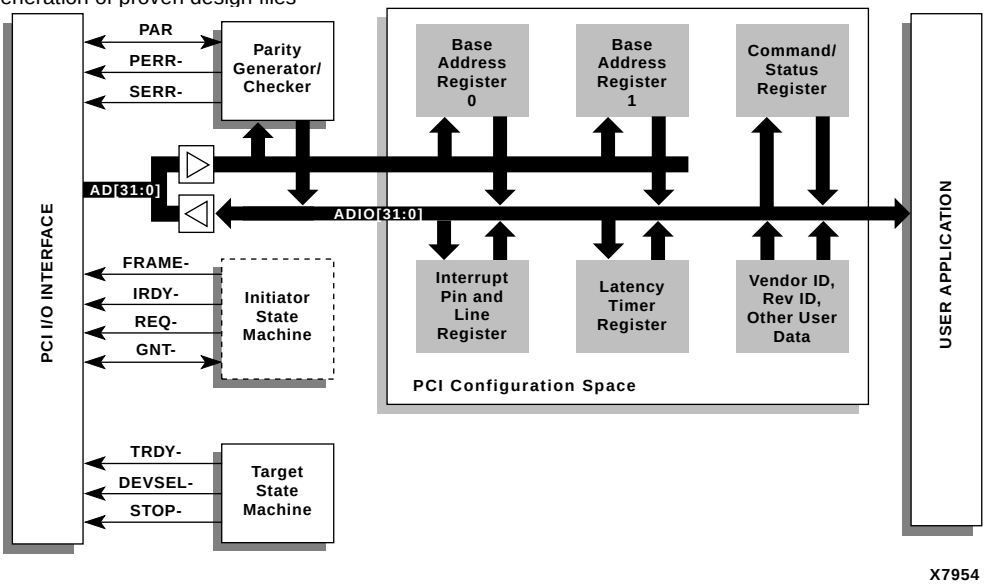


Figure 1: LogiCORE PCI32 SpartanXL Interface Block Diagram (one BAR only in XCS20XL)

The PCI Compliance Checklists, found in the Xilinx PCI Databook, have additional details. Other features that enable efficient implementation of a complete PCI system in the SpartanXL family includes:

- Select-RAM™ memory: on-chip ultra-fast RAM with synchronous write option and dual-port RAM option. Used in the PCI32 SpartanXL Interface to implement the FIFO.
- Individual output enable for each I/O
- Internal 3-state bus capability
- 8 global low-skew clock or signal distribution networks
- IEEE 1149.1-compatible boundary scan logic support

See *Spartan FPGA Data Sheet* for more details.

The module is carefully optimized for best possible performance and utilization in the SpartanXL FPGA architecture. When implemented in the XCS30, more than 50% of the FPGA's resources remain for integrating a unique back-end interface and other system functions into a fully programmable one-chip solution. When implemented in the XCS40, more than 65% of the FPGA's resources remain for integrating a unique back-end interface and other system functions into a fully programmable one-chip solution.

Smart-IP Technology - guaranteed timing

Drawing on the architectural advantages of Xilinx FPGAs, new Xilinx Smart-IP technology ensures highest performance, predictability, repeatability, and flexibility in PCI designs. The Smart-IP technology is incorporated in every LogiCORE PCI Core.

Xilinx Smart-IP technology leverages the Xilinx architectural advantages, such as look-up tables (LUTs), distributed RAM, and segmented routing, as well as floorplanning information, such as logic mapping and relative location constraints. This technology provides the best physical layout, predictability, and performance. Additionally, these predetermined features allow for significantly reduced compile times over competing architectures.

PCI Cores made with Smart-IP technology are unique by maintaining their performance and predictability regardless of the device size.

To guarantee the critical setup, hold, and min. and max. clock-to-out timing, the PCI core is delivered with Smart-IP constraint files that are unique for a device and package combination. These constraint files guide the implementation tools so that the critical paths always are within PCI specification. Retargeting the PCI core to an unsupported device will void the guarantee of timing. Contact one of the Xilinx XPERTs partners for support of unlisted devices and packages. See the XPERTs section in chapter 7 of the Xilinx PCI Data Book for contact information.

Functional Description

The LogiCORE PCI32 SpartanXL Interface is partitioned into five major blocks, plus the user application, shown in Figure 1. Each block is described below.

PCI I/O Interface Block

The I/O interface block handles the physical connection to the PCI bus including all signaling, input and output synchronization, output three-state controls, and all request-grant handshaking for bus mastering.

Parity Generator/Checker

Generates/checks even parity across the AD bus, the CBE lines, and the PAR signal. Reports data parity errors via PERR- and address parity errors via SERR-.

Target State Machine

This block manages control over the PCI32 SpartanXL Interface for Target functions. The states implemented are a subset of equations defined in "Appendix B" of the *PCI Local Bus Specification*. The controller is a high-performance state machine using state-per-bit (one-hot) encoding for maximum performance. State-per-bit encoding has narrower and shallower next-state logic functions that closely match the Xilinx FPGA architecture.

Initiator State Machine

This block manages control over the PCI32 SpartanXL Interface for Initiator functions. The states implemented are a subset of equations defined in "Appendix B" of the *PCI Local Bus Specification*. The Initiator Control Logic also uses state-per-bit encoding for maximum performance.

PCI Configuration Space

This block provides the first 64 bytes of Type 0, version 2.1, Configuration Space Header (CSH) (see Table 1) to support software-driven "Plug-and Play" initialization and configuration. This includes Command, Status, and two Base Address Registers (BARs). These BARs illustrate how to implement memory- or I/O-mapped address spaces. Each BAR sets the base address for the interface and allows the system software to determine the addressable range required by the interface. Using a combination of Configurable Logic Block (CLB) flip-flops for the read/write registers and CLB look-up tables for the read-only registers results in optimized packing density and layout.

With this release, the hooks for extending configuration space has been built into the backend interface. Setting the CapPtr and bit 15 of the Status Register allows the user to implement functions such as Advanced Configuration and Power Interface (ACPI) in the backend design.

User Application with Optional Burst FIFOs

The LogiCORE PCI32 SpartanXL Interface provides a simple, general-purpose interface with a 32-bit data path and latched address for de-multiplexing the PCI address/data bus. The general-purpose user interface allows the rest of the device to be used in a wide range of applications.

Typically, the user application contains burst FIFOs to increase PCI system performance (An Application Note is available, please see the *Xilinx Documents* section). An on-chip read/write FIFO, built from the on-chip synchronous dual-port RAM (SelectRAM™) available in SpartanXL devices, supports data transfers in excess of 33 MHz.

Table 1: PCI Configuration Space Header

31 16	15 0	
Device ID	Vendor ID	00h
Status	Command	04h
Class Code	Rev ID	08h
BIST / Header Type / Latency Timer	*Cache Line Size*	0Ch
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 (BAR5)		20h
Base Address Register 5 (BAR5)		24h
Cardbus CIS Pointer		28h
Subsystem ID	Subsystem Vendor ID	2Ch
Expansion ROM Base Address		30h
Reserved	CapPtr	34h
Reserved		38h
Max_Lat / Min_Gnt / Interrupt Pin	Interrupt Line	3Ch
Reserved		40h-FFh

Note:
Italicized address areas are not implemented in the LogiCORE PCI32 SpartanXL Interface default configuration. These locations return zero during configuration read accesses.

Interface Configuration

The LogiCORE PCI32 SpartanXL Interface can easily be configured to fit unique system requirements using Xilinx web-based PCI Configuration and Download Tool. The following customization is supported by the LogiCORE product and described in accompanying documentation.

- Initiator and target functionality
- Base Address Register configuration (1-2 Registers in XCS30XL and XCS40XL, 1 BAR only in XCS20XL, size and mode of BAR)
- Configuration Space Header ROM
- Initiator and target state machine (e.g., termination conditions, transaction types and request/transaction arbitration)
- Burst functionality
- User Application including FIFO (back-end design)

Supported PCI Commands

Table 2 illustrates the PCI bus commands supported by the LogiCORE PCI32 SpartanXL Interface. The compliance checklist later in this data book have more details on supported and unsupported commands.

Table 2: PCI Bus Commands

CBE [3:0]	Command	PCI Master	PCI Slave
0000	Interrupt Acknowledge	No[1]	Ignore
0001	Special Cycle	No[1]	Ignore
0010	I/O Read	Yes	Yes
0011	I/O Write	Yes	Yes
0100	Reserved	Ignore	Ignore
0101	Reserved	Ignore	Ignore
0110	Memory Read	Yes	Yes
0111	Memory Write	Yes	Yes
1000	Reserved	Ignore	Ignore
1001	Reserved	Ignore	Ignore
1010	Configuration Read	Yes	Yes
1011	Configuration Write	Yes	Yes
1100	Memory Read Multiple	Yes	Yes
1101	Dual Address Cycle	No[1]	Ignore
1110	Memory Read Line	Yes	Yes
1111	Memory Write Invalidate	No[1]	Yes

Note:
1. The Initiator can present these commands, however, they either require additional user-application logic to support them or have not been thoroughly tested.

Burst Transfer

The PCI bus derives its performance from its ability to support burst transfers. The performance of any PCI application depends largely on the size of the burst transfer. A FIFO to support PCI burst transfer can efficiently be implemented using the SpartanXL on-chip RAM feature, SelectRAM™. Each SpartanXL CLB supports two 16x1 RAM blocks. This corresponds to 32 bits of single-ported RAM or 16 bits of dual-ported RAM, with simultaneous read/write capability.

Bandwidth

The Xilinx PCI32 SpartanXL Interface supports a sustained bandwidth of up to 132 MBytes/sec. The design can be configured to take advantage of the ability of the LogiCORE PCI32 Interface to do very long bursts. Since the FIFO does not have a fixed size, a burst can go on for as long as the chipset arbiter will allow. Furthermore, since the FIFOs and the DMA are decoupled from the proven core, a designer can modify these functions without affecting the critical PCI timing.

The flexible Xilinx backend, combined with support for many different PCI features, gives users a solution that can be used in many high-performance applications. Xilinx is able to support different depths of FIFOs as well as dual port FIFOs, synchronous or asynchronous FIFOs, and multiple FIFOs. The user is not restricted to one DMA engine, hence, a DMA that fits a specific application can be designed.

The theoretical maximum bandwidth of a 32-bit, 33 MHz PCI bus is 132 MBytes. How close you get to this maximum bandwidth will depend on several factors, including the PCI design used, PCI chipset, the processor's ability to keep up with your data stream, the maximum capability of your PCI design, and other traffic on the PCI bus. Older chipsets and processors will tend to allow less bandwidth than newer ones.

In the Zero wait-state mode, no wait-states are inserted either while sourcing data or receiving data. This allows a 100% burst transfer rate in both directions with full PCI compliance. No additional wait-states are inserted in response to a wait-state from another agent on the bus, as required by the PCI V 2.2 specification. Either IRDY or TRDY is kept asserted until the current data phase ends, as required by PCI V 2.2 Specification.

In this version of the PCI Interface, based on the Xilinx V3.0 PCI Interface, the end of initiator transaction wait-state has been removed.

See Table 3 for PCI bus transfer rates for various operations in Zero wait-state mode.

Table 3: LogiCORE PCI32 SpartanXL Transfer Rates

Zero Wait-State Mode	
Operation	**Transfer Rate**
Initiator Write (PCI ← LogiCORE)	3-1-1-1
Initiator Read (PCI → LogiCORE)	4-1-1-1
Target Write (PCI→ LogiCORE)	5-1-1-1
Target Read (PCI ← LogiCORE)	6-1-1-1

Note: Initiator Read and Target Write operations have effectively the same bandwidth for burst transfer.

Timing Specification

The SpartanXL family, together with the LogiCORE PCI32 Interface enables design of fully compliant PCI systems. Backend design can affect the maximum speed your design is capable of. Factors in your back-end designs that can affect timing include loading of hot signals coming directly from the PCI bus, gate count and floor planning. Table 4 shows the key timing parameters for the LogiCORE PCI32 SpartanXL Interface that must be met for full PCI compliance.

Verification Methods

Xilinx has developed a testbench with numerous vectors to test the Xilinx PCI design; this is included with the LogiCORE PCI32 SpartanXL Interfaces. A version of this testbench is also used internally by the Xilinx PCI team to verify the PCI32 Interfaces. Additionally, the PCI32 Interfaces have been tested in hardware for electrical, functional and timing compliance.

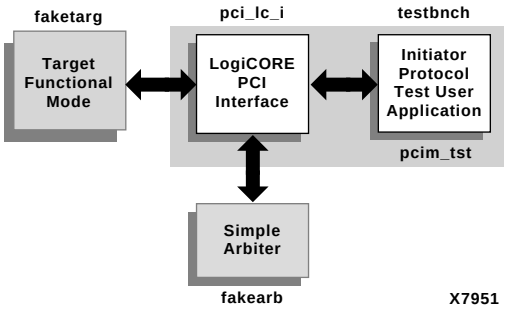
Table 4. Advanced Timing Parameters [ns]

Parameter	Ref.	PCI Spec.		LogiCORE PCI32, XCSXL-4	
		Min	Max	Min	Max
CLK Cycle Time		30	∞	30[1]	∞
CLK High Time		11		11	
CLK Low Time		11		11	
CLK to Bus Signals Valid[3]	T_{ICK-OF}	2	11	2[2]	9.6
CLK to REQ# and GNT# Valid[3]	T_{ICK-OF}	2	12	2[2]	9.6
Tri-state to Active		2		2[2]	
CLK to Tri-state			28		28[1]
Bus Signal Setup to CLK (IOB)	T_{PSU}		7		7
Bus Signal Setup to CLK (CLB)			7		7[1]
GNT# Setup to CLK	T_{PSU}		10		5.2
Input Hold Time After CLK (IOB)	T_{PH}		0		0
Input Hold Time After CLK (CLB)			0		0[2]
RST# to Tri-state			40		40[2]

Notes:
1. Controlled by TIMESPECs, included in product
2. Verified by analysis and bench-testing
3. IOB configured for Fast slew rate

The testbench shipped with the interface verifies the PCI interface functions according to the test scenarios specified in *PCI Compliance Checklist, V 2.1*; see Figure 2. This testbench consists of 28 test scenarios, each designed to test a specific PCI bus operation. Refer to the checklists chapter in this databook for a complete list of scenarios.

Figure 2. PCI Protocol Testbench



Ping Reference Design

The Xilinx LogiCORE PCI “PING” Application Example, delivered in VHDL and Verilog, has been developed to provide an easy-to-understand example which demonstrates many of the principles and techniques required to successfully use a LogiCORE PCI32 Spartan Interface in a System-on-a-Chip solution.

Synthesizable PCI Bridge Design Example

Synthesizable PCI bridge design examples, delivered in Verilog and VHDL, are available to demonstrate how to interface with the LogiCORE PCI32 Spartan Interface and provide a modular foundation upon which to base other designs. See separate data sheet for details.

Device Utilization

The Target-Only and Target/Initiator options require a variable amount of CLB resources for the PCI32 Spartan Interface. The core includes a switch to force the entire deletion of unused Base Address Registers.

Utilization can vary widely, depending on the configuration choices made by the designer. Options that can affect the size of the core are:

- Initiator vs. Target-Only. The Initiator requires about 12 CLBs more than the target (not set in the cfg file; set at the time the core is generated).
- Number of Base Address Registers Used. Turning off any unused BARs will save on resources.
- Size of the BARs. Setting the BAR to a smaller size requires more flip-flops. A smaller address space requires more flip-flops to decode.
- Decode Speed. Medium decode requires slightly more logic than slow decode.
- Latency timer. Disabling the latency timer will save a few resources. It must be enabled for bursting.

Recommended Design Experience

The LogiCORE PCI32 Spartan Interface is pre-implemented allowing engineering focus at the unique back-end functions of a PCI design. Regardless, PCI is a high-performance system that is challenging to implement in any technology, ASIC or FPGA. Therefore, we recommend previous experience with building high-performance, pipelined FPGA designs using Xilinx implementation software, TIMESPECs, and guide files. The challenge to implement a complete PCI design including back-end functions varies depending on configuration and functionality of your application. Contact your local Xilinx representative for details on your specific design requirements.

PCI32 Spartan Master & Slave Interface

May, 1999 Data Sheet

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-377-3259
E-mail: Techsupport: hotline@xilinx.com
Feedback: logicore@xilinx.com
URL: www.xilinx.com

Introduction

With Xilinx LogiCORE PCI32 Spartan Master & Slave Interface, a designer can build a customizable, low-cost 32-bit, 33MHz fully PCI compliant system in a Spartan-family FPGA.

Features

- Fully 2.1 PCI compliant 32 bit, 33MHz PCI Interface
  - Master (Initiator/Target)
  - Slave (Target-only)
- Pre-defined implementation for predictable timing in Xilinx Spartan FPGAs (see *LogiCORE Facts* for listing of supported devices)
- Incorporates Xilinx Smart-IP Technology
- 5 V Operation with Spartan devices
- Zero wait-state burst operation
- Fully verified design
  - Tested with the Xilinx internal testbench
  - Tested in hardware (silicon proven)
- Configurable on-chip dual-port FIFOs can be added for maximum burst speed (see *Xilinx Documents* section)
- Programmable single-chip solution with customizable back-end functionality
- Supported Initiator functions
  - Initiate Memory Read, Memory Write, Memory Read Multiple (MRM), Memory Read Line (MRL) commands
  - Initiate I/O Read, I/O Write commands
  - Initiate Configuration Read, Configuration Write commands
  - Bus Parking

LogiCORE™ Facts

Core Specifics	
Device Family	XCS Spartan
CLBs Used[1]	152 - 268
IOBs Used	53
System Clock f_{max}	0 – 33MHz
Device Features Used	Bi-directional data buses SelectRAM™ (optional user FIFO) Boundary scan (optional)

Supported Devices/Resources Remaining		
	I/O	**CLB[1]**
XCS30 PQ208	107	308 - 424
XCS30 PQ240	141	308 - 424
XCS40 PQ208	107	516 - 632
XCS40 PQ240	141	516 - 632

Provided with Core	
Documentation	*PCI32 User's Guide* *PCI Data Book*
Design File Formats	VHDL, Verilog Simulation Models NGO Netlist[2]
Constraint Files	M1 User Constraint File (UCF) M1 Guide files
Verification Tools	VHDL and Verilog Testbench
Core Symbols	VHDL, Verilog
Reference designs	Synthesizable PCI Bridge Design

Design Tool Requirements	
Xilinx Core Tools	M1.4
Entry/Verification Tools[4]	VHDL, Verilog

Support

Xilinx provides technical support for this LogiCORE product when used as described in the User's Guide or supporting Application Notes. Xilinx cannot guarantee timing, functionality, or support of the product if implemented in devices not listed above, or customized beyond that referenced in the product documentation.

Notes:
1. The exact number of CLBs depends on user configuration of the core and level of resource sharing with adjacent logic. Factors that can affect the size of the design are number and size of the BARs, and medium vs. slow decode. These numbers include a 16 x 32 FIFO.
2. Available on Xilinx Home Page, in the LogiCORE PCI Lounge: www.xilinx.com/products/logicore/pci/pci_sol.htm
3. See Xilinx Home Page for supported EDA tools

Features (cont.)

- Supported Target functions
  - Type 0 Configuration Space Header
  - Up to 2 Base Address Registers (memory or I/O with adjustable block size from 16 bytes to 2 GBytes, slow decode speed)
  - Parity Generation (PAR), Parity Error Detection (PERR# and SERR#
  - ACPI Configuration Registers (backend module)
  - Memory Read, Memory Write, Memory Read Multiple (MRM), Memory Real Line (MRL), Memory Write, Invalidate (MWI) commands
  - I/O Read, I/O Write commands
  - Configuration Read, Configuration Write commands
  - 32-bit data transfers, burst transfers with linear address ordering
  - Target Abort, Target Retry, Target Disconnect
  - Full Command/Status Register
- Available for configuration and download on the web
  - Web-based configuration
  - Generation of proven design files

Applications

- PCI add-in boards such as graphic cards, video adapters, LAN adapters and data acquisition boards.
- Embedded applications within telecommunication and industrial systems.
- CompactPCI boards,
- Other applications that need PCI

General Description

The LogiCORE™ PCI32 Spartan Master and Slave Interfaces are pre-implemented and fully tested modules for Xilinx Spartan FPGAs (see *LogiCORE Facts* for listing of supported devices). The pin-out and the relative placement of the internal Configurable Logic Blocks (CLBs) are predefined. Critical paths are controlled by TimeSpecs and guide files to ensure that timing is always met. This significantly reduces engineering time required to implement the PCI portion of your design. Resources can instead be focused on the unique back-end logic in the FPGA and the system level design. As a result, the LogiCORE PCI products can cut your development time by several months.

Xilinx Spartan Series FPGAs enables designs of fully PCI compliant systems. The devices meet all required electrical and timing parameters including AC output drive characteristics, input capacitance specifications (10pF), 7 ns setup and 0 ns hold to system clock, and 11 ns system clock to output. These devices meet all specifications for 5 V PCI.

The PCI Compliance Checklists, both device and protocol, and waveforms for this core, can be found in 1998 *Xilinx PCI Data Book* at:

`www.xilinx.com/products/logicore/pci/docs/pci_databook_5_98.pdf`

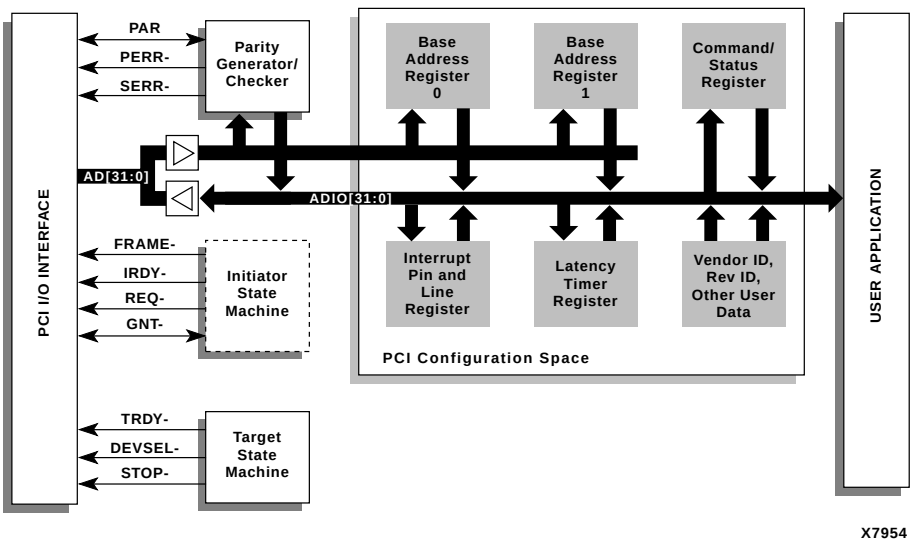


Figure 1: LogiCORE PCI32 Spartan Interface Block Diagram

Other features that enable efficient implementation of a complete PCI system in the Spartan family includes:

- Select-RAM™ memory: on-chip ultra-fast RAM with synchronous write option and dual-port RAM option. Used in the PCI32 Spartan Interface to implement the FIFO.
- Individual output enable for each I/O
- Internal 3-state bus capability
- 8 global low-skew clock or signal distribution networks
- IEEE 1149.1-compatible boundary scan logic support

See *Spartan FPGA Data Sheet* for more details.

The module is carefully optimized for best possible performance and utilization in the Spartan FPGA architecture. When implemented in the XCS30, more than 50% of the FPGA's resources remain for integrating a unique back-end interface and other system functions into a fully programmable one-chip solution. When implemented in the XCS40, more than 65% of the FPGA's resources remain for integrating a unique back-end interface and other system functions into a fully programmable one-chip solution.

Smart-IP Technology

Drawing on the architectural advantages of Xilinx FPGAs, new Xilinx Smart-IP technology ensures highest performance, predictability, repeatability, and flexibility in PCI designs. The Smart-IP technology is incorporated in every LogiCORE PCI Core.

Xilinx Smart-IP technology leverages the Xilinx architectural advantages, such as look-up tables (LUTs), distributed RAM, and segmented routing, and floorplanning information, such as logic mapping and relative location constraints. This technology provides the best physical layout, predictability, and performance. Additionally, these predetermined features allow for significantly reduced compile times over competing architectures.

The PCI32 Spartan Interface can parameterized, allowing for design flexibility in which users can create the exact PCI interface needed. PCI Cores made with Smart-IP technology are unique by maintaining their performance and predictability regardless of the device size.

Functional Description

The LogiCORE PCI32 Spartan Interface is partitioned into five major blocks, plus the user application, shown in Figure 1. Each block is described below.

PCI I/O Interface Block

The I/O interface block handles the physical connection to the PCI bus including all signaling, input and output synchronization, output three-state controls, and all request-grant handshaking for bus mastering.

Parity Generator/Checker

Generates/checks even parity across the AD bus, the CBE lines, and the PAR signal. Reports data parity errors via PERR- and address parity errors via SERR-.

Target State Machine

This block manages control over the PCI32 Spartan Interface for Target functions. The states implemented are a subset of equations defined in "Appendix B" of the *PCI Local Bus Specification*. The controller is a high-performance state machine using state-per-bit (one-hot) encoding for maximum performance. State-per-bit encoding has narrower and shallower next-state logic functions that closely match the Xilinx FPGA architecture.

Initiator State Machine

This block manages control over the PCI32 Spartan Interface for Initiator functions. The states implemented are a subset of equations defined in "Appendix B" of the *PCI Local Bus Specification*. The Initiator Control Logic also uses state-per-bit encoding for maximum performance.

PCI Configuration Space

This block provides the first 64 bytes of Type 0, version 2.1, Configuration Space Header (CSH) (see Table 1) to support software-driven "Plug-and Play" initialization and configuration. This includes Command, Status, and two Base Address Registers (BARs). These BARs illustrate how to implement memory- or I/O-mapped address spaces. Each BAR sets the base address for the interface and allows the system software to determine the addressable range required by the interface. Using a combination of Configurable Logic Block (CLB) flip-flops for the read/write registers and CLB look-up tables for the read-only registers results in optimized packing density and layout.

With this release, the hooks for extending configuration space has been built into the backend interface. Setting the CapPtr and bit 15 of the Status Register allows the user to implement functions such as Advanced Configuration and Power Interface (ACPI) in the backend design.

User Application with Optional Burst FIFOs

The LogiCORE PCI32 Spartan Interface provides a simple, general-purpose interface with a 32-bit data path and latched address for de-multiplexing the PCI address/data bus. The general-purpose user interface allows the rest of the device to be used in a wide range of applications.

Typically, the user application contains burst FIFOs to increase PCI system performance (An Application Note is available, please see the *Xilinx Documents* section). An on-chip read/write FIFO, built from the on-chip synchronous dual-port RAM (SelectRAM™) available in Spartan devices, supports data transfers in excess of 33 MHz

Table 1: PCI Configuration Space Header

31	16	15	0	
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Rev ID	08h
BIST	Header Type	Latency Timer	*Cache Line Size*	0Ch
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 (BAR5)				20h
Base Address Register 5 (BAR5)				24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			CapPtr	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch
Reserved				40h-FFh

Note:
Italicized address areas are not implemented in the LogiCORE PCI32 Spartan Interface default configuration. These locations return zero during configuration read accesses.

Interface Configuration

The LogiCORE PCI32 Spartan Interface can easily be configured to fit unique system requirements using Xilinx web-based PCI Configuration and Download Tool. The following customization is supported by the LogiCORE product and described in accompanying documentation.

- Initiator and target functionality
- Base Address Register configuration (1 - 2 Registers, size and mode)
- Configuration Space Header ROM
- Initiator and target state machine (e.g., termination conditions, transaction types and request/transaction arbitration)
- Burst functionality
- User Application including FIFO (back-end design)

Table 2: PCI Bus Commands

CBE [3:0]	Command	PCI Master	PCI Slave
0000	Interrupt Acknowledge	No[1]	Ignore
0001	Special Cycle	No[1]	Ignore
0010	I/O Read	Yes	Yes
0011	I/O Write	Yes	Yes
0100	Reserved	Ignore	Ignore
0101	Reserved	Ignore	Ignore
0110	Memory Read	Yes	Yes
0111	Memory Write	Yes	Yes
1000	Reserved	Ignore	Ignore
1001	Reserved	Ignore	Ignore
1010	Configuration Read	Yes	Yes
1011	Configuration Write	Yes	Yes
1100	Memory Read Multiple	Yes	Yes
1101	Dual Address Cycle	No[1]	Ignore
1110	Memory Read Line	Yes	Yes
1111	Memory Write Invalidate	No[1]	Yes

Note:
1. The Initiator can present these commands, however, they either require additional user-application logic to support them or have not been thoroughly tested.

Supported PCI Commands

Table 2 illustrates the PCI bus commands supported by the LogiCORE PCI32 Spartan Interface. The compliance checklist later in this data book have more details on supported and unsupported commands.

Burst Transfer

The PCI bus derives its performance from its ability to support burst transfers. The performance of any PCI application depends largely on the size of the burst transfer. A FIFO to support PCI burst transfer can efficiently be implemented using the Spartan on-chip RAM feature, SelectRAM™. Each Spartan CLB supports two 16x1 RAM blocks. This corresponds to 32 bits of single-ported RAM or 16 bits of dual-ported RAM, with simultaneous read/write capability.

Bandwidth

The Xilinx PCI32 Spartan Interface supports a sustained bandwidth of up to 132 MBytes/sec. See the Xilinx web for the supported device/speed grade/wait-states mode combinations.The design can be configured to take advantage of the ability of the LogiCORE PCI32 Interface to do very long bursts. Since the FIFO isn't a fixed size, burst can go on as long as the chipset arbiter will allow. Furthermore, since the FIFOs and DMA are decoupled from the proven core, a designer can modify these functions without effecting the critical PCI timing.

The flexible Xilinx backend, combined with support for many different PCI features, gives users a solution that lends itself to being used in many high-performance applications. Xilinx is able to support different depths of FIFOs as well as dual port FIFOs, synchronous or asynchronous FIFOs and multiple FIFOs. The user is not locked into one DMA engine, hence, a DMA that fits a specific application can be designed.

The theoretical maximum bandwidth of a 32 bit, 33 MHz PCI bus is 132 MB/s. How close you get to this maximum will depend on several factors, including the PCI design used, PCI chipset, the processor's ability to keep up with your data stream, the maximum capability of your PCI design and other traffic on the PCI bus. Older chipsets and processors will tend to allow less bandwidth than newer ones.

In the Zero wait-state mode, no wait-states are inserted either while sourcing data or receiving data. This allows a 100% burst transfer rate in both directions with full PCI compliance. No additional wait-states are inserted in response to a wait-state from another agent on the bus. Either IRDY or TRDY is kept asserted until the current data phase ends, as required by the V2.1 PCI Specification.

In one wait-state mode, the LogiCORE PCI32 Spartan Interface automatically inserts a wait-state when sourcing data (Initiator Write, Target Read) during a burst transfer. In this mode, the LogiCORE PCI32 Spartan Interface can accept data at 100% burst transfer rate and supply data at 50%.

See Table 3 for a PCI bus transfer rates for various operations in either zero or one wait-state mode.

Table 3: LogiCORE PCI32 Spartan Transfer Rates

Zero Wait-State Mode	
Operation	**Transfer Rate**
Initiator Write (PCI ← LogiCORE)	3-1-1-2
Initiator Read (PCI → LogiCORE)	4-1-1-2
Target Write (PCI→ LogiCORE)	5-1-1-1
Target Read (PCI ← LogiCORE)	6-1-1-1
One Wait-State Mode	
Operation	**Transfer Rate**
Initiator Write (PCI ← LogiCORE)	3-2-2-2
Initiator Read (PCI → LogiCORE)	4-1-1-2
Target Write (PCI→ LogiCORE)	5-1-1-1
Target Read (PCI ← LogiCORE)	6-2-2-2

Note: Initiator Read and Target Write operations have effectively the same bandwidth for burst transfer.

Timing Specification

The XCS family, together with the LogiCORE PCI32 Spartan Interface enables design of fully compliant PCI systems. Backend design can affect the maximum speed your design is capable of. Factors in your back-end designs that can affect timing include loading of hot signals coming directly from the PCI bus, gate count and floor planning. Table 4 shows the key timing parameters for the LogiCORE PCI32 Spartan Interface that must be met for full PCI compliance.

Verification Methods

Xilinx has developed a testbench with numerous vectors to test the Xilinx PCI design; this is included with the LogiCORE PCI32 Spartan Master and Slave Interfaces A version of this testbench is also used internally by the Xilinx PCI team to verify the PCI32 Interfaces. Additionally, the PCI32 Interfaces have been tested in hardware for electrical, functional and timing compliance.

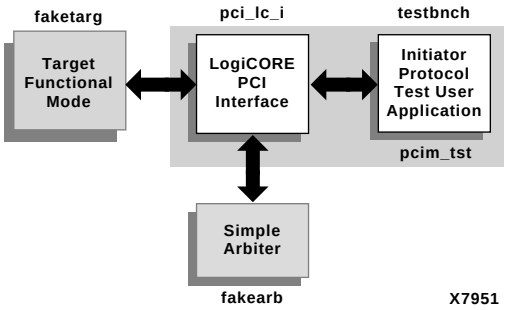
Table 4. Advanced Timing Parameters [ns]

Parameter	Ref.	PCI Spec.		LogiCORE PCI32, XCS-4	
		Min	Max	Min	Max
CLK Cycle Time		30	∞	30[1]	∞
CLK High Time		11		11	
CLK Low Time		11		11	
CLK to Bus Signals Valid[3]	T_{ICKOF}	2	11	2[2]	9.6
CLK to REQ# and GNT# Valid[3]	T_{ICKOF}	2	12	2[2]	9.6
Tri-state to Active		2		2[2]	
CLK to Tri-state			28		28[1]
Bus Signal Setup to CLK (IOB)	T_{PSU}		7		7
Bus Signal Setup to CLK (CLB)			7		7[1]
GNT# Setup to CLK	T_{PSU}		10		5.2
Input Hold Time After CLK (IOB)	T_{PH}		0		0
Input Hold Time After CLK (CLB)			0		0[2]
RST# to Tri-state			40		40[2]

Notes:
1. Controlled by TIMESPECs, included in product
2. Verified by analysis and bench-testing
3. IOB configured for Fast slew rate

The testbench shipped with the interface verifies the PCI interface functions according to the test scenarios specified in the *PCI Local Bus Specification, V2.1*; see Figure 2. This testbench consists of 28 test scenarios, each designed to test a specific PCI bus operation. Refer to the checklists chapter in this databook for a complete list of scenarios.

Figure 2. PCI Protocol Testbench



Ping Reference Design

The Xilinx LogiCORE PCI “PING” Application Example, delivered in VHDL and Verilog, has been developed to provide an easy-to-understand example which demonstrates many of the principles and techniques required to successfully use a LogiCORE PCI32 Spartan Interface in a System On A Chip solution.

Synthesizable PCI Bridge Design Example

Synthesizable PCI bridge design examples, delivered in Verilog and VHDL, are available to demonstrate how to interface to the LogiCORE PCI32 Spartan Interface and provides a modular foundation upon which to base other designs. See separate data sheet for details.

Device Utilization

The Target-Only and Target/Initiator options require a variable amount of CLB resources for the PCI32 Spartan Interface. The core includes a switch to force the entire deletion of unused Base Address Registers.

Utilization can vary widely, depending on the configuration choices made by the designer. Options that can affect the size of the core are:

- Initiator vs. Target-Only. The Initiator requires about 12 CLBs more than the target (not set in the cfg file; set at the time the core is generated).
- Number of Base Address Registers Used. Turning off any unused BARs will save on resources.
- Size of the BARs. Setting the BAR to a smaller size requires more flip-flops. A smaller address space requires more flip-flops to decode.
- Decode Speed. Medium decode requires slightly more logic than slow decode.
- Latency timer. Disabling the latency timer will save a few resources. It must be enabled for bursting.

Recommended Design Experience

The LogiCORE PCI32 Spartan Interface is pre-implemented allowing engineering focus at the unique back-end functions of a PCI design. Regardless, PCI is a high-performance system that is challenging to implement in any technology, ASIC or FPGA. Therefore, we recommend previous experience with building high-performance, pipelined FPGA designs using Xilinx implementation software, TIMESPECs, and guide files. The challenge to implement a complete PCI design including back-end functions varies depending on configuration and functionality of your application. Contact your local Xilinx representative for a closer review and estimation for your specific requirements.

Reference DESIGN

Synthesizable PCI Bridge Design Examples

May, 1999 Data Sheet

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-377-3259
E-mail: Techsupport: hotline@xilinx.com
Feedback: logicore@xilinx.com
URL: www.xilinx.com/pci

Introduction

This synthesizable PCI bridge designs are a set of re-usable Reference Designs for use with the LogiCORE PCI64 and PCI32 Interfaces. They are delivered in Verilog and VHDL and have been tested with various devices. These examples demonstrate how to interface to the PCI core and provide a modular foundation upon which to base other designs. The Reference Designs can be easily modified to remove select portions of functionality. The facts table lists the set of features and specifics for each design.

General Description

Part of or all of the design is available at no cost to all registered LogiCORE PCI Interface customers, who can download it from the LogiCORE PCI Lounges at

www.xilinx.com/pci

See the Ordering Information chapter for details.

These designs are general purpose data transfer engines to be used with the LogiCORE PCI Interfaces. Figure 1 presents a block diagram of the bridge design. Typically, the user will customize the local interface to conform to a particular peripheral bus (ISA, VME, i960) or attach to a memory device. The design is modular so that unused portions may be removed. Some versions are subsets of the complete design, and do not contain parts of the target functionality as indicated in the facts table.

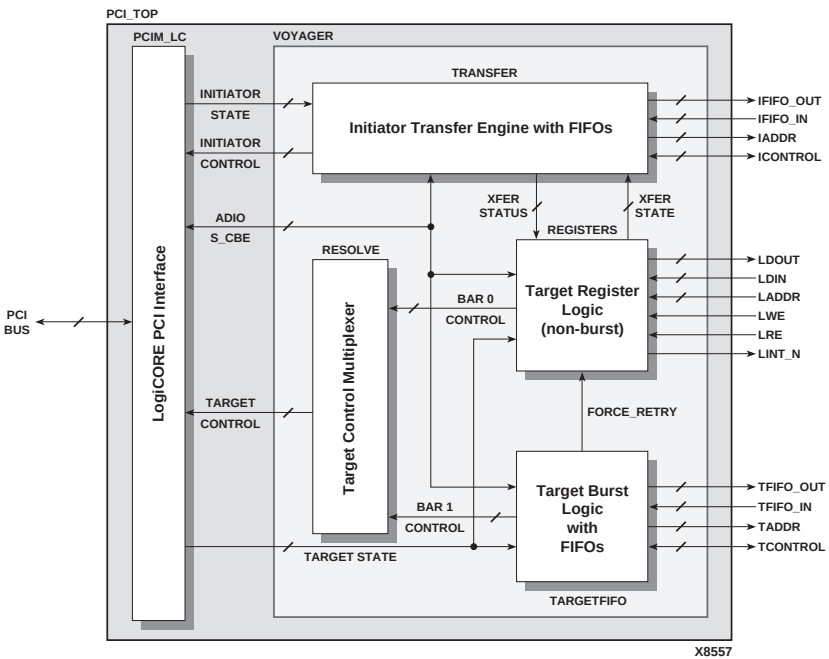


Figure 1: Synthesizable PCI Bridge Block diagram

Design Example Facts

Features	SB01	SB02	SB03	SB07	SB08
Initiator Functions					
Separate read and write FIFOs (unidirectional)	✔	✔	✔	✔	✔
Block data transfer engine (DMA)	✔	✔	✔	✔	✔
Programmable Burst sizes fixed by transfer counter	✔	✔	✔	✔	✔
Auto data delivery (handles terminations)	✔	✔	✔	✔	✔
Discard counter to prevent deadlock	✔	✔	✔	✔	✔
Initiator address counter	✔	✔	✔	✔	✔
Target Functions					
BAR 0 - Supports single data phase transfers	✔	✔	✔	TBD	TBD
BAR 0 - Region 1 demonstrates doorbells	✔				
BAR 0 - Region 2 demonstrates mailboxes	✔				
BAR 0 - Region 3 demonstrates long latency accesses	✔				
BAR 0 - Region 4 contains control registers for initiator	✔	✔	✔		
BAR 1 - Separate read and write FiFOs (unidirectional)	✔				
BAR 1 - Delayed completion discard after time-out	✔				
BAR 1 - Generates target abort on address wrap	✔				
Target address counter	✔	✔	✔		
Target functions independent of initiator	✔	✔	✔		
Bus Width					
32 Bit	✔	✔	✔		✔
64 Bit				✔	
Bus Frequency					
33 MHz	✔	✔	✔	✔	✔
66 MHz				✔	
HDL Support					
Verilog	✔	✔	✔	✔	✔
VHDL	✔	✔	✔		
Supported Families/LogiCORE Product					
SpartanXL / PCI32 V3.0			✔		
XLA / PCI32 V2.0	✔	✔	✔		
Virtex / PCI32 V3.0					✔
Virtex / PCI64 V3.0				✔	
Device Features Used					
Distributed RAM FIFOs	✔	✔	✔		
Block RAM FIFOs				✔	✔
Resource Utilization					
CLBs Used[1]	Up to 990	Up to 530	Up to 530	TBD	TBD

1. The CLB count includes the full design and PCI interface. Actual count depends on the implemented feature set. The bridge design does not use any I/O.

Functional Description

This design example supports target functionality in two memory spaces. Initiator functionality is controlled by writing into registers. The local bus interface signals are distinct for each block in the design, allowing blocks to be added or removed. Data transfer is pipelined for high clock rate. The functional description listed here describes the entire synthesizable bridge design, certain versions will contain a subset of this functionality as listed in the facts table.

BAR0 Configuration

BAR0 is configured as a 4 kilobyte MEM space which maps to a number of registers. This space does not support multiple data phase transfers. All accesses to this space terminate with target disconnect with data.

This space is logically divided into four regions based on functionality. The four regions, and the functions of the registers, are discussed below.

Region One: Doorbells

Register DBELL_P1 is a PCI-to-local doorbell. A PCI agent may create an interrupt on the local side by setting any bit of the register. A PCI agent is permitted to read back the status of this register with no side-effects.

When the local side services the interrupt, it reads this register to determine the cause of the interrupt, then clears the interrupt by writing a one to that bit. The local side may read this register without side-effects.

Similarly, DBELL_L1 is a local-to-PCI doorbell. To prevent spurious interrupts, an interrupt may not be cleared by the agent that requested it. The recipient of the interrupt must clear the interrupt. To enforce this, doorbell register bits may not be cleared from the requesting side. Before doorbell interrupts may occur, the doorbell interrupt enable bits in the CONTROL register must be set.

Region Two: Mailboxes

Register MBOX_P1 is a PCI-to-local mailbox. A PCI agent may deliver mail to an empty mailbox for a local agent to pick up. When a PCI agent writes to this register, the data is registered and a “full” flag is set. Subsequent writes to a full mailbox have no effect. The PCI agent may not read back delivered mail. Reads of the mailbox from the PCI bus side return the state of the full flag (replicated in all bits).

When the local side reads the mailbox, the “full” flag is cleared. Subsequent reads of an empty mailbox return the last valid data present in the mailbox.

Similarly, MBOX_L1 is a local-to-PCI mailbox. The “full” flag may be monitored in two ways. Mailbox “full” flags are always observable in the CONTROL register, so both PCI agents and local agents may poll the CONTROL register to watch for new messages. Optionally, full mailboxes may create interrupts. Interrupts are created on the recipient’s side, and are cleared by reading the mailbox. Before mailbox interrupts may occur, the mailbox interrupt enable bits in the CONTROL register must be set.

Region Three: Bounded Latency Accesses

The two registers in this region are used for demonstrating bounded latency non-burst accesses. This type of access may be used in situations where the user application has a short latency with a known upper bound of 16 PCI clocks from the time the initiator asserts FRAME#. This is done by inserting wait states until the target is capable of completing the transaction.

Register BL_CTRL controls the initial latency of read and write operations for itself and BL_DATA. Only the least-significant four bits of the register are implemented, and the register is only accessible from the PCI bus. The local side has no access to this register, so local reads will return all zeroes and writes have no effect.

The second register, BL_DATA, is a general purpose, read/write register that responds according to the settings in BL_CTRL. This data register is only accessible from the PCI bus. The local side has no access to this register, so local reads will return all zeroes and writes have no effect.

Region Four: Control Registers

The first three registers in this region control the initiator transfer engine.

Register XFER_LEN is used to indicate the length of the data block to be transferred. The low half of the register is not implemented. The high half is implemented as a loadable 16-bit counter.

This register is accessible from both the PCI bus and local bus, and may be read at any time. Status of transfers in progress may be obtained by reading this register. With each successful transfer, the counter decrements.

Register XFER_PADR contains the current PCI bus address for transfers performed by the transfer engine. Depending on the direction of the transfer, this address may be a source or destination.

This register is accessible from both the PCI bus and local bus, and may be read at any time. Status of transfers in progress may be obtained by reading this register. With each successful transfer, the address increments.

Register XFER_LADR contains the current local address for transfers performed by the transfer engine. Depending on the direction of the transfer, this address may be a source or destination. Only the low half of this register is implemented.

This register is accessible from both the PCI bus and local bus, and may be read at any time. Status of transfers in progress may be obtained by reading this register. With each successful transfer, the address increments.

These registers must not be written to while the initiator is active. To ensure this does not occur, writes to these register are disabled while the initiator is active.

BAR1 Configuration

BAR1 is configured as a 64 kilobyte MEM space which maps to the target FIFOs. This space supports multiple data phase transfers. Transfers beyond the end of the address space result in target abort. For all other accesses, this region will respond according to how it is accessed. Consult the PCI specification regarding delayed transactions. Data transfer between the local side and PCI bus is achieved using retries and delayed transactions as needed.

Posted Writes

The target performs posted writes. On writes to an idle target, the FIFO accepts incoming data until it is full or the write transaction has ended, whichever occurs first. In the event of a full FIFO, the target issues a disconnect. After the PCI transaction is complete, the target empties the FIFO by writing the data out to the local side until the FIFO is empty. To achieve this, the target latches the destination address for use during write out.

On writes to a busy target (the FIFO is still busy from the previous transaction) the target responds with retry, without putting the request in a retry queue.

Prefetched Reads

For reads, the target may not anticipate the length of the transaction or have the data available in time. For this reason, the target puts the transaction in a retry queue and responds with a retry termination. Then the target prefetches data to fill the FIFO. When the initiator returns to retry the transaction, the data will be available.

If the initiator returns to retry the transaction, and does not completely empty the FIFO, the FIFO is flushed after the transaction is complete. If the initiator does empty the FIFO, and attempts to read more, the target issues a disconnect.

If the initiator never retries the original transaction, deadlock may occur. For this reason, there exists a discard timer that signals a waiting delayed completion should be discarded. This timer times out after 32,768 PCI clocks. This period may be shortened to allow simulation of this event in a reasonable amount of time.

Register File Interface

The operation of this block is synchronous to the PCI clock. This block contains all the control and status registers discussed in the functional description. The local bus access port is defined in Table 1.

Table 1: Local Bus Register Interface

Name	Direction	Function
LWE	input	Write enable for registers
LRE	input	Read enable for registers
LADDR	input	Address input
LDIN	input	Data input
LDOUT	output	Data output
LINT_N	output	Active low local interrupt

Target FIFO Interface

The operation of this block is synchronous to the PCI clock. This block is interfaced to two FIFOs; one is the target read (TRF) FIFO, and the other is the target write (TWF) FIFO.

The FIFOs are identical, but data flows in opposite directions. Table 2 lists the signals used in the interface.

Table 2: Local Bus Target Fifo Interface

Name	Direction	Function
TRF_LD TWF_ST	output output	Data requested or available
TRF_ADDR TWF_ADDR	output output	Transfer starting address
TRF_AF, TWF_AE	output	Transfer almost done flag
TRF_WR, TWF_RD	input	FIFO write and read enable
TRF_DIN TWF_DOUT	input output	Data transfer ports

Initiator FIFO Interface

The operation of this block is controlled by the contents of registers in the register block. This block is interfaced to two FIFOs, similar to the memory interface block. One is the initiator read (IRF) FIFO, and the other is the initiator write (IWF) FIFO.

The FIFOs are identical, but data flows in opposite directions. A description of a FIFO follows in the FIFO section. Table 3 lists the signals used in the interface.

Table 3: Local Bus Initiator Fifo Interface

Name	Direction	Function
IWF_LD IRF_ST	output output	Data requested or available
IF_ADDR	output	Transfer starting address
IWF_AF, IRF_AE	output	Transfer almost done flag
IWF_WR, IRF_RD	input	FIFO write and read enable
IWF_DIN IRF_DOUT	input output	Data transfer ports

Pinout

The register file and FIFO interface pinouts are not fixed to specific FPGA I/O pads, allowing flexibility in customization. The PCI bus specific signals are constrained as part of the LogiCORE PCI32 implementation.

As shipped, all of the register file and FIFO interface signals are brought off-chip, but it is not necessary that any interface signals be brought off chip at all in single FPGA designs.

Core Modifications

Modifications can be done to remove the initiator functionality or selected portions of the target functionality. The full design may be expanded as needed or reduced to a very small subset of the original design. The PCI interface itself is also configurable by the designer.

Verification Methods

This design example includes a system level testbench that simulates a four-slot PCI system. This simulation testbench includes a behavioral host bridge (with programmable arbiter) capable of generating burst transactions and a programmable behavioral target.

Recommended Design Experience

The challenge to implement a complete PCI design varies depending on configuration and functionality of your application. We recommend previous experience with building high-performance, pipelined FPGA designs using the Xilinx implementation software and familiarity with either VHDL or Verilog.

Reference Design License

This design is covered under the Xilinx Reference License Agreement. You can find a copy of this license agreement in the Xilinx *PCI Data Book* in chapter 9, and on-line in the PCI Master and PCI64 Lounges.

Nallatech Limited

10-14 Market Street
Kilsyth, Glasgow
G65 0BD
Scotland
Phone: +1 44 7020 986532
Fax: +1 44 7020 986534
E-mail: info@nallatech.com
Website: www.nallatech.com

Introduction

This board allows designers to quickly evaluate the performance of Xilinx's 64 bit / 66 MHz PCI Core including data throughput capabilities across the PCI bus to an on-board SDRAM SODIMM module.

Expansion capability is provided through an interface standard known as DIME. Two DIME module sites are present on the PCI card providing users with the ability to build custom systems from a variety of solutions provided by this royalty free, open module standard.

Features

- Universal PCI Card
- PCI 2.2 Compliant
- Supports 3.3V and 5V VIO voltages using auto-sensing Circuitry
- Supports 32 & 64 bit PCI in frequencies up to 66 MHz
- Requires only a 5V supply from the Host Motherboard
- Configured over SelectMap Interface from Flash Memory via 95144XL CPLD
- Flash Memory reconfigured via Xilinx MultiLINX Download Cable
- 50Mbytes per second configuration rate
- CPLD Flash booting source files included with board
- Direct Virtex Configuration using MultiLINX Download Cable
- Image Processing Demonstration Bitstream
- Image Processing Demonstration Software
- PCB Design Files for PCI Interface available
- 4M x 64 (32Mbytes) SDRAM in SODIMM Socket
- LEDs showing Power Good for 5V, 3.3V, 2.5V & 1.8V
- LEDs Indicating Vio level and whether 3.3V or 5V Bitstream has been loaded
- 2 DIME Module Sites for Customization and Expansion
- 50MHz and Programmable Oscillator

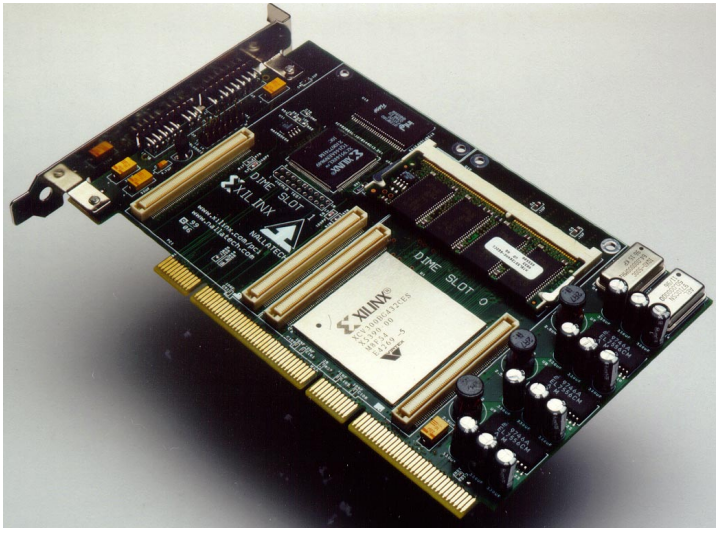


Figure 1: Nallatech PCI64 Prototyping Board

Options

All options are available directly from Nallatech.

- DSP and Image Processing Core, Compliant with DIME Distributed Image Processing Level 1 Standard
- JTAG FPGA Configuration Software (for FPGAs on DIME Modules)
- Reprogram Bitstream in Boot Flash Via PCI Interface (for easy field upgrades)
- Ballyvision - NTSC/PAL Video Capture and Display DIME Module
- Ballyblue - Dual V1000 Virtex FPGA DIME Module for over 2 Million Gates
- Ballytest - DIME Connector Breakout Module
- Ballydiff - Low Voltage Differential Signalling, LVDS, DIME Module
- Custom DIME Module Design Service is Available

General Description

The Nallatech PCI64 Prototyping Card is provided with the Xilinx PCI64/66 Design Kit, available from Xilinx. This card allows designers to quickly evaluate the performance of the Xilinx PCI64/66 LogiCORE design in their system. The firmware and software provided clearly demonstrate the capabilities of the LogiCORE design along with the performance enhancements that the Virtex offers for DSP applications.

Further, the Nallatech PCI64 board demonstrates how to build a universal PCI card. A universal PCI interface requires the inclusion of diode clamps to the 3.3 V rail for a 3.3 V signaling environment, and the exclusion of these in a 5 V signaling environment. To accomplish this, the Virtex FPGA must load different bitstreams depending on the signalling environment. This card demonstrates one way of achieving this.

The Xilinx PCI64 Design Kit provides the user with a complete PCI design example. This includes

- Xilinx 64 bit, 66 Mhz PCI LogiCORE
- Xilinx PCI Bridge interface Core
- Xilinx SDRAM Interface Core
- An image processing demonstration design including a Gaussian Noise Generator and 3 x 3 convolvers

The sources are available for users that have valid licenses for the various cores.

The block diagram, seen in Figure 2, shows the basic interconnectivity of the various interfaces to the Virtex FPGA.

Two pieces of software demonstrate the performance of the PCI bus and the DSP processing capabilities of the Virtex FPGA.

The Virtex FPGA is configured at power on over its Select Map interface from an high speed Flash memory. An XC95144 CPLD controls the Flash and the transfer of data to the FPGAs Select Map port. This data is transferred over the selectmap port at 50MBytes/Sec, thus allowing the FPGA to be configured in a few milliseconds. This is a 50x improvement over the fast serial mode transfer rate.

Two DIME Module sites provide the user with access to an ever growing variety of interfaces and data processing nodes. In fact full system solutions can be developed with just this card and one or two DIME modules. Therefore the user often need not bother with the development of custom PCBs for a solution to their system problem. For more details on available DIME modules and more information on the standard, go to the Nallatech web site,

`www.nallatech.com`

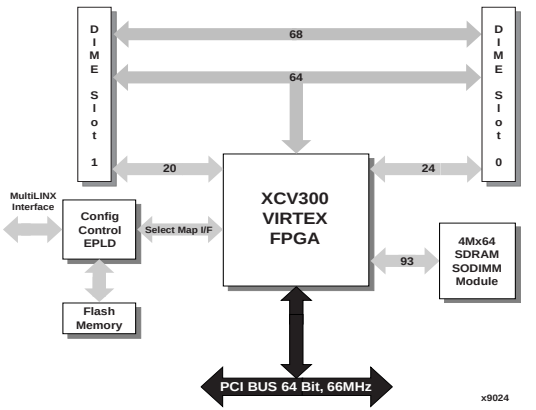


Figure 2: PCI64 Card Block Diagram

Configuration

The FPGA must be configured in less than 100 mS in order for the PCI interface to be up and running when the PCI bus is scanned for available devices. In order to achieve this the Select Map configuration mode is utilized together with an Intel Fast Flash memory. An XC95144 CPLD is used to control the booting process.

The Flash memory can be erased and reprogrammed with new bitstreams using the Xilinx MultiLINX Download Cable. The CPLD, under the control of either the PCI signalling voltage (Vio) or the MultiLINX cable, if connected, will enter one of several modes:

Default Mode (no MultiLINX cable connected)

- Load 3.3 V PCI bitstream from Flash Memory
- Load 5 V PCI bitstream from Flash Memory

Program Mode (MultiLINX cable connected)

- Erase 3.3 V PCI bitstream
- Erase 5 V PCI bitstream
- Program 3.3 V PCI bitstream
- Program 5 V PCI bitstream
- Pass-through mode for direct programming of Virtex

The default mode demonstrates how a universal PCI card can be built. The Program mode gives the user control over the flash chip.

Alternatively, Nallatech have a product that can be added to a user's PCI interface that allows the Flash Configuration Memory to be reprogrammed directly over the PCI interface using a software Utility. This allows users of the PCI logicore to easily integrate Firmware Field upgrades into their products.

Software

Two software programs are included, demonstrating the performance of the PCI bus and the DSP processing capabilities of the Virtex FPGA.

The first program provides a GUI interface to the PCI Bridge Design part of the PCI LogiCORE. The SDRAM is connected to the back of this bridge and DMA transfers to and from this memory can be performed and the transfer rate is displayed. This interface also provides the user with the basic operation of the Xilinx PCI Bridge Design.

Another GUI-based program allows the user to download an image to the SDRAM memory. Noise can then be added to the image and functions such as convolution or edge detection can be performed on the image. The resultant image is then stored in another part of the SDRAM for collection by the GUI. The function is also performed on the PC and the time to complete the functions is logged for comparison. Figure 3 shows the user interface to the Image Processing Demonstration program.

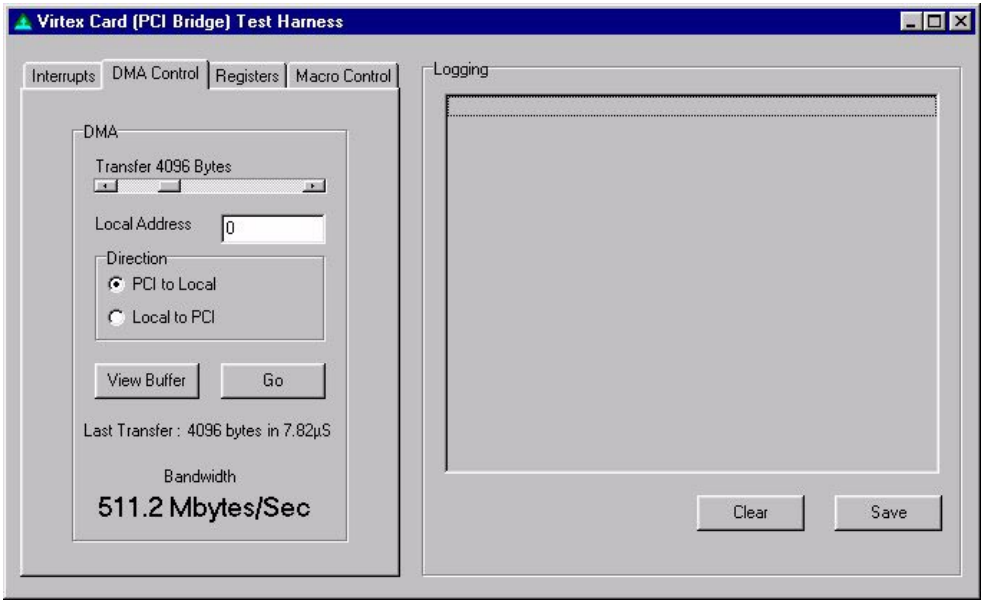


Figure 3: PCI Bridge Design Demo

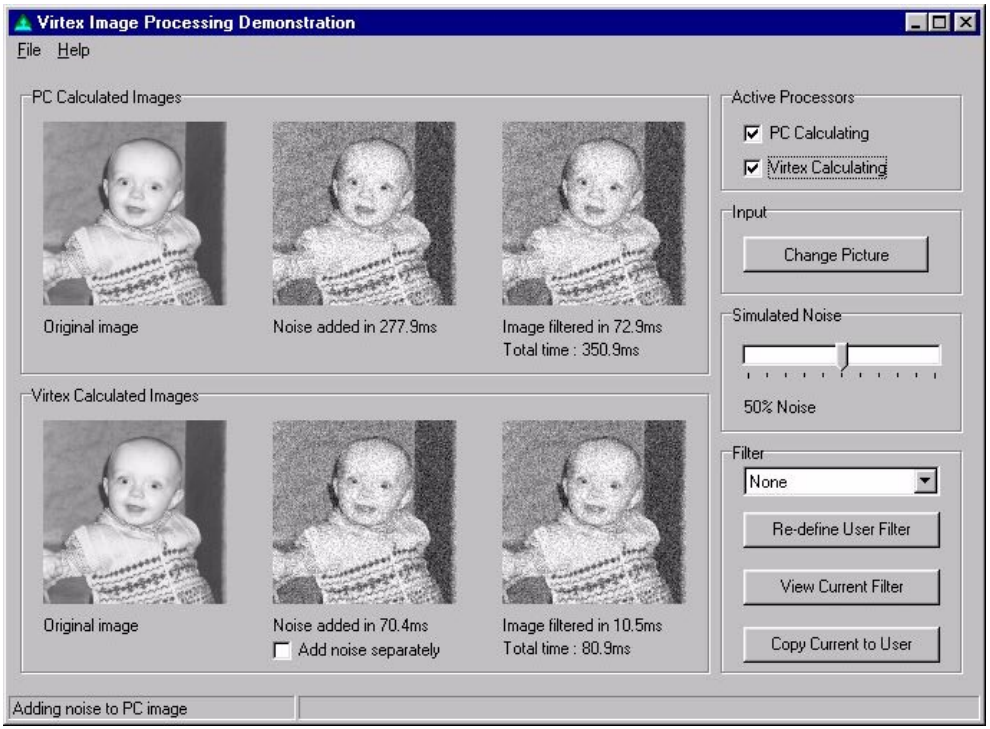


Figure 4: GUI for Image Processing Demo

VIRTUAL COMPUTER CORPORATION

Virtual Computer Corporation

6925 Canby Ave. #103
Reseda, CA 91335 USA
Phone: +1 818-342-8294
Fax: +1 818-342-0240
E-mail: info@vcc.com
Website: www.vcc.com

Introduction

To facilitate rapid development of PCI Designs using the Xilinx LogiCORE PCI32 Interface, VCC has developed the HotPCI Spartan Prototyping Board. This allows the designer to quickly develop, modify, and test Xilinx LogiCORE PCI32 designs in-system. The HotPCI Board is provided with the Xilinx PCI32 Design Kit and should be used in conjunction with the LogiCORE PCI32 Spartan Interface.

Features

The Spartan HotPCI board comes standard with the following hardware and features:

- 5 V PCI 2.1 Compliance
- Xilinx Spartan XCS40-4, reconfigurable via PCI bus or Xchecker cable
- Xilinx XC95108-10 for reconfiguration management
- 8x128K bytes of fast SRAM organized as 2 Independent Banks of 32-bit RAM (four 8-bit x 128K)
- Configuration Flash 128KB for initial configuration
- Configuration RAM Cache 128KB for reconfiguration
- Programmable Clock Generator (360KHz to 100 MHz)
- Mezzanine Connectors for daughter cards
- LEDs to indicate configuration finished (DONE), 5 V, and 3.3 V
- 3 Split Power Planes & 1 Ground Plane
- 4 Signal Layers
- Xchecker/Download Cable Module
- PCI Demo bitstream
- Backend design example files for RAM interface
- PCB design files for the PCI Interface only
- Reference drivers for Windows 95 and Windows NT

Options

All options are available directly from VCC. See the Ordering Information chapter for more details.

- Programmable Voltage Control Module (3.3v to 1.8v)
- License for the Configuration Cache Manager™
- Prototyping Daughter Card
- XC40125 Extended Logic Daughter card
- XC6264 RPU Daughter Card
- HotPCI Card with a Xilinx XC4062XL FPGA

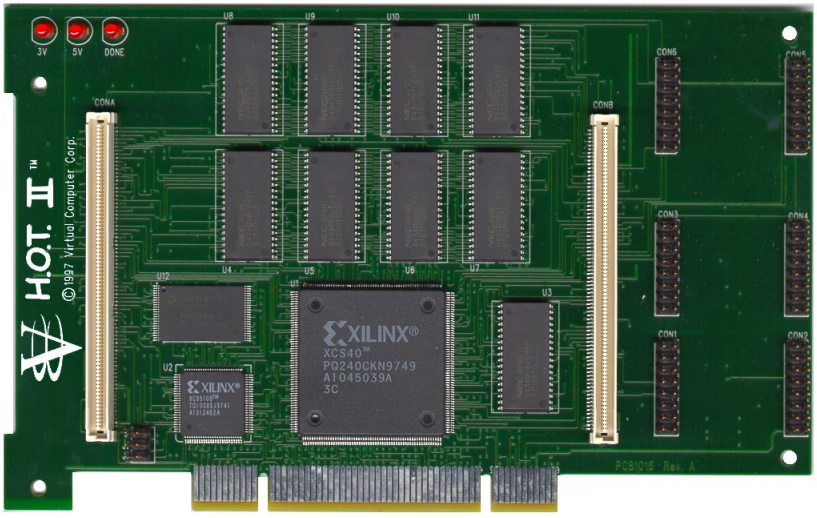


Figure 1: Spartan HotPCI Spartan Prototyping Board

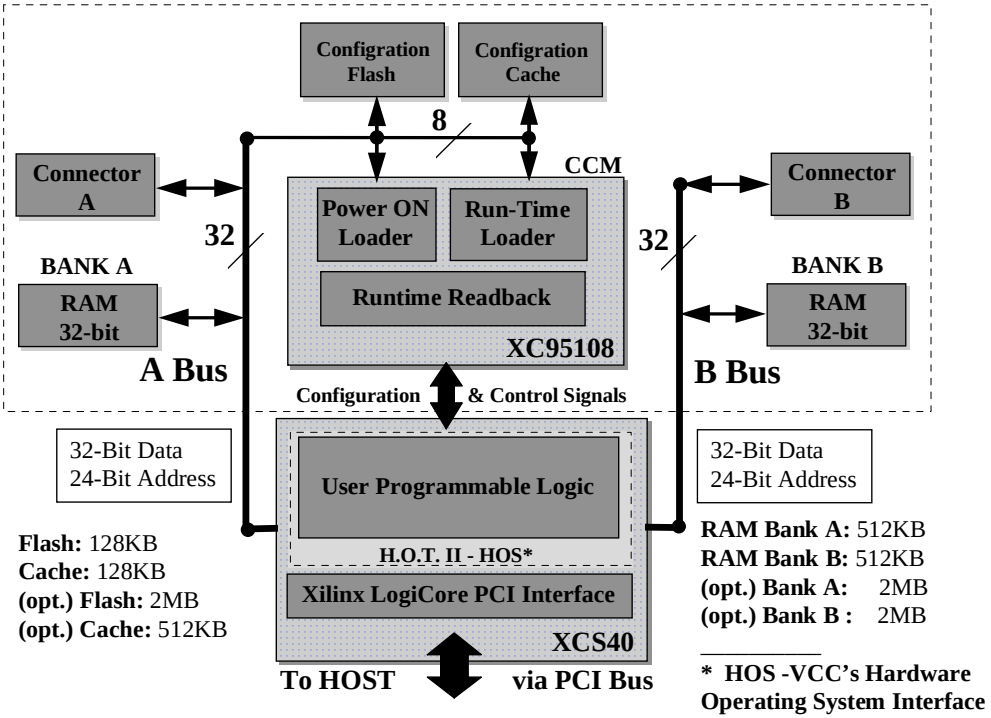


Figure 2: Spartan HotPCI Block Diagram

General Description

The Spartan HotPCI board is a PCI bus based general purpose PCI Prototyping Board, which is provided with the Xilinx PCI32 Design Kit (see the Ordering Information chapter of this data book for details). When used in conjunction with the LogiCORE PCI32 Spartan Interface, designers can quickly modify and test their PCI designs in-system. It features the XCS40 FPGA as the PCI bus interface chip, and a single bank of SRAM plus a VCC proprietary reconfiguration system, the Configuration Cache Manager™ (CCM). The Spartan HotPCI Spartan Prototyping Board is valuable for evaluating, customizing, and verifying the Xilinx LogiCORE PCI32 product line. The mezzanine daughter card connectors offer expandability for prototyping additional features or extending the programmable logic capabilities of the HotPCI board. A series of daughter cards are available from VCC, including a prototyping card for wire-wrap projects, an extended logic card with the XC40125, and a XC6200 Reconfigurable Processing Unit (RPU) card offering microsecond dynamic partial reconfiguration.

Software

The HotPCI Spartan Prototyping Board is supplied with drivers made by Vireo Software, Inc. The Xilinx PCI32 Design Kit also includes driver development tools from Vireo. Together with VCC's HotPCI Board, Vireo's development tools allow easy customization and prototyping of a complete PCI system. See the *Driver::Works Windows Device Driver Development Kit Version 2.0* and the *VtoolsD Windows Device Driver Development Kit Version 3.0* data sheets for more details on the driver development tools. Included with the board is also a bitstream and a Windows '95 application CD that allow demonstration of the card.

Functional Description

The Spartan XCS40 FPGA contains the Xilinx LogiCORE PCI32 Interface Macro and the backend design. VCC supplies a customized backend that allows users to communicate with two fully independent 32-bit banks of RAM and the Configuration Cache Manager™ (CCM). The CCM controls the Run-Time Reconfiguration (RTR) behavior of the system.

The HotPCI board has two independent buses, each with 32-bit data and 24-bit address. There is a daughter board I/O connector for each of these two buses.

Configuration with the CCM

At power-on the FPGA is set to slave serial mode; then the CCM obtains the configuration data from flash and loads the FPGA. Through the PCI bus, the user can load a new

configuration into the Configuration Ram Cache. The user then writes to the CCM to start the reconfiguration of the FPGA. During this time access to the board is disabled by the driver. When the FPGA comes back on-line it signals the driver, which reloads the PCI Header information into the LogiCORE PCI Core. A 128KB Configuration Cache RAM can hold 3 XCS40 configurations. See Figure 2 for a block diagram of the board.

Configuration with an Xchecker cable

Configuration can also be performed through an Xchecker cable or download cable. The supplied Xchecker module occupies one of the daughter card I/O connectors. The PCI bus of the host computer must be reset when this occurs.

Compuware® NUMEGA™

Compuware NuMega

9 Townsend West
Nashua, NH 03063
Phone: 1 800-4NUMEGA (1 800 468-6342)
+1 603 578-8400
Fax: +1 603 578-8401
E-mail: customer_service@numega.com
Technical support:
www.numega.com/support/support.shtml
Website: www.numega.com

Introduction

To facilitate rapid development of PCI Designs using the Xilinx LogiCORE PCI32 Interface, Vireo is providing the DriverWorks Windows Device Driver Development Kit. This kit includes an interactive GUI Wizard that runs in conjunction with the Microsoft Visual C++ 4.2 and later. Provided at no extra cost with the Xilinx PCI Design Kit, is a full-featured version of DriverWorks licensed for prototyping fully functional drivers and testing them with the HotPCI board. An unrestricted license is available directly from Vireo.

Support

Support for DriverWorks is provided only from Vireo. See Vireo's home page for contact instructions and other details.

Features

- Windows NT support
- Windows 98 support
- DriverWizard™, Vireo's Code Generation Wizard Interface. The Wizard includes automated support for all PCI functions, including:
  - PCI Configuration
  - DMA
  - Mapped Memory
  - Interrupt handling
  - IO Ports
  - Application interface
  - Registry interface
  - Plug and Play handling
- Support for MSVC++ 4.2 and later
- C++ Class Library for NT/WDM driver development
- Full library source code is included
- Dozens of sample drivers, with full source code
- Full Plug and Play support
- Driver Access Architecture (DAA) supports portability between Windows NT, Windows 95, Windows 98, and Win32 Driver Model (WDM)
- DriverMonitor™ - monitor driver activity without a debugger.
- Ready-to-use examples tested on real-world hardware
- Full technical support through Vireo
- RISC platform support
- More than 700 pages of printed and online documentation

Description

DriverWorks is a next-generation environment for device driver development based on a powerful and flexible C++ class library coupled with a powerful code generation wizard.

Over time, Windows application development has evolved to class libraries such as MFC and development tools such Microsoft's Application Wizard. Vireo provides a similar environment for Windows NT and WDM device driver development.

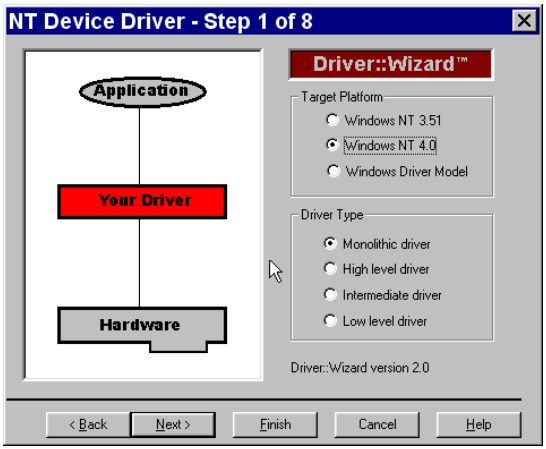


Figure 1: DriverWizard GUI (earlier version shown)

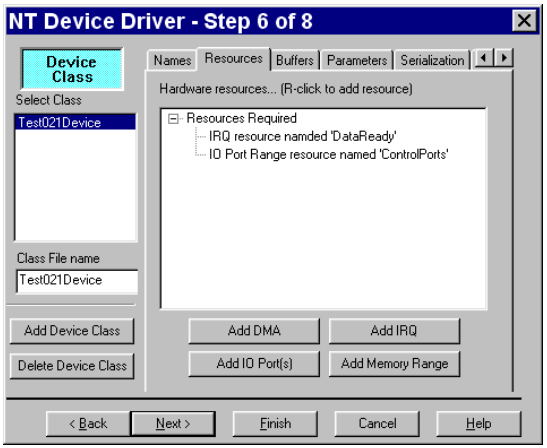


Figure 2: DriverWizard GUI (earlier version shown)

The DriverWorks class library offers thousands of lines of tested code that reduce many complex tasks to simple library calls. In fact, DriverWorks offers by far the most complete device driver library available.

DriverWorks also ships with complete examples that are designed to be used as a basis for further development.

DriverWorks also includes Vireo's unique DriverWizard technology, shown in Figures 1 & 2. DriverWizard guides you through a series of steps that identify many characteristics of your device. DriverWizard then generates source code tailored to your driver. The DriverWorks class library, framework, and Wizard provide access to tens of thousands of lines of working, debugged code that will allow you to develop your device drivers quickly.

DriverWorks implements Vireo's Device Access Architecture (DAA) interfaces. Using DAA, device driver source code can be easily ported between Windows 95, Windows 98, and all versions of Windows NT. Drivers written with DAA provide optimal performance on each platform while at the same time offering a common set of objects and interfaces that provide source code portability with no limitations or overhead.

DriverWorks requires Microsoft Visual C++ version 4.2 or later, and the Microsoft NT DDK, or the Windows 98 DDK. DriverWorks drivers have been tested on both Alpha and Intel single and dual processor platforms.

DriverWorks incorporates years of class-library design experience into a clean, object-oriented system that accurately reflects the underlying system architecture while avoiding the use of arcane C++ language features.

The DriverMonitor tool, shown in Figure 3, provides a unique workbench for loading, testing, tracing, and unloading your device driver.

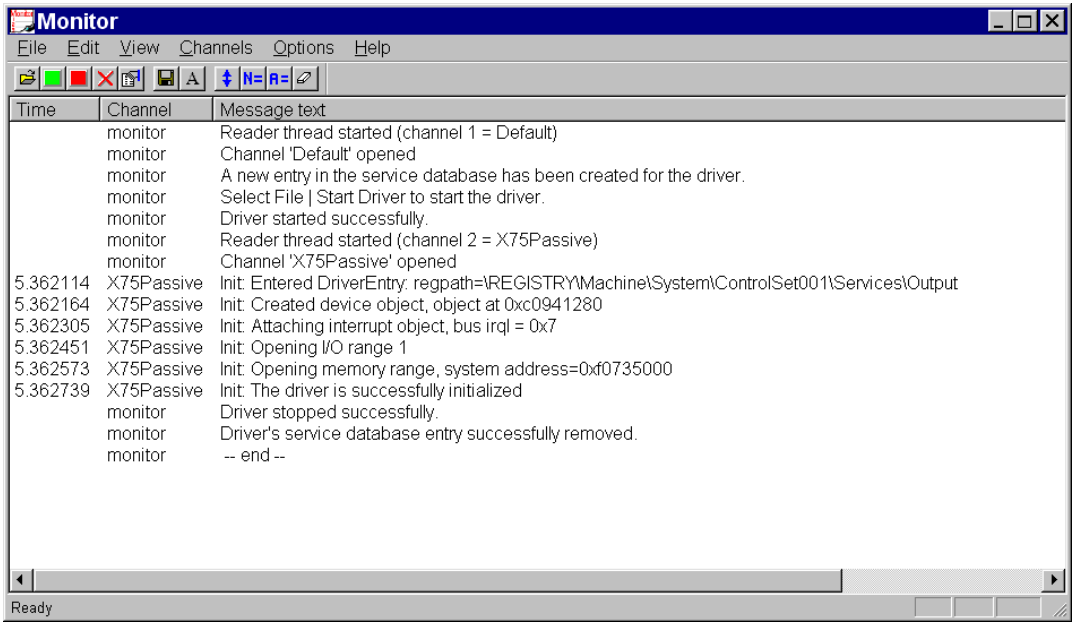


Figure 3: DriverMonitor Interface

Licensing

The version of DriverWorks included in the Xilinx PCI Design Kit is fully functional and includes all libraries and software. It is licensed for use in driver development and prototyping only. Vireo offers Xilinx PCI customers the opportunity to purchase a royalty-free distribution license. Contact Vireo for pricing and details.

Vireo provides free bug fixes available for immediate download. Timely new versions provide support for a new compiler versions, and operating system revisions. Vireo also provides new examples and bug fixes on a regular basis.

Technical support on this product is available *only* through Vireo Software Inc.

Compuware® NUMEGA™

Compuware NuMega

9 Townsend West
Nashua, NH 03063
Phone: 1 800-4NUMEGA (1 800 468-6342)
+1 603 578-8400
Fax: +1 603 578-8401
E-mail: customer_service@numega.com
Technical support:
www.numega.com/support/support.shtml
Website: www.numega.com

Introduction

To facilitate rapid development of PCI Designs using the Xilinx LogiCORE PCI Interface, Vireo is providing the VtoolsD Windows Device Driver Development Kit. This kit includes an interactive GUI Wizard that allows the creation of a drive driver framework with a few simple selection and mouse clicks. Both the Microsoft Visual C++ 4.2 and later and the Borland C++ 4.x and later compilers are supported. Provided at no extra cost with the Xilinx PCI Design Kit, is a full-featured, fully functional version of VtoolsD licensed for prototyping drivers and testing them with the HotPCI board.

Support

Support for VtoolsD is provided only from Vireo. See Vireo's home page for contact information and other details.

Features

- Windows 95, 98, 3.X Support
- VtoolsD Interface
- Works with MS or Borland C/C++ compilers
- More than 50 sample drivers
  - Over 2 dozen example drivers written in C
  - Over 1 dozen example drivers written in C++
- Detailed on-line and printed documentation
- C and C++ system interfaces
- C Run Time Library
- C++ Class Library
- Includes complete source code for all libraries
- Thunks and wraps for every VMM/ VxD service and handler
- Microsoft DDK components bundled with VtoolsD for Windows 95
- Debug kernel executables and symbol tables
- WDEB386 system-level debugger for VxDs
- More than 1900 online help topic pages
- DDK documentation and help files
- Supports Driver Access Architecture (DAA)
- QuickVxD Wizard for quick device driver framework development.
- Driver Access Architecture (DAA) supports portability between Windows NT, Windows 95, Windows 98, and Win32 Driver Model (WDM)
- Complete access to over 900 interfaces from C/C++
- More than 60 classes designed for VxD operation
- More than 80 ANSI-compatible C Run Time Library functions
- DriverMonitor™ - monitor driver activity without a debugger.
- QuickVxD source code generator - a VxD Wizard
- Microsoft and Borland compiler support
- Dynamic VxD Loader
- VxD Viewer

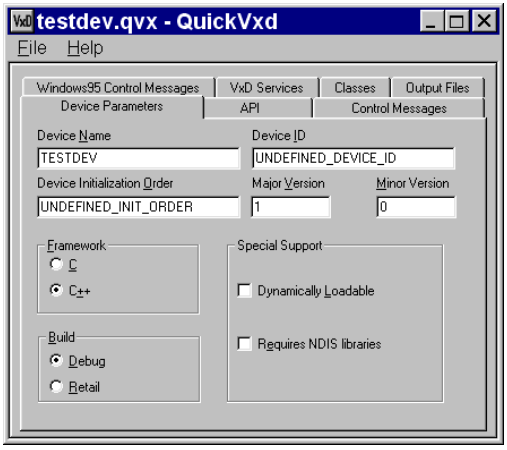


Figure 1: VtoolsD™ QuickVxD GUI

Description

VtoolsD is the easiest and fastest way to build Virtual Device Drivers (VxDs) for Microsoft Windows. Designed for both novice and experienced VxD developers, VtoolsD provides the comprehensive C or C++ solution for all VxD development challenges. Shipping since July 1994, VtoolsD is a mature, professional product used by thousands of developers world wide.

VtoolsD supports all of the system interfaces that the Microsoft DDK provides, plus an additional set of services provided by the VtoolsD libraries. VtoolsD can be used to write any kind of VxD, and makes that easier than it would be using the DDK.

DriverWorks implements Vireo's Device Access Architecture (DAA) interfaces. Using DAA, device driver source code can be easily ported between Windows 95, Windows 98, and all versions of Windows NT. Drivers written with DAA provide optimal performance on each platform while at the same time offering a common set of objects and interfaces that provide source code portability with no limitations or overhead.

The DriverMonitor tool, shown in Figure 2, provides a unique workbench for loading, testing, tracing, and unloading your device driver.

VtoolsD requires either MSVC++ 4.2 or later or the Borland C++ 4.X and later compilers. The Microsoft DDK is not required to use VtoolsD.

Licensing

The version of VtoolsD, included in the Xilinx PCI Design Kit, is fully functional and includes all libraries and software. It is licensed for use in driver development and prototyping only. Vireo offers Xilinx PCI customers the opportunity to purchase a royalty-free distribution license. Contact Vireo for pricing and details.

Vireo provides free bug fixes available for immediate download. Timely new versions provide support for a new compiler versions, and operating system revisions. Vireo also provides new examples and bug fixes on a regular basis.

Technical support on this product is available *only* through Vireo Software Inc.

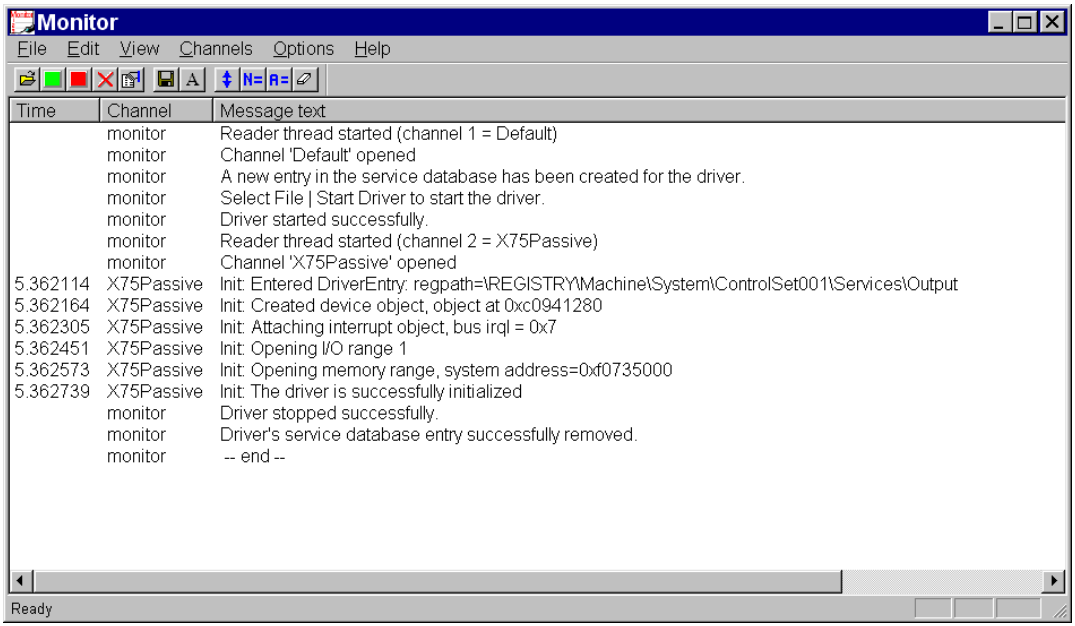


Figure 2: DriverMonitor Interface

Reference DESIGN

Synthesizable PCI Power Management Design Example

May, 1999 (Version 1.0) Data Sheet

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-377-3259
E-mail: Techsupport: hotline@xilinx.com
Feedback: logicore@xilinx.com
URL: www.xilinx.com/pci

Introduction

The synthesizable PCI Power Management design is a Reference Design for use with the LogiCORE PCI32 Spartan/Spartan XL and PCI32 4000/4000XLA . It is delivered in both Verilog and VHDL. This example shows how the capabilities linked list structure may be implemented along with the LogiCORE PCI32 Interface and the user backend function in the Xilinx FPGA.

This design is demonstrated as an add-on module to the Xilinx ping design. Please refer to the PCI application note *Ping Application Example* for details (available from Xilinx's customer-only PCI lounge).

This data sheet assumes that you are familiar with the PCI V2.2 specification and the PCI Power Management V1.1 specification. If you are not familiar with this information, please review the *PCI Local Bus Specification*, Revision 2.2 and the *PCI Bus Power Management Interface Specification*, Version 1.1 available from the PCI Special Interest Group (www.pcisig.com).

Features

- Demonstrates how to implement the capabilities linked list in the LogiCORE PCI32 Interfaces
- Demonstrates how to implement the user-defined configuration space register
- Supports for up to 4 PCI function power management states (D0-D3)
- Supports PMCSR_BSE register for PCI bridge specific functionality
- Supports optional Data register to report state-dependent operating data

Reference Design Facts

Design Specifics	
Tested LogiCORE PCI Interfaces	PCI32 4000 V2.0.2 PCI32 Spartan V2.0.3 PCI32 4000XLA V3.0 PCI32 SpartanXL V3.0
Tested Devices[1]	XC4013XLT-1 PQ240 XC4028XLT-1 HQ240 XC4013XLA-09 PQ208 XCS30-4 PQ240 XCS30XL-4 PQ240
CLBs Used[3]	Up to 58
IOBs Used[3]	2
Tested Platforms[2]	
Workstation Flows	Verilog XL 2.6, VSS 9802, FPGA Compiler and A1.5iSP1
PC Flows	MTI ModelSim PE/Plus V4.7h, Foundation Express 3.1 and F1.5i with Service Pack 1
Xilinx Core Tools	M1.5isp1
Provided with the Reference Design	
Documentation	*Synthesizable PCI Power Management Application Note*
Design File Formats	VHDL, Verilog
Verification Tools	VHDL\Verilog Testbench
Symbols	VHDL, Verilog
Support	
This Reference Design is provided as is under the Reference Design License Agreement, refer to chapter 9 in the *Xilinx PCI Databook*.	

Notes:
1. Listed are the devices that Xilinx used to verify the design. Other devices may be used.
2. Listed are the design tools that Xilinx used to verify the design.
3. The CLB and IOB counts do not include the PCI interface and the Ping application design. Actual CLB number depends on the core & device used.

General Description

This design implements an add-on PCI power management module in Xilinx FPGA and is intended for use with the LogiCORE PCI Interface V2.0 or V3.0. The power management module interfaces with the LogiCORE PCI32 interface and the backend application design (ping application in this case). It is also compatible with Xilinx's synthesizable PCI bridge design examples.

This design contains banks of user-defined registers to implement the PCI configuration space from address 0x40 to 0xFF. This register bank is compliant to the capabilities linked list structure defined by both the PCI spec and the Power Management spec. Since **Vaux** pin is not supported in this reference design, **PME#** event from $D3_{cold}$ is not supported.

This design does not define any Device-Class Power Management policy, which falls outside the scope of the Power Management spec.

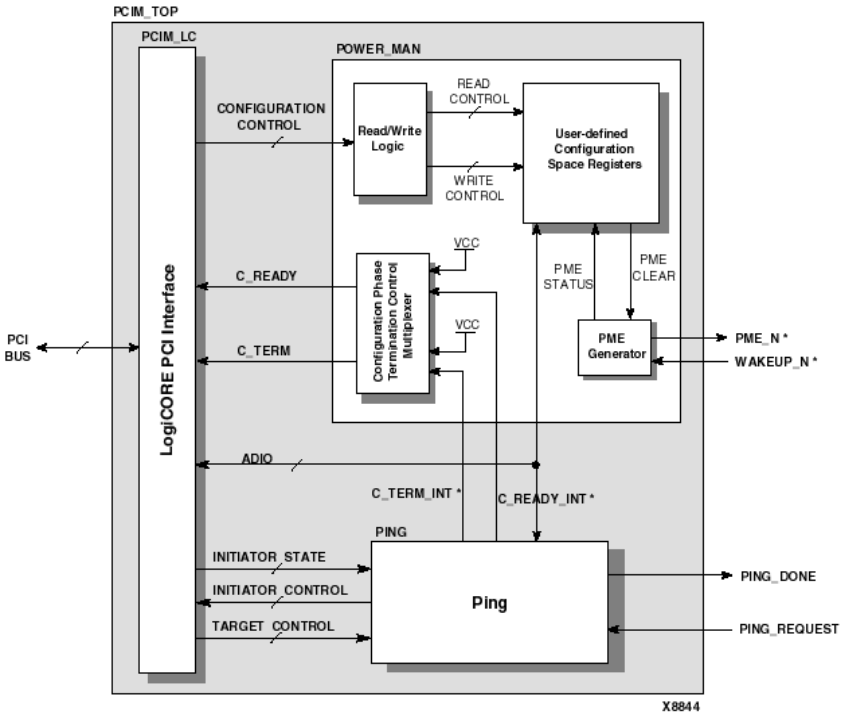


Figure 3: Synthesizable PCI Power Management Block Diagram

Functional Description

The Power Management spec defines a standard for four basic power management operations: Capabilities Reporting, Power Status Reporting, Setting Power States, and System Wakeup. These four operations are supported in this reference design.

Xilinx's LogiCORE PCI32 Interface allows user to enable the Capabilities bit (Bit 4) in the Status register and to specify the address of the capabilities linked list in the 1-byte Cap_Ptr field of the PCI Configuration Header. Refer to the *File Modification* section for details.

Since the PCI32 Interface provides only the PCI Configuration Header (Offset 0x00 to 0x3F), a User-defined Configuration Register is needed to implement registers from address offset 0x40 to 0xFF. This design implements an 8-byte register to implement the Power Management register block. This design also supports the RTL-level logic for the PME generator.

Capabilities Linked List

In order for the software to determine if a specific PCI function is designed to support Capabilities Linked List, Bit 4 of the Status register needs to be set and the first address of

the linked list needs to be set to the Cap_Ptr register (Offset 0x34 for Header Type 0 & 1 devices) during design time.

This can be done during the web download of the PCI32 Interface or by editing the configuration file (`cfg.v` or `cfg.vhd`). For each capabilities linked list, the first byte of each entry contains the ID for that capability (**01h** for Power Management). The next byte (*Next_Item_Ptr*) contains the pointer to the **absolute offset** in the function's PCI Configuration Space for the next item in the list. Items must be **DWORD aligned**. The last item in the list must have its *Next_Item_Ptr* set to **null**.

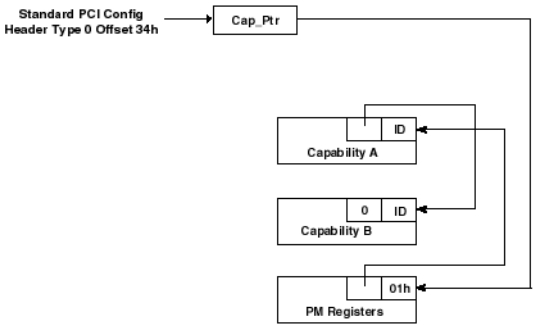


Figure 4: Capabilities Linked List

This design only implements the Power Management capabilities linked list register.

Power Management Register Block

Figure 3 illustrates the PCI Power Management Register Block definition. The first 2 bytes (*Capabilities ID* [offset = 0] and *Next Item Pointer* [offset = 1]) are used for the linked list infrastructure. The next 4 bytes (*PMC* [offset = 2], and *PMCSR* registers [offset = 4]) are required for compliance with the Power Management spec. The next byte (bridge support *PMCSR_BSE* extensions [offset = 6]) is required only for bridge functions, and the remaining byte *Data* register [offset = 8] is optional for any class of function.

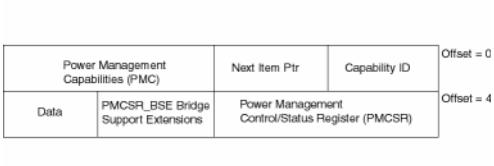


Figure 5: Power Management Register Block

This design implements the complete register block with user-defined configuration space (refer to next section).

Refer to the Power Management spec for detailed description of this register.

Capability ID and Next Item Pointer

Both the *Cap_ID* and the *Next_Item_Ptr* are 8-bit read-only registers and their values are set by in the HDL code (`power_man.v` or `power_man.vhd`).

PMC

This is a 16-bit read-only register which provides information on the capabilities of the PCI function related to power management. The information in this register is static and known at design time.

The register value is set in the HDL code. This design supports the optional **D1** and **D2** power states, PME# can be asserted from **D0**, **D1**, **D2** and **$D3_{hot}$**, and no PCI clock is required for the function to generate PME#.

PMCSR

This 16-bit read-write register is used to managed the PCI function's power management state as well as to enable/monitor PMEs.

Bits	Read/Write	Description
15	Read/Write-Clear	PME_Status
14:13	Read Only	Data_Scale
12:9	Read/Write	Data_Select
8	Read/Write	PME_En
7:2	Read Only	Reserved
1:0	Read/Write	PowerState

PMCSR_BSE

This is an 8-bit read-only register implement in this design to support PCI bridge functionality in PCI-to-PCI bridges. The value is set to **C0h** in the HDL code. System designer can modify PMCSR_BSE[7:6] for the specific bridge support or remove this register from the HDL code if the PCI function is not a PCI-to-PCI bridge.

Bits	Description
7	BPCC_En (Bus Powered/Clock Control Enable)
6	B2_B3# (B2/B3 support for $D3_{hot}$)
5:0	Reserved

Data

This is an optional 8-bit read-only register that provides a mechanism for the PCI function to report state dependent operating data such as power consumed or heat dissipation. Typically the data returned through the *Data* register is a static copy of the PCI function's worst case "DC characteristics" data sheet. This data, when made available to the system software, could then be used to intelligently make decisions about power budgeting, cooling requirements, etc.

If *Data* register is implemented, then the **Data_Select** and **Data_Scale** fields of the *PMCSR* register must also be implemented.

This register is implemented as a 16x8 ROM in the HDL code and its value can be modified by user.

User-defined Configuration Space

The PCI spec defines a 256-byte configuration space and Xilinx's LogiCORE PCI32 Interface implements only the PCI Configuration Header (first 40 bytes, offset 0x00 to 0x3F). Configuration space with address offset 0x40 to 0xFF can be added by the user. Please refer to the User Definable Configuration Space section in the Configuration Transfers chapter of the PCI32 User's Guide.

The Power Management register block is implemented using this user-defined configuration space (address 0x50 in this design, can be changed by user). Access to this region is controlled by the "*User Config Space Enable*" switch in the configuration file. This switch can be enabled during the web download or by editing the configuration file.

This design allows different configuration data phase control conditions between the Power Management register block and the backend design (ping example design in this case). Output signals **C_TERM** and **C_READY** from the backend design will go to the power_man instead of driving the PCI interface, as is normally the case in the ping design example. These two signals are named in the top-level wrapper file as **C_TERM_INT** and **C_READY_INT** respectively). Since this design will disconnect with data on the first data phase during configuration cycles, it will drive these two signals HIGH all the time. The *Configuration Phase Termination Control Multiplexer* will multiplex between these signals from ping and power_man with control signal **c_switch**. The multiplexer output will then drive the **C_TERM** and **C_READY** of the PCI32 interface.

Writing to a read-only register or unimplemented address in the upper configuration space will be ignored but the transaction will be completed gracefully. Reading unimplemented address in the upper configuration space will return zeros.

PME Generation

This optional PME# is an asynchronous signal. Two pins (*PME_N* and *WAKEUP_N*) are added to implement this circuit. When the PCI function is programmed into a lower power consumption state, it can request a change in its current power management state and/or to indicate that a PME (power management event) has occurred. The PME in this design is the *WAKEUP_N* signal assertion by the testbench.

Once PME# is asserted, this design will continue to drive the signal low until software explicitly clears the *PME_En* bit or clears the *PME_Status* bit. Since **3.3Vaux** pin is not support in this design, PME# cannot be asserted from $\mathbf{D3_{cold}}$ state and it will be system software's responsibility to bring it back to *D0 uninitialized* state.

Pinout

The new *PME_N* and *WAKEUP_N* pins are not fixed to specific FPGA I/O pads. The PCI bus specific signals are constrained as part of the LogiCORE PCI32 implementation.

Core Modifications

This design is released with all the RTL source codes and run scripts for simulation and M1 implementation. It does not include any PCI32 core related files (e.g. PCI32 core netlist, ucf files, guide files, etc.) that are located in the *src* directory of the PCI32 download.

Although this design is released with all the Ping backend design, files are different from those in the Ping example. Under the */example/source* directory, a new file (`power_man`) is added and modifications are made to `ping_tb`, `pcim_lc/pcis_lc`, `cfg` and `stimulus` files.

To run the Power Management reference design, download the design from Xilinx's PCI32 lounge. Unzipping the download creates the *example_pm* directory. Move the *example_pm* directory under the same directory where *example* directory is located. The *example_pm* directory contains all the design files and run scripts for this reference design.

Below is a description of the file differences.

The cfg file

- The Cap List Enable and the User Config Space Enable switches are enabled
- Capability List Pointer address is programmed to 0x50 (this address can be set to 0x40 to 0xFF by user)

The pcim_top/pcis_top file

- Add PME_N and WAKEUP_N ports
- Instantiate of the power_man module
- Reconnect the C_READY and C_TERM ports (refer to Figure 1). Signals from backend design which used to drive C_READY & C_TERM will connect to power_man and power_man will drive C_READY & C_TERM in the PCI32 interface
- Instantiate IBUF and OBUF for WAKEUP_N and PME_N respectively for the Express flows

Users can enable the Power Management capabilities linked list in their own designs during web download or by editing the `cfg` file under the *src/xpci* directory. Then include the `power_man` module and modify the `pcim_top/pcis_top` file.

Web download

- Enable the Cap List Enable box (note that the User

Config Space box will also be enabled)
- Double click on the Cap List Ptr field (address 34h) and enter the Cap List Ptr address

Editing the cfg file

- Enable the Capability List Enable switch and the User Config Space Enable switch
- Enter the Capability List Pointer address

Verification Methods

A simulation testbench is provided in both Verilog and VHDL. This testbench first configures this design, then it generates PCI transactions to test the Power Management capabilities linked list, the I/O and memory registers.

Recommended Design Experience

Previous experience with Xilinx's PCI design flow, Verilog and VHDL is recommended to user of this reference design.

Xilinx PCI

LogiCORE PCI Supported Virtex FPGAs

May, 1999 Product Overview

Features

- Fast, high-density Field-Programmable Gate Arrays
  - Densities from 50k to 1M system gates
  - System performance up to 200 MHz
  - 66-MHz PCI Compliant
  - Hot-swappable for Compact PCI
- Multi-standard SelectIO™ interfaces
  - 16 high-performance interface standards
  - Connects directly to ZBTRAM devices
- Built-in clock-management circuitry
  - Four dedicated delay-locked loops (DLLs) for advanced clock control
  - Four primary low-skew global clock distribution nets, plus 24 secondary global nets
- Hierarchical memory system
  - LUTs configurable as 16-bit RAM, 32-bit RAM, 16-bit dual-ported RAM, or 16-bit Shift Register
  - Configurable synchronous dual-ported 4k-bit RAMs
  - Fast interfaces to external high-performance RAMs
- Flexible architecture that balances speed and density
  - Dedicated carry logic for high-speed arithmetic
  - Dedicated multiplier support
  - Cascade chain for wide-input functions
  - Abundant registers/latches with clock enable, and dual synchronous/asynchronous set and reset
  - Internal 3-state bussing
  - IEEE 1149.1 boundary-scan logic
  - Die-temperature sensing device
- Supported by FPGA Foundation™ and Alliance Development Systems
  - Complete support for Unified Libraries, Relationally Placed Macros, and Design Manager
  - Wide selection of PC and workstation platforms
- SRAM-based in-system configuration
  - Unlimited reprogrammability
  - Four programming modes
- 0.22-µm five-layer metal process
- 100% factory tested

Description

The Virtex FPGA family delivers high-performance, high-capacity programmable logic solutions. Dramatic increases in silicon efficiency result from optimizing the new architecture for place-and-route efficiency and exploiting an aggressive 5-layer-metal 0.22-µm CMOS process. These advances make Virtex FPGAs powerful and flexible alternatives to mask-programmed gate arrays. The Virtex family comprises the nine members shown in Table 1.

Building on experience gained from previous generations of FPGAs, the Virtex family represents a revolutionary step forward in programmable logic design. Combining a wide variety of programmable system features, a rich hierarchy of fast, flexible interconnect resources, and advanced process technology, the Virtex family delivers a high-speed and high-capacity programmable logic solution that enhances design flexibility while reducing time-to-market.

Table 1: Virtex Field-Programmable Gate Array Family Members.

Device	System Gates	CLB Array	Logic Cells	Maximum Available I/O	BlockRAM Bits	Max Select RAM Bits
XCV50	57,906	16x24	1,728	180	32,768	24,576
XCV100	108,904	20x30	2,700	180	40,960	38,400
XCV150	164,674	24x36	3,888	260	49,152	55,296
XCV200	236,666	28x42	5,292	284	57,344	75,264
XCV300[1]	322,970	32x48	6,912	316	65,536	98,304
XCV400	468,252	40x60	10,800	404	81,920	153,600
XCV600	661,111	48x72	15,552	500	98,304	221,184
XCV800	888,439	56x84	21,168	514	114,688	301,056
XCV1000[2]	1,124,022	64x96	27,648	514	131,072	393,216

1. Verified by Xilinx for PCI64/66, PCI64/33, and PCI32/33
2. Verified by Xilinx for PCI64/66 and PCI64/33

See the Xilinx PCI Retargeting Guide for 33 MHz PCI support of other devices listed here.

For the complete Virtex Product Specification, see www.xilinx.com

LogiCORE PCI32 Supported Spartan and SpartanXL FPGAs

May, 1999 Product Overview

Introduction

The Spartan™ Series is the first high-volume production FPGA solution to deliver all the key requirements for ASIC replacement up to 40,000 gates. These requirements include high performance, on-chip RAM, Core Solutions and prices that, in high volume, approach, and in many cases are equivalent to mask programmed ASIC devices.

The Xilinx Spartan series is the result of more than thirteen years of FPGA design experience and feedback from thousands of customers. By streamlining the Spartan feature set, leveraging advanced hybrid process technologies and focusing on total cost management, the Spartan series delivers the key features required by ASIC and other high volume logic users while avoiding the initial cost, long development cycles, and inherent risk of conventional ASICs. The Spartan Series currently has 10 members; only the devices supported by PCI are shown in Figure 1.

Spartan Series Features

Note: The Spartan Series devices described in this product overview include the 5V Spartan™ family of devices and the 3.3V SpartanXL™ family of devices.

- Next generation ASIC replacement technology
  - First ASIC replacement FPGA for high-volume production with on-chip RAM
  - Advanced process technology
  - Density up to 1862 logic cells or 40,000 system gates
  - Streamlined feature set based on XC4000 architecture
  - System performance beyond 80 MHz
  - Broad set of AllianceCORE™ and LogiCORE™ pre-defined solutions available
  - Unlimited reprogrammability
  - Low cost
- System level features
  - Fully 3.3 V and 5 V PCI compliant
  - Available in both 5.0 Volt and 3.3 Volt versions
  - On-chip SelectRAM™ memory
  - Low power segmented routing architecture
  - Full readback capability for program verification and internal node observability
  - Dedicated high-speed carry logic
  - Internal 3-state bus capability
  - 8 global low-skew clock or signal distribution networks
  - IEEE 1149.1-compatible boundary scan logic
- Versatile I/O and packaging
  - Low cost plastic packages available in all densities
  - Footprint compatibility in common packages
  - Individually programmable output slew-rate control maximizes performance and reduces noise
  - Zero input register hold time simplifies system timing
- Fully supported by powerful Xilinx development system
  - Foundation series: Fully integrated, shrink-wrap software
  - Alliance series: Over 100 PC and engineering workstation third-party development systems supported
  - Fully automatic mapping, placement, and routing
  - Interactive design editor for design optimization

Table 1: LogiCORE PCI32 supported Spartan and SpartanXL Series Field Programmable Gate Arrays

Device	Logic Cells	Max System Gates	Typical Gate Range (Logic and RAM)*	CLB Matrix	Total CLBs	Number of Flip-Flops	Max. User I/O
XCS20XL	950	20,000	7,000 - 20,000	20 x 20	400	1,120	160
XCS30 & XCS30XL	1368	30,000	10,000 - 30,000	24 x 24	576	1,536	192
XCS40 & XCS40XL	1862	40,000	13,000 - 40,000	28 x 28	784	2,016	224

* Max values of Typical Gate Range include 20-30% of CLBs used as RAM.

For the complete Spartan and SpartanXL Product Specification, see www.xilinx.com

Additional SpartanXL Features

- 3.3V supply for low power with 5V tolerant I/Os
- Power down input
- Higher performance
- Faster carry logic
- More flexible high-speed clock network
- Universal PCI Interface capability in SpartanXL
- Latch capability in Configurable Logic Blocks
- Input fast capture latch
- Optional mux or 2-input function generator on outputs
- 12 mA or 24 mA output drive
- 5V/3.3V PCI compliant
- Enhanced Boundary Scan
- Express Mode configuration

Universal PCI Interfaces

A Universal PCI Interface is one capable of being plugged into both a 3.3 V and 5 V PCI bus. Implementing a universal PCI interface in SpartanXL devices requires loading of different bitstreams to change I/O characteristics. For 3.3 V PCI electrical compliance, the 5 V drive strength must be turned off and the 3.3 V clamp diodes enabled. These changes are done in the bitstream. See the *SpartanXL Implementation Guide* for details on generating these bitstreams. It is the designer's responsibility to monitor Vio and load the appropriate bitstream depending on Vio.

When the 3.3 V PCI clamp diodes are enabled, the SpartanXL device loses 5 V tolerance. This is due to the 3.3 V clamp diode being tied to the 3.3 V supply in the device. The designer must insure that a full 5 V is not applied when these clamp diodes are enabled.

Design Methodology

The LogiCORE PCI Interfaces are highly optimized for the Virtex, XC4000XLA, and SpartanXL FPGAs. Easy to use tools provide easy implementation of the PCI interface. Pre-placed and pre-routed guide files along with .ucf files guarantee timing performance on critical control signals like IRDY#, TRDY#, and FRAME#.

The user connects the LogiCORE PCI Interface to other modules to complete the design. For example, to complete a PCI adapter card interface using the XC4013XLA, a designer first configures the PCI core using the intuitive graphical user interface on the Xilinx web site, and downloads proven PCI design. See Figure 2 for the GUI. A designer would then create either a VHDL or a Verilog description, using the LogiCORE PCI32 Interface component together with the required user application. The user can then simulate the core with the customer design using provided VHDL and Verilog simulation models. Next, the user compiles the design using Xilinx Alliance or Foundation Series software for the targeted FPGA. At this point the design can be resimulated or downloaded to the target device. See Figure 1 for a complete design flow.

LogiCORE PCI Configuration

To support multiple design environments, the LogiCORE PCI interface can be configured and downloaded from the web complete with a netlist, constraints, simulation model, and testbench in VHDL or Verilog. This methodology is described in our on-line documentation. For more details see the *Xilinx PCI Design Guide*.

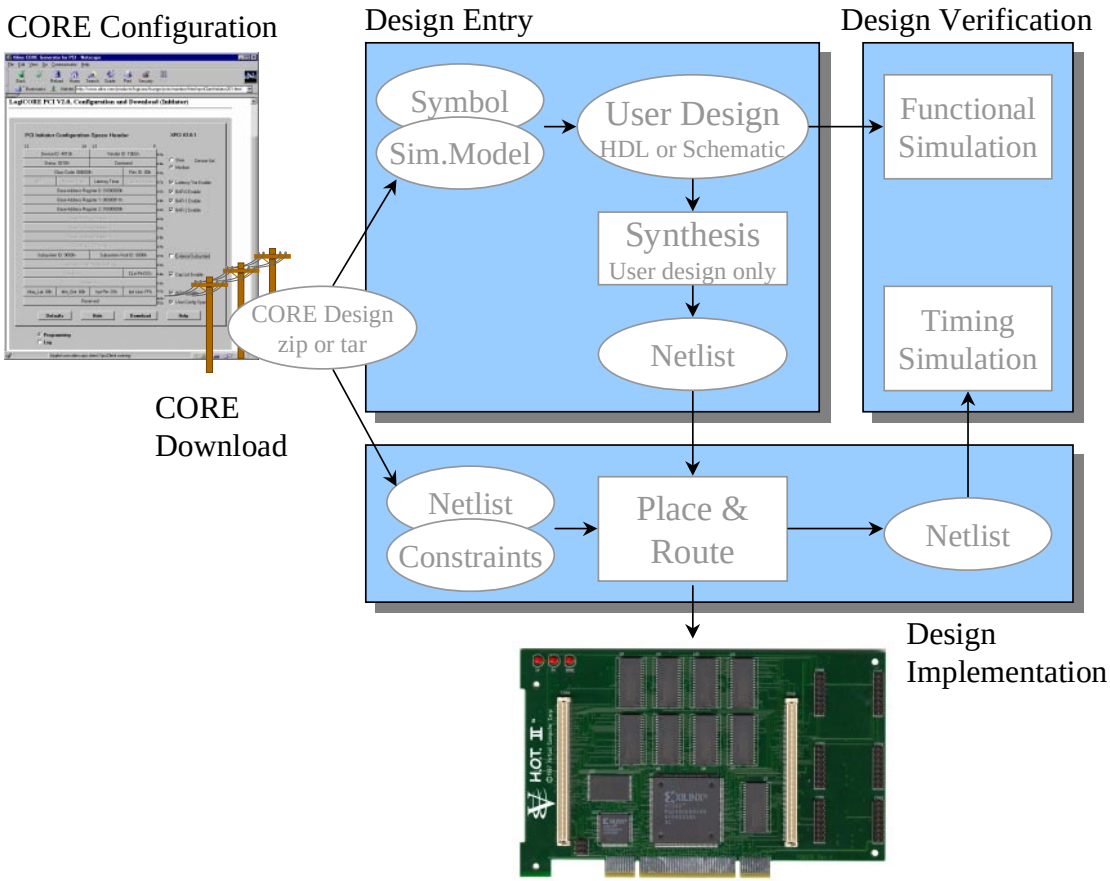


Figure 1: PCI32 Interface Design and Simulation Methodology

This LogiCORE PCI configuration methodology allows HDL users to modify some attributes of the PCI macro. After configuring the macro, the user can download the design files in a PC or UNIX format. Using the macro in a HDL flow is described in either the *LogiCORE PCI Virtex, XLA,* or *SpartanXL Implementation Guide*.

For users that cannot access the configuration tool on the web, the core can be configured locally by editing a single VHDL or Verilog file.

Core Configuration in VHDL and Verilog

Since the core is a blackbox instantiation, the configuration file applies the user's configuration information to the simulation model and implementation netlist. If using an VHDL or Verilog synthesis tool, two configuration options are available. The LogiCORE PCI interface can be configured and downloaded from the web or the user can simply edit the configuration file. To modify the design for either implementation or simulation in Verilog, edit the cfg.v file. If using VHDL, edit the cfg.vhd file included with the design files. This file supplies the configuration information to both the implementation black box, which is instantiated in your design, and the functional simulation model. The *LogiCORE PCI Design Guide* has more details on the contents of the cfg file and how to edit this file.

Selectable Options

Enable 66 MHz (Virtex PCI64 only)

The Enable 66 MHz Checkbox allows the user to set the value of the 66 MHz capable bit in the Status register. This option should be checked for all 66 MHz designs.

Latency Timer

The Latency Timer Checkbox allows the user to enable or disable the Latency Timer and its associated register in configuration space. If the latency timer register is enabled, the PCI bus host will write a value to this register at system initialization. The value written to this register determines the minimum number of clocks guaranteed to the initiator during bus transactions. See the LogiCORE PCI Design Guide for more information.

Base Address Register Enable

The Base Address Register Enable Checkboxes allow the user to enable or disable any of the three base address registers. The user should enable at least one base address register. Each enabled base address register may then be configured by the user to allocate memory space or I/O space.

External Subsystem

The External Subsystem Checkbox allows the user to disable the Subsystem ID and Subsystem Vendor ID fields. Subsystem ID and Subsystem Vendor ID uniquely identify the add-in board or subsystem containing the PCI device. The option to disable the Subsystem ID and Subsystem Vendor ID fields is provided to enable the user to design an application which dynamically provides this information to the LogiCORE PCI64 interface, rather than have it defined in the configuration file. When this option is enabled, the Subsystem ID and Subsystem Vendor ID must be supplied to the interface using the `SUB_DATA[31:0]` bus.

Cap List Enable

The Cap List Enable Checkbox allows the user to enable the Capabilities List Pointer located at address 34h in configuration space. This pointer provides an offset into user configuration space for the location of the first item in a capabilities list. Enabling this feature also sets the capabilities bit in the Status register and enables User Configuration Space.

INTA# Enable

The INTA# Enable Checkbox enables the Interrupt Pin field and the Interrupt Line register. If the user chooses to enable this feature, these allow the device driver and operating system to recognize that the device is connected to the INTA# interrupt pin.

User Config Space

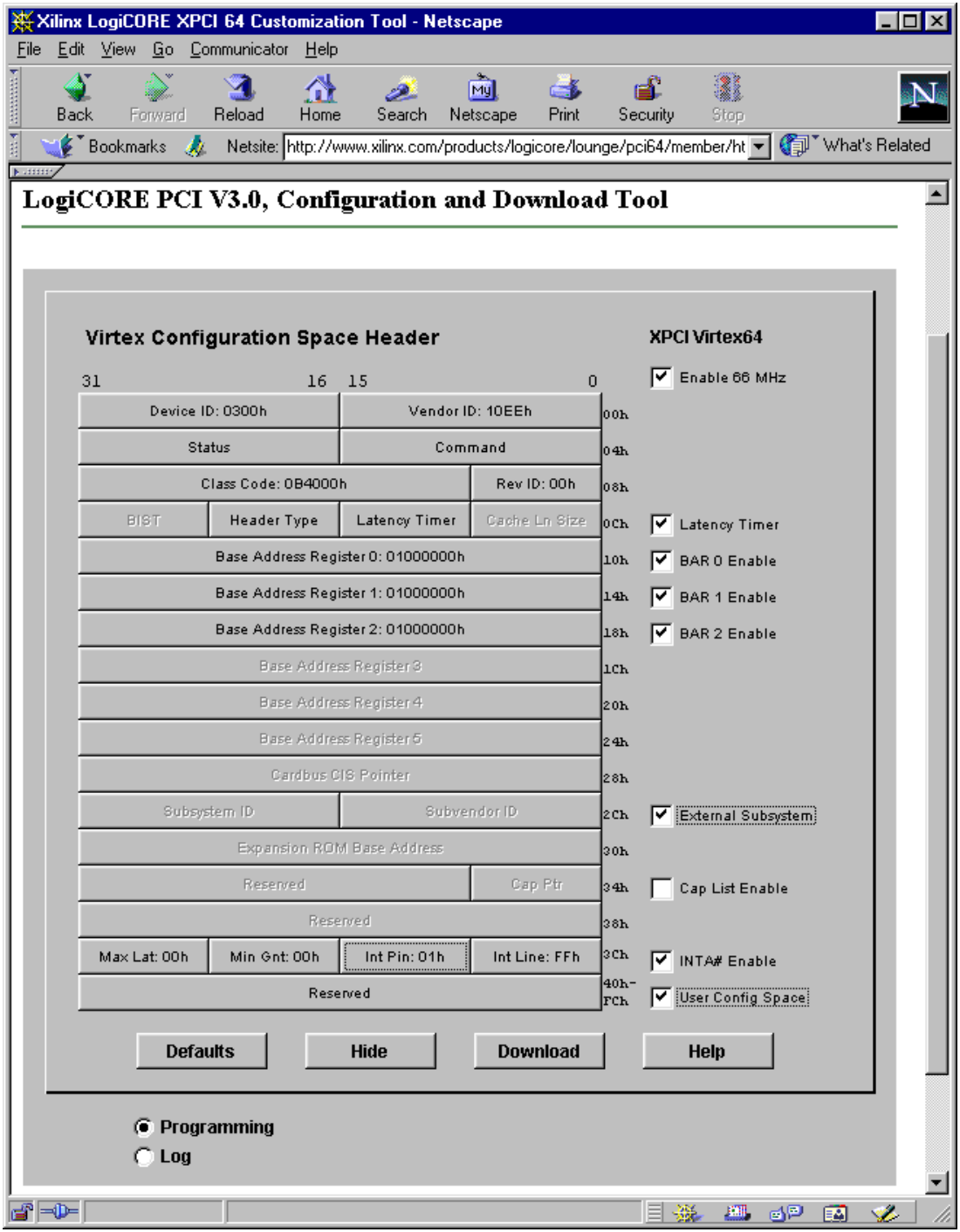
The User Config Space Checkbox enables the user to define and control configuration space addresses 40h through FCh. This checkbox is also selected when the Cap List Enable checkbox is checked because the Capabilities List must reside in the 40h - FCh address range of configuration space. However, the user may enable this option by itself in order to allocate space for internal configuration registers. The responsibility for managing data transfer to and from this space belongs to the user once the this feature is enabled. The LogiCORE PCI Design Guide contains examples of interfacing to user configuration space.

Core Features

Base Address Registers

The LogiCORE interface supports up to three BARs. The user is free to use any combination of BAR desired. Use of BAR 0 is recommend since some PCs will not recognize a interface that doesn't have a BAR 0. The user may enable or disable the BAR settings using the configuration tool on the web or by editing the cfg file. Each BAR has several attributes that can be changed. Refer to the *LogiCORE PCI Design Guide* for more details.

Figure 2: PCI Configuration and Download GUI (Demo available at www.xilinx.com/pci)



Virtex
PCI Compliance Checklist

Component Product Information

Date	May, 1999
Vendor Name	Xilinx, Inc.
Vendor Street Address	2100 Logic Drive
Vendor City, State, Zip	San Jose, CA 95124 U.S.A
Vendor Phone Number	+1 408-559-7778
Vendor Contact, Title	Per Holmberg LogiCORE Marketing Manager
Product Name	Virtex
Product Model Number	XCVxxx
Product Revision Level	

Component Electrical Checklist

This checklist applies to the following Component/Manufacturer: Virtex/ Xilinx, Inc.

All items were verified over the following range of junction temperatures: ____min ____max

OR

All items were verified over the following range of CASE temperatures: - 5 ° C min 85 ° C max

5 V Signaling

Type	Description	Yes or N/A
CE1.	Component supports 5V signaling environment? if “na”, skip to section “3.3V Signaling” below.	yes ✓ na ___
CE2.	Component operates over voltage range 5V +/- 5%? “na” allowed for components that support 5V signaling, but draw power from a supply other than Vcc 5V.	na ___ yes ✓ no___
CE3.	Voltages between 2.0V and Vcc+0.5V are recognized as logic high?	yes ✓ no___
CE4.	Voltages between -0.5V and 0.8V are recognized as logic low?	yes ✓ no___
CE5.	All inputs sink less than 70uA when pulled to 2.7V DC?	yes ✓ no___
CE6.	All inputs source less than 70uA when pulled to 0.5V DC?	yes ✓ no___
CE7.	All outputs drive to 2.4V (min) in the high state while sourcing 2mA?	yes ✓ no___
CE8.	All outputs drive to 0.55V (max) in the low state, sinking 3 or 6*mA?	yes ✓ no___
CE9.	Outputs source at least 44mA at 1.4V in the high state? *proven at: 3.0 Vcc= min, ___process=worst/slow, ___ junction temp= 85°C(max)* *by: ___ SPICE simulation, ✓ device characterization, ___ other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE10.	Outputs source no more than 142mA at 3.1V in the high state? *proven at: 3.6 Vcc=max, ___ process=best/fast, ___ junction temp= -5°C(min)* *by: ___ SPICE simulation, ✓ device characterization, ___ other:_____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE11.	Outputs sink at least 95mA at 2.2V in the low state? *proven at: 3.6 Vcc=max, __ process=worst/slow, ___ junctiontemp= 85°C(max)* *by: ___ SPICE simulation, ✓ device characterization, ___ other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE12.	Outputs sink no more than 206mA at 0.71V in the low state? *proven at: 3.6 Vcc=max, ___ process=best/fast, ___ junction temp= -5°C(min)* *by: ___ SPICE simulation, ___ device characterization, ___ other:_____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___

Type	Description	Yes or N/A
CE13.	REQ#, GNT# outputs source at least 22mA at 1.4V in the high state? *proven at: 3.6 Vcc=max, ___ process=worst/slow, ___ junction temp= 85°C* *(max) by: ___ SPICE simulation, ✓ device characterization, ___ other:_____*	yes ✓ no___
CE14.	REQ#, GNT# outputs sink at least 47mA at 2.2V in the low state? *proven at: 3.0 Vcc=min, ___ process=worst/slow, ___ junction temp=85°C (max)* *by: ___ SPICE simulation, ✓ device characterization, ___ other:_____*	na ___ yes ✓ no___
CE15.	Clamps on all signals source at least 25mA at -1V, and 91mA at -2V? *proven by: __ SPICE simulation, ✓ device characterization, other:_____*	na ___ yes ✓ no___
CE16.	Unloaded rise times are no lower than 1 V/nS between 0.4 and 2.4V? The unloaded maximum rise time is: 2 V/nS (measured at pin)	yes ✓ no___
CE17.	Unloaded fall times are no lower than 1 V/nS between 2.4 and 0.4V? The unloaded maximum fall time is: 2.5 V/nS (measured at pin)	yes ✓ no___

3.3 V Signaling

Type	Description	Pass/NA
CE18.	Component supports 3.3V signaling environment?	yes ✓ na___
	if “na”, skip to section “Loading and Device Protection” below.	
CE19.	Component operates over voltage range 3.3V +/- 0.3V?	yes ✓ no___
CE20.	Voltages between 0. 5Vcc and Vcc+0.5V are recognized as logic high?	yes ✓ no___
CE21.	Voltages between -0.5V and 0.3Vcc are recognized as logic low?	yes ✓ no___
CE22.	All inputs sink/source less than 10 uA at any voltage from 0V to Vcc?	yes ✓ no___
CE23.	All outputs drive to 0.9Vcc (min) in the high state while sourcing 500uA?	yes ✓ no___
CE24.	All outputs drive to 0.1Vcc (max) in the low state, sinking 1500uA?	yes ✓ no___
CE25.	Outputs source at least 36mA at 0.9V in the high state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 75 ° C (max) by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, RST#, and SERR#	yes ✓ no___
CE26.	Outputs source no more than 115mA at 2.5V in the high state? *proven at: ✓ Vcc=3.6V, ___ process=best/fast, ___ junction temp= 25 ° C (min) by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, RST#, and SERR#	yes ✓ no___
CE27.	Outputs sink at least 48mA at 1.8V in the low state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 75 ° C (max) by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE28.	Outputs sink no more than 137mA at 0.65V in the low state? *proven at: ✓ Vcc=3.6V, ___ process=best/fast, ✓ junction temp= 25 ° C (min) by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE29.	REQ#, GNT# outputs source at least 18mA at 0.9V in the high state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 75 ° C (max) by: ___ SPICE simulation, ✓ device characterization, other: _____*	yes ✓ no___
CE30.	REQ#, GNT# outputs sink at least 24mA at 1.8V in the low state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 75 ° C (max) by: ___ SPICE simulation, ✓ device characterization, other: _____*	yes ✓ no __
CE31.	Clamps on all signals source at least 25mA at -1V, and 91mA at -2V? *proven by: ___ SPICE simulation, ✓ device characterization, other:_____*	yes ✓ no___
CE32.	Clamps on all signals sink at least 25mA at Vcc+1V, and 91mA at Vcc+2V? *proven by: ___ SPICE simulation, ✓ device characterization, other:_____*	na ___ yes ✓ no___

Type	Description	Pass/NA
CE33.	Unloaded rise times are no lower than 1 V/nS between 0.2Vcc and 0.6Vcc? The unloaded maximum rise time is: 1.54 V/nS (measured at pin)	na ___ yes ✓ no___
CE34.	Unloaded fall times are no lower than 1 V/nS between 0.6Vcc and 0.2Vcc? The unloaded maximum fall time is: 3.3 V/nS (measured at pin)	yes ✓ no___

Loading and Device Protection

Type	Description	Yes/No
CE35.	Capacitance on all PCI signals (except CLK, IDSEL) is less than or equal to 10 pF?	yes ✓ no___
CE36.	Capacitance on CLK signal is between 5 and 12 pF?	yes ✓ no___
CE37.	Capacitance on IDSEL signal is less than 8 pF? *capacitance guaranteed by: device characterization* ✓ *other* _____. The maximum inductance on any PCI pin is: *15.9* nH.	yes ✓ no ___
CE38.	Read, understand section “Maximum AC Ratings and Device Protection”? ✓ *believe to be non-issue given technology used proven robustness when exposed to prescribed test condition.*	yes ✓ no___

Timing Specification

Type	Description	Pass / N/A
CE39.	Component is operational at any frequency between DC and 33 MHz? Notes: “na” implies component intended for motherboard use only. To satisfy this requirement, designs are allowed to require software to place the component in the proper state before stopping the clock and return it to an operational state after restarting the clock.	yes ✓ na___
CE40.	Component is operational with a CLK High Time of 11 nS for 33 Mhz PCI, 6 ns for 66 Mhz PCI?	na ___ yes ✓ no___
CE41.	Component is operational with a CLK Low Time of 11 nS for 33 Mhz PCI, 6 ns for 66 Mhz PCI?	na ___ yes ✓ no___
CE42.	All bussed signals are driven valid between 2 and 11 nS after CLK for 33 Mhz PCI, between 2 and 6 ns for 66 Mhz PCI?	yes ✓ no___
CE43.	REQ# and GNT# signals are driven valid between 2 and 12 nS after CLK for 33 Mhz PCI, between 2 and 6 ns for 66 Mhz PCI?	yes ✓ no___
CE44.	All Tri-state signals become active no earlier than 2 nS after CLK?	yes ✓ no___
CE45.	All Tri-state signals float no later than 28 nS after CLK for 33 Mhz PCI, no later than 14 nS for 66 Mhz PCI?	yes ✓ no___
CE46.	All bussed inputs require no more than 7 nS setup to CLK for 33 Mhz PCI, no more than 3 nS for 66 Mhz PCI?	yes ✓ no___
CE47.	REQ# requires no more than 12 nS setup to CLK for 33 Mhz PCI, no more than 5 nS for 66 Mhz PCI?	na ___ yes ✓ no___
CE48.	GNT# requires no more than 10 nS setup to CLK for 33 Mhz PCI, no more than 5 nS for 66 Mhz PCI?	na ___ yes ✓ no___
CE49.	All inputs require no more than 0 nS of hold time after CLK?	yes ✓ no___
CE50.	All outputs are Tri-stated within 40 nS after RST# goes low? *all timings (CE39 through CE50 verified by (check all that apply)* *____ static timing design tools (MOTIVE, QTV, QuickPath, Veritime...)* *____ dynamic timing design tools (Verilog, Qsim, Quicksim, ViewSim, VHDL, ...)* *✓ silicon AC testing* *____other_____* *NOTE: Maximum and minimum timings assume different output loadings for both 5.0V and 3.3V parts. See PCI Spec Rev 2.1 page 134 note #2.*	yes ✓ no___

64-bit Components

Type	Description	Pass or N/A
	Component is 32-bit only, this section is	NA__
CE51.	Component senses, during RST# active, its connection to 64-bit wires?	yes ✓ no___
CE52.	64-bit input signals will be stable when not connected?	yes ✓ no__

Explanations:

CE51: The Virtex device must be configured prior to the deassertion of RST#.
CE52: The user must follow recommendations explained in the documentation.

This section should be used to clarify any answers on checklist items above. Please key explanation to item number.

Component Product Information

Date	May, 1999
Vendor Name	Xilinx, Inc.
Vendor Street Address	2100 Logic Drive
Vendor City, State, Zip	San Jose, CA 95124 U.S.A
Vendor Phone Number	+1 408-559-7778
Vendor Contact, Title	Per Holmberg LogiCORE Marketing Manager
Product Name	XC4000XLA
Product Model Number	XC4xxxXLA
Product Revision Level	

Component Electrical Checklist

This checklist applies to the following Component/Manufacturer: XC4000XLA / Xilinx, Inc.

All items were verified over the following range of junction temperatures: ____min ____max

OR

All items were verified over the following range of CASE temperatures: -10 °C min 125 °C max

5 V Signaling

Type	Description	Yes or N/A
CE1.	Component supports 5V signaling environment? if “na”, skip to section “3.3V Signaling” below.	yes✓ na__
CE2.	Component operates over voltage range 5V +/- 5%? “na” allowed for components that support 5V signaling, but draw power from a supply other than Vcc 5V.	na✓ yes ___ no___
CE3.	Voltages between 2.0V and Vcc+0.5V are recognized as logic high?	yes ✓ no___
CE4.	Voltages between -0.5V and 0.8V are recognized as logic low?	yes ✓ no___
CE5.	All inputs sink less than 70uA when pulled to 2.7V DC?	yes ✓ no___
CE6.	All inputs source less than 70uA when pulled to 0.5V DC?	yes ✓ no___
CE7.	All outputs drive to 2.4V (min) in the high state while sourcing 2mA?	yes ✓ no___
CE8.	All outputs drive to 0.55V (max) in the low state, sinking 3 or 6*mA?	yes ✓ no___
CE9.	Outputs source at least 44mA at 1.4V in the high state? *proven at: 3.0 Vcc= min, ___ process=worst/slow, ___ junction temp=125 °C ____ (max) by: ___ SPICE simulation, ✓ device characterization, ___ other:_____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE10.	Outputs source no more than 142mA at 3.1V in the high state? *proven at: 3.6 Vcc=max, ___ process=best/fast, ___ junction temp= -10 °C min)by: ___ SPICE simulation, ✓ device characterization, ___ other:_____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE11.	Outputs sink at least 95mA at 2.2V in the low state? *proven at: 3.6 Vcc=max, ___ process=worst/slow, ___ junction temp=125 °C (max) by: ___ SPICE simulation, ___ device characterization, ___ other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE12.	Outputs sink no more than 206mA at 0.71V in the low state? *proven at: 3.6 Vcc=max, ___ process=best/fast, ___ junction temp=-10 °C (min) by: ___ SPICE simulation, ✓ device characterization, ___ other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___

Type	Description	Yes or N/A
CE13.	REQ#, GNT# outputs source at least 22mA at 1.4V in the high state? *proven at: 3.6 Vcc=max, ___ process=worst/slow, ___ junction temp=125 °C (max) by: ___ SPICE simulation, __ device characterization, ___ other:_____*	yes ✓ no___
CE14.	REQ#, GNT# outputs sink at least 47mA at 2.2V in the low state? *proven at: 3.0 Vcc=min, ___ process=worst/slow, ___ junction temp=125 °C (max) by: ___ SPICE simulation, __ device characterization, ___ other:_____*	na ___ yes ✓ no___
CE15.	Clamps on all signals source at least 25mA at -1V, and 91mA at -2V? *proven by: __ SPICE simulation, ✓ device characterization, other:_____*	na ___ yes ✓ no___
CE16.	Unloaded rise times are no lower than 1 V/nS between 0.4 and 2.4V? The unloaded maximum rise time is: 2.3V/nS (measured at pin)	yes ✓ no___
CE17.	Unloaded fall times are no lower than 1 V/nS between 2.4 and 0.4V? The unloaded maximum fall time is: 2.1V/nS (measured at pin)	yes ✓ no___

3.3 V Signaling

Type	Description	Pass/NA
CE18.	Component supports 3.3V signaling environment?	yes ✓ na___
	if “na”, skip to section “Loading and Device Protection” below.	
CE19.	Component operates over voltage range 3.3V +/- 0.3V?	yes ✓ no___
CE20.	Voltages between 0. 5Vcc and Vcc+0.5V are recognized as logic high?	yes ✓ no___
CE21.	Voltages between -0.5V and 0.3Vcc are recognized as logic low?	yes ✓ no___
CE22.	All inputs sink/source less than 10 uA at any voltage from 0V to Vcc?	yes ✓ no___
CE23.	All outputs drive to 0.9Vcc (min) in the high state while sourcing 500uA?	yes ✓ no___
CE24.	All outputs drive to 0.1Vcc (max) in the low state, sinking 1500uA?	yes ✓ no___
CE25.	Outputs source at least 36mA at 0.9V in the high state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 85 ° C (max) by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, RST#, and SERR#	yes ✓ no___
CE26.	Outputs source no more than 115mA at 2.5V in the high state? *proven at: ✓ Vcc=3.6V, ___ process=best/fast, ✓ junction temp= 25 ° C (min) by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, RST#, and SERR#	yes ✓ no___
CE27.	Outputs sink at least 48mA at 1.8V in the low state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 85 ° C (max) by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE28.	Outputs sink no more than 137mA at 0.65V in the low state? *proven at: ✓ Vcc=3.6V, ___ process=best/fast, ✓ junction temp= 25 ° C (min) by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE29.	REQ#, GNT# outputs source at least 18mA at 0.9V in the high state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 85 ° C (max) by: ___ SPICE simulation, ✓ device characterization, other: _____*	yes ✓ no___
CE30.	REQ#, GNT# outputs sink at least 24mA at 1.8V in the low state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 85 ° C (max) by: ___ SPICE simulation, ✓ device characterization, other: _____*	yes ✓ no __
CE31.	Clamps on all signals source at least 25mA at -1V, and 91mA at -2V? *proven by: ___ SPICE simulation, ✓ device characterization, other: _____*	yes ✓ no___
CE32.	Clamps on all signals sink at least 25mA at Vcc+1V, and 91mA at Vcc+2V? *proven by: ___ SPICE simulation, ✓ device characterization, other: _____*	na ___ yes ✓ no___

Type	Description	Pass/NA
CE33.	Unloaded rise times are no lower than 1 V/nS between 0.2Vcc and 0.6Vcc? The unloaded maximum rise time is: 1.55 V/nS (measured at pin)	na ___ yes ✓ no___
CE34.	Unloaded fall times are no lower than 1 V/nS between 0.6Vcc and 0.2Vcc? The unloaded maximum fall time is: 1.43 V/nS (measured at pin)	yes ✓ no___

Loading and Device Protection

Type	Description	Yes/No
CE35.	Capacitance on all PCI signals (except CLK, IDSEL) is less than or equal to 10 pF?	yes ✓ no___
CE36.	Capacitance on CLK signal is between 5 and 12 pF?	yes ✓ no___
CE37.	Capacitance on IDSEL signal is less than 8 pF? *capacitance guaranteed by: device characterization* ✓ *other* _____. The maximum inductance on any PCI pin is: *15.9* nH.	yes ✓ no ___
CE38.	Read, understand section “Maximum AC Ratings and Device Protection”? ✓ *believe to be non-issue given technology used proven robustness when exposed to prescribed test condition.*	yes ✓ no___

Timing Specification

Type	Description	Pass / N/A
CE39.	Component is operational at any frequency between DC and 33 MHz? Notes: “na” implies component intended for motherboard use only. To satisfy this requirement, designs are allowed to require software to place the component in the proper state before stopping the clock and return it to an operational state after restarting the clock.	yes ✓ na___
CE40.	Component is operational with a CLK High Time of 11 nS for 33 Mhz PCI, 6 ns for 66 Mhz PCI?	na ___ yes ✓ no___
CE41.	Component is operational with a CLK Low Time of 11 nS for 33 Mhz PCI, 6 ns for 66 Mhz PCI?	na ___ yes ✓ no___
CE42.	All bussed signals are driven valid between 2 and 11 nS after CLK for 33 Mhz PCI, between 2 and 6 ns for 66 Mhz PCI?	yes ✓ no___
CE43.	REQ# and GNT# signals are driven valid between 2 and 12 nS after CLK for 33 Mhz PCI, between 2 and 6 ns for 66 Mhz PCI?	yes ✓ no___
CE44.	All Tri-state signals become active no earlier than 2 nS after CLK?	yes ✓ no___
CE45.	All Tri-state signals float no later than 28 nS after CLK for 33 Mhz PCI, no later than 14 nS for 66 Mhz PCI?	yes ✓ no___
CE46.	All bussed inputs require no more than 7 nS setup to CLK for 33 Mhz PCI, no more than 3 nS for 66 Mhz PCI?	yes ✓ no___
CE47.	REQ# requires no more than 12 nS setup to CLK for 33 Mhz PCI, no more than 5 nS for 66 Mhz PCI?	na ___ yes ✓ no___
CE48.	GNT# requires no more than 10 nS setup to CLK for 33 Mhz PCI, no more than 5 nS for 66 Mhz PCI?	na ___ yes ✓ no___
CE49.	All inputs require no more than 0 nS of hold time after CLK?	yes ✓ no___
CE50.	All outputs are Tri-stated within 40 nS after RST# goes low? *all timings (CE39 through CE50 verified by (check all that apply)* ____ *static timing design tools (MOTIVE, QTV, QuickPath, Veritime...)* ____ *dynamic timing design tools (Verilog, Qsim, Quicksim, ViewSim, VHDL, ...)* ✓ *silicon AC testing* ____*other*_____ *NOTE: Maximum and minimum timings assume different output loadings for both 5.0V and 3.3V parts. See PCI Spec Rev 2.1 page 134 note #2.*	yes ✓ no___

64-bit Components

Type	Description	Pass or N/A
	Component is 32-bit only, this section is	NA ✓
CE51.	Component senses, during RST# active, its connection to 64-bit wires?	yes ___ no___
CE52.	64-bit input signals will be stable when not connected?	yes ___ no__

Explanations:

This section should be used to clarify any answers on checklist items above. Please key explanation to item number.

Component Product Information

Date	May, 1999
Vendor Name	Xilinx, Inc.
Vendor Street Address	2100 Logic Drive
Vendor City, State, Zip	San Jose, CA 95124 U.S.A
Vendor Phone Number	+1 408-559-7778
Vendor Contact, Title	Per Holmberg LogiCORE Marketing Manager
Product Name	SpartanXL
Product Model Number	XCSxxxXL
Product Revision Level	

Component Electrical Checklist

This checklist applies to the following Component/Manufacturer: Spartan-XL / Xilinx, Inc.

All items were verified over the following range of junction temperatures: ____min ____max

OR

All items were verified over the following range of CASE temperatures: -10 ° C min +125 ° C max

5 V Signaling

Type	Description	Yes or N/A
CE1.	Component supports 5V signaling environment?	yes ✓ na __
	if “na”, skip to section “3.3V Signaling” below.	
CE2.	Component operates over voltage range 5V +/- 5%?	na___ yes ✓ no___
	“na” allowed for components that support 5V signaling, but draw power from a supply other than Vcc 5V.	
CE3.	Voltages between 2.0V and Vcc+0.5V are recognized as logic high?	yes ✓ no___
CE4.	Voltages between -0.5V and 0.8V are recognized as logic low?	yes ✓ no___
CE5.	All inputs sink less than 70uA when pulled to 2.7V DC?	yes ✓ no___
CE6.	All inputs source less than 70uA when pulled to 0.5V DC?	yes ✓ no___
CE7.	All outputs drive to 2.4V (min) in the high state while sourcing 2mA?	yes ✓ no___
CE8.	All outputs drive to 0.55V (max) in the low state, sinking 3 or 6*mA?	yes ✓ no___
CE9.	Outputs source at least 44mA at 1.4V in the high state? *proven at: 3.0 Vcc= min, ___ process=worst/slow, ✓ junction temp= 85 °C max) by: ___ SPICE simulation, ✓device characterization, other:_____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE10.	Outputs source no more than 142mA at 3.1V in the high state? *proven at: 3.6 Vcc=max, ___ process=best/fast, ✓ junction temp= -5 °C (min)by: ___ SPICE simulation, ✓ device characterization, ___ other:_____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE11.	Outputs sink at least 95mA at 2.2V in the low state? *proven at: 3.6 Vcc=max, ___ process=worst/slow, ✓ junction temp = 85 °C (max) by: ___ SPICE simulation,✓ device characterization, ___ other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE12.	Outputs sink no more than 206mA at 0.71V in the low state? *proven at: 3.6 Vcc=max, ___ process=best/fast,✓ junction temp = -5 °C (min) by: ___ SPICE simulation, ✓ device characterization, ___ other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE13.	REQ#, GNT# outputs source at least 22mA at 1.4V in the high state? *proven at: 3.6 Vcc=max, ___ process=worst/slow, ✓ junction temp = 85 °C (max) by: ___ SPICE simulation, ✓ device characterization, ___ other_____*	yes ✓ no___
CE14.	REQ#, GNT# outputs sink at least 47mA at 2.2V in the low state? *proven at: 3.0 Vcc=min, ___ process=worst/slow, ✓ junction temp= 85 °C (max) by: ___ SPICE simulation, ✓ device characterization, ___ other_____*	na ___ yes ✓ no___

Type	Description	Yes or N/A
CE15.	Clamps on all signals source at least 25mA at -1V, and 91mA at -2V? *proven by:__ SPICE simulation, ✓ device characterization, other:_____*	na ___ yes ✓ no___
CE16.	Unloaded rise times are no lower than 1 V/nS between 0.4 and 2.4V? The unloaded maximum rise time is: *22* V/nS (measured at pin)	yes ✓ no___
CE17.	Unloaded fall times are no lower than 1 V/nS between 2.4 and 0.4V? The unloaded maximum fall time is: *2.3* V/nS (measured at pin)	yes ✓ no___

3.3 V Signaling

Type	Description	Pass/NA
CE18.	Component supports 3.3V signaling environment?	yes ✓ na___
	if “na”, skip to section “Loading and Device Protection” below.	
CE19.	Component operates over voltage range 3.3V +/- 0.3V?	yes ✓ no___
CE20.	Voltages between 0. 5Vcc and Vcc+0.5V are recognized as logic high?	yes ✓ no___
CE21.	Voltages between -0.5V and 0.3Vcc are recognized as logic low?	yes ✓ no___
CE22.	All inputs sink/source less than 10 uA at any voltage from 0V to Vcc?	yes ✓ no___
CE23.	All outputs drive to 0.9Vcc (min) in the high state while sourcing 500uA?	yes ✓ no___
CE24.	All outputs drive to 0.1Vcc (max) in the low state, sinking 1500uA?	yes ✓ no___
CE25.	Outputs source at least 36mA at 0.9V in the high state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 75 °C max)* *by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, RST#, and SERR#	yes ✓ no___
CE26.	Outputs source no more than 115mA at 2.5V in the high state? *proven at: ✓ Vcc=3.6V, ___ process=best/fast, ✓ junction temp = 25 °C (min)* *by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, RST#, and SERR#	yes ✓ no___
CE27.	Outputs sink at least 48mA at 1.8V in the low state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp = 75 °C (max)* *by: ___ SPICE simulation, ✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE28.	Outputs sink no more than 137mA at 0.65V in the low state? *proven at: ✓ Vcc=3.6V, ✓ process=best/fast, ___ junction temp= 25 °C (min) by: ___ SPICE simulation,✓ device characterization, other: _____* NOTE: applies to all outputs except REQ#, GNT#, CLK, and RST#	yes ✓ no___
CE29.	REQ#, GNT# outputs source at least 18mA at 0.9V in the high state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 75 °C (max)* *by: ___ SPICE simulation, ✓ device characterization, other: _____*	yes ✓ no___
CE30.	REQ#, GNT# outputs sink at least 24mA at 1.8V in the low state? *proven at: ✓ Vcc=3.0V, ___ process=worst/slow, ✓ junction temp= 75 °C(max)* *by: ___ SPICE simulation, ✓ device characterization, other: _____*	yes ✓ no __
CE31.	Clamps on all signals source at least 25mA at -1V, and 91mA at -2V? *proven by: ___ SPICE simulation, ✓ device characterization, other:_____*	yes ✓ no___
CE32.	Clamps on all signals sink at least 25mA at Vcc+1V, and 91mA at Vcc+2V? *proven by: ___ SPICE simulation, ✓ device characterization, other:_____*	na ___ yes ✓ no___

Type	Description	Pass/NA
CE33.	Unloaded rise times are no lower than 1 V/nS between 0.2Vcc and 0.6Vcc? The unloaded maximum rise time is: 1.47 V/nS (measured at pin)	na ___ yes ✓ no___
CE34.	Unloaded fall times are no lower than 1 V/nS between 0.6Vcc and 0.2Vcc? The unloaded maximum fall time is: 1.5 V/nS (measured at pin)	yes ✓ no___

Loading and Device Protection

Type	Description	Yes/No
CE35.	Capacitance on all PCI signals (except CLK, IDSEL) is less than or equal to 10 pF?	yes ✓ no___
CE36.	Capacitance on CLK signal is between 5 and 12 pF?	yes ✓ no___
CE37.	Capacitance on IDSEL signal is less than 8 pF? *capacitance guaranteed by: device characterization* ✓ *other* _____. The maximum inductance on any PCI pin is: *15.9* nH.	yes ✓ no ___
CE38.	Read, understand section “Maximum AC Ratings and Device Protection”? ✓ *believe to be non-issue given technology used proven robustness when exposed to prescribed test condition.*	yes ✓ no___

Timing Specification

Type	Description	Pass / N/A
CE39.	Component is operational at any frequency between DC and 33 MHz? Notes: “na” implies component intended for motherboard use only. To satisfy this requirement, designs are allowed to require software to place the component in the proper state before stopping the clock and return it to an operational state after restarting the clock.	yes ✓ na___
CE40.	Component is operational with a CLK High Time of 11 nS for 33 Mhz PCI, 6 ns for 66 Mhz PCI?	na ___ yes ✓ no___
CE41.	Component is operational with a CLK Low Time of 11 nS for 33 Mhz PCI, 6 ns for 66 Mhz PCI?	na ___ yes ✓ no___
CE42.	All bussed signals are driven valid between 2 and 11 nS after CLK for 33 Mhz PCI, between 2 and 6 ns for 66 Mhz PCI?	yes ✓ no___
CE43.	REQ# and GNT# signals are driven valid between 2 and 12 nS after CLK for 33 Mhz PCI, between 2 and 6 ns for 66 Mhz PCI?	yes ✓ no___
CE44.	All Tri-state signals become active no earlier than 2 nS after CLK?	yes ✓ no___
CE45.	All Tri-state signals float no later than 28 nS after CLK for 33 Mhz PCI, no later than 14 nS for 66 Mhz PCI?	yes ✓ no___
CE46.	All bussed inputs require no more than 7 nS setup to CLK for 33 Mhz PCI, no more than 3 nS for 66 Mhz PCI?	yes ✓ no___
CE47.	REQ# requires no more than 12 nS setup to CLK for 33 Mhz PCI, no more than 5 nS for 66 Mhz PCI?	na ___ yes ✓ no___
CE48.	GNT# requires no more than 10 nS setup to CLK for 33 Mhz PCI, no more than 5 nS for 66 Mhz PCI?	na ___ yes ✓ no___
CE49.	All inputs require no more than 0 nS of hold time after CLK?	yes ✓ no___
CE50.	All outputs are Tri-stated within 40 nS after RST# goes low? *all timings (CE39 through CE50 verified by (check all that apply)* ____ *static timing design tools (MOTIVE, QTV, QuickPath, Veritime...)* ____ *dynamic timing design tools (Verilog, Qsim, Quicksim, ViewSim, VHDL, ...)* ✓ *silicon AC testing* ____*other*_____ *NOTE: Maximum and minimum timings assume different output loadings for both 5.0V and 3.3V parts. See PCI Spec Rev 2.1 page 134 note #2.*	yes ✓ no___

64-bit Components

Type	Description	Pass or N/A
	Component is 32-bit only, this section is	NA ✓
CE51.	Component senses, during RST# active, its connection to 64-bit wires?	yes ___ no__
CE52.	64-bit input signals will be stable when not connected?	yes __ no__

Explanations:

This section should be used to clarify any answers on checklist items above. Please key explanation to item number.

Component Product Information

Date	May, 1999
Vendor Name	Xilinx, Inc.
Vendor Street Address	2100 Logic Drive
Note Vendor City, State, Zip	San Jose, CA 95124 U.S.A
Vendor Phone Number	+1 408-559-7778
Vendor Contact, Title	Per Holmberg LogiCORE Marketing Manager
Product Name	LogiCORE PCI32 and PCI64 Interfaces
Product Model Number	DO-DI-PCI64, DO-DI-PCI32 XC4000XLA FPGA Spartan-XL FPGA
Product Revision Level	All cores based on V3.0

Component Configuration Checklist

Organization

Type	Description	Yes or No
CO1.	Does each PCI resource have a configuration space based on the 256 byte template defined in section 6.1, with a predefined 64 byte header and a 192 byte device specific region?	yes ✓ no___
CO2.	Do all functions in the device support the Vendor ID, Device ID, Command, Status, Header Type and Class Code fields in the header? See figure 6-1.	yes ✓ no___
CO3.	Is the configuration space available for access at all times?	yes ✓ no___
CO4.	Are writes to reserved registers or read only bits completed normally and the data discarded?	yes ✓ no___
CO5.	Are reads to reserved or unimplemented registers, or bits, completed normally and a data value of 0 returned?	yes ✓ no___
CO6.	Is the vendor ID a number allocated by the PCI SIG? **End users must use their own Vendor ID.**	yes ✓ no___
CO7.	Does the Header Type field have a valid encoding?	yes ✓ no___
CO8.	Do multi-byte transactions access the appropriate registers and are the registers in “little endian” order?	yes ✓ no___
CO9.	Are all READ ONLY register values within legal ranges? For example, the Interrupt Pin register must only contain values 0-4.	yes ✓ no___
CO10.	Is the class code in compliance with the definition in Appendix D?	yes ✓ no___
CO11.	Is the predefined header portion of configuration space accessible as bytes, words, and dwords?	yes ✓ no___
CO12.	Is the device a multifunction device?	yes __ no ✓
CO13.	If the device is multifunction, are config space accesses to unimplemented functions ignored?	yes ___ no___ N/A ✓

Indicate either N/A (Not Applicable) or Implemented by placing a check in the appropriate box. Grayed areas indicate invalid selections. This table should be completed for each function in a multifunction device.

Location	Name	Required/Optional	N/A	Implemented
00h-01h	Vendor ID	Required		✓
02h-03h	Device ID	Required		✓
04h-05h	Command	Required		✓
06h-07h	Status	Required		✓
08h	Revision ID	Required		✓
09h-0Bh	Class Code	Required		✓
0Ch	Cache Line Size	Required by master devices/functions that can generate Memory Write and Invalidate	✓	
0Dh	Latency Timer	Required by master devices/functions that can burst more than two data phases		✓
0Eh	Header Type	If the device is multi-functional, then bit 7 must be set to a 1. The remaining bits are required to have a defined value.		✓
0Fh	BIST	Optional	✓	
10h-27h	Base Address Registers	1 or more required for any address allocation.		✓
28h-2Bh	Cardbus CIS Pointer	Optional	✓	
2CH=2Dh	Subsystem Vendor ID	Optional		✓
2Eh-2Fh	Subsystem ID	Optional		✓
30h-2Fh	Expansion ROM Base Address	Required for devices/functions that have expansion ROM.	✓	
34h-3Bh	Reserved			✓
3Ch	Interrupt Line	Required by devices/functions that use an interrupt pin.		✓
3Dh	Interrupt Pin	Required by devices/functions that use an interrupt pin.		✓
3Eh	Min_Gnt	Optional		✓
3Fh	Max_Lat	Optional		✓

Device Control

This section should be completed individually for all functions in a multifunction device.

Type	Description	Yes/No
DC1.	When the command register is loaded with a 0000h is the device/function logically disconnected from the PCI, with the exception of configuration accesses? (Devices in BOOT code path are exempt).	yes ✓ no___
DC2.	Is the device/function disabled after the assertion of PCI RST#? (Devices in BOOT code path are exempt).	yes ✓ no___

In the following tables for Command and Status Registers, an "✓" in the "Target" or "Master" columns, indicates that applying the bit is appropriate. "N/A" indicates that applying the bit is not applicable, but must return a 0 when read.

Bit	Name	Required/Optional	Target	Master
0	I/O Space	Required if device/function has registers mapped into I/O space.	✓	N/A
1	Memory Space	Required if device/function responds to memory space accesses.	✓	N/A
2	Bus Master	Required	N/A	✓
3	Special Cycles	Required for devices/functions that can respond to Special Cycles.	N/A	N/A
4	Memory Write and Invalidate Enable	Required for devices/functions that generate Memory Write and Invalidate cycles.	N/A	N/A
5	VGA Palette snoop	Required for VGA or graphical devices/functions that snoop VGA color palette.	N/A	N/A
6	Parity Error Response	Required unless exempted per section 3.7.2.	✓	✓
7	Wait cycle control	Optional	N/A	N/A
8	SERR# enable	Required if device/function has SERR# pin.	✓	N/A
9	Fast Back-to-Back Enable	Required if Master device/function can support fast back-to-back cycles among different targets.	N/A	N/A
10-15	Reserved		✓	✓

Device Status

This section should be completed individually for all functions in a multifunction device.

Device Status Questions

Type	Description	Yes/No
DS1.	Do all implemented read/write bits in the Status reset to 0?	yes ✓ no___
DS2.	Are read/write bits set to a 1 exclusively by the device/function?	yes ✓ no___
DS3.	Are read/write bits reset to a 0 when PCIRST# is asserted?	yes ✓ no___
DS4.	Are read/write bits reset to a 0 by writing a 1 to the bit?	yes ✓ no___

Bit	Name	Required/Optional	Target	Master
0-4	Reserved	Required	✓	✓
5	66 Mhz Capable	Required for 66Mhz capable devices	✓	✓
6	UDF Supported	Optional	N/A	N/A
7	Fast Back-to-Back Capable	Optional	N/A	N/A
8	Data Parity Detected	Required	N/A	✓
9-10	DEVSEL Timing	Required	✓	N/A
11	Signaled Target Abort	Required for devices/functions that are capable of signaling target abort	✓	N/A
12	Received Target Abort	Required	N/A	✓
13	Received Master Abort	Required	N/A	✓
14	Signaled System Error	Required for devices/functions that are capable of asserting SERR#	✓	✓
15	Detected Parity Error	Required unless exempted per section 3.7.2	✓	✓

Base Addresses

This section should be completed individually for all functions in a multifunction device

Type	Description	Yes/No
BA1.	If the device/function uses expansion ROM, does it implement the Expansion ROM Base Address Register? **The expansion ROM base address is not supported.**	yes ___ no ↙
BA2.	Do all Base Address registers asking for IO space request 256 bytes or less?	yes ↙ no ___
BA3.	If the device/function has an Expansion ROM Base Address register, does the memory enable bit in the Command register have precedence over the enable bit in the Expansion ROM base Address register? **The expansion ROM base address is not supported.**	yes ___ no ↙
BA4.	Does the device/function use any address space (memory or IO) other than that assigned using Base Address registers? (i.e.; Does the device/function hard-decode any addresses?) Note: If the answer is yes, you must list decoded addresses as explanations at the end of this section.	yes ___ no ↙
BA5.	Does the device/function decode all 32-bits of IO space?	yes ↙ no ___
BA6.	If the device/function has an Expansion ROM Base Address register, is the size of the memory space requested 16MB or smaller?	yes ___ no___

VGA Devices

VGA Devices (fill in this section only if component is VGA device)

Type	Description	Yes/No
VG1.	Is palette snoop implemented, including bit in Command register?	yes ___ no___
VG2.	Is Expansion ROM Base Address register implemented and provide full relocatability of the expansion ROM? (The device must NOT do a hard decode of 0C0000h).	yes ___ no ___
VG3.	Does the device come up disabled? (Bottom three bits of Command register must be initialized to zero on power-up and PCIRST#).	yes ___ no ___
VG4.	Does Class Code field indicate VGA device? (value of 030000h).	yes ___ no___
VG5.	Does the device hard-decode only standard ISA VGA addresses and their aliases? (IO addresses 3B0h through 3BBh, 3C0h through 3DFh, Memory addresses 0A0000h through 0BFFFFh)	yes ___ no___
VG6.	Does the device use Base Address Registers to allocate needed space other than standard ISA VGA Addresses? (e.g. for a linear FRAME buffer)	yes ___ no___

General Component Protocol Checklist (Master)

The following checklist is to filled out as a general verification of the IUT's protocol compliance. This checklist applies to all master operations.

Test #	Description	Yes or No
MP1.	All Sustained Tri-State signals are driven high for one clock before being Tri-Stated. (2.1)	yes ✓ no___
MP2.	IUT always asserts all byte enables during each data phase of a Memory Write Invalidate cycle. (3.1.1) **Memory Write and Invalidate command not supported**	yes ___ no ✓
MP3.	IUT always uses Linear Burst Ordering for Memory Write Invalidate cycles. (3.1.1) **Memory Write and Invalidate command not supported.**	yes ___ no ✓
MP4.	IUT always drives IRDY# when data is valid during a write transaction. (3.2.1)	yes ✓ no ___
MP5.	IUT only transfers data when both IRDY# and TRDY# are asserted on the same rising clock edge. (3.2.1)	yes ✓ no ___
MP6.	Once the IUT asserts IRDY# it never changes FRAME# until the current data phase completes. (3.2.1)	yes ✓ no___
MP7.	Once the IUT asserts IRDY# it never changes IRDY# until the current data phase completes. (3.2.1)	yes ✓ no ___
MP8.	IUT never uses reserved burst ordering (AD[1::0] = “01”. (3.2.2) **Value driven onto AD bus is controlled by the user application.**	yes ✓ no __
MP9.	IUT never uses reserved burst ordering (AD[1::0] = “11”. (3.2.2) **Value driven onto AD bust is controlled by the user application.**	yes ✓ no __
MP10.	IUT always ignores configuration command unless IDSEL is asserted and AD[1::0] are “00”. (3.2.2)	yes ✓ no ____
MP11.	The IUT's AD lines are driven to stable values during every address and data phase. (3.2.4)	yes ✓ no ___
MP12.	The IUT's C/BE# output buffers remain enabled from the first clock of the data phase through the end of the transaction. (3.3.1)	yes ✓ no ___
MP13.	The IUT's C/BE# lines contain valid Byte Enable information during the entire data phase. (3.3.1) **The values on the C/BE# pins are driven by the user application.**	yes ✓ no ___
MP14.	IUT never deasserts FRAME# unless IRDY# is asserted or will be asserted (3.3.3.1)	yes ✓ no ___
MP15.	IUT never deasserts IRDY# until at least one clock after FRAME# is deasserted. (3.3.3.1)	yes ✓ no ___
MP16.	Once the IUT deasserts FRAME# it never reasserts FRAME# during the same transaction. (3.3.3.1)	yes ✓ no ___
MP17.	IUT never terminates with master abort once target has asserted DEVSEL#. (3.3.3.1)	yes ✓ no ___
MP18.	IUT never signals master abort earlier than 5 clocks after FRAME# was first sampled asserted. (3.3.3.1)	yes ✓ no ___
MP19.	IUT always repeats an access exactly as the original when terminated by retry. (3.3.3.2.2) **The retry process is controlled by logic in the user’s application. The user application must repeat the transaction to meet this requirement.**	yes ✓ no ___

Test #	Description	Yes or No
MP20.	IUT never starts cycle unless GNT# is asserted. (3.4.1)	yes ↙ no ___
MP21.	IUT always Tri-States C/BE# and AD within one clock after GNT# negation when bus is idle and FRAME# is negated. (3.4.3)	yes ↙ no___
MP22.	IUT always drives C/BE# and AD within eight clocks of GNT# assertion when bus is idle. (3.4.3)	yes ↙ no___
MP23.	IUT always asserts IRDY# within eight clocks on all data phases. (3.5.2) **The user application must assert M_READY appropriately to meet this requirement.**	yes ↙ no ___
MP24.	IUT always begins lock operation with a read transaction. (3.6) **LOCK# function not supported.**	yes ___ no ↙
MP25.	IUT always releases LOCK# when access is terminated by target-abort or master-abort. (3.6) **LOCK# function not supported.**	yes ___ no ↙
MP26.	IUT always deasserts LOCK# for minimum of one idle cycle between consecutive lock operations. (3.6) **LOCK# function not supported.**	yes ___ no↙
MP27.	IUT always uses Linear Burst Ordering for configuration cycles. (3.7.4)	yes ↙ no ___
MP28.	IUT always drives PAR within one clock of C/BE# and AD being driven. (3.8.1)	yes ↙ no ___
MP29.	IUT always drives PAR such that the number of “1”s on AD[31::0],C/BE[3:0], and PAR equals an even number. (3.8.1)	yes ↙ no ___
MP30.	IUT always drives PERR# (when enabled) active two clocks after data when data parity error is detected. (3.8.2.1)	yes ↙ no ___
MP31.	IUT always drives PERR (when enabled) for a minimum of 1 clock for each data phase that a parity error is detected. (3.8.2.1)	yes ↙ no ___
MP32.	IUT always holds FRAME# asserted for cycle following DUAL command. (3.10.1) **Dual Address command not supported.**	yes ___ no ↙
MP33.	IUT never generates DUAL cycle when upper 32-bits of address are zero. (3.10.1)	yes ___ no ↙

General Component Protocol Checklist (Target)

The following checklist is to filled out as a general verification of the IUT's protocol compliance. This checklist applies to all target operations.

General Component Protocol Checklist (Target)

Test #	Description	Pass or N/A
TP1.	All Sustained Tri-State signals are driven high for one clock before being Tri-Stated.	yes ✓ no ___
TP2.	IUT never reports PERR# until it has claimed the cycle and completed a data phase. (2.2.5)	yes ✓ no ___
TP3.	IUT never aliases reserved commands with other commands. (3.1.1)	yes ✓ no ___
TP4.	32-bit addressable IUT treats DUAL command as reserved. (3.1.1)	yes ✓ no ___
TP5.	Once IUT has asserted TRDY# it never changes TRDY# until the data phase completes. (3.2.1)	yes ✓ no ___
TP6.	Once IUT has asserted TRDY# it never changes DEVSEL# until the data phase completes. (3.2.1)	yes ✓ no ___
TP7.	Once IUT has asserted TRDY# it never changes STOP# until the data phase completes. (3.2.1)	yes ✓ no___
TP8.	Once IUT has asserted STOP# it never changes STOP# until the data phase completes. (3.2.1)	yes ✓ no ___
TP9.	Once IUT has asserted STOP# it never changes TRDY# until the data phase completes. (3.2.1)	yes ✓ no ___
TP10.	Once IUT has asserted STOP# it never changes DEVSEL# until the data phase completes. (3.2.1)	yes ✓ no ___
TP11.	IUT only transfers data when both IRDY# and TRDY# are asserted on the same rising clock edge. (3.2.1)	yes ✓ no ___
TP12.	IUT always asserts TRDY# when data is valid on a read cycle. (3.2.1)	yes ✓ no ___
TP13.	IUT always signals target-abort when unable to complete the entire IO access as defined by the byte enables. (3.2.2) **This function is implemented in the user application. Only the user application could determine if the byte enables were valid for the selected I/O devices.**	yes ✓ no ___
TP14.	IUT never responds to reserved encodings. (3.2.2)	yes ✓ no ___
TP15.	IUT always ignores configuration command unless IDSEL is asserted and AD[1::0] are “00”. (3.2.2)	yes ✓ no ___
TP16.	IUT always disconnects after the first data phase when reserved burst mode is detected. (3.2.2)	yes ✓ no ___
TP17.	The IUT's AD lines are driven to stable values during every address and data phase. (3.2.4)	yes ✓ no ___
TP18.	The IUT's C/BE# output buffers remain enabled from the first clock of the data phase through the end of the transaction. (3.3.1)	yes ✓ no ___
TP19.	IUT never asserts TRDY# during turnaround cycle on a read. (3.3.1)	yes ✓ no ___

General Component Protocol Checklist (Target) (Continued)

Test #	Description	Pass or N/A
TP20.	IUT always deasserts TRDY#,STOP#, and DEVSEL# the clock following the completion of the last data phase. (3.3.3.2)	yes ↙ no ___
TP21.	IUT always signals disconnect when burst crosses resource boundary. (3.3.3.2) **This function would be implemented in the user application.**	yes ↙ no __
TP22.	IUT always deasserts STOP# the cycle immediately following FRAME# being deasserted. (3.3.3.2.1)	yes ↙ no ___
TP23.	Once the IUT has asserted STOP# it never deasserts STOP# until FRAME# is negated. (3.3.3.2.1)	yes ↙ no ___
TP24.	IUT always deasserts TRDY# before signaling target-abort. (3.3.3.2.1)	yes ↙ no ___
TP25.	IUT never deasserts STOP# and continues the transaction. (3.3.3.2.1)	yes ↙ no ___
TP26.	IUT always completes initial data phase within 16 clocks. (3.5.1.1)	yes ↙ no ___
TP27.	IUT always locks minimum of 16 bytes. (3.6) **LOCK# function not supported.**	yes ___ no ↙
TP28.	IUT always issues DEVSEL# before any other response. (3.7.1)	yes ↙ no ___
TP29.	Once IUT has asserted DEVSEL# it never deasserts DEVSEL# until the last data phase has competed except to signal target-abort. (3.7.1)	yes ↙ no ___
TP30.	IUT never responds to special cycles. (3.7.2)	yes ↙ no ___
TP31.	IUT always drives PAR within one clock of C/BE# and AD being driven. (3.8.1)	yes ↙ no ___
TP32.	IUT always drives PAR such that the number of “1”s on AD[31::0],C/BE[3:0], and PAR equals an even number. (3.8.1)	yes ↙ no ___

Component Protocol Checklist for a Master Device

Definition: IUT is an acronym for “Implementation Under Test”.

Test Scenario: 1.1. PCI Device Speed Tests

Memory Transactions

Test #	Description	Pass	N/A
If IUT does not implement memory transactions mark 1 through 10 N/A.			
1	Data transfer after write to fast memory slave.	✓	
2	Data transfer after read from fast memory slave.	✓	
3	Data transfer after write to medium memory slave.	✓	
4	Data transfer after read from medium memory slave.	✓	
5	Data transfer after write to slow memory slave.	✓	
6	Data transfer after read from slow memory slave.	✓	
7	Data transfer after write to subtractive memory slave.	✓	
8	Data transfer after read from subtractive memory slave.	✓	
9	Master abort bit set after write to slower than subtractive memory slave.	✓	
10	Master abort bit set after read from slower than subtractive memory slave.	✓	

I/O Transactions

Test #	Description	Pass	N/A
If IUT does not implement I/O transactions mark 11 through 20 N/A.			
11	Data transfer after write to fast I/O slave.	✓	
12	Data transfer after read from fast I/O slave.	✓	
13	Data transfer after write to medium I/O slave.	✓	
14	Data transfer after read from medium I/O slave.	✓	
15	Data transfer after write to slow I/O slave.	✓	
16	Data transfer after read from slow I/O slave.	✓	
17	Data transfer after write to subtractive I/O slave.	✓	
18	Data transfer after read from subtractive I/O slave.	✓	
19	Master abort bit set after write to slower than subtractive I/O slave.	✓	
20	Master abort bit set after read from slower than subtractive I/O slave.	✓	

Configuration Transactions:

Test #	Description	Pass	N/A
If IUT does not implement configuration transactions mark 21 through 30 N/A.			
21	Data transfer after write to fast config slave.	✓	
22	Data transfer after read from fast config slave.	✓	
23	Data transfer after write to medium config slave.	✓	
24	Data transfer after read from medium config slave.	✓	
25	Data transfer after write to slow config slave.	✓	

Test #	Description	Pass	N/A
26	Data transfer after read from slow config slave.	✓	
27	Data transfer after write to subtractive config slave.	✓	
28	Data transfer after read from subtractive config slave.	✓	
29	Master abort bit set after write to slower than subtractive config slave.	✓	
30	Master abort bit set after read from slower than subtractive config slave.	✓	

Interrupt Acknowledge Transactions:

Test #	Description	Pass	N/A
If IUT does not implement interrupt transactions mark 31 through 35 N/A.			
31	Data transfer after interrupt from fast memory slave.	✓	
32	Data transfer after interrupt from medium memory slave.	✓	
33	Data transfer after interrupt from slow memory slave.	✓	
34	Data transfer after interrupt from subtractive memory slave.	✓	
35	Master abort bit set for interrupt from slower than subtractive memory slave.	✓	

Special Cycle Transactions:

Test #	Description	Pass	N/A
If IUT does not implement special cycle transactions mark 36 through 37 N/A.			
36	Data transfer after special transaction to slave.	✓	
37	Master abort bit is not set after special transaction.	✓	

Test Scenario: 1.2. PCI Bus Target Abort Cycles

Memory Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory transactions mark 1 through 16 N/A.			
1	Target Abort bit set after write to fast memory slave.	✓	
2	IUT does not repeat the write transaction.	✓	
3	IUT's Target Abort bit set after read from fast memory slave.	✓	
4	IUT does not repeat the read transaction.	✓	
5	Target Abort bit set after write to medium memory slave.	✓	
6	IUT does not repeat the write transaction.	✓	
7	IUT's Target Abort bit set after read from medium memory slave.	✓	
8	IUT does not repeat the read transaction.	✓	
9	Target Abort bit set after write to slow memory slave.	✓	
10	IUT does not repeat the write transaction.	✓	
11	IUT's Target Abort bit set after read from slow memory slave.	✓	
12	IUT does not repeat the read transaction.	✓	
13	Target Abort bit set after write to subtractive memory slave.	✓	
14	IUT does not repeat the write transaction.	✓	

Test #	Description	Pass	N/A
15	IUT's Target Abort bit set after read from subtractive memory slave.	✓	
16	IUT does not repeat the read transaction.	✓	

I/O Transactions:

Test #	Description	Pass	N/A
If IUT does not implement I/O transactions mark 17 through 32 N/A.			
17	Target Abort bit set after write to fast I/O slave.	✓	
18	IUT does not repeat the write transaction.	✓	
19	IUT's Target Abort bit set after read from fast I/O slave.	✓	
20	IUT does not repeat the read transaction.	✓	
21	Target Abort bit set after write to medium I/O slave.	✓	
22	IUT does not repeat the write transaction.	✓	
23	IUT's Target Abort bit set after read from medium I/O slave.	✓	
24	IUT does not repeat the read transaction.	✓	
25	Target Abort bit set after write to slow I/O slave.	✓	
26	IUT does not repeat the write transaction.	✓	
27	IUT's Target Abort bit set after read from slow I/O slave.	✓	
28	IUT does not repeat the read transaction.	✓	
29	Target Abort bit set after write to subtractive I/O slave.	✓	
30	IUT does not repeat the write transaction.	✓	
31	IUT's Target Abort bit set after read from subtractive I/O slave.	✓	
32	IUT does not repeat the read transaction.	✓	

Configuration Transactions:

Test #	Description	Pass	N/A
If IUT does not implement configuration transactions mark 33 through 48 N/A.			
33	Target Abort bit set after write to fast config slave	✓	
34	IUT does not repeat the write transaction.	✓	
35	IUT's Target Abort bit set after read from fast config slave.	✓	
36	IUT does not repeat the read transaction.	✓	
37	Target Abort bit set after write to medium config slave.	✓	
38	IUT does not repeat the write transaction.	✓	
39	IUT's Target Abort bit set after read from medium config slave.	✓	
40	IUT does not repeat the read transaction.	✓	
41	Target Abort bit set after write to slow config slave.	✓	
42	IUT does not repeat the write transaction.	✓	
43	IUT's Target Abort bit set after read from slow config slave.	✓	
44	IUT does not repeat the read transaction.	✓	
45	Target Abort bit set after write to subtractive config slave.	✓	
46	IUT does not repeat the write transaction.	✓	

Test #	Description	Pass	N/A
47	IUT's Target Abort bit set after read from subtractive config slave.	✓	
48	IUT does not repeat the read transaction.	✓	

Interrupt Acknowledge Transactions:

Test #	Description	Pass	N/A
If IUT does not implement interrupt transactions mark 49 through 56 N/A.			
49	IUT's Target Abort bit set after interrupt acknowledge from fast slave.	✓	
50	IUT does not repeat the interrupt acknowledge transaction.	✓	
51	IUT's Target Abort bit set after interrupt acknowledge from medium slave.	✓	
52	IUT does not repeat the interrupt acknowledge transaction.	✓	
53	IUT's Target Abort bit set after interrupt acknowledge from slow slave.	✓	
54	IUT does not repeat the interrupt acknowledge transaction.	✓	
55	IUT's Target Abort bit set after interrupt acknowledge from subtractive slave.	✓	
56	IUT does not repeat the interrupt acknowledge transaction.	✓	

Test Scenario: 1.3. PCI Bus Target Retry Cycles

Memory Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory transactions mark 1 through 8 N/A.			
1	Data transfer after write to fast memory slave.	✓	
2	Data transfer after read from fast memory slave.	✓	
3	Data transfer after write to medium memory slave.	✓	
4	Data transfer after read from medium memory slave.	✓	
5	Data transfer after write to slow memory slave.	✓	
6	Data transfer after read from slow memory slave.	✓	
7	Data transfer after write to subtractive memory slave.	✓	
8	Data transfer after read from subtractive memory slave.	✓	

I/O Transactions:

Test #	Description	Pass	N/A
If IUT does not implement I/O transactions mark 9 through 16 N/A.			
9	Data transfer after write to fast I/O slave.	✓	
10	Data transfer after read from fast I/O slave.	✓	
11	Data transfer after write to medium I/O slave.	✓	
12	Data transfer after read from medium I/O slave.	✓	
13	Data transfer after write to slow I/O slave.	✓	
14	Data transfer after read from slow I/O slave.	✓	
15	Data transfer after write to subtractive I/O slave.	✓	
16	Data transfer after read from subtractive I/O slave.	✓	

Configuration Transactions:

Test #	Description	Pass	N/A
If IUT does not implement configuration transactions mark 17 through 24 N/A.			
17	Data transfer after write to fast config slave.	✓	
18	Data transfer after read from fast config slave.	✓	
19	Data transfer after write to medium config slave.	✓	
20	Data transfer after read from medium config slave.	✓	
21	Data transfer after write to slow config slave.	✓	
22	Data transfer after read from slow config slave.	✓	
23	Data transfer after write to subtractive config slave.	✓	
24	Data transfer after read from subtractive config slave.	✓	

Interrupt Acknowledge Transactions:

Test #	Description	Pass	N/A
If IUT does not implement interrupt transactions mark 25 through 28 N/A.			
25	Data transfer after interrupt acknowledge from fast slave.	✓	
26	Data transfer after interrupt acknowledge from medium slave.	✓	
27	Data transfer after interrupt acknowledge from slow slave.	✓	
28	Data transfer after interrupt acknowledge from subtractive slave.	✓	

Test Scenario: 1.4. PCI Bus Single Data Phase Disconnect Cycles

Memory Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory transactions mark 1 through 8 N/A.			
1	Data transfer after write to fast memory slave.	✓	
2	Data transfer after read from fast memory slave.	✓	
3	Data transfer after write to medium memory slave.	✓	
4	Data transfer after read from medium memory slave.	✓	
5	Data transfer after write to slow memory slave.	✓	
6	Data transfer after read from slow memory slave.	✓	
7	Data transfer after write to subtractive memory slave.	✓	
8	Data transfer after read from subtractive memory slave.	✓	

I/O Transactions:

Test	Description	Pass	N/A
If IUT does not implement I/O transactions mark 9 through 16 N/A.			
9	Data transfer after write to fast I/O slave.	✓	
10	Data transfer after read from fast I/O slave.	✓	
11	Data transfer after write to medium I/O slave.	✓	
12	Data transfer after read from medium I/O slave.	✓	
13	Data transfer after write to slow I/O slave.	✓	

Test	Description	Pass	N/A
14	Data transfer after read from slow I/O slave.	✓	
15	Data transfer after write to subtractive I/O slave.	✓	
16	Data transfer after read from subtractive I/O slave.	✓	

Configuration Transactions:

Test #	Description	Pass	N/A
If IUT does not implement configuration transactions mark 17 through 24 N/A.			
17	Data transfer after write to fast config slave.	✓	
18	Data transfer after read from fast config slave.	✓	
19	Data transfer after write to medium config slave.	✓	
20	Data transfer after read from medium config slave.	✓	
21	Data transfer after write to slow config slave.	✓	
22	Data transfer after read from slow config slave.	✓	
23	Data transfer after write to subtractive config slave.	✓	
24	Data transfer after read from subtractive config slave.	✓	

Interrupt Acknowledge Transactions:

Test #	Description	Pass	N/A
If IUT does not implement interrupt transactions mark 25 through 28 N/A.			
25	Data transfer after interrupt acknowledge from fast slave.	✓	
26	Data transfer after interrupt acknowledge from medium slave.	✓	
27	Data transfer after interrupt acknowledge from slow slave.	✓	
28	Data transfer after interrupt acknowledge from subtractive slave.	✓	

Test Scenario: 1.5. PCI Bus Multi-Data Phase Target Abort Cycles

Memory Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory transactions mark 1 through 16 N/A.			
1	Target Abort bit set after write to fast memory slave.	✓	
2	IUT does not repeat the write transaction.	✓	
3	IUT's Target Abort bit set after read from fast memory slave.	✓	
4	IUT does not repeat the read transaction.	✓	
5	Target Abort bit set after write to medium memory slave.	✓	
6	IUT does not repeat the write transaction.	✓	
7	IUT's Target Abort bit set after read from medium memory slave.	✓	
8	IUT does not repeat the read transaction.	✓	
9	Target Abort bit set after write to slow memory slave.	✓	
10	IUT does not repeat the write transaction.	✓	
11	IUT's Target Abort bit set after read from slow memory slave.	✓	
12	IUT does not repeat the read transaction.	✓	

Test #	Description	Pass	N/A
13	Target Abort bit set after write to subtractive memory slave.	✓	
14	IUT does not repeat the write transaction.	✓	
15	IUT's Target Abort bit set after read from subtractive memory slave.	✓	
16	IUT does not repeat the read transaction.	✓	

Dual Address Cycle Transactions:

Test #	Description	Pass	N/A
If IUT does not implement dual address transactions mark 17 through 32 N/A.			
17	Target Abort bit set after write to fast memory slave.		✓
18	IUT does not repeat the write transaction.		✓
19	IUT's Target Abort bit set after read from fast memory slave.		✓
20	IUT does not repeat the read transaction.		✓
21	Target Abort bit set after write to medium memory slave.		✓
22	IUT does not repeat the write transaction.		✓
23	IUT's Target Abort bit set after read from medium memory slave.		✓
24	IUT does not repeat the read transaction.		✓
25	Target Abort bit set after write to slow memory slave.		✓
26	IUT does not repeat the write transaction.		✓
27	IUT's Target Abort bit set after read from slow memory slave.		✓
28	IUT does not repeat the read transaction.		✓
29	Target Abort bit set after write to subtractive memory slave.		✓
30	IUT does not repeat the write transaction.		✓
31	IUT's Target Abort bit set after read from subtractive memory slave.		✓
32	IUT does not repeat the read transaction.		✓

Configuration Transactions:

Test #	Description	Pass	N/A
If IUT does not implement configuration transactions mark 33 through 48 N/A.			
33	Target Abort bit set after write to fast config. slave.	✓	
34	IUT does not repeat the write transaction.	✓	
35	IUT's Target Abort bit set after read from fast config. slave.	✓	
36	IUT does not repeat the read transaction.	✓	
37	Target Abort bit set after write to medium config. slave.	✓	
38	IUT does not repeat the write transaction.	✓	
39	IUT's Target Abort bit set after read from medium config. slave.	✓	
40	IUT does not repeat the read transaction.	✓	
41	Target Abort bit set after write to slow config. slave.	✓	
42	IUT does not repeat the write transaction.	✓	
43	IUT's Target Abort bit set after read from slow config. slave.	✓	
44	IUT does not repeat the read transaction.	✓	

Test #	Description	Pass	N/A
45	Target Abort bit set after write to subtractive config. slave.	✓	
46	IUT does not repeat the write transaction.	✓	
47	IUT's Target Abort bit set after read from subtractive config. slave.	✓	
48	IUT does not repeat the read transaction.	✓	

Memory Read Multiple Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory read multiple transactions mark 49 through 56 N/A.			
49	IUT's Target Abort bit set after read from fast memory slave.	✓	
50	IUT does not repeat the read transaction.	✓	
51	IUT's Target Abort bit set after read from medium memory slave.	✓	
52	IUT does not repeat the read transaction.	✓	
53	IUT's Target Abort bit set after read from slow memory slave.	✓	
54	IUT does not repeat the read transaction.	✓	
55	IUT's Target Abort bit set after read from subtractive memory slave.	✓	
56	IUT does not repeat the read transaction.	✓	

Memory Read Line Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory read line transactions mark 57 through 64 N/A.			
57	IUT's Target Abort bit set after read from fast memory slave.	✓	
58	IUT does not repeat the read transaction.	✓	
59	IUT's Target Abort bit set after read from medium memory slave.	✓	
60	IUT does not repeat the read transaction.	✓	
61	IUT's Target Abort bit set after read from slow memory slave.	✓	
62	IUT does not repeat the read transaction.	✓	
63	IUT's Target Abort bit set after read from subtractive memory slave.	✓	
64	IUT does not repeat the read transaction.	✓	

Memory Write and Invalidate Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory write and invalidate transactions mark 65 through 72 N/A.			
65	Target Abort bit set after write to fast memory slave.		✓
66	IUT does not repeat the write transaction.		✓
67	Target Abort bit set after write to medium memory slave.		✓
68	IUT does not repeat the write transaction.		✓
69	Target Abort bit set after write to slow memory slave.		✓
70	IUT does not repeat the write transaction.		✓
71	IUT's Target Abort bit set after read from slow memory slave.		✓
72	IUT does not repeat the write transaction.		✓

Test Scenario: 1.6. PCI Bus Multi-Data Phase Retry Cycles

Memory Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory transactions mark 1 through 8 N/A.			
1	Data transfer after write to fast memory slave.	✓	
2	Data transfer after read from fast memory slave.	✓	
3	Data transfer after write to medium memory slave.	✓	
4	Data transfer after read from medium memory slave.	✓	
5	Data transfer after write to slow memory slave.	✓	
6	Data transfer after read from slow memory slave.	✓	
7	Data transfer after write to subtractive memory slave.	✓	
8	Data transfer after read from subtractive memory slave.	✓	

I/O Transactions:

Test #	Description	Pass	N/A
If IUT does not implement I/O transactions mark 9 through 16 N/A.			
9	Data transfer after write to fast I/O slave.	✓	
10	Data transfer after read from fast I/O slave.	✓	
11	Data transfer after write to medium I/O slave.	✓	
12	Data transfer after read from medium I/O slave.	✓	
13	Data transfer after write to slow I/O slave.	✓	
14	Data transfer after read from slow I/O slave.	✓	
15	Data transfer after write to subtractive I/O slave.	✓	
16	Data transfer after read from subtractive I/O slave.	✓	

Configuration Transactions:

Test #	Description	Pass	N/A
If IUT does not implement configuration transactions mark 17 through 24 N/A.			
17	Data transfer after write to fast config. slave.	✓	
18	Data transfer after read from fast config. slave.	✓	
19	Data transfer after write to medium config. slave.	✓	
20	Data transfer after read from medium config. slave.	✓	
21	Data transfer after write to slow config. slave.	✓	
22	Data transfer after read from slow config. slave.	✓	
23	Data transfer after write to subtractive config. slave.	✓	
24	Data transfer after read from subtractive config. slave.	✓	

Memory Read Multiple Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory read multiple transactions mark 25 through 28 N/A.			
25	Data transfer after memory read multiple from fast slave.	✓	

Test #	Description	Pass	N/A
26	Data transfer after memory read multiple from medium slave.	✓	
27	Data transfer after memory read multiple from slow slave.	✓	
28	Data transfer after memory read multiple from subtractive slave.	✓	

Memory Read Line Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory read line transactions mark 29 through 32 N/A.			
29	Data transfer after memory read line from fast slave.	✓	
30	Data transfer after memory read line from medium slave.	✓	
31	Data transfer after memory read line from slow slave.	✓	
32	Data transfer after memory read line from subtractive slave.	✓	

Memory Write and Invalidate Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory write and invalidate transactions mark 33 through 36 N/A.			
33	Data transfer after memory write and invalidate to fast slave.		✓
34	Data transfer after memory write and invalidate to medium slave.		✓
35	Data transfer after memory write and invalidate to slow slave.		✓
36	Data transfer after memory write and invalidate to subtractive slave.		✓

Test Scenario: 1.7. PCI Bus Multi-Data Phase Disconnect Cycles

Memory Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory transactions mark 1 through 8 N/A.			
1	Data transfer after write to fast memory slave.	✓	
2	Data transfer after read from fast memory slave.	✓	
3	Data transfer after write to medium memory slave.	✓	
4	Data transfer after read from medium memory slave.	✓	
5	Data transfer after write to slow memory slave.	✓	
6	Data transfer after read from slow memory slave.	✓	
7	Data transfer after write to subtractive memory slave.	✓	
8	Data transfer after read from subtractive memory slave.	✓	

I/O Transactions:

Test #	Description	Pass	N/A
If IUT does not implement I/O transactions mark 9 through 16 N/A.			
9	Data transfer after write to fast I/O slave.	✓	
10	Data transfer after read from fast I/O slave.	✓	
11	Data transfer after write to medium I/O slave.	✓	
12	Data transfer after read from medium I/O slave.	✓	
13	Data transfer after write to slow I/O slave.	✓	

Test #	Description	Pass	N/A
14	Data transfer after read from slow I/O slave.	✓	
15	Data transfer after write to subtractive I/O slave.	✓	
16	Data transfer after read from subtractive I/O slave.	✓	

Configuration Transactions:

Test #	Description	Pass	N/A
If IUT does not implement configuration transactions mark 17 through 24 N/A.			
17	Data transfer after write to fast config. slave.	✓	
18	Data transfer after read from fast config. slave.	✓	
19	Data transfer after write to medium config. slave.	✓	
20	Data transfer after read from medium config. slave.	✓	
21	Data transfer after write to slow config. slave.	✓	
22	Data transfer after read from slow config. slave.	✓	
23	Data transfer after write to subtractive config. slave.	✓	
24	Data transfer after read from subtractive config. slave.	✓	

Memory Read Multiple Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory read multiple transactions mark 25 through 28 N/A.			
25	Data transfer after memory read multiple from fast slave.	✓	
26	Data transfer after memory read multiple from medium slave.	✓	
27	Data transfer after memory read multiple from slow slave.	✓	
28	Data transfer after memory read multiple from subtractive slave.	✓	

Memory Read Line Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory read line transactions mark 29 through 32 N/A.			
29	Data transfer after memory read line from fast slave.	✓	
30	Data transfer after memory read line from medium slave.	✓	
31	Data transfer after memory read line from slow slave.	✓	
32	Data transfer after memory read line from subtractive slave.	✓	

Memory Write and Invalidate Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory write and invalidate transactions mark 33 through 36 N/A.			
33	Data transfer after memory write and invalidate to fast slave.		✓
34	Data transfer after memory write and invalidate to medium slave.		✓
35	Data transfer after memory write and invalidate to slow slave.		✓
36	Data transfer after memory write and invalidate to subtractive slave.		✓

Test Scenario: 1.8. Multi-Data Phase and TRDY# Cycles

Memory Transactions:

Test #	Description	Pass	N/A
If IUT does not implement multi-data phase memory transactions mark 1 through 12 N/A.			
1	Verify that data is written to primary target when TRDY# is released after 2nd rising clock edge and asserted on 3rd rising clock edge after FRAME#	✓	
2	Verify that data is read from primary target when TRDY# is released after 2nd rising clock edge and asserted on 3rd rising clock edge after FRAME#	✓	
3	Verify that data is written to primary target when TRDY# is released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#	✓	
4	Verify that data is read from primary target when TRDY# is released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#	✓	
5	Verify that data is written to primary target when TRDY# is released after 3rd rising clock edge and asserted on 5th rising clock edge after FRAME#	✓	
6	Verify that data is read from primary target when TRDY# is released after 3rd rising clock edge and asserted on 5th rising clock edge after FRAME#	✓	
7	Verify that data is written to primary target when TRDY# is released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#	✓	
8	Verify that data is read from primary target when TRDY# is released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#	✓	
9	Verify that data is written to primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#	✓	
10	Verify that data is read from primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#	✓	
11	Verify that data is written to primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#	✓	
12	Verify that data is read from primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#	✓	

Dual Address Cycle Transactions:

Test #	Description	Pass	N/A
If IUT does not implement dual address cycle transactions mark 13 through 24 N/A.			
13	Verify that data is written to primary target when TRDY# released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#		✓
14	Verify that data is read from primary target when TRDY# released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#		✓
15	Verify that data is written to primary target when TRDY# released after 4th rising clock edge and asserted on 5th rising clock edge after FRAME#		✓
16	Verify that data is read from primary target when TRDY# released after 4th rising clock edge and asserted on 5th rising clock edge after FRAME#		✓
17	Verify that data is written to primary target when TRDY# released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#		✓
18	Verify that data is read from primary target when TRDY# released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#		✓

Test #	Description	Pass	N/A
19	Verify that data is written to primary target when TRDY# released after 5th rising clock edge and asserted on 7th rising clock edge after FRAME#		✓
20	Verify that data is read from primary target when TRDY# released after 5th rising clock edge and asserted on 7th rising clock edge after FRAME#		✓
21	Verify that data is written to primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#		✓
22	Verify that data is read from primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#		✓
23	Verify that data is written to primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#		✓
24	Verify that data is read from primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#		✓

Memory Read Multiple Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory read multiple transactions mark 25 through 30 N/A.			
25	Verify that data is read from primary target when TRDY# released after 2nd rising clock edge and asserted on 3rd rising clock edge after FRAME#	✓	
26	Verify that data is read from primary target when TRDY# released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#	✓	
27	Verify that data is read from primary target when TRDY# released after 3rd rising clock edge and asserted on 5th rising clock edge after FRAME#	✓	
28	Verify that data is read from primary target when TRDY# released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#	✓	
29	Verify that data is read from primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#	✓	
30	Verify that data is read from primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#	✓	

Memory Read Line Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory read line transactions mark 31 through 36 N/A.			
31	Verify that data is read from primary target when TRDY# released after 2nd rising clock edge and asserted on 3rd rising clock edge after FRAME#	✓	
32	Verify that data is read from primary target when TRDY# released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#	✓	
33	Verify that data is read from primary target when TRDY# released after 3rd rising clock edge and asserted on 5th rising clock edge after FRAME#	✓	
34	Verify that data is read from primary target when TRDY# released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#	✓	

Test #	Description	Pass	N/A
35	Verify that data is read from primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#	✓	
36	Verify that data is read from primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#	✓	

Memory Write and Invalidate Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory write and invalidate transactions mark 37 through 42 N/A.			
37	Verify that data is written to primary target when TRDY# released after 2nd rising clock edge and asserted on 3rd rising clock edge after FRAME#		✓
38	Verify that data is written to primary target when TRDY# released after 3rd rising clock edge and asserted on 4th rising clock edge after FRAME#		✓
39	Verify that data is written to primary target when TRDY# released after 3rd rising clock edge and asserted on 5th rising clock edge after FRAME#		✓
40	Verify that data is written to primary target when TRDY# released after 4th rising clock edge and asserted on 6th rising clock edge after FRAME#		✓
41	Verify that data is written to primary target when TRDY# alternately released for one clock cycle and asserted for one clock cycle after FRAME#		✓
42	Verify that data is written to primary target when TRDY# alternately released for two clock cycles and asserted for two clock cycles after FRAME#		✓

Test Scenario: 1.9. Bus Data Parity Error Single Cycles

Memory Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory transactions mark 1 through 3 N/A.			
1	Verify the IUT sets Data Parity Error Detected bit when Primary Target asserts PERR# on IUT memory write	✓	
2	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT memory read	✓	
3	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT memory read	✓	

I/O Transactions:

Test #	Description	Pass	N/A
If IUT does not implement I/O transactions mark 4 through 6 N/A.			
4	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT I/O write	✓	
5	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT I/O read	✓	
6	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT I/O read	✓	

Configuration Transactions:

Test #	Description	Pass	N/A
If IUT does not implement configuration transactions mark 7 through 9 N/A.			
7	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT config write	✓	
8	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT config read	✓	
9	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT config read	✓	

Test Scenario: 1.10. Bus Data Parity Error Multi-Data Phase Cycles

Memory Transactions:

Test #	Description	Pass	N/A
If IUT does not implement memory transactions mark 1 through 3 N/A.			
1	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT multi data phase memory write	✓	
2	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT multi data phase memory read	✓	
3	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT Memory multi data phase read	✓	

Dual Address Cycle Transactions:

Test #	Description	Pass	N/A
If IUT does not implement dual address transactions mark 4 through 6 N/A.			
4	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT dual address multi data phase write		✓
5	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT dual address multi data phase read		✓
6	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT dual address multi data phase read		✓

Configuration Transactions:

Test #	Description	Pass	N/A
If IUT does not implement configuration transactions mark 7 through 9 N/A.			
7	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT config multi-data phase write	✓	
8	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT config multi-data phase read	✓	
9	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT config multi-data phase read	✓	

Memory Read Multiple:

Test #	Description	Pass	N/A
If IUT does not implement memory read multiple transactions mark 10 through 11 N/A.			
10	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT memory read multiple data phase.	✓	
11	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT memory read multiple data phase.	✓	

Memory Read Line:

Test #	Description	Pass	N/A
If IUT does not implement memory read line transactions mark 12 through 13 N/A.			
12	Verify that PERR# is active two clocks after the first data phase (which had odd parity) on IUT memory read line data phase.	✓	
13	Verify the IUT sets Parity Error Detected bit when odd parity is detected on IUT memory read line data phase.	✓	

Memory Write and Invalidate:

Test #	Description	Pass	N/A
If IUT does not implement memory write and invalidate transactions mark 14 N/A.			
14	Verify the IUT sets Parity Error Detected bit when Primary Target asserts PERR# on IUT memory write and invalidate data phase.		✓

Test Scenario: 1.11. Bus Master Timeout

Test #	Description	Pass	N/A
1	Memory write transaction terminates before 4 data phases completed	✓	
2	Memory read transaction terminates before 4 data phases completed	✓	
3	Config write transaction terminates before 4 data phases completed	✓	
4	Config read transaction terminates before 4 data phases completed	✓	
5	Memory read multiple transaction terminates before 4 data phases	✓	
6	Memory read line transaction terminates before 4 data phases	✓	
7	Dual Address write transaction terminates before 4 data phases complete		✓
8	Dual Address read transaction terminates before 4 data phases complete		✓
If IUT does not support cache coherent transactions mark 9 N/A.			
9	Memory write invalidate terminates on line boundary		✓

Test Scenario: 1.12. Target Lock

Test #	Description	Pass	N/A
If IUT does not support locked transactions mark 1 through 5 N/A.			
1	IUT does not perform bus transaction (read lock) on locked resource		✓
2	IUT does establish lock after lock is released		✓
3	IUT does release lock after write to primary target		✓
4	IUT does not establish lock when it detects retry		✓
5	IUT does not establish lock when it detects target abort		✓

Test Scenario: 1.13. PCI Bus Master Parking

Test #	Description	Pass	N/A
Verify that the IUT is able to drive PCI bus to stable conditions if it is idle and GNT# is asserted.			
1	IUT drives AD[31::00] to stable values within eight PCI Clocks of GNT#.	✓	
2	IUT drives C/BE[3::0]# to stable values within eight PCI Clocks of GNT#.	✓	
3	IUT drives PAR one clock cycle after IUT drives AD[31::0]	✓	
Verify that the IUT will tri-state the bus when GNT# is not asserted.			
4	IUT Tri-states AD[31::00] and C/BE[3::0] and PAR when GNT# is released.	✓	

Test Scenario: 1.14. PCI Bus Master Arbitration

Test #	Description	Pass	N/A
Verify that the IUT is able to complete bus transaction when GNT# is deasserted coincident with FRAME# asserted.			
1	IUT completes transaction when de-asserting GNT# is coincident with asserting FRAME#.	✓	

Test Scenario 1.x Explanations.

Explanations
Scenario 1.1: The LogiCORE initiator detects and reports a master abort. However, the initiator deasserts FRAME# and IRDY# one cycle later than specified in the *PCI Local Bus Specification*. Otherwise, the initiator treats master abort as required. A one cycle latency should not adversely affect most designs.
Scenario 1.2: The LogiCORE initiator will not retry the transaction unless directed to do so by the user application. It is the responsibility of the user application to monitor for target aborts and act appropriately.
Scenario 1.3: The LogiCORE initiator will retry the transaction if directed to do so by the user application. It is the responsibility of the user application to monitor for retries and act appropriately.
Scenario 1.5: The LogiCORE initiator will not retry the transaction unless directed to do so by the user application. It is the responsibility of the user application to monitor for target aborts and act appropriately.
Scenario 1.6: The LogiCORE initiator will retry the transaction if directed to do so by the user application. It is the responsibility of the user application to monitor for retries and act appropriately.
Scenario 1.7: The core will always issue a disconnect with data on the first data phase of a configuration transaction. For extended configuration space transactions, the user application must perform a disconnect with data on the first data phase. Multi-data phase configuration transactions are not supported.
Scenario 1.14: The LogiCORE initiator will not begin a transaction if GNT# is asserted for a single cycle and then deasserted. In this case, the initiator will request the bus again. This behavior is transparent to the user application.

This section should be used to clarify any answers on checklist items above. Please key explanation to item number.

Component Protocol Checklist for a Target Device

Definition: IUT is an acronym for “Implementation Under Test”.

Test Scenario: 2.1. Target Reception of an Interrupt Cycle.

Test #	Description	Pass	N/A
If IUT does not respond to Interrupt Acknowledge bus transactions mark 1 through 2 N/A			
1	IUT generates interrupts when programmed	✓	
2	IUT clears interrupts when serviced (may include driver specific actions)	✓	

Test Scenario: 2.2. Target Reception of a Special Cycle.

Test #	Description	Pass	N/A
If IUT does not implement Special Cycles mark 1 through 2 N/A			
1	No DEVSEL# assertion by IUT after Special Cycle	✓	
2	IUT receives encoded special cycle		✓

Test Scenario: 2.3. Target Detection of Address and Data Parity Error for Special Cycle.

Test #	Description	Pass	N/A
1	IUT reports address parity error via SERR#	✓	
2	IUT reports data parity error via SERR#		✓
3	IUT keeps SERR# active for at least one clock	✓	

Test Scenario: 2.4. Target Reception of I/O Cycles with Legal and Illegal Byte Enables.

Test #	Description	Pass	N/A
If IUT does not support I/O cycles mark 1 through 4 N/A or if IUT claims all 32 bits during an I/O cycle mark 1 and 2 N/A			
1	IUT asserts TRDY# following 2nd rising edge from FRAME# on all legal BE’s		✓
2	IUT terminates with target abort for each illegal BE		✓
If IUT supports target disconnect check the following			
3	IUT asserts STOP#		✓
4	IUT deasserts STOP# after FRAME# deassertion		✓

Test Scenario: 2.5. Target Ignores Reserved Commands.

Test #	Description	Pass	N/A
1	IUT does not respond to reserved commands	✓	
2	Initiator detects master abort for each transfer	✓	
For 32-bit targets			
3	IUT does not respond to 64bit cycle (dual address)	✓	

Test Scenario: 2.6. Target Receives Configuration Cycles.

Test #	Description	Pass	N/A
1	IUT responds to all type 0 configuration read/write cycles appropriately	✓	
2	IUT does not respond to type 0 configuration cycles with IDSEL inactive	✓	
If IUT does not support type 1 configuration cycles mark 3 through 5 N/A			
Test #	**Description**	**Pass**	**N/A**
3	IUT responds to all type 1 configuration read/write cycles appropriately		✓
4	IUT responds to all type 0 configuration read/write cycles appropriately		✓
5	IUT does not respond (master abort) on illegal configuration cycle types		✓

Test Scenario: 2.7. Target Receives I/O Cycles with Address and Data Parity Errors.

Test #	Description	Pass	N/A
If IUT does not support I/O cycles mark all N/A			
1	IUT reports address parity error via SERR# during I/O read/write cycles	✓	
2	IUT reports data parity error via PERR# during I/O write cycles	✓	

Test Scenario: 2.8. Target Configuration Cycles with Address and Data Parity Errors.

Test #	Description	Pass	N/A
1	IUT reports address parity error via SERR# during configuration read/ write cycles	✓	
2	IUT reports data parity error via PERR# during configuration write cycles	✓	

Test Scenario: 2.9. Target Receives Memory Cycles.

Test #	Description	Pass	N/A
If IUT does not interface to a memory subsystem mark all N/A			
1	IUT completes single memory read and write cycles appropriately	✓	
If IUT does not interface to main system memory or memory is not cacheable mark 2 to 4 N/A			
2	IUT completes memory read line cycles appropriately	✓	
3	IUT completes memory read multiple cycles appropriately	✓	
4	IUT completes memory write and invalidate cycles appropriately	✓	
5	IUT disconnects after one cycle on reserved memory operations	✓	
6	IUT disconnects on burst transactions that cross address boundaries		✓

Test Scenario: 2.10. Target Gets Memory Cycles with Address and Data Parity Errors.

Test #	Description	Pass	N/A
If IUT does not interface to a memory subsystem mark 1 to 2 N/A			
1	IUT reports address parity error via SERR# during all memory read and write cycles	✓	
2	IUT reports data parity error via PERR# during all memory write cycles	✓	

Test Scenario: 2.11. Target Gets Fast Back to Back Cycles.

Test #	Description	Pass	N/A
1	IUT responds to back to back memory writes appropriately		✓
2	IUT responds to memory write followed by memory read appropriately		✓
If IUT does not implement the “Fast Back-to-Back Bit” then mark 3 and 4 N/A			
3	IUT responds to back to back memory writes with 2nd write selecting IUT		✓
4	IUT responds to memory write followed by memory read with read selecting IUT		✓

Test Scenario: 2.12. Target Performs Exclusive Access Cycles.

Test #	Description	Pass	N/A
If the IUT does not implement LOCK# mark all N/A			
1	IUT responds to exclusive access by initiator and accepts LOCK#		✓
2	IUT responds with retry when second initiator attempts an access		✓
3	IUT responds to access releasing LOCK# by initiator		✓
4	IUT responds to access by second initiator		✓

Test Scenario: 2.13. Target Gets Cycles with IRDY# Used for Data Stepping.

Test #	Description	Pass	N/A
1	IUT responds appropriately with a wait state inserted on phase 1 of 3 data phases	✓	
2	IUT responds appropriately with a wait state inserted on phase 2 of 3 data phases	✓	
3	IUT responds appropriately with a wait state inserted on phase 3 of 3 data phases	✓	
4	IUT responds appropriately with a wait state inserted on all of 3 data phases	✓	

Test Scenario 2.x Explanations.

Explanations
Scenario 2.4: The LogiCORE target does not automatically generate target abort or disconnect during illegal transfers. However, this behavior can be implemented in the user application.
Scenario 2.6: The LogiCORE target does not support burst transfers in or out of its configuration space.
Scenario 2.9: The LogiCORE target does not automatically generate target abort when a burst transaction crosses an address boundary. However, this behavior can be implemented in the user application.

This section should be used to clarify any answers on checklist items above. Please key explanation to item number.

Pinout and Configuration

Layout Considerations

The PQ pinouts Xilinx supplies follow the PCI-SIG suggested pinout and aligns the PCI data path (ADIO[31:0]) along the horizontal long lines in the FPGA. The horizontal Long-lines support internal 3-state busses. Various registers, such as the Base Address Registers, are aligned vertically, in columns.

Since the BG432 package used by the XC4062XLA is cavity down, the LogiCORE PCI Interface is placed on the other side of the die.

Compatibility Considerations

Here are a few issues to consider in design your PCB.

- If you are migrating from an XC4000XLT to an XC4000XLA device, all pinouts are identical except the Vtt pins become regular I/Os.
- Spartan and SpartanXL pinouts are identical for the same packages.
- Spartan/SpartanXL PQ208 and XC4000XLA PQ208 PCI pinouts do not match, due to a different bondout on the Spartan/SpartanXL PQ208.
- The SpartanXL PQ240 and the XC4000XLA PQ/HQ240 PCI pinouts match.

Pinout Tables

The following pinout tables list the PCI pin assignment for these supported packages:

- Table 1 XC4013XLAPQ208
- Table 2 XC4013XLAPQ240
- Table 3 XC4028XLAPQ240
- Table 4 XC4062XLAPQ240
- Table 5 XC4062XLABG432
- Table 6 XCS20TQ144
- Table 7 XCS30PQ208
- Table 8 XCS30PQ240
- Table 9 XCS40PQ208
- Table 10 XCS40PQ240
- Table 10 XCV300BG432 (64 bit)

For each pin, both the PCI function and the fundamental device pin function are listed. Those shown in bold italics are dedicated pins for configuring the FPGA device using one of the serial configuration modes. Pins without a PCI function listed are available as additional user I/O.

Note: If there are conflicts between these tables and the constraints file, the constraints file has precedence.

Due to the complexity of the Virtex I/O, the V300 pinout table will only list the PCI specific pins. Refer to the Virtex data sheet for details on the remaining pins.

Configuration Mode

The LogiCORE PCI Interface is designed to use Serial Master Mode or Slave Mode for configuring the device. An external serial configuration device, such as the Xchecker cable or an embedded processor is required for Serial Slave Mode.

Use of the XC4000XLA or SpartanXL fast configuration mode is recommended to minimize the FPGA power-up configuration time. The fast mode is set as part of the MakeBits options in the XDM profile read in during the design compilation phase.

Please refer to the Xilinx Programmable Logic Data Book for additional information.

The Xchecker is useful during debugging. However, the PCI system needs to be held reset while the FPGA's bit-stream is loaded. If using an embedded processor, the user needs to insure that configuration will not be delayed by interrupts to the processor, and that it is capable of configuring it prior to the assertion of IDSEL. Currently the v2.1 PCI Specification does not specify a time requirement for the initial asserting of IDSEL after power-on. Currently, there is a proposal for the v2.2 PCI Specification to require 2^25 clocks before configuring the PCI bus after power-on. Until this is required in the PCI Specification, Xilinx recommends use of the Master Serial fast mode to assure the FPGA is configured at the time of IDSEL assertion.

Pinout for the XC4013XLA PQ208

Table 1: Pinout for the XC4013XLA PQ208

Pin Function	PCI Function	PQ208
N.C.	N.C.	P1
GND	GND	P2
N.C.	N.C.	P3
I/O, GCK1 (A16)	PCLK	P4
I/O (A17)	AD23	P5
I/O	AD22	P6
I/O	AD21	P7
I/O, TDI	TDI	P8
I/O, TCK	TCK	P9
I/O	AD20	P10
I/O	AD19	P11
I/O	AD18	P12
I/O	AD17	P13
GND	GND	P14
I/O	AD16	P15
I/O	CBE2	P16
I/O, TMS	TMS	P17
I/O		P18
I/O	GNT-	P19
I/O	FRAME-	P20
I/O	IRDY-	P21
I/O	TRDY-	P22
I/O	DEVSEL-	P23
I/O	STOP-	P24
GND	GND	P25
VCC	VCC	P26
I/O	PERR-	P27
I/O	SERR-	P28
I/O	PAR	P29
I/O	REQ-	P30
I/O		P31
I/O		P32
I/O		P33
I/O		P34
I/O	CBE1	P35
I/O	AD15	P36
GND	GND	P37
I/O	AD14	P38
I/O	AD13	P39
I/O	AD12	P40
I/O	AD11	P41
I/O	AD10	P42
I/O	AD9	P43
I/O	AD8	P44
I/O		P45

Table 1: Pinout for the XC4013XLA PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O		P46
I/O, GCK2		P47
O (M1)	**M1**	**P48**
GND	GND	P49
I (M0)	**M0**	**P50**
N.C.	N.C.	P51
N.C.	N.C.	P52
N.C.	N.C.	P53
N.C.	N.C.	P54
VCC	VCC	P55
I (M2)	**M2**	**P56**
I/O, GCK3		P57
I/O (HDC)	**HDC**	**P58**
I/O	CBE0	P59
I/O	AD7	P60
I/O	AD6	P61
I/O (LDC-)	**LDC-**	**P62**
I/O	AD5	P63
I/O	AD4	P64
I/O	AD3	P65
I/O	AD2	P66
GND	GND	P67
I/O	AD1	P68
I/O		P69
I/O	AD0	P70
I/O		P71
I/O		P72
I/O		P73
I/O		P74
I/O		P75
I/O		P76
I/O (INIT-)	**INIT-**	**P77**
VCC	VCC	P78
GND	GND	P79
I/O		P80
I/O		P81
I/O		P82
I/O		P83
I/O		P84
I/O		P85
I/O		P86
I/O		P87
I/O		P88
I/O		P89
GND	GND	P90
I/O		P91

Table 1: Pinout for the XC4013XLA PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O		P92
I/O		P93
I/O		P94
I/O		P95
I/O		P96
I/O		P97
I/O		P98
I/O		P99
I/O, GCK4		P100
GND	GND	P101
N.C.	N.C.	P102
DONE	**DONE**	**P103**
N.C.	N.C.	P104
N.C.	N.C.	P105
VCC	VCC	P106
N.C.	N.C.	P107
PROGRAM-	**PROGRAM-**	**P108**
I/O (D7)	RST-	P109
I/O, GCK5		P110
I/O		P111
I/O		P112
I/O (D6)		P113
I/O		P114
I/O		P115
I/O		P116
I/O		P117
I/O		P118
GND	GND	P119
I/O		P120
I/O		P121
I/O (D5)		P122
I/O (CS0-)		P123
I/O		P124
I/O		P125
I/O		P126
I/O		P127
I/O (D4)		P128
I/O		P129
VCC	VCC	P130
GND	GND	P131
I/O (D3)		P132
I/O (RS-)		P133
I/O		P134
I/O		P135
I/O		P136
I/O		P137

Table 1: Pinout for the XC4013XLA PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O (D2)		P138
I/O		P139
I/O		P140
I/O		P141
GND	GND	P142
I/O		P143
I/O		P144
I/O		P145
I/O		P146
I/O (D1)		P147
I/O (RCLK, RDY/ BUSY)		P148
I/O		P149
I/O		P150
I/O (D0, DIN)	**DIN**	**P151**
I/O, GCK6 (DOUT)	**DOUT**	**P152**
CCLK	CCLK	P153
VCC	VCC	P154
N.C.	N.C.	P155
N.C.	N.C.	P156
N.C.	N.C.	P157
N.C.	N.C.	P158
O, TDO	TDO	P159
GND	GND	P160
I/O (A0, WS-)		P161
I/O, GCK7 (A1)		P162
I/O		P163
I/O		P164
I/O (CS1, A2)		P165
I/O (A3)		P166
I/O		P167
I/O		P168
I/O		P169
I/O		P170
GND	GND	P171
I/O		P172
I/O		P173
I/O (A4)		P174
I/O (A5)		P175
I/O		P176
I/O		P177
I/O(A21)		P178
I/O(A20)		P179
I/O (A6)		P180
I/O (A7)		P181
GND	GND	P182

Table 1: Pinout for the XC4013XLA PQ208 (Continued)

Pin Function	PCI Function	PQ208
VCC	VCC	P183
I/O (A8)		P184
I/O (A9)		P185
I/O (A19)		P186
I/O (A18)		P187
I/O		P188
I/O		P189
I/O (A10)		P190
I/O (A11)	AD31	P191
I/O		P192
I/O	AD30	P193
GND	GND	P194
I/O	AD29	P195
I/O	AD28	P196
I/O	AD27	P197
I/O	AD26	P198
I/O (A12)	AD25	P199
I/O (A13)	AD24	P200
I/O	CBE3	P201
I/O	IDSEL	P202
I/O (A14)		P203
I/O, GCK8 (A15)		P204
VCC	VCC	P205
N.C.	N.C.	P206
N.C.	N.C.	P207
N.C.	N.C.	P208

Pinout for the XC4013XLA PQ240

Table 2: Pinout for the XC4013XLA PQ240

Pin Function	PCI Function	PQ240
GND	GND	P1
I/O, GCK1 (A16)	PCLK	P2
I/O (A17)	AD23	P3
I/O		P4
I/O	AD22	P5
I/O, TDI	TDI	P6
I/O, TCK	TCK	P7
I/O	AD21	P8
I/O	AD20	P9
I/O	AD19	P10
I/O	AD18	P11
I/O		P12
I/O		P13
GND	GND	P14
I/O	AD17	P15
I/O		P16
I/O, TMS	TMS	P17
I/O		P18
VCC		P19
I/O	AD16	P20
I/O	CBE2	P21
GND‡	GND	P22
I/O	GNT-	P23
I/O	FRAME-	P24
I/O	IRDY-	P25
I/O	TRDY-	P26
I/O	DEVSEL-	P27
I/O	STOP-	P28
GND	GND	P29
VCC	VCC	P30
I/O	PERR-	P31
I/O	SERR-	P32
I/O	PAR	P33
I/O	REQ-	P34
I/O		P35
I/O		P36
GND‡	GND	P37
I/O		P38
I/O		P39
VCC	VCC	P40
I/O		P41
I/O		P42
I/O	CBE1	P43
I/O	AD15	P44
GND	GND	P45

Table 2: Pinout for the XC4013XLA PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P46
I/O		P47
I/O	AD14	P48
I/O	AD13	P49
I/O	AD12	P50
I/O	AD11	P51
I/O	AD10	P52
I/O	AD9	P53
I/O	AD8	P54
I/O		P55
I/O		P56
I/O, GCK2		P57
O (M1)	**M1**	**P58**
GND	GND	P59
I (M0)	**M0**	**P60**
VCC	VCC	P61
I (M2)	**M2**	**P62**
I/O, GCK3		P63
I/O (HDC)	**HDC**	**P64**
I/O	CBE0	P65
I/O	AD7	P66
I/O	AD6	P67
I/O (LDC-)	**LDC-**	**P68**
I/O	AD5	P69
I/O	AD4	P70
I/O	AD3	P71
I/O	AD2	P72
I/O		P73
I/O		P74
GND	GND	P75
I/O	AD1	P76
I/O		P77
I/O	AD0	P78
I/O		P79
VCC	VCC	P80
I/O		P81
I/O		P82
GND‡	GND	P83
I/O		P84
I/O		P85
I/O		P86
I/O		P87
I/O		P88
I/O (INIT-)	**INIT-**	**P89**
VCC	VCC	P90
GND	GND	P91

Table 2: Pinout for the XC4013XLA PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P92
I/O		P93
I/O		P94
I/O		P95
I/O		P96
I/O		P97
GND‡	GND	P98
I/O		P99
I/O		P100
VCC	VCC	P101
I/O		P102
I/O		P103
I/O		P104
I/O		P105
GND	GND	P106
I/O		P107
I/O		P108
I/O		P109
I/O		P110
I/O		P111
I/O		P112
I/O		P113
I/O		P114
I/O		P115
I/O		P116
I/O		P117
I/O, GCK4		P118
GND	GND	P119
DONE	**DONE**	**P120**
VCC	VCC	P121
PROGRAM-	**PROGRAM-**	**P122**
I/O (D7)	RST-	P123
I/O, GCK5		P124
I/O		P125
I/O		P126
I/O		P127
I/O		P128
I/O (D6)		P129
I/O		P130
I/O		P131
I/O		P132
I/O		P133
I/O		P134
GND	GND	P135
I/O		P136
I/O		P137

Table 2: Pinout for the XC4013XLA PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P138
I/O		P139
VCC	VCC	P140
I/O (D5)		P141
I/O (CS0-)		P142
GND‡	GND	P143
I/O		P144
I/O		P145
I/O		P146
I/O		P147
I/O (D4)		P148
I/O		P149
VCC	VCC	P150
GND	GND	P151
I/O (D3)		P152
I/O (RS-)		P153
I/O		P154
I/O		P155
I/O		P156
I/O		P157
GND‡	GND	P158
I/O (D2)		P159
I/O		P160
VCC	VCC	P161
I/O		P162
I/O		P163
I/O		P164
I/O		P165
GND	GND	P166
I/O		P167
I/O		P168
I/O		P169
I/O		P170
I/O		P171
I/O		P172
I/O (D1)		P173
I/O (RCLK, RDY/ BUSY)		P174
I/O		P175
I/O		P176
I/O (D0, DIN)	**DIN**	**P177**
I/O, GCK6 (DOUT)	**DOUT**	**P178**
CCLK	CCLK	P179
VCC	VCC	P180
O, TDO	TDO	P181
GND	GND	P182

Table 2: Pinout for the XC4013XLA PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O (A0, WS-)		P183
I/O, GCK7 (A1)		P184
I/O		P185
I/O		P186
I/O (CS1, A2)		P187
I/O (A3)		P188
I/O		P189
I/O		P190
I/O		P191
I/O		P192
I/O		P193
I/O		P194
N.C.	N.C.	P195
GND	GND	P196
I/O		P197
I/O		P198
I/O		P199
I/O		P200
VCC	VCC	P201
I/O (A4)		P202
I/O (A5)		P203
GND‡	GND	P204
I/O		P205
I/O		P206
I/O (A21)		P207
I/O (A20)		P208
I/O (A6)		P209
I/O (A7)		P210
GND	GND	P211
VCC	VCC	P212
I/O (A8)		P213

Table 2: Pinout for the XC4013XLA PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O (A9)		P214
I/O (A19)		P215
I/O (A18)		P216
I/O		P217
I/O		P218
GND‡	GND	P219
I/O (A10)		P220
I/O (A11)		P221
VCC	VCC	P222
I/O		P223
I/O	AD31	P224
I/O		P225
I/O	AD30	P226
GND	GND	P227
I/O	AD29	P228
I/O	AD28	P229
I/O	AD27	P230
I/O	AD26	P231
I/O (A12)	AD25	P232
I/O (A13)	AD24	P233
I/O		P234
I/O		P235
I/O	CBE3	P236
I/O	IDSEL	P237
I/O (A14)		P238
I/O, GCK8 (A15)		P239
VCC	VCC	P240

‡ Pins marked with this symbol are used for Ground connections on some revisions of the device. These pins may not physically connect to anything on the current device revision. However, they should be externally connected to Ground, if possible.

Pinout for the XC4028XLA HQ240

Table 3: Pinout for the XC4028XLA HQ240

Pin Function	PCI Function	HQ240
GND	GND	P1
I/O, GCK1 (A16)	PCLK	P2
I/O (A17)	AD23	P3
I/O		P4
I/O	AD22	P5
I/O, TDI	TDI	P6
I/O, TCK	TCK	P7
I/O	AD21	P8
I/O	AD20	P9
I/O	AD19	P10
I/O	AD18	P11
I/O		P12
I/O		P13
GND	GND	P14
I/O	AD17	P15
I/O		P16
I/O, TMS	TMS	P17
I/O		P18
VCC		P19
I/O	AD16	P20
I/O	CBE2	P21
GND	GND	P22
I/O	GNT-	P23
I/O	FRAME-	P24
I/O	IRDY-	P25
I/O	TRDY-	P26
I/O	DEVSEL-	P27
I/O	STOP-	P28
GND	GND	P29
VCC	VCC	P30
I/O	PERR-	P31
I/O	SERR-	P32
I/O	PAR	P33
I/O	REQ-	P34
I/O		P35
I/O		P36
GND	GND	P37
I/O		P38
I/O		P39
VCC	VCC	P40
I/O		P41
I/O		P42
I/O	CBE1	P43
I/O	AD15	P44
GND	GND	P45

Table 3: Pinout for the XC4028XLA HQ240 (Continued)

Pin Function	PCI Function	HQ240
I/O		P46
I/O		P47
I/O	AD14	P48
I/O	AD13	P49
I/O	AD12	P50
I/O	AD11	P51
I/O	AD10	P52
I/O	AD9	P53
I/O	AD8	P54
I/O		P55
I/O		P56
I/O, GCK2		P57
O (M1)	**M1**	**P58**
GND	GND	P59
I (M0)	**M0**	**P60**
VCC	VCC	P61
I (M2)	**M2**	**P62**
I/O, GCK3		P63
I/O (HDC)	**HDC**	**P64**
I/O	CBE0	P65
I/O	AD7	P66
I/O	AD6	P67
I/O (LDC-)	**LDC-**	**P68**
I/O	AD5	P69
I/O	AD4	P70
I/O	AD3	P71
I/O	AD2	P72
I/O		P73
I/O		P74
GND	GND	P75
I/O	AD1	P76
I/O		P77
I/O	AD0	P78
I/O		P79
VCC	VCC	P80
I/O		P81
I/O		P82
GND	GND	P83
I/O		P84
I/O		P85
I/O		P86
I/O		P87
I/O		P88
I/O (INIT-)	**INIT-**	**P89**
VCC	VCC	P90
GND	GND	P91

Table 3: Pinout for the XC4028XLA HQ240 (Continued)

Pin Function	PCI Function	HQ240
I/O		P92
I/O		P93
I/O		P94
I/O		P95
I/O		P96
I/O		P97
GND	GND	P98
I/O		P99
I/O		P100
VCC	VCC	P101
I/O		P102
I/O		P103
I/O		P104
I/O		P105
GND	GND	P106
I/O		P107
I/O		P108
I/O		P109
I/O		P110
I/O		P111
I/O		P112
I/O		P113
I/O		P114
I/O		P115
I/O		P116
I/O		P117
I/O, GCK4		P118
GND	GND	P119
DONE	**DONE**	**P120**
VCC	VCC	P121
PROGRAM-	**PROGRAM-**	**P122**
I/O (D7)	RST-	P123
I/O, GCK5		P124
I/O		P125
I/O		P126
I/O		P127
I/O		P128
I/O (D6)		P129
I/O		P130
I/O		P131
I/O		P132
I/O		P133
I/O		P134
GND	GND	P135
I/O		P136
I/O		P137

Table 3: Pinout for the XC4028XLA HQ240 (Continued)

Pin Function	PCI Function	HQ240
I/O		P138
I/O		P139
VCC	VCC	P140
I/O (D5)		P141
I/O (CS0-)		P142
GND	GND	P143
I/O		P144
I/O		P145
I/O		P146
I/O		P147
I/O (D4)		P148
I/O		P149
VCC	VCC	P150
GND	GND	P151
I/O (D3)		P152
I/O (RS-)		P153
I/O		P154
I/O		P155
I/O		P156
I/O		P157
GND	GND	P158
I/O (D2)		P159
I/O		P160
VCC	VCC	P161
I/O		P162
I/O		P163
I/O		P164
I/O		P165
GND	GND	P166
I/O		P167
I/O		P168
I/O		P169
I/O		P170
I/O		P171
I/O		P172
I/O (D1)		P173
I/O (RCLK, RDY/ BUSY)		P174
I/O		P175
I/O		P176
I/O (D0, DIN)	**DIN**	**P177**
I/O, GCK6 (DOUT)	**DOUT**	**P178**
CCLK	CCLK	P179
VCC	VCC	P180
O, TDO	TDO	P181
GND	GND	P182

Table 3: Pinout for the XC4028XLA HQ240 (Continued)

Pin Function	PCI Function	HQ240
I/O (A0, WS-)		P183
I/O, GCK7 (A1)		P184
I/O		P185
I/O		P186
I/O (CS1, A2)		P187
I/O (A3)		P188
I/O		P189
I/O		P190
I/O		P191
I/O		P192
I/O		P193
I/O		P194
I/O		P195
GND	GND	P196
I/O		P197
I/O		P198
I/O		P199
I/O		P200
VCC	VCC	P201
I/O (A4)		P202
I/O (A5)		P203
GND‡	GND	P204
I/O		P205
I/O		P206
I/O (A21)		P207
I/O (A20)		P208
I/O (A6)		P209
I/O (A7)		P210
GND	GND	P211
VCC	VCC	P212
I/O (A8)		P213
I/O (A9)		P214
I/O (A19)		P215
I/O (A18)		P216
I/O		P217
I/O		P218
GND‡	GND	P219
I/O (A10)		P220
I/O (A11)		P221
VCC	VCC	P222
I/O		P223
I/O	AD31	P224
I/O		P225
I/O	AD30	P226
GND	GND	P227
I/O	AD29	P228

Table 3: Pinout for the XC4028XLA HQ240 (Continued)

Pin Function	PCI Function	HQ240
I/O	AD28	P229
I/O	AD27	P230
I/O	AD26	P231
I/O (A12)	AD25	P232
I/O (A13)	AD24	P233
I/O		P234
I/O		P235
I/O	CBE3	P236
I/O	IDSEL	P237
I/O (A14)		P238
I/O, GCK8 (A15)		P239
VCC	VCC	P240

Pinout for the XC4062XLA HQ240

Table 4: Pinout for the XC4062XLA HQ240

Pin Function	PCI Function	HQ240
GND	GND	P1
I/O, GCK1 (A16)	PCLK	P2
I/O (A17)	AD23	P3
I/O		P4
I/O	AD22	P5
I/O, TDI	TDI	P6
I/O, TCK	TCK	P7
I/O	AD21	P8
I/O	AD20	P9
I/O	AD19	P10
I/O	AD18	P11
I/O		P12
I/O		P13
GND	GND	P14
I/O	AD17	P15
I/O		P16
I/O, TMS	TMS	P17
I/O		P18
VCC		P19
I/O	AD16	P20
I/O	CBE2	P21
GND	GND	P22
I/O	GNT-	P23
I/O	FRAME-	P24
I/O	IRDY-	P25
I/O	TRDY-	P26
I/O	DEVSEL-	P27
I/O	STOP-	P28
GND	GND	P29
VCC	VCC	P30
I/O	PERR-	P31
I/O	SERR-	P32
I/O	PAR	P33
I/O	REQ-	P34
I/O		P35
I/O		P36
GND	GND	P37
I/O		P38
I/O		P39
VCC	VCC	P40
I/O		P41
I/O		P42
I/O	CBE1	P43
I/O	AD15	P44
GND	GND	P45

Table 4: Pinout for the XC4062XLA HQ240 (Continued)

Pin Function	PCI Function	HQ240
I/O		P46
I/O		P47
I/O	AD14	P48
I/O	AD13	P49
I/O	AD12	P50
I/O	AD11	P51
I/O	AD10	P52
I/O	AD9	P53
I/O	AD8	P54
I/O		P55
I/O		P56
I/O, GCK2		P57
O (M1)	**M1**	**P58**
GND	GND	P59
I (M0)	**M0**	**P60**
VCC	VCC	P61
I (M2)	**M2**	**P62**
I/O, GCK3		P63
I/O (HDC)	**HDC**	**P64**
I/O	CBE0	P65
I/O	AD7	P66
I/O	AD6	P67
I/O (LDC-)	**LDC-**	**P68**
I/O	AD5	P69
I/O	AD4	P70
I/O	AD3	P71
I/O	AD2	P72
I/O		P73
I/O		P74
GND	GND	P75
I/O	AD1	P76
I/O		P77
I/O	AD0	P78
I/O		P79
VCC	VCC	P80
I/O		P81
I/O		P82
GND	GND	P83
I/O		P84
I/O		P85
I/O		P86
I/O		P87
I/O		P88
I/O (INIT-)	**INIT-**	**P89**
VCC	VCC	P90
GND	GND	P91

Table 4: Pinout for the XC4062XLA HQ240 (Continued)

Pin Function	PCI Function	HQ240
I/O		P92
I/O		P93
I/O		P94
I/O		P95
I/O		P96
I/O		P97
GND	GND	P98
I/O		P99
I/O		P100
VCC	VCC	P101
I/O		P102
I/O		P103
I/O		P104
I/O		P105
GND	GND	P106
I/O		P107
I/O		P108
I/O		P109
I/O		P110
I/O		P111
I/O		P112
I/O		P113
I/O		P114
I/O		P115
I/O		P116
I/O		P117
I/O, GCK4		P118
GND	GND	P119
DONE	**DONE**	**P120**
VCC	VCC	P121
PROGRAM-	**PROGRAM-**	**P122**
I/O (D7)	RST-	P123
I/O, GCK5		P124
I/O		P125
I/O		P126
I/O		P127
I/O		P128
I/O (D6)		P129
I/O		P130
I/O		P131
I/O		P132
I/O		P133
I/O		P134
GND	GND	P135
I/O		P136
I/O		P137

Table 4: Pinout for the XC4062XLA HQ240 (Continued)

Pin Function	PCI Function	HQ240
I/O		P138
I/O		P139
VCC	VCC	P140
I/O (D5)		P141
I/O (CS0-)		P142
GND	GND	P143
I/O		P144
I/O		P145
I/O		P146
I/O		P147
I/O (D4)		P148
I/O		P149
VCC	VCC	P150
GND	GND	P151
I/O (D3)		P152
I/O (RS-)		P153
I/O		P154
I/O		P155
I/O		P156
I/O		P157
GND	GND	P158
I/O (D2)		P159
I/O		P160
VCC	VCC	P161
I/O		P162
I/O		P163
I/O		P164
I/O		P165
GND	GND	P166
I/O		P167
I/O		P168
I/O		P169
I/O		P170
I/O		P171
I/O		P172
I/O (D1)		P173
I/O (RCLK, RDY/ BUSY)		P174
I/O		P175
I/O		P176
I/O (D0, DIN)	**DIN**	**P177**
I/O, GCK6 (DOUT)	**DOUT**	**P178**
CCLK	CCLK	P179
VCC	VCC	P180
O, TDO	TDO	P181
GND	GND	P182

Table 4: Pinout for the XC4062XLA HQ240 (Continued)

Pin Function	PCI Function	HQ240
I/O (A0, WS-)		P183
I/O, GCK7 (A1)		P184
I/O		P185
I/O		P186
I/O (CS1, A2)		P187
I/O (A3)		P188
I/O		P189
I/O		P190
I/O		P191
I/O		P192
I/O		P193
I/O		P194
I/O		P195
GND	GND	P196
I/O		P197
I/O		P198
I/O		P199
I/O		P200
VCC	VCC	P201
I/O (A4)		P202
I/O (A5)		P203
GND‡	GND	P204
I/O		P205
I/O		P206
I/O (A21)		P207
I/O (A20)		P208
I/O (A6)		P209
I/O (A7)		P210
GND	GND	P211
VCC	VCC	P212
I/O (A8)		P213

Table 4: Pinout for the XC4062XLA HQ240 (Continued)

Pin Function	PCI Function	HQ240
I/O (A9)		P214
I/O (A19)		P215
I/O (A18)		P216
I/O		P217
I/O		P218
GND‡	GND	P219
I/O (A10)		P220
I/O (A11)		P221
VCC	VCC	P222
I/O		P223
I/O	AD31	P224
I/O		P225
I/O	AD30	P226
GND	GND	P227
I/O	AD29	P228
I/O	AD28	P229
I/O	AD27	P230
I/O	AD26	P231
I/O (A12)	AD25	P232
I/O (A13)	AD24	P233
I/O		P234
I/O		P235
I/O	CBE3	P236
I/O	IDSEL	P237
I/O (A14)		P238
I/O, GCK8 (A15)		P239
VCC	VCC	P240

‡ Pins marked with this symbol are used for Ground connections on some revisions of the device. These pins may not physically connect to anything on the current device revision. However, they should be externally connected to Ground, if possible.

Pinout for the XC4062XLA BG432

Table 5: Pinout for the XC4062XLA BG432

Pin Function	PCI Function	BG432
I/O, GCK1 (A16)		D29
I/O (A17)		C30
I/O		E28
I/O		E29
I/O, TDI	TDI	D30
I/O, TCK	TCK	D31
I/O		F28
I/O		F29
I/O		E30
I/O		E31
I/O		G28
I/O		G29
I/O		F30
I/O		F31
I/O		H28
I/O		H29
I/O		G30
I/O		H30
I/O		J28
I/O		J29
I/O		H31
I/O		J30
I/O		K28
I/O		K29
I/O, TMS	TMS	K30
I/O		K31
I/O		L29
I/O		L30
I/O		M30
I/O		M28
I/O		M29
I/O		M31
I/O		N31
I/O		N28
I/O		N29
I/O		N30
I/O		P30
I/O		P28
I/O		P29
I/O		R31
I/O		R30
I/O		R28
I/O		R29
I/O		T31
I/O		T30

Table 5: Pinout for the XC4062XLA BG432

Pin Function	PCI Function	BG432
I/O		T29
I/O		U31
I/O		U30
I/O		U28
I/O		U29
I/O		V30
I/O		V29
I/O		V28
I/O		W31
I/O		W30
I/O		W29
I/O		W28
I/O		Y31
I/O		Y30
I/O		Y29
I/O		Y28
I/O		AA30
I/O		AA29
I/O		AB31
I/O		AB30
I/O		AB29
I/O		AB28
I/O		AC30
I/O		AC29
I/O		AC28
I/O		AD31
I/O		AD30
I/O		AD29
I/O		AD28
I/O		AE30
I/O		AE29
I/O		AF31
I/O		AE28
I/O		AF30
I/O		AF29
I/O		AG31
I/O		AF28
I/O		AG30
I/O		AG29
I/O		AH31
I/O		AG28
I/O		AH30
I/O, GCK2		AJ30
O (M1)	**M1**	**AH29**
I (M0)	**M0**	**AH28**
I (M2)	**M2**	**AJ28**
I/O, GCK3		AK29

Table 5: Pinout for the XC4062XLA BG432

Pin Function	PCI Function	BG432
I/O (HDC)	**HDC**	**AH27**
I/O		AK28
I/O		AJ27
I/O		AL28
I/O (LDC-)	**LDC-**	**AH26**
I/O		AK27
I/O		AJ26
I/O		AL27
I/O		AH25
I/O		AK26
I/O		AL26
I/O		AH24
I/O		AJ25
I/O		AK25
I/O		AJ24
I/O		AH23
I/O		AK24
I/O		AL24
I/O		AH22
I/O		AJ23
I/O		AK23
I/O		AJ22
I/O		AK22
I/O		AL22
I/O		AJ21
I/O		AH20
I/O		AK21
I/O		AJ20
I/O		AH19
I/O		AK20
I/O		AJ19
I/O		AL20
I/O		AH18
I/O		AK19
I/O		AJ18
I/O		AL19
I/O		AK18
I/O		AH17
I/O		AJ17
I/O		AK17
I/O		AL17
I/O		AJ16
I/O (INIT-)	**INIT-**	**AK16**
I/O		AL16
I/O		AH15
I/O		AL15
I/O		AJ15

Table 5: Pinout for the XC4062XLA BG432

Pin Function	PCI Function	BG432
I/O		AK15
I/O		AJ14
I/O		AH14
I/O		AK14
I/O		AL13
I/O		AK13
I/O		AJ13
I/O		AH13
I/O		AL12
I/O		AK12
I/O		AJ12
I/O		AK11
I/O		AH12
I/O		AJ11
I/O		AL10
I/O		AK10
I/O		AJ10
I/O		AK9
I/O		AL8
I/O		AH10
I/O		AJ9
I/O		AK8
I/O		AJ8
I/O		AH9
I/O		AK7
I/O		AL6
I/O		AJ7
I/O		AH8
I/O		AK6
I/O		AL5
I/O		AH7
I/O		AJ6
I/O		AK5
I/O		AL4
I/O		AH6
I/O		AJ5
I/O		AK4
I/O		AH5
I/O		AK3
I/O, GCK4		AJ4
DONE	**DONE**	**AH4**
PROGRAM-	**PROGRAM-**	**AH3**
I/O (D7)		AJ2
I/O, GCK5	PCLK	AG4
I/O	AD0	AG3
I/O	AD1	AH2
I/O	AD2	AH1

Table 5: Pinout for the XC4062XLA BG432

Pin Function	PCI Function	BG432
I/O	AD3	AF4
I/O	AD4	AF3
I/O	AD5	AG2
I/O	AD6	AG1
I/O	AD7	AE4
I/O		AE3
I/O		AF2
I/O (D6)		AF1
I/O	CBE0	AD4
I/O	AD8	AD3
I/O	AD9	AE2
I/O	AD10	AD2
I/O		AC4
I/O	AD11	AC3
I/O	AD12	AD1
I/O	AD13	AC2
I/O	AD14	AB4
I/O	AD15	AB3
I/O	CBE1	AB2
I/O	REQ-	AB1
I/O		AA3
I/O (D5)		AA2
I/O (CS0-)		Y2
I/O		Y4
I/O		Y3
I/O		Y1
I/O	PAR	W1
I/O	SERR-	W4
I/O		W3
I/O	PERR-	W2
I/O		V2
I/O	STOP-	V4
I/O	DEVSEL-	V3
I/O	TRDY-	U1
I/O	IRDY-	U2
I/O	FRAME-	U4
I/O	GNT-	U3
I/O (D4)		T1
I/O	CBE2	T2
I/O (D3)		T3
I/O (RS-)		R1
I/O	AD16	R2
I/O	AD17	R4
I/O	AD18	R3
I/O	AD19	P2
I/O	AD20	P3
I/O	AD21	P4

Table 5: Pinout for the XC4062XLA BG432

Pin Function	PCI Function	BG432
I/O		N1
I/O		N2
I/O	AD22	N3
I/O	AD23	N4
I/O	IDSEL	M1
I/O	CBE3	M2
I/O		M3
I/O		M4
I/O (D2)		L2
I/O		L3
I/O		K1
I/O	AD24	K2
I/O	AD25	K3
I/O	AD26	K4
I/O	AD27	J2
I/O	AD28	J3
I/O	AD29	J4
I/O	AD30	H1
I/O	AD31	H2
I/O		H3
I/O		H4
I/O		G2
I/O		G3
I/O		F1
I/O (D1)		G4
I/O (RCLK, RDY/ BUSY)		F2
I/O		F3
I/O		E1
I/O		F4
I/O		E2
I/O		E3
I/O		D1
I/O		E4
I/O		D2
I/O (D0, DIN)	**DIN**	**C2**
I/O, GCK6 (DOUT)	**DOUT**	**D3**
CCLK	CCLK	D4
O, TDO	TDO	C4
I/O (A0, WS-)		B3
I/O, GCK7 (A1)		D5
I/O		B4
I/O		C5
I/O		A4
I/O		D6
I/O		B5

Table 5: Pinout for the XC4062XLA BG432

Pin Function	PCI Function	BG432
I/O		C6
I/O (CS1, A2)		A5
I/O (A3)		D7
I/O		B6
I/O		A6
I/O		D8
I/O		C7
I/O		B7
I/O		D9
I/O		B8
I/O		A8
I/O		D10
I/O		C9
I/O	I/O	B9
I/O		C10
I/O		B10
I/O		A10
I/O		C11
I/O		D12
I/O		B11
I/O		C12
I/O		D13
I/O		B12
I/O		C13
I/O		A12
I/O		D14
I/O		B13
I/O (A4)		C14
I/O (A5)		A13
I/O		B14
I/O		D15
I/O (A21)		C15
I/O (A20)		B15
I/O		A15
I/O		C16
I/O (A6)		B16
I/O (A7)		A16
I/O (A8)		D17
I/O (A9)		A17
I/O		C17
I/O		B17
I/O (A19)		C18
I/O (A18)		D18
I/O		B18
I/O		A19
I/O (A10)		B19
I/O (A11)		C19

Table 5: Pinout for the XC4062XLA BG432

Pin Function	PCI Function	BG432
I/O		D19
I/O		A20
I/O		B20
I/O		C20
I/O		B21
I/O		D20
I/O		C21
I/O		A22
I/O		B22
I/O		C22
I/O		B23
I/O		A24
I/O		D22
I/O		C23
I/O		B24
I/O		C24
I/O		D23
I/O		B25
I/O		A26
I/O		C25
I/O (A12)		D24
I/O (A13)		B26
I/O		A27
I/O		D25
I/O		C26
I/O		B27
I/O		A28
I/O		D26
I/O		C27
I/O		B28
I/O		D27
I/O		B29
I/O (A14)		C28
I/O, GCK8 (A15)		D28

BG432

VCC Pins						
A1	A11	A21	A31	C3	C29	D11
D21	L1	L4	L28	L31	AA1	AA4
AA28	AA31	AH11	AH21	AJ3	AJ29	AL1
AL11	AL21	AL31	-	-	-	-
GND Pins						
A2	A3	A7	A9	A14	A18	A23
A25	A29	A30	B1	B2	B30	B31
C1	C31	D16	G1	G31	J1	J31
P1	P31	T4	T28	V1	V31	AC1
AC31	AE1	AE31	AH16	AJ1	AJ31	AK1
AK2	AK30	AK31	AL2	AL3	AL7	AL9
AL14	AL18	AL23	AL25	AL29	AL30	-
N.C. Pins						
C8	-	-	-	-	-	-

Pinout for the XCS20 TQ144

Table 6: Pinout for the XCS20 TQ144

Pin Function	PCI Function	PQ208
GND	GND	P1
I/O, GCK1	PCLK	P2
I/O	AD20	P3
I/O	AD19	P4
I/O	AD18	P5
I/O, TDI	TDI	P6
I/O, TCK	TCK	P7
GND	GND	P8
I/O	AD17	P9
I/O	AD16	P10
I/O, TMS	TMS	P11
I/O	CBE2	P12
I/O	FRAME-	P13
I/O	IRDY-	P14
I/O	TRDY-	P15
I/O	DEVSEL-	P16
GND	GND	P17
VCC	VCC	P18
I/O	STOP-	P19
I/O	PERR-	P20
I/O	SERR-	P21
I/O	PAR	P22
I/O	GNT-	P23
I/O	REQ-	P24
I/O	CBE1	P25
I/O	AD15	P26
GND	GND	P27
I/O	AD14	P28
I/O	AD13	P29
I/O	AD12	P30
I/O	AD11	P31
I/O	AD10	P32
I/O, GCK2	AD9	P33
M1		**P34**
GND	GND	P35
M0		**P36**
VCC	VCC	P37
PWRDWN		**P38**
GCK3	AD8	P39
I/O(HDC)		**P40**
I/O	CBE0	P41
I/O	AD7	P42
I/O	AD6	P43
I/O(LDC-)		**P44**
GND	GND	P45

Table 6: Pinout for the XCS20 TQ144 (Continued)

Pin Function	PCI Function	PQ208
I/O	AD5	P46
I/O	AD4	P47
I/O	AD3	P48
I/O	AD2	P49
I/O	AD1	P50
I/O	AD0	P51
I/O		P52
I/O (INIT-)	**INIT-**	**P53**
VCC	VCC	P54
GND	GND	P55
I/O		P56
I/O		P57
I/O		P58
I/O		P59
I/O		P60
I/O		P61
I/O		P62
I/O		P63
GND	GND	P64
I/O		P65
I/O		P66
I/O		P67
I/O		P68
I/O		P69
I/O, GCK4		P70
GND	GND	P71
DONE	**DONE**	**P72**
VCC	VCC	P73
PROGRAM		**P74**
I/O (D7)	RST-	P75
I/O, GCK5		P76
I/O		P77
I/O		P78
I/O (D6)		P79
I/O		P80
GND	GND	P81
I/O		P82
I/O		P83
I/O (D5)		P84
I/O		P85
I/O		P86
I/O		P87
I/O (D4)		P88
I/O		P89
VCC	VCC	P90
GND	GND	P91

Table 6: Pinout for the XCS20 TQ144 (Continued)

Pin Function	PCI Function	PQ208
I/O (D3)		P92
I/O		P93
I/O		P94
I/O		P95
I/O (D2)		P96
I/O		P97
I/O		P98
I/O		P99
GND	GND	P100
I/O (D1)		P101
I/O		P102
I/O		P103
I/O		P104
I/O (D0, DIN)	**DIN**	**P105**
I/O, GCK6 (DOUT)	**DOUT**	**P106**
CCLK	**CCLK**	**P107**
VCC	VCC	P108
O, TDO	TDO	P109
GND	GND	P110
I/O		P111
I/O, GCK7		P112
I/O		P113
I/O		P114
I/O (CS1)		P115
I/O		P116
I/O		P117
GND	GND	P118
I/O		P119
I/O		P120
I/O		P121
I/O		P122
I/O		P123
I/O		P124
I/O		P125
I/O		P126
GND	GND	P127
VCC	VCC	P128
I/O		P129
I/O	AD31	P130
I/O	AD30	P131
I/O	AD29	P132
I/O	AD28	P133
I/O	AD27	P134
I/O	AD26	P135
I/O	AD25	P136
GND	GND	P137

Table 6: Pinout for the XCS20 TQ144 (Continued)

Pin Function	PCI Function	PQ208
I/O	AD24	P138
I/O	CBE3	P139
I/O	IDSEL	P140
I/O	AD23	P141
I/O	AD22	P142
I/O, GCK8	AD21	P143
VCC	VCC	P144

Pinout for the XCS30 PQ208

Table 7: Pinout for the XCS30 PQ208

Pin Function	PCI Function	PQ208
GND	GND	P1
I/O, PGCK1	PCLK	P2
I/O		P3
I/O		P4
I/O	AD23	P5
I/O, TDI	TDI	P6
I/O, TCK	TCK	P7
I/O	AD22	P8
I/O	AD21	P9
I/O	AD20	P10
I/O	AD19	P11
I/O	AD18	P12
GND	GND	P13
I/O	AD17	P14
I/O	AD16	P15
I/O, TMS	TMS	P16
I/O		P17
VCC	VCC	P18
I/O	GNT-	P19
I/O	FRAME-	P20
I/O	IRDY-	P21
I/O	TRDY-	P22
I/O	DEVSEL-	P23
I/O	STOP-	P24
GND	GND	P25
VCC	VCC	P26
I/O	PERR-	P27
I/O	SERR-	P28
I/O	PAR	P29
I/O	REQ-	P30
I/O		P31
I/O	CBE2	P32
VCC	VCC	P33
I/O		P34
I/O	CBE1	P35
I/O	AD15	P36
I/O	AD14	P37
GND	GND	P38
I/O	AD13	P39
I/O	AD12	P40
I/O	AD11	P41
I/O	AD10	P42
I/O	AD9	P43
I/O	AD8	P44
I/O		P45

Table 7: Pinout for the XCS30 PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O		P46
I/O		P47
I/O		P48
I/O, SGCK2		P49
N.C.	N.C.	P50
GND	GND	P51
MODE	**MODE**	P52
VCC	VCC	P53
N.C.	N.C.	P54
I/O, PGCK2		P55
I/O (HDC)	**HDC**	**P56**
I/O		P57
I/O	I/O	P58
I/O	CBE0	P59
I/O (LDC-)	**LDC-**	**P60**
I/O	AD7	P61
I/O	AD6	P62
I/O	AD5	P63
I/O	AD4	P64
I/O	AD3	P65
GND	GND	P66
I/O	AD2	P67
I/O	AD1	P68
I/O	AD0	P69
I/O		P70
VCC	VCC	P71
I/O		P72
I/O		P73
I/O		P74
I/O		P75
I/O		P76
I/O (INIT-)	**INIT-**	**P77**
VCC	VCC	P78
GND	GND	P79
I/O		P80
I/O		P81
I/O		P82
I/O		P83
I/O		P84
I/O		P85
VCC	VCC	P86
I/O		P87
I/O		P88
I/O		P89
I/O	I/O	P90
GND	GND	P91

Table 7: Pinout for the XCS30 PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O		P92
I/O		P93
I/O		P94
I/O		P95
I/O		P96
I/O		P97
I/O		P98
I/O		P99
I/O		P100
I/O		P101
I/O, SGCK3		P102
GND	GND	P103
DONE	**DONE**	**P104**
VCC	VCC	P105
PROGRAM-	**PROGRAM-**	**P106**
I/O	I/O	P107
I/O, PGCK3		P108
I/O	RST-	P109
I/O		P110
I/O		P111
I/O		P112
I/O		P113
I/O		P114
I/O		P115
I/O		P116
I/O		P117
GND	GND	P118
I/O		P119
I/O		P120
VCC	VCC	P121
I/O		P122
I/O		P123
I/O		P124
I/O		P125
I/O		P126
I/O		P127
I/O		P128
I/O		P129
VCC	VCC	P130
GND	GND	P131
I/O		P132
I/O		P133
I/O		P134
I/O		P135
I/O		P136
I/O		P137

Table 7: Pinout for the XCS30 PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O		P138
I/O		P139
VCC	VCC	P140
I/O		P141
I/O		P142
GND	GND	P143
I/O		P144
I/O		P145
I/O		P146
I/O		P147
I/O		P148
I/O		P149
I/O		P150
I/O		P151
I/O		P152
I/O (DIN)	**DIN**	**P153**
I/O, SGCK4 (DOUT)	**DOUT**	**P154**
CCLK	CCLK	P155
VCC	VCC	P156
O, TDO	TDO	P157
GND	GND	P158
I/O		P159
I/O, PGCK4		P160
I/O		P161
I/O		P162
I/O		P163
I/O		P164
I/O		P165
I/O		P166
I/O		P167
I/O		P168
I/O		P169
GND	GND	P170
I/O		P171
I/O		P172
VCC	VCC	P173
I/O		P174
I/O		P175
I/O		P176
I/O		P177
I/O		P178
I/O		P179
I/O		P180
I/O		P181
GND	GND	P182
VCC	VCC	P183

Table 7: Pinout for the XCS30 PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O		P184
I/O		P185
I/O		P186
I/O		P187
I/O		P188
I/O		P189
I/O		P190
I/O	AD31	P191
VCC	VCC	P192
I/O	AD30	P193
I/O	AD29	P194
GND	GND	P195
I/O	AD28	P196
I/O	AD27	P197
I/O	AD26	P198
I/O	AD25	P199
I/O	AD24	P200
I/O	CBE3	P201
I/O	IDSEL	P202
I/O		P203
I/O		P204
I/O		P205
I/O		P206
I/O, SGCK1		P207
VCC	VCC	P208

Pinout for the XCS30 PQ240

Table 8: Pinout for the XCS30 PQ240

Pin Function	PCI Function	PQ240
GND	GND	P1
I/O, PGCK1	PCLK	P2
I/O	AD23	P3
I/O		P4
I/O	AD22	P5
I/O, TDI	TDI	P6
I/O, TCK	TCK	P7
I/O	AD21	P8
I/O	AD20	P9
I/O	AD19	P10
I/O	AD18	P11
I/O		P12
I/O		P13
GND	GND	P14
I/O	AD17	P15
I/O		P16
I/O, TMS	TMS	P17
I/O		P18
VCC		P19
I/O	AD16	P20
I/O	CBE2	P21
GND‡	GND	P22
I/O	GNT-	P23
I/O	FRAME-	P24
I/O	IRDY-	P25
I/O	TRDY-	P26
I/O	DEVSEL-	P27
I/O	STOP-	P28
GND	GND	P29
VCC	VCC	P30
I/O	PERR-	P31
I/O	SERR-	P32
I/O	PAR	P33
I/O	REQ-	P34
I/O		P35
I/O		P36
GND‡	GND	P37
I/O		P38
I/O		P39
VCC	VCC	P40
I/O		P41
I/O		P42
I/O	CBE1	P43
I/O	AD15	P44
GND	GND	P45

Table 8: Pinout for the XCS30 PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P46
I/O		P47
I/O	AD14	P48
I/O	AD13	P49
I/O	AD12	P50
I/O	AD11	P51
I/O	AD10	P52
I/O	AD9	P53
I/O	AD8	P54
I/O		P55
I/O		P56
I/O, SGCK2		P57
N.C.		P58
GND	GND	P59
MODE	**MODE**	**P60**
VCC	VCC	P61
N.C.	N.C.	P62
I/O, PGCK2		P63
I/O (HDC)	**HDC**	**P64**
I/O	CBE0	P65
I/O	AD7	P66
I/O	AD6	P67
I/O (LDC-)	**LDC-**	**P68**
I/O	AD5	P69
I/O	AD4	P70
I/O	AD3	P71
I/O	AD2	P72
I/O		P73
I/O		P74
GND	GND	P75
I/O	AD1	P76
I/O		P77
I/O	AD0	P78
I/O		P79
VCC	VCC	P80
I/O		P81
I/O		P82
GND‡	GND	P83
I/O		P84
I/O		P85
I/O		P86
I/O		P87
I/O		P88
I/O (INIT-)	**INIT-**	**P89**
VCC	VCC	P90
GND	GND	P91

Table 8: Pinout for the XCS30 PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P92
I/O		P93
I/O		P94
I/O		P95
I/O		P96
I/O		P97
GND‡	GND	P98
I/O		P99
I/O		P100
VCC	VCC	P101
I/O		P102
I/O		P103
I/O		P104
I/O		P105
GND	GND	P106
I/O		P107
I/O		P108
I/O		P109
I/O		P110
I/O		P111
I/O		P112
I/O		P113
I/O		P114
I/O		P115
I/O		P116
I/O		P117
I/O, SGCK3		P118
GND	GND	P119
DONE	**DONE**	**P120**
VCC	VCC	P121
PROGRAM-	**PROGRAM-**	**P122**
I/O	RST-	P123
I/O, PGCK3		P124
I/O		P125
I/O		P126
I/O		P127
I/O		P128
I/O		P129
I/O		P130
I/O		P131
I/O		P132
I/O		P133
I/O		P134
GND	GND	P135
I/O		P136
I/O		P137

Table 8: Pinout for the XCS30 PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P138
I/O		P139
VCC	VCC	P140
I/O		P141
I/O		P142
GND‡	GND	P143
I/O		P144
I/O		P145
I/O		P146
I/O		P147
I/O		P148
I/O		P149
VCC	VCC	P150
GND	GND	P151
I/O		P152
I/O		P153
I/O		P154
I/O		P155
I/O		P156
I/O		P157
GND‡	GND	P158
I/O		P159
I/O		P160
VCC	VCC	P161
I/O		P162
I/O		P163
I/O		P164
I/O		P165
GND	GND	P166
I/O		P167
I/O		P168
I/O		P169
I/O		P170
I/O		P171
I/O		P172
I/O		P173
I/O		P174
I/O		P175
I/O		P176
I/O (DIN)	**DIN**	**P177**
I/O, SGCK4 (DOUT)	**DOUT**	**P178**
CCLK	CCLK	P179
VCC	VCC	P180
O, TDO	TDO	P181
GND	GND	P182
I/O		P183

Table 8: Pinout for the XCS30 PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O, PGCK4		P184
I/O		P185
I/O		P186
I/O		P187
I/O		P188
I/O		P189
I/O		P190
I/O		P191
I/O		P192
I/O		P193
I/O		P194
N.C.	N.C.	P195
GND	GND	P196
I/O		P197
I/O		P198
I/O		P199
I/O	I/O	P200
VCC	VCC	P201
I/O		P202
I/O		P203
GND‡	GND	P204
I/O		P205
I/O		P206
I/O		P207
I/O		P208
I/O		P209
I/O		P210
GND	GND	P211
VCC	VCC	P212
I/O		P213
I/O		P214
I/O		P215

Table 8: Pinout for the XCS30 PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P216
I/O		P217
I/O		P218
GND‡	GND	P219
I/O		P220
I/O		P221
VCC	VCC	P222
I/O		P223
I/O	AD31	P224
I/O		P225
I/O	AD30	P226
GND	GND	P227
I/O	AD29	P228
I/O	AD28	P229
I/O	AD27	P230
I/O	AD26	P231
I/O	AD25	P232
I/O	AD24	P233
I/O		P234
I/O		P235
I/O	CBE3	P236
I/O	IDSEL	P237
I/O		P238
I/O, SGCK1		P239
VCC	VCC	P240

‡ Pins marked with this symbol are used for Ground connections on some revisions of the device. These pins may not physically connect to anything on the current device revision. However, they should be externally connected to Ground, if possible.

Pinout for the XCS40 PQ208

Table 9: Pinout for the XCS40 PQ208

Pin Function	PCI Function	PQ208
GND	GND	P1
I/O, PGCK1	PCLK	P2
I/O		P3
I/O		P4
I/O	AD23	P5
I/O, TDI	TDI	P6
I/O, TCK	TCK	P7
I/O	AD22	P8
I/O	AD21	P9
I/O	AD20	P10
I/O	AD19	P11
I/O	AD18	P12
GND	GND	P13
I/O	AD17	P14
I/O	AD16	P15
I/O, TMS	TMS	P16
I/O		P17
VCC	VCC	P18
I/O	GNT-	P19
I/O	FRAME-	P20
I/O	IRDY-	P21
I/O	TRDY-	P22
I/O	DEVSEL-	P23
I/O	STOP-	P24
GND	GND	P25
VCC	VCC	P26
I/O	PERR-	P27
I/O	SERR-	P28
I/O	PAR	P29
I/O	REQ-	P30
I/O		P31
I/O	CBE2	P32
VCC	VCC	P33
I/O		P34
I/O	CBE1	P35
I/O	AD15	P36
I/O	AD14	P37
GND	GND	P38
I/O	AD13	P39
I/O	AD12	P40
I/O	AD11	P41
I/O	AD10	P42
I/O	AD9	P43
I/O	AD8	P44
I/O		P45

Table 9: Pinout for the XCS40 PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O		P46
I/O		P47
I/O		P48
I/O, SGCK2		P49
N.C.	N.C.	P50
GND	GND	P51
MODE	**MODE**	P52
VCC	VCC	P53
N.C.	N.C.	P54
I/O, PGCK2		P55
I/O (HDC)	**HDC**	**P56**
I/O		P57
I/O	I/O	P58
I/O	CBE0	P59
I/O (LDC-)	**LDC-**	**P60**
I/O	AD7	P61
I/O	AD6	P62
I/O	AD5	P63
I/O	AD4	P64
I/O	AD3	P65
GND	GND	P66
I/O	AD2	P67
I/O	AD1	P68
I/O	AD0	P69
I/O		P70
VCC	VCC	P71
I/O		P72
I/O		P73
I/O		P74
I/O		P75
I/O		P76
I/O (INIT-)	**INIT-**	**P77**
VCC	VCC	P78
GND	GND	P79
I/O		P80
I/O		P81
I/O		P82
I/O		P83
I/O		P84
I/O		P85
VCC	VCC	P86
I/O		P87
I/O		P88
I/O		P89
I/O	I/O	P90
GND	GND	P91

Table 9: Pinout for the XCS40 PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O		P92
I/O		P93
I/O		P94
I/O		P95
I/O		P96
I/O		P97
I/O		P98
I/O		P99
I/O		P100
I/O		P101
I/O, SGCK3		P102
GND	GND	P103
DONE	**DONE**	**P104**
VCC	VCC	P105
PROGRAM-	**PROGRAM-**	**P106**
I/O	I/O	P107
I/O, PGCK3		P108
I/O	RST-	P109
I/O		P110
I/O		P111
I/O		P112
I/O		P113
I/O		P114
I/O		P115
I/O		P116
I/O		P117
GND	GND	P118
I/O		P119
I/O		P120
VCC	VCC	P121
I/O		P122
I/O		P123
I/O		P124
I/O		P125
I/O		P126
I/O		P127
I/O		P128
I/O		P129
VCC	VCC	P130
GND	GND	P131
I/O		P132
I/O		P133
I/O		P134
I/O		P135
I/O		P136
I/O		P137

Table 9: Pinout for the XCS40 PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O		P138
I/O		P139
VCC	VCC	P140
I/O		P141
I/O		P142
GND	GND	P143
I/O		P144
I/O		P145
I/O		P146
I/O		P147
I/O		P148
I/O		P149
I/O		P150
I/O		P151
I/O		P152
I/O (DIN)	**DIN**	**P153**
I/O, SGCK4 (DOUT)	**DOUT**	**P154**
CCLK	CCLK	P155
VCC	VCC	P156
O, TDO	TDO	P157
GND	GND	P158
I/O		P159
I/O, PGCK4		P160
I/O		P161
I/O		P162
I/O		P163
I/O		P164
I/O		P165
I/O		P166
I/O		P167
I/O		P168
I/O		P169
GND	GND	P170
I/O		P171
I/O		P172
VCC	VCC	P173
I/O		P174
I/O		P175
I/O		P176
I/O		P177
I/O		P178
I/O		P179
I/O		P180
I/O		P181
GND	GND	P182
VCC	VCC	P183

Table 9: Pinout for the XCS40 PQ208 (Continued)

Pin Function	PCI Function	PQ208
I/O		P184
I/O		P185
I/O		P186
I/O		P187
I/O		P188
I/O		P189
I/O		P190
I/O	AD31	P191
VCC	VCC	P192
I/O	AD30	P193
I/O	AD29	P194
GND	GND	P195
I/O	AD28	P196
I/O	AD27	P197
I/O	AD26	P198
I/O	AD25	P199
I/O	AD24	P200
I/O	CBE3	P201
I/O	IDSEL	P202
I/O		P203
I/O		P204
I/O		P205
I/O		P206
I/O, SGCK1		P207
VCC	VCC	P208

Pinout for the XCS40 PQ240

Table 10: Pinout for the XCS40 PQ240

Pin Function	PCI Function	PQ240
GND	GND	P1
I/O, PGCK1	PCLK	P2
I/O	AD23	P3
I/O		P4
I/O	AD22	P5
I/O, TDI	TDI	P6
I/O, TCK	TCK	P7
I/O	AD21	P8
I/O	AD20	P9
I/O	AD19	P10
I/O	AD18	P11
I/O		P12
I/O		P13
GND	GND	P14
I/O	AD17	P15
I/O		P16
I/O, TMS	TMS	P17
I/O		P18
VCC		P19
I/O	AD16	P20
I/O	CBE2	P21
GND‡	GND	P22
I/O	GNT-	P23
I/O	FRAME-	P24
I/O	IRDY-	P25
I/O	TRDY-	P26
I/O	DEVSEL-	P27
I/O	STOP-	P28
GND	GND	P29
VCC	VCC	P30
I/O	PERR-	P31
I/O	SERR-	P32
I/O	PAR	P33
I/O	REQ-	P34
I/O		P35
I/O		P36
GND‡	GND	P37
I/O		P38
I/O		P39
VCC	VCC	P40
I/O		P41
I/O		P42
I/O	CBE1	P43
I/O	AD15	P44
GND	GND	P45

Table 10: Pinout for the XCS40 PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P46
I/O		P47
I/O	AD14	P48
I/O	AD13	P49
I/O	AD12	P50
I/O	AD11	P51
I/O	AD10	P52
I/O	AD9	P53
I/O	AD8	P54
I/O		P55
I/O		P56
I/O, SGCK2		P57
N.C.		P58
GND	GND	P59
MODE	**MODE**	**P60**
VCC	VCC	P61
N.C.	N.C.	P62
I/O, PGCK2		P63
I/O (HDC)	**HDC**	**P64**
I/O	CBE0	P65
I/O	AD7	P66
I/O	AD6	P67
I/O (LDC-)	**LDC-**	**P68**
I/O	AD5	P69
I/O	AD4	P70
I/O	AD3	P71
I/O	AD2	P72
I/O		P73
I/O		P74
GND	GND	P75
I/O	AD1	P76
I/O		P77
I/O	AD0	P78
I/O		P79
VCC	VCC	P80
I/O		P81
I/O		P82
GND‡	GND	P83
I/O		P84
I/O		P85
I/O		P86
I/O		P87
I/O		P88
I/O (INIT-)	**INIT-**	**P89**
VCC	VCC	P90
GND	GND	P91

Table 10: Pinout for the XCS40 PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P92
I/O		P93
I/O		P94
I/O		P95
I/O		P96
I/O		P97
GND‡	GND	P98
I/O		P99
I/O		P100
VCC	VCC	P101
I/O		P102
I/O		P103
I/O		P104
I/O		P105
GND	GND	P106
I/O		P107
I/O		P108
I/O		P109
I/O		P110
I/O		P111
I/O		P112
I/O		P113
I/O		P114
I/O		P115
I/O		P116
I/O		P117
I/O, SGCK3		P118
GND	GND	P119
DONE	**DONE**	**P120**
VCC	VCC	P121
PROGRAM-	**PROGRAM-**	**P122**
I/O	RST-	P123
I/O, PGCK3		P124
I/O		P125
I/O		P126
I/O		P127
I/O		P128
I/O		P129
I/O		P130
I/O		P131
I/O		P132
I/O		P133
I/O		P134
GND	GND	P135
I/O		P136
I/O		P137

Table 10: Pinout for the XCS40 PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P138
I/O		P139
VCC	VCC	P140
I/O		P141
I/O		P142
GND‡	GND	P143
I/O		P144
I/O		P145
I/O		P146
I/O		P147
I/O		P148
I/O		P149
VCC	VCC	P150
GND	GND	P151
I/O		P152
I/O		P153
I/O		P154
I/O		P155
I/O		P156
I/O		P157
GND‡	GND	P158
I/O		P159
I/O		P160
VCC	VCC	P161
I/O		P162
I/O		P163
I/O		P164
I/O		P165
GND	GND	P166
I/O		P167
I/O		P168
I/O		P169
I/O		P170
I/O		P171
I/O		P172
I/O		P173
I/O		P174
I/O		P175
I/O		P176
I/O (DIN)	**DIN**	**P177**
I/O, SGCK4 (DOUT)	**DOUT**	**P178**
CCLK	CCLK	P179
VCC	VCC	P180
O, TDO	TDO	P181
GND	GND	P182
I/O		P183

Table 10: Pinout for the XCS40 PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O, PGCK4		P184
I/O		P185
I/O		P186
I/O		P187
I/O		P188
I/O		P189
I/O		P190
I/O		P191
I/O		P192
I/O		P193
I/O		P194
N.C.	N.C.	P195
GND	GND	P196
I/O		P197
I/O		P198
I/O		P199
I/O	I/O	P200
VCC	VCC	P201
I/O		P202
I/O		P203
GND‡	GND	P204
I/O		P205
I/O		P206
I/O		P207
I/O		P208
I/O		P209
I/O		P210
GND	GND	P211
VCC	VCC	P212
I/O		P213

Table 10: Pinout for the XCS40 PQ240 (Continued)

Pin Function	PCI Function	PQ240
I/O		P214
I/O		P215
I/O		P216
I/O		P217
I/O		P218
GND‡	GND	P219
I/O		P220
I/O		P221
VCC	VCC	P222
I/O		P223
I/O	AD31	P224
I/O		P225
I/O	AD30	P226
GND	GND	P227
I/O	AD29	P228
I/O	AD28	P229
I/O	AD27	P230
I/O	AD26	P231
I/O	AD25	P232
I/O	AD24	P233
I/O		P234
I/O		P235
I/O	CBE3	P236
I/O	IDSEL	P237
I/O		P238
I/O, SGCK1		P239
VCC	VCC	P240

‡ Pins marked with this symbol are used for Ground connections on some revisions of the device. These pins may not physically connect to anything on the current device revision. However, they should be externally connected to Ground, if possible.

Pinout for the XCV300 BG432

Table 11: Pinout for the PCI64 Interface in a XCV300 BG432

PCI Function	BG432
AD63	N4
AD62	N3
AD61	N1
AD60	P3
AD59	AB4
AD58	AC2
AD57	AD1
AD56	AC3
AD55	AC4
AD54	AD2
AD53	AE2
AD52	AD3
AD51	AD4
AD50	AF2
AD49	AE3
AD48	AE4
AD47	AG1
AD46	AG2
AD45	AF3
AD44	AF4
AD43	AH1
AD42	AH2
AD41	AG3
AD40	AJ4
AD39	AK3
AD38	AH5
AD37	AK4
AD36	AJ5
AD35	AH6
AD34	AL4
AD33	AK5
AD32	AJ6
AD31	D7
AD30	A5
AD29	C6
AD28	B5
AD27	D6
AD26	A4
AD25	C5
AD24	D2
AD23	E4
AD22	D1
AD21	E2
AD20	F4

Table 11: Pinout for the PCI64 Interface in a XCV300 BG432

PCI Function	BG432
AD19	E1
AD18	F3
AD17	F2
AD16	G4
AD15	G3
AD14	G2
AD13	H4
AD12	H3
AD11	H2
AD10	H1
AD9	J4
AD8	J3
AD7	J2
AD6	K1
AD5	L3
AD4	L2
AD3	M4
AD2	M3
AD1	M2
AD0	M1
PCLK	A16
RST	B6
REQ	P2
IDSEL	R3
DEVSEL	R4
GNT	R2
STOP	R1
IRDY	T3
TRDY	T2
FRAME	U3
ACK64	U4
REQ64	U2
CBE3	U1
CBE2	V3
CBE1	V2
CBE0	W3
PERR	W4
PAR	W1
PAR64	Y1
CBE7	Y3
CBE6	Y4
CBE5	Y2
CBE4	AA2
SERR	AA3
INTA	**A6**

Resources

PCI Special Interest Group (PCI-SIG) Publications

The PCI-SIG publishes various PCI specifications and related documents. Most publications cost US$25 plus applicable shipping charges.

- PCI Local Bus Specification, Rev. 2.2
- PCI Compliance Checklist v2.1 (available via the Web)
- PCI System Design Guide v1.0

Contact:

PCI Special Interest Group
2575 NE Kathryn St. #17
Hillsboro, OR 97124
Phone: +1 800-433-5177 (within USA)
Phone: +1 503-693-6232 (worldwide)
Fax: +1 503-693-8344
E-Mail: info@pcisig.com
URL: www.pcisig.com

PCI and FPGA XPERT Partners

Listed below are design centers and design consultants that have experience with the LogiCORE PCI Products.

Nallatech Limited
Nallatech are experts in complete electronics based systems design. This includes the skills required for software/hardware partitioning and the development of Distributed Parallel Processing Systems.
Nallatech specializes in:

- Real time image processing systems
- DSP
- DSP algorithm development targeting FPGA's
- Military designs, specifically obsolescence upgrades
- High speed data throughput
- Complex PCI Logicore designs

10-14 Market Street
Kilsyth
Glasgow
G65 0BD
Scotland
Phone: +1 44 7020 986532
Fax: +1 44 7020 986534
Email: info@nallatech.com
URL: www.nallatech.com

Multi Video Designs
Multi Video Designs (MVD) offer complete consultant design and training services to their clients. MVD's experience in offering design consultancy services has offered then a broad experience with Xilinx related designs.

Edgard Garcia
106 Av. des Guis
38130 Plaisance du Touch
France
Phone: +1 (33) 05 62 13 52 32
Fax: +1 (33) 05 61 06 72 60
Email: edgard.garcia@mvd-fpga.com
URL: www.mvd-fpga.com

Memec Design Services
Memec Design Services is dedicated to bringing you the best Xilinx design engineering services and FPGA library modules available in the market today. Memec Design Service is also an AllianceCORE member. Memec is a global services company with offices in U.S, Europe, Hong Kong and China. They provide PCI services that include customization and integration.

Headquarters:
1819 S. Dobson Rd. Ste. 203
Mesa, AZ 85202, USA
Phone: +1 888-360-9044 - inside the USA
+1 602-491-4311 - outside the USA
Fax: +1 602-491-4907
E-mail: southwest@memecdesign.com
URL: www.memecdesign.com

Branch Offices:
2460 N. First Street
Suite 170
San Jose, CA 95131
Phone: +1408-952-7018
Fax: +1 408-952-7059
Email: west@memecdesign.com

30 Nagog Park
Acton, MA 01720
Phone: +1 978-266-9193
Fax: +1 978-266-9194
Email: northeast@memecdesign.com

Memec Design Services
% Insight World Trade Center
Montecito No. 38
Piso 19, OFNA.12.Col.Napoles,
C.P. 03810, Mexico City D.F.
Phone: +1 525-488-0119
Fax: +1 525-488-0179
Email: mexico@memecdesign.com

4835 University Square
Suite 19
Huntsville, AL 35816
Phone: +1 256-830-5732
Fax: +1 256-830-5787
Email: southeast@memecdesign.com

555 Legget Drive, Suite 305
Kanata, Ontario K2K2X3
Phone: +1 613-271-2028
Fax: +1 613-599-4867
Email: canada@memecdesign.com

Unit 3520, Tower 1, Metroplaza
Hing Fong Road, Kwai Fong
N.T., Hong Kong
Phone: +1 852-2410-2720
FAX: +1 852-2481-6937

Rm A801, Bao Hua Building
Hua Qiang North Road
Shenzhen, P.R.China

Comit Systems
Comit Systems is a software and systems engineering company. They offer design services along with a set of efficiently implemented libraries for Xilinx devices. Comit Systems is also an AllianceCORE member. Comit Systems have done numerous turnkey designs and have extensive experience with Xilinx PCI Core, both in integration and customization.

3375 Scott Blvd, Suite 330
Santa Clara California 95054
Phone: +1 408-988-7966
Fax: +1 408-988-2133
E-mail: preeth@comit.com
URL: www.comit.com

Dark Room Technologies, Inc.
Dark Room Technologies provides all levels of consulting from routing an FPGA to complete board-level solutions, including debugged prototypes. Our tools include the full Viewlogic tool suite with VHDL modeling capability, Abel, XAbel, PC design and layout, C cross compilers, 192-channel 200-MHz logic analysis systems, FCC prescreening, T1 and DS-3 Test and Diagnostic equipment, and SMD assembly, inspection and rework equipment. We can do code development and bring the design through FCC EMI testing and verification.

Dark Room Technologies, Inc. has been specializing in Xilinx FPGA design consulting since 1987, and doing product development and engineering consulting since 1976. We have done over 50 FPGA design including interfaces to SCSI, PCI, ISA/EISA, Turbo channel, ATM, T1, DS-3, and most micro and embedded processors, such as StrongARM, i960, IDT4640 and the R3000.

Austin Franklin
126 Poor Farm Road
Harvard MA, 01451, USA
Phone: +1 508-772-9928
Fax: +1 508-772-4287
E-mail: info@darkroom.com
URL: www.darkroom.com

Supporting PCI Tools

Nallatech Limited
Provides the PCI64 PCI Prototyping Board
Allan Cantle
10-14 Market Street
Kilsyth, Glasgow
G65 0BD
Scotland
Phone: +1 44 7020 986532
Fax: +1 44 7020 986534
Email: a.cantle@nallatech.com
URL: www.nallatech.com

VCC Corporation
Provides HotPCI Rapid Prototyping Board
6925 Canby Ave. #103
Reseda, CA 91335 USA
Phone: +1 818-342-8294
Fax: +1 818-342-0240
E-mail: info@vcc.com
Website: www.vcc.com

Compuware Numega (formerly, Vireo Software Inc.)
Provides drivers and driver development tools.
30 Monument Square, Suite 135
Concord, MA 01742 USA
Phone: +1 978-369-3380
Fax: +1 978-318-6946
E-mail: customer_service@numega.com
Website: www.numega.com

PCI Reference Books

There are many reference books available on PCI. The following are a few that the LogiCORE development team found useful.

PCI System Architecture by Tom Shanley and Don Anderson. ISBN 1-881609-08-1. An excellent general reference book on PCI. This book is included with the LogiCORE PCI product.

Contact:

Mindshare Press
2202 Buttercup Dr.
Richardson, TX 75082
Phone: +1 214-231-2216
Fax: +1 214-783-4715

Distributed by:

Computer Literacy Bookshops, Inc.
P.O. Box 641897
San Jose, CA 95164
Phone: +1 408-435-0744
Fax: +1 408-435-1823
E-mail: contact-us@fatbrain.com
URL: www.fatbrain.com

PCI Hardware and Software, 4th Edition by Edward Solari & George Willse. ISBN 0-929392-59-0. Everything that you ever wanted to know about PCI systems design, and more.

Contact:

Annabooks
11848 Bernardo Center Drive
Suite 110
San Diego, CA 92128
Phone: +1 619-673-0870
+1 800-462-1042
Fax: +1 619-673-1432
E-mail: info@annabooks.com
URL: www.annabooks.com

Xilinx Documents

See the Xilinx web page at

www.xilinx.com/pci

for available literature.

IMPORTANT! Be sure to visit the Xilinx WebLINX web site for the latest information and application notes using the LogiCORE PCI interface.

LogiCORE User's Lounge

The LogiCORE User's Lounge web site provides a quick and convenient way to obtain the latest updates, documentation, design tips, application notes, and utilities. The Lounge web site is open to registered LogiCORE users. To register, point your Internet browser software to:

www.xilinx.com/pci

Waveforms

Waveforms

These waveforms demonstrate the operation the Xilinx V3.0 PCI core, including the states of the backend signals. The 64-bit transactions only apply to the PCI64 Virtex Interface. In these examples, the PCI V2.2 Specification names, such as FRAME# and IRDY# are used for the PCI Bus signals. Due to the limitations of design tools, the Xilinx PCI core has slightly different names for the PCI Bus signals. These names are listed in the Signal Description chapter of the Xilinx *PCI Design Guide*.

These waveforms were created with the Xilinx Internal PCI Testbench. See specific sections for explanations regarding these waveforms.

Target Configuration Read

Figure 1 represents a Configuration Read transaction. In this transaction, data from configuration space (in this case the devices and vendor IDs) is driven onto the ADIO[31:0] bus. The “cfg.v/.vhd” file is the source of the configuration space setting.

Address Phase

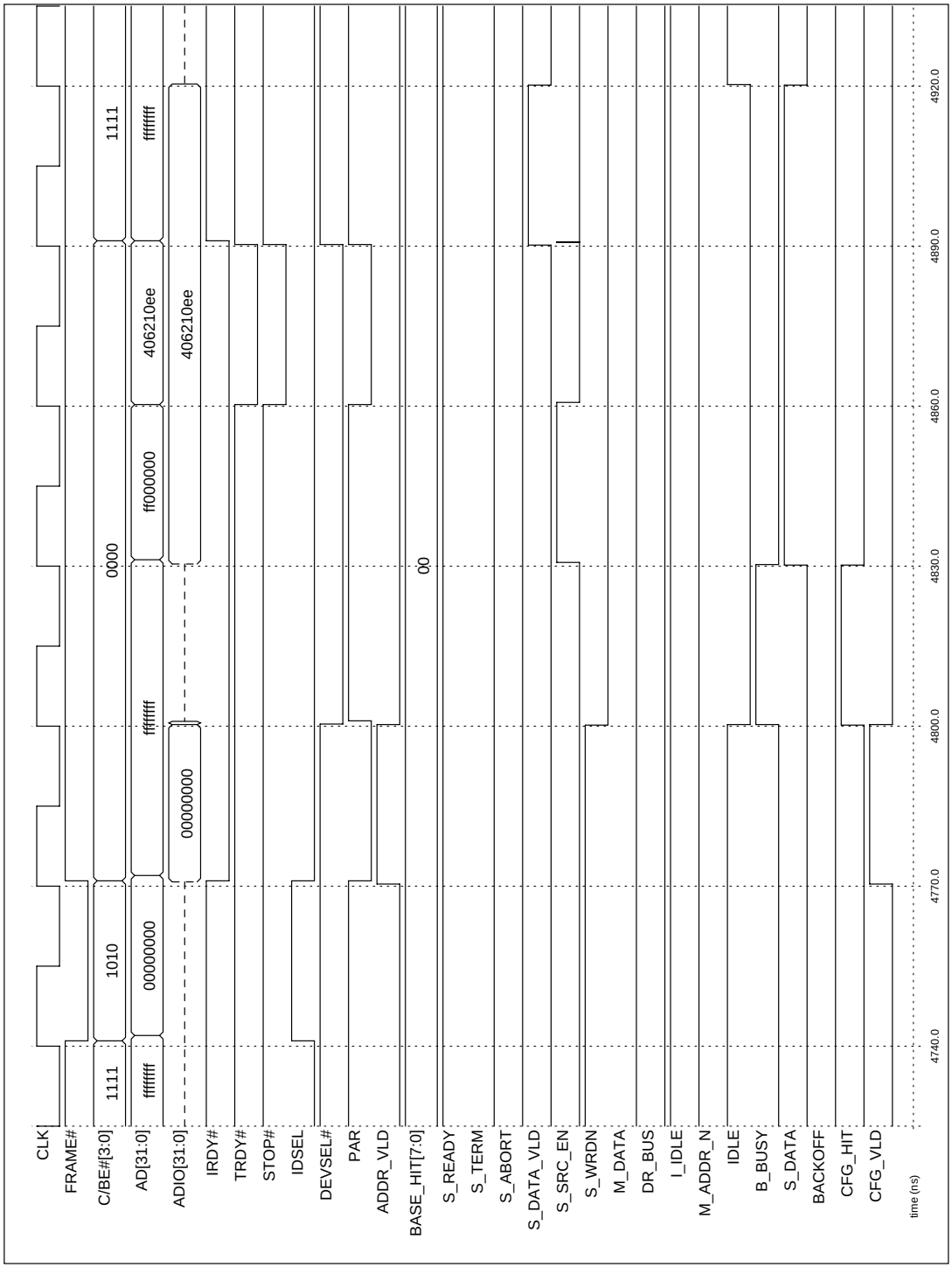
The host asserts IDSEL and puts the address on the PCI bus. The address appears on the ADIO bus one cycle later.

Data Phase

Since the address is in the first 64 bytes of configuration, the PCI core automatically responds to this transactions. The PCI core asserts CFG_VLD and ADDR_VLD, indicating the Target has a valid configuration cycle address on the ADIO bus. CFG_HIT and DEVSEL# are then asserted.

The transaction continues and completes in accordance with a single cycle read. The Xilinx PCI core always terminates configuration transactions with a “disconnect with data” where DEVSEL#, STOP#, and TRDY# are all asserted since burst configuration cycles are not supported in Xilinx PCI Interfaces. For details on using extended configuration space (addresses from 40h to FFh) refer to the Xilinx *PCI Design Guide*.

Figure 1: Target Configuration Read



Target Configuration Write

Figure 2 represents a Configuration Write transaction. In this transaction, data from the host is written into Base Address Register 1.

Address Phase

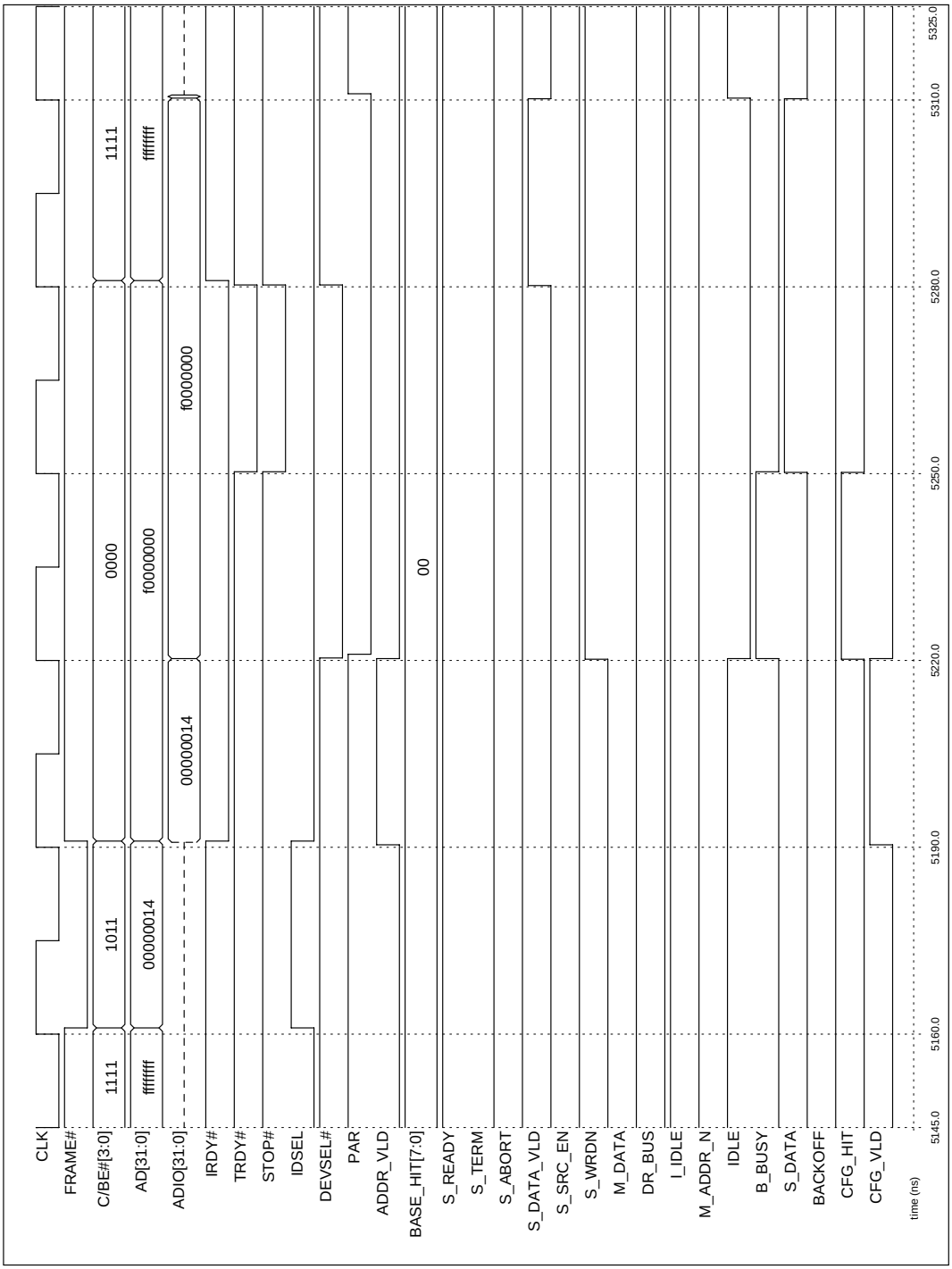
The host asserts IDSEL and puts the address on the PCI bus. The address appears on the ADIO bus one cycle later.

Data Phase

Since the address is in the first 64 bytes of configuration, the PCI core automatically responds to this transactions. The PCI core asserts CFG_VLD and ADDR_VLD, indicating the Target has a valid configuration cycle address on the ADIO bus. CFG_HIT and DEVSEL# are then asserted.

The transaction continues and completes in accordance with a single cycle write. The Xilinx PCI core always terminates configuration transactions with a “disconnect with data” where DEVSEL#, STOP#, and TRDY# are all asserted since burst configuration cycles are not supported in Xilinx PCI Interfaces. For details on using extended configuration space (addresses from 40h to FFh) refer to the Xilinx *PCI Design Guide*.

Figure 2: Target Configuration Write



Initiator 32-bit Single Memory Read

Figure 3 represents a single cycle Memory Read transaction. This consists of an address phase followed immediately by a single data phase.

Requesting the PCI Bus

REQUEST is asserted by the user's logic for one clock cycle to generate a REQ# signal to the PCI Arbiter. The PCI Interface asserts FRAME# only if GNT# is asserted for more than one clock cycle.

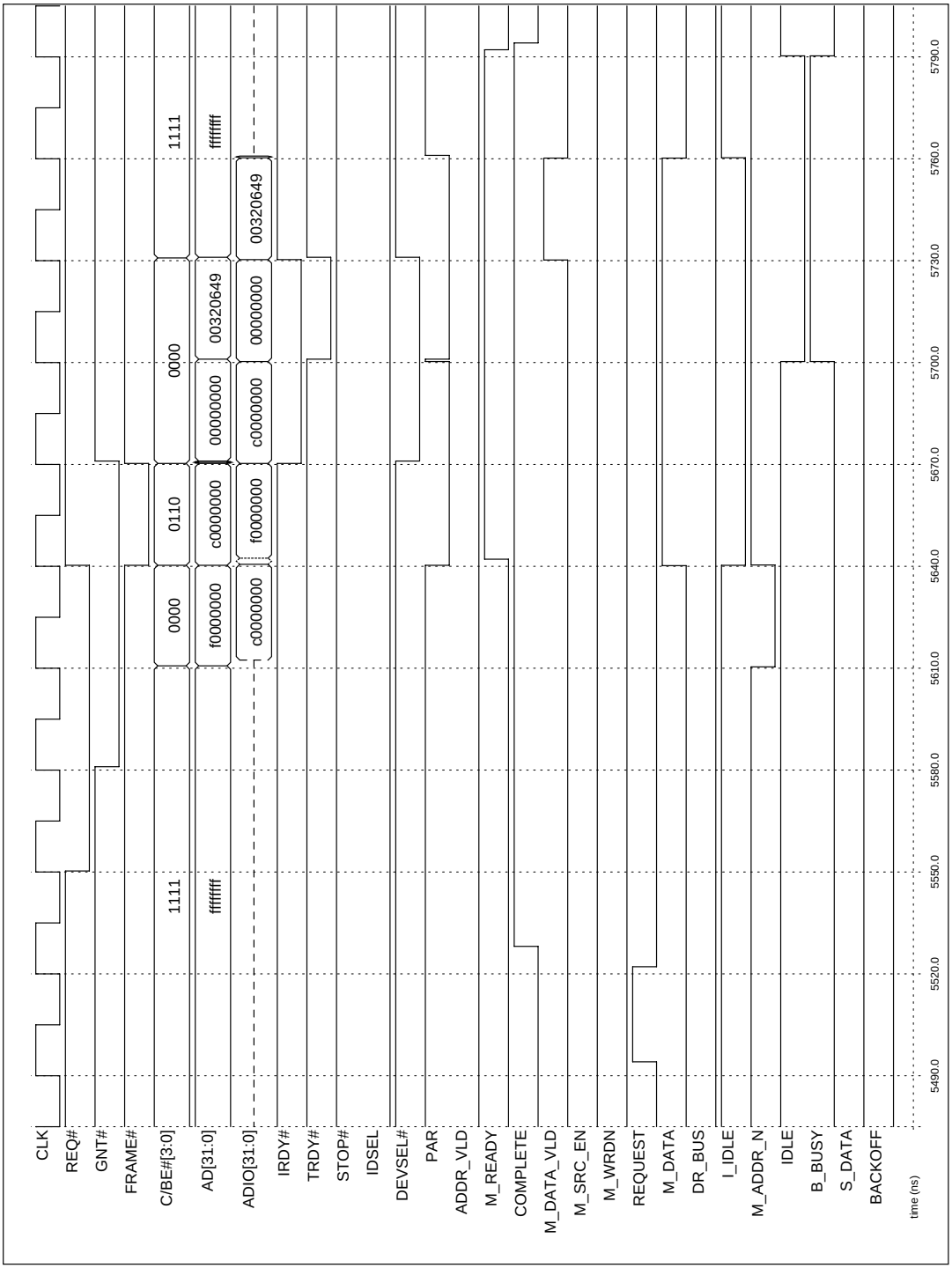
Address Phase

The M_ADDR_N signal is used to drive the output-enable of the tri-state buffers (BUFTs) which enable the valid address onto the internal ADIO bus one CLK cycle before the assertion of FRAME#. Once the Initiator asserts FRAME#, it drives the AD bus with the address from the ADIO bus, and drives a Memory Read command (0110), supplied by the user to the backend (M_CBE[3:0], not shown), onto the C/BE# lines.

Data Phase

The M_DATA phase begins on the CLK cycle immediately following the M_ADDR_N phase. During M_DATA, the byte enables are presented on M_CBE. One clock cycle after FRAME# and REQ64# are asserted, the Initiator asserts IRDY#, and the fast decode Target asserts DEVSEL#. Since this is a Read, there is a one cycle bus turnaround after the address phase; TRDY# can not be asserted until after the turnaround cycle. The Target is now suppling data onto the PCI bus, while the Initiator is supplying the Byte enables. Since this is only a single Dword transfer, COMPLETE and M_READY are both asserted at the beginning of the transaction, and FRAME# is deasserted after one cycle to indicate the last data will occur. The Target asserts TRDY#, which starts the data transfer and completes with the deassertion of IRDY#, TRDY#, and DEVSEL#. M_DATA_VLD indicates that the data is valid on AD bus and should be latched by the Initiator register, in the user's backend. Notice that this is the cycle after TRDY# and IRDY# were asserted. The signal M_WRDN is asserted low to represent data is being read.

Figure 3: Initiator 32-bit Single Memory Read



Initiator 32-bit Single Memory Write

Figure 4 represents a single cycle Memory Write transaction. This consists of an address phase followed immediately by a single data phase.

Requesting the PCI Bus

REQUEST is asserted for one clock cycle to generate a REQ# signal to the PCI Arbiter. The PCI Interface asserts FRAME# only if GNT# is asserted for more than one clock cycle.

Address Phase

The M_ADDR_N signal is used to drive the output-enable of the 3-state buffers (BUFTs) which enable the valid address onto the internal ADIO bus one CLK cycle before the assertion of FRAME#.

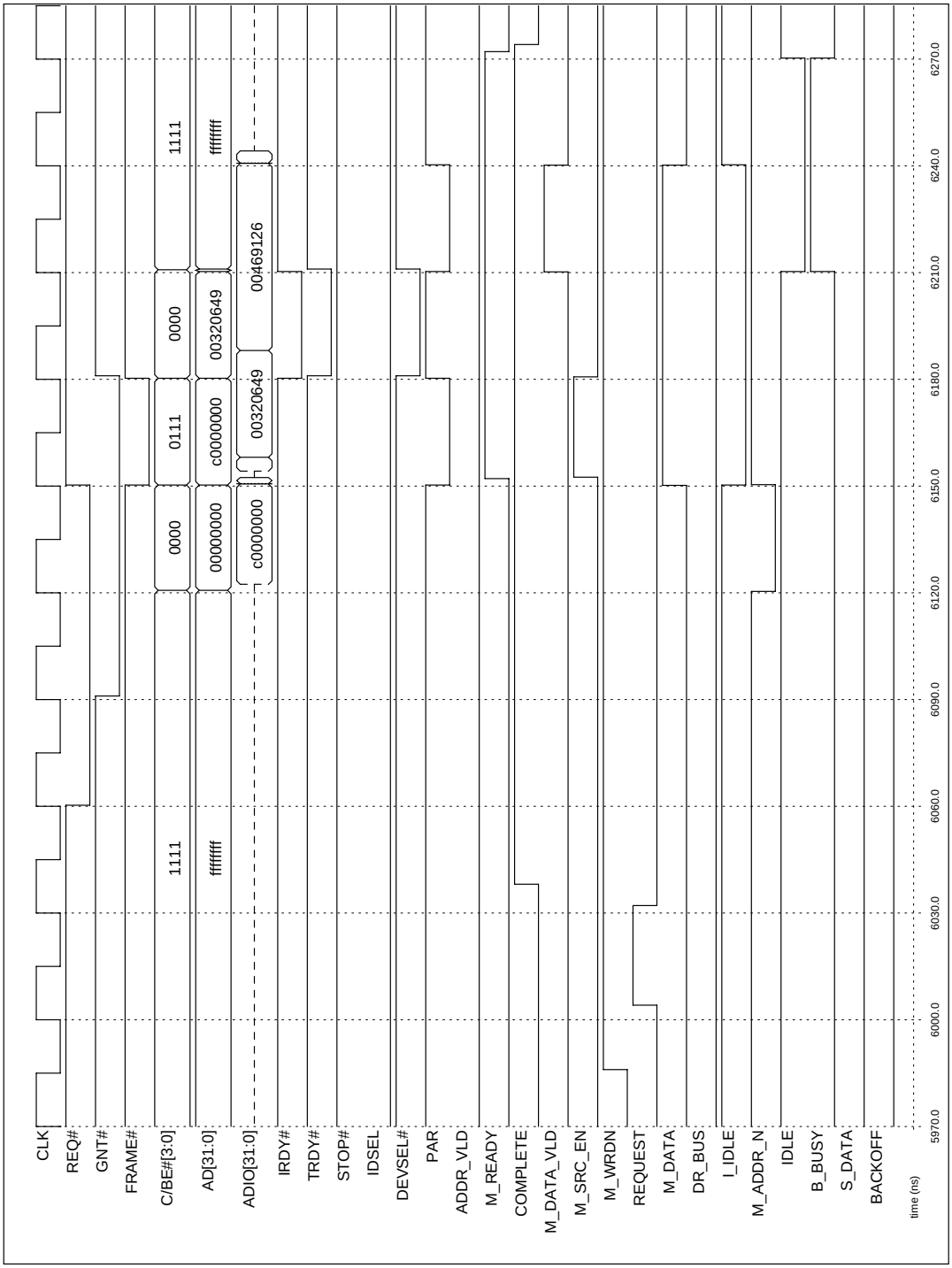
Once the Initiator asserts FRAME#, it drives the AD lines with the address put on the ADIO bus during the previous CLK cycle, and drives a Memory Write command (0111), supplied by the user to the backend (M_CBE[3:0], not shown), onto the C/BE# lines.

Data Phase

The M_DATA phase begins on the CLK cycle immediately following the M_ADDR_N phase. During M_DATA, the byte enables are presented on M_CBE. Data is placed on the ADIO bus from a register in the user design. The output enable of the register is controlled by M_DATA (conditional upon being the write state!).

One clock cycle after FRAME# is asserted, the Initiator asserts IRDY#, and on the next CLK cycle, the fast decode Target asserts both TRDY# and DEVSEL#. Since this is only a single Dword transfer, COMPLETE and M_READY are both asserted at the beginning of the transaction, and FRAME# is deasserted after one cycle to indicate the last data is present on the AD[31:0] bus. With both IRDY# and TRDY# asserted, the data transfer completes and we see M_DATA_VLD on the next cycle, indicating the Target accepted the data. The transfer completes with the deassertion of IRDY#, TRDY#, and DEVSEL# and the bus has a turnaround cycle. The signal M_WRDN is asserted high to represent data is being written.

Figure 4: Initiator 32-bit Single Memory Write



Initiator 32-bit Burst Memory Read Multiple

Figure 5 represents a burst cycle Memory Read Multiple transaction. This consists of a single address phase followed by two or more data phases.

Requesting the PCI Bus

REQUEST is asserted for one clock cycle to generate a REQ# signal to the PCI Arbiter. The PCI Interface asserts FRAME# only if GNT# is asserted for more than one clock cycle.

Address Phase

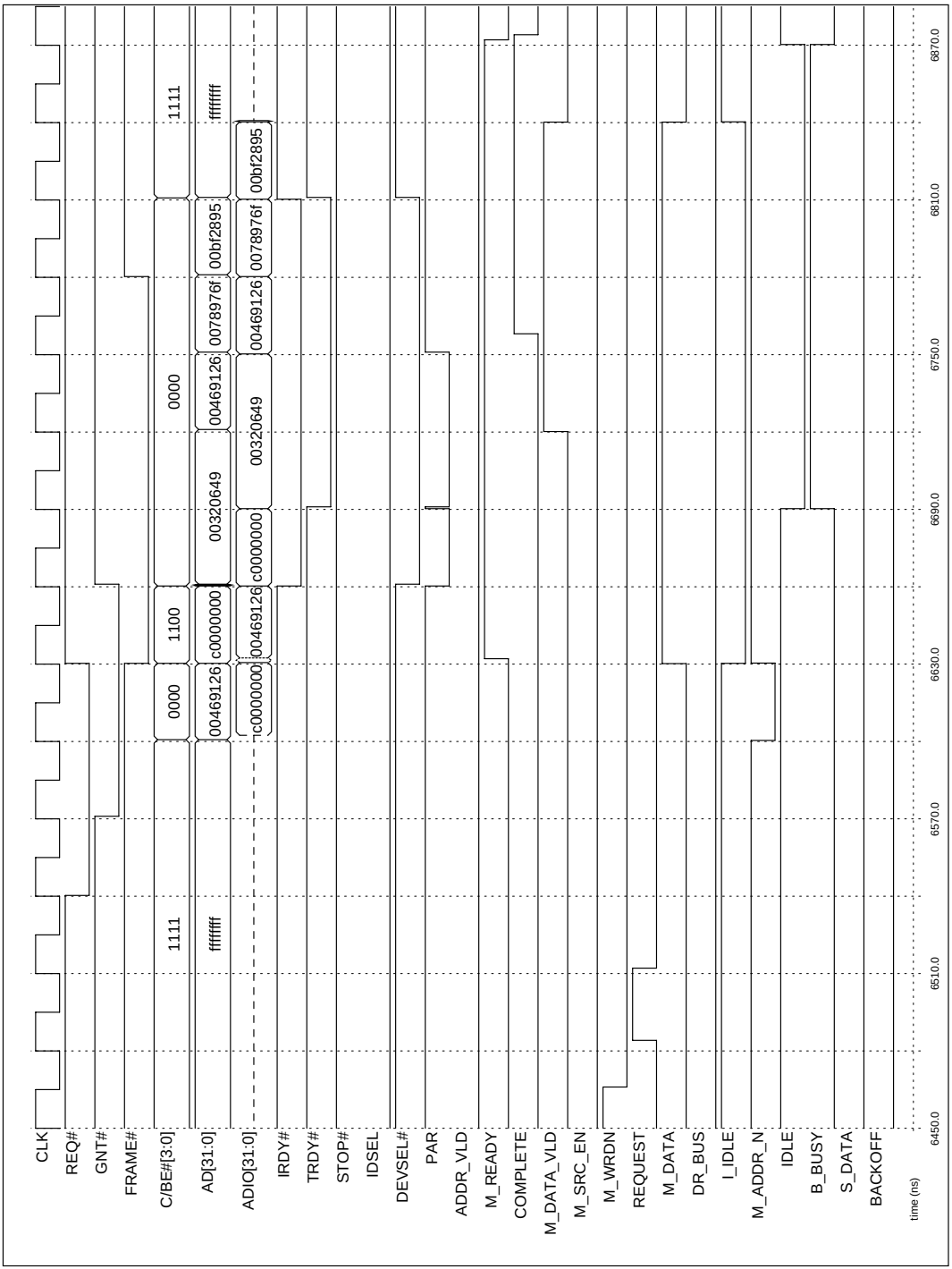
The M_ADDR_N signal is used to drive the output-enable of the 3-state buffers (BUFTs) which enable the valid address onto the internal ADIO bus one CLK cycle before the assertion of FRAME#. Once the Initiator asserts FRAME#, it drives the AD lines with the address put on the ADIO bus during the previous CLK cycle, and drives a Memory Read Multiple command (1100), supplied by the user to the backend (M_CBE[3:0], not shown), onto the C/BE# lines.

Data Phase

The M_DATA phase begins on the CLK cycle immediately following the M_ADDR_N phase. During M_DATA, the byte enables are presented on M_CBE. One clock cycle after FRAME# and REQ64# are asserted, the Initiator asserts IRDY#, and the fast decode Target asserts DEVSEL#. Since this is a Read, there is a one cycle bus turnaround after the address phase; TRDY# can not be asserted until after the turnaround cycle. The Target is now suppling data onto the PCI bus, while the Initiator is supplying the Byte enables. Since this is a multi-Dword transfer, COMPLETE is initially deasserted and M_READY is asserted at the beginning of the transaction, and FRAME# stays asserted until the next to last data phase.

M_DATA_VLD indicates the data is present on the AD bus and is used to advance the Initiator address pointer. With both IRDY# and TRDY# asserted, the data transfer occurs without inserted wait states from either agent. COMPLETE is asserted on the next to last data phase, since the Initiator has to deassert FRAME# at the beginning of the last data phase. The transfer completes with the deassertion of IRDY#, TRDY#, and DEVSEL# and the bus has a turnaround cycle. The signal M_WRDN is asserted low to represent data is being read.

Figure 5: Initiator 32-bit Burst Memory Read Multiple



Initiator 32-bit Burst Memory Write

Figure 6 represents a burst cycle Memory Write transaction. This consists of a single address phase followed by four data phases.

Requesting the PCI Bus

REQUEST is asserted for one clock cycle to generate a REQ# signal to the PCI Arbiter. The PCI Interface asserts FRAME# only if GNT# is asserted for more than one clock cycle.

Address Phase

The M_ADDR_N signal is used to drive the output-enable of the 3-state buffers (BUFTs) which enable the valid address onto the internal ADIO bus one CLK cycle before the assertion of FRAME#. Once the Initiator asserts FRAME#, it drives the AD lines with the address put on the ADIO bus during the previous CLK cycle, and drives a Memory Write command (0111), supplied by the user to the backend (M_CBE[3:0], not shown), onto the C/BE# lines.

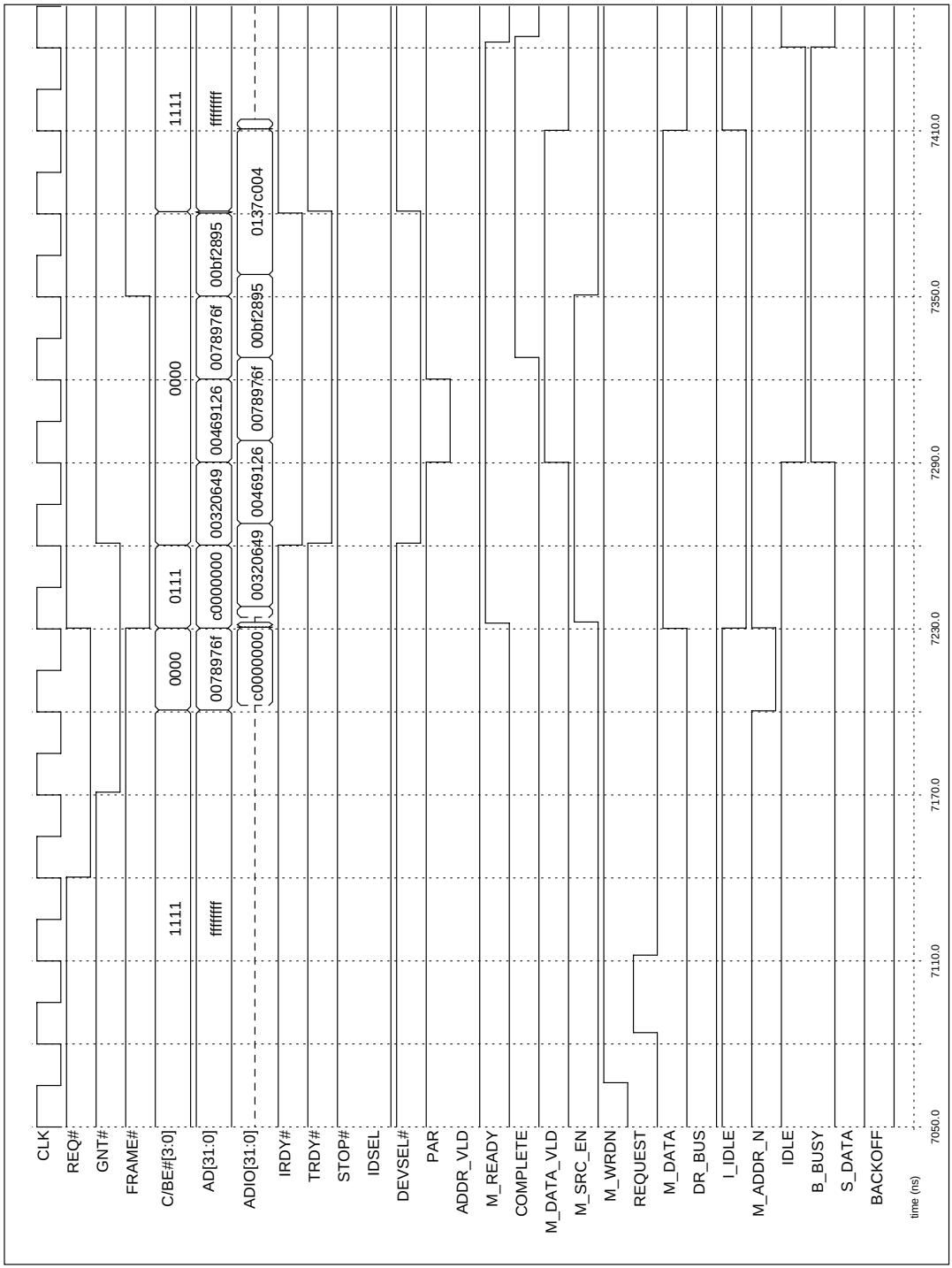
Data Phase

The M_DATA phase begins on the CLK cycle immediately following the M_ADDR_N phase. During M_DATA, the byte enables are presented on M_CBE. Data is placed on the ADIO bus from a FIFO in the user design. The output enable of the FIFO is controlled by M_DATA (conditional upon being the write state!).

One clock cycle after FRAME# is asserted, the Initiator asserts IRDY#, and the fast decode Target asserts DEVSEL# and TRDY#. Since this is a multi-Dword transfer, COMPLETE is initially deasserted and M_READY is asserted at the beginning of the transaction. COMPLETE is asserted during the next to last data phase and FRAME# is deasserted at the beginning of the last data phase.

M_DATA_VLD indicates the data was taken by the Target and M_SRC_EN is used to advance the Initiator address pointer to read the next data from the FIFO. With both IRDY# and TRDY# asserted, the data transfer occurs without inserted wait states from either agent. The transfer completes with the deassertion of IRDY#, TRDY#, and DEVSEL# and the bus has a turnaround cycle. The signal M_WRDN is asserted high to represent data is being written.

Figure 6: Initiator 32-bit Burst Memory Write



Initiator 32-bit Burst Memory Write with Disconnect

Figure 7 represents a burst cycle Memory Write transaction with a Target termination. This consists of a single address phase followed by four data phases but only three Dwords were transferred because of the disconnect with data.

Requesting the PCI Bus

REQUEST is asserted for one clock cycle to generate a REQ# signal to the PCI Arbiter. The PCI Interface asserts FRAME# only if GNT# is asserted for more than one clock cycle.

Address Phase

The M_ADDR_N signal is used to drive the output-enable of the 3-state buffers (BUFTs) which enable the valid address onto the internal ADIO bus one CLK cycle before the assertion of FRAME#. Once the Initiator asserts FRAME#, it drives the AD lines with the address put on the ADIO bus during the previous CLK cycle, and drives a Memory Write command (0111), supplied by the user to the backend (M_CBE[3:0], not shown), onto the C/BE# lines.

Data Phase

The M_DATA phase begins on the CLK cycle immediately following the M_ADDR_N phase. During M_DATA, the byte enables are presented on M_CBE. Data is placed on the ADIO bus from a FIFO in the user design. The output enable of the FIFO is controlled by M_DATA (conditional upon being the write state!).

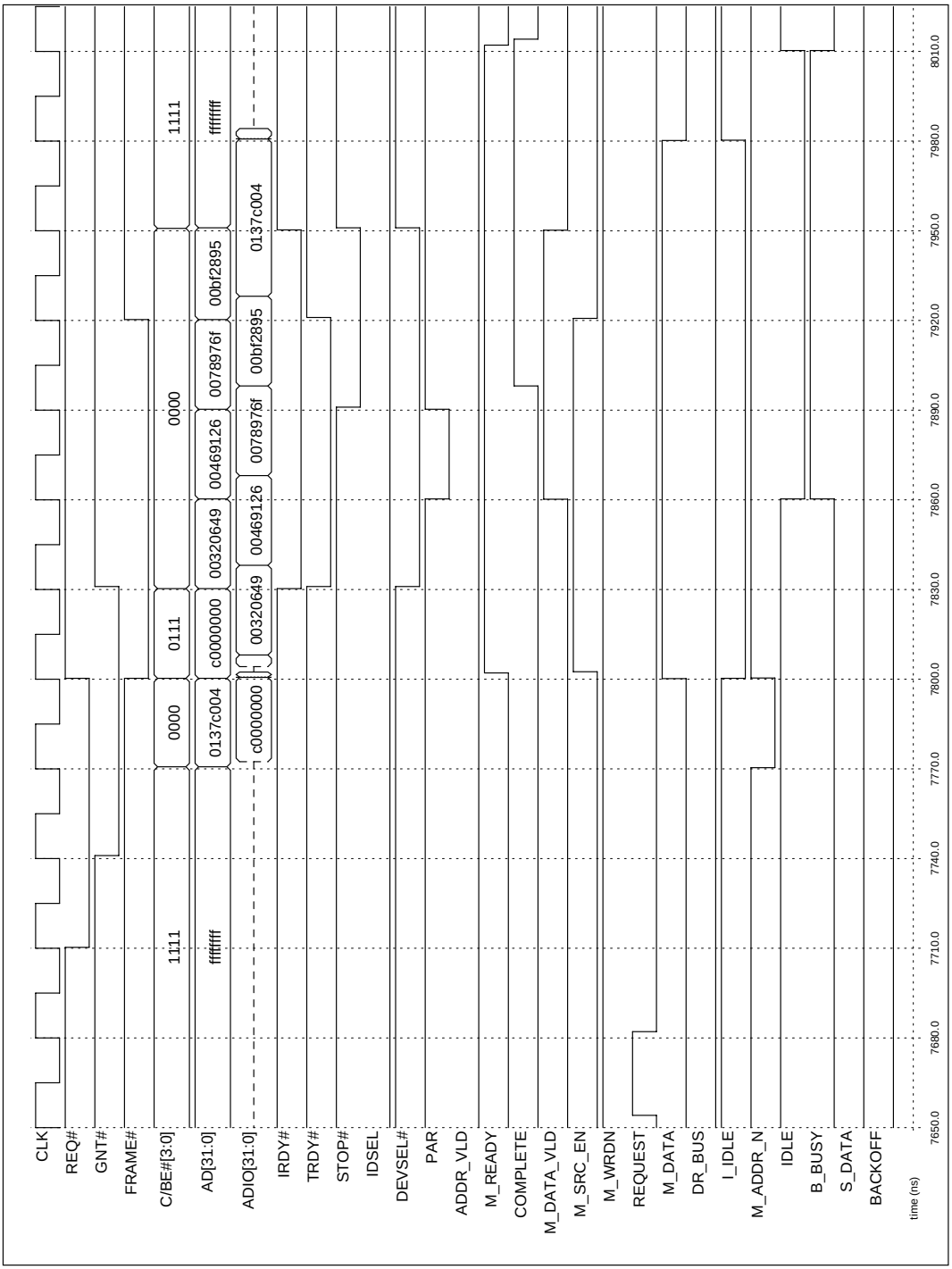
One clock cycle after FRAME# is asserted, the Initiator asserts IRDY#, and the fast decode Target asserts DEVSEL# and TRDY#. Since this is a multi-Dword transfer, COMPLETE is initially deasserted and M_READY is asserted at the beginning of the transaction. COMPLETE is asserted during the next to last data phase and FRAME# is deasserted at the beginning of the last data phase.

M_DATA_VLD indicates the data was taken by the Target and M_SRC_EN is used to advance the Initiator address pointer to read the next data from the FIFO. With both IRDY# and TRDY# asserted, the data transfer occurs without inserted wait states from either agent.

The Target signalled a disconnect by asserting STOP#, and kept TRDY# asserted so that the third Dword was transferred. The last data phase does not have a data transfer because TRDY# is not asserted. M_SRC_EN is asserted for four cycles and M_DATA_VLD is asserted for only three cycles. Since there is a difference between the number of data transfers anticipated (four), and the number that occurred (three), the user design may have to perform a backup of the FIFO. Details on how to do this are documented in the Xilinx *PCI Design Guide*.

The transfer completes with the deassertion of IRDY#, STOP#, and DEVSEL# and the bus has a turnaround cycle. The signal M_WRDN is asserted high to represent data is being written.

Figure 7: Initiator 32-bit Burst Memory Write with Disconnect



Target 32-bit Single Memory Read

Figure 8 represents a single cycle Target Memory Read transaction. In this transaction, data is placed on the ADIO bus from a data register in the user design.

Address Phase

The Initiator drives FRAME# and the address onto the AD lines, and drives a Memory Read command (0110) onto C/BE#. The Initiator signals that it only wants to transfer one Dword by deasserting FRAME# and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture this address.

Data Phase

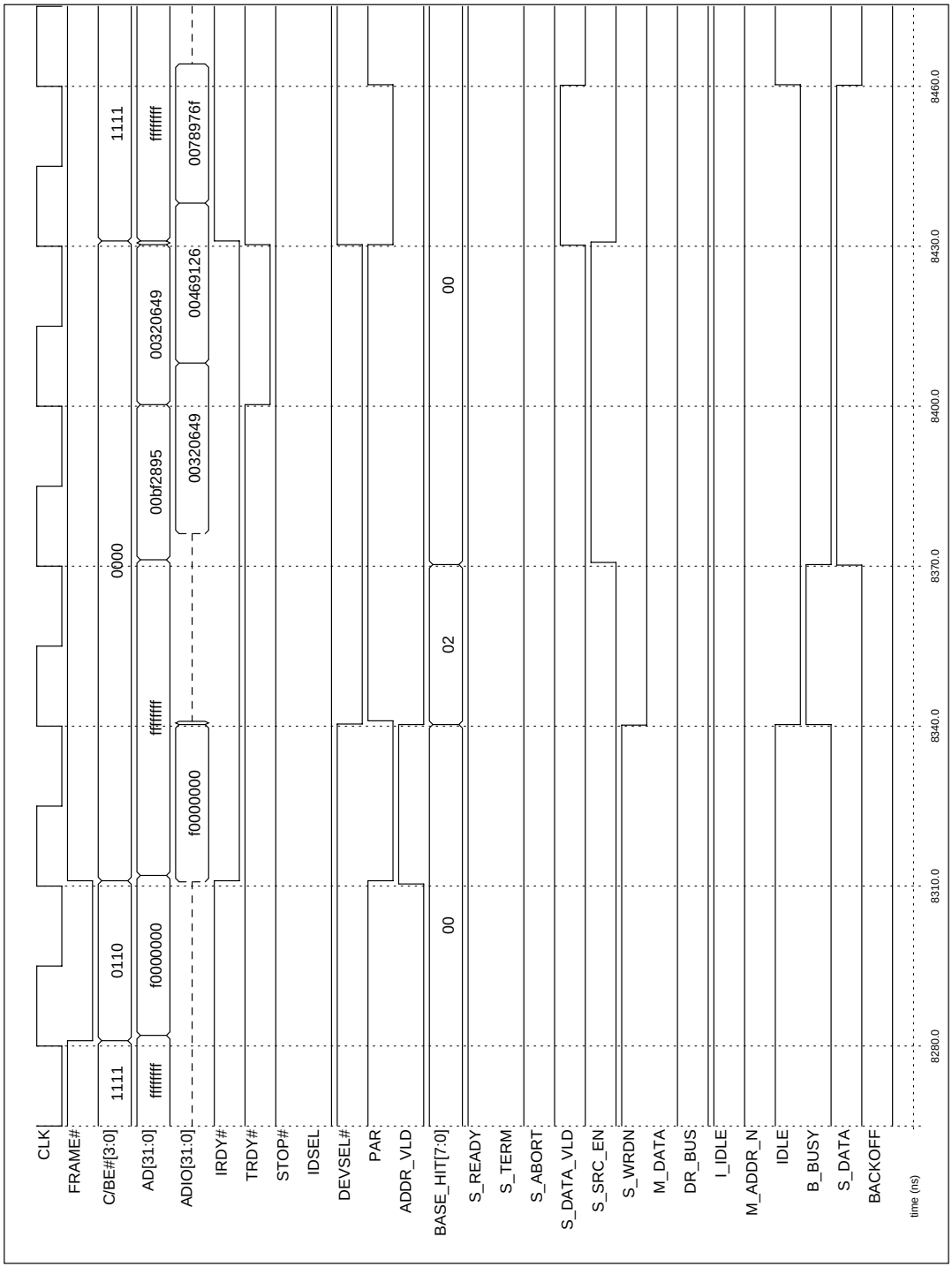
The PCI Interface sees that the bus is no longer idle and asserts Target state machine signals, IDLE and B_BUSY. After B_BUSY, the Target state machine enters the S_DATA state, indicating that the backend will begin transferring data. DEVSEL# is asserted as a medium decode speed. The PCI Interface always adds one cycle of TRDY# latency after DEVSEL# assertion. Since this is a read, TRDY# is not asserted for one extra cycle beyond the usual TRDY# latency due to the Read transaction turnaround cycle where the Initiator stops driving AD and the Target begins driving AD.

BASE_HIT[x] is asserted for one cycle to indicate to the backend logic that it is the target of an access. The signal S_WRDN is asserted low to represent data is being read from the Target.

S_READY is asserted and S_TERM is deasserted, indicating that the backend is able to transfer more than one Dword. The PCI core deasserts TRDY# and DEVSEL# after one transfer, because the Initiator deasserts FRAME#.

S_SRC_EN is asserted for two cycles and S_DATA_VLD is asserted for one cycle. Since there is a difference between the number of data transfers anticipated (two), and the number that occurred (one), the user design may have to perform a backup of the FIFO. Details on how to do this are documented in the Xilinx *PCI Design Guide*.

Figure 8: Target 32-bit Single Memory Read



Target 32-bit Single Memory Write

Figure 9 represents a single cycle Target Memory Write transaction. In this transaction, data is captured from the ADIO bus to a data register in the user design.

Address Phase

The Initiator drives FRAME# and the address onto the AD lines, and drives a Memory Write command (0111) onto C/BE#. The Initiator signals that it only wants to transfer one Dword by deasserting FRAME# and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture this address.

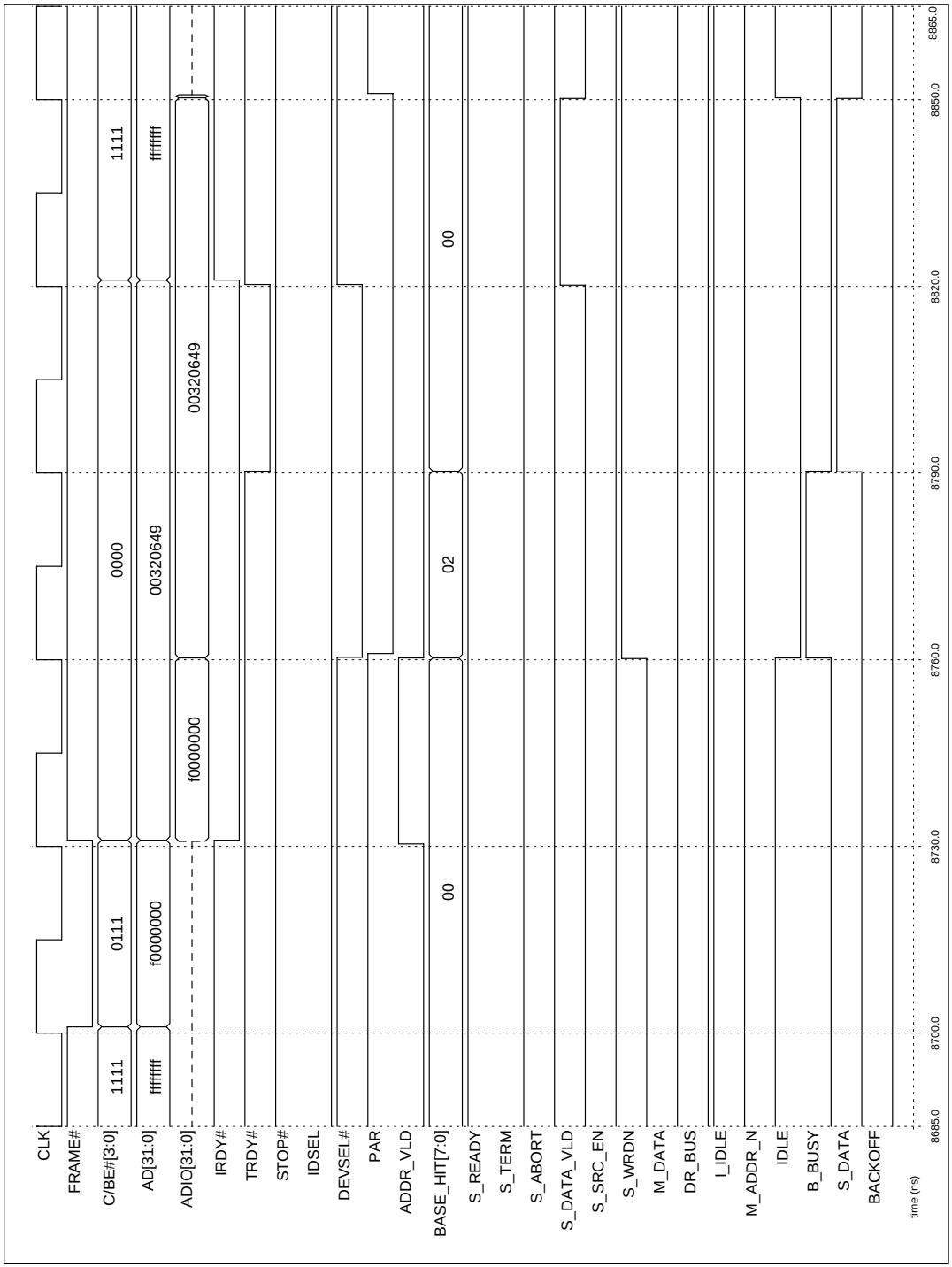
Data Phase

The PCI Interface sees that the bus is no longer idle and asserts Target state machine signals, IDLE and B_BUSY. After B_BUSY, the Target state machine enters the S_DATA state, indicating that the backend will begin transferring data. DEVSEL# is asserted as a medium decode speed. The PCI Interface always adds one cycle of TRDY# latency after DEVSEL# assertion.

BASE_HIT[x] is asserted for one cycle to indicate to the backend logic that it is the target of an access. The signal S_WRDN is asserted high to represent that data is being written to the target.

S_READY is asserted and S_TERM is deasserted, indicating that the backend is able to transfer more than one Dword. The PCI core deasserts TRDY# and DEVSEL# after one transfer, because the Initiator deasserts FRAME#.

Figure 9: Target 32-bit Single Memory Write



Target 32-bit Burst Memory Read Multiple

Figure 10 represents a Target Memory Read Multiple transaction. In this transaction, data is placed on the ADIO bus from a FIFO in the user design.

Address Phase

The Initiator drives FRAME# and the address onto the AD lines, and drives a Memory Read Multiple command (1100) onto C/BE#. The Initiator signals that it wants to transfer multiple Dwords by keeping FRAME# asserted and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture a copy of this address into the address counter.

Data Phase

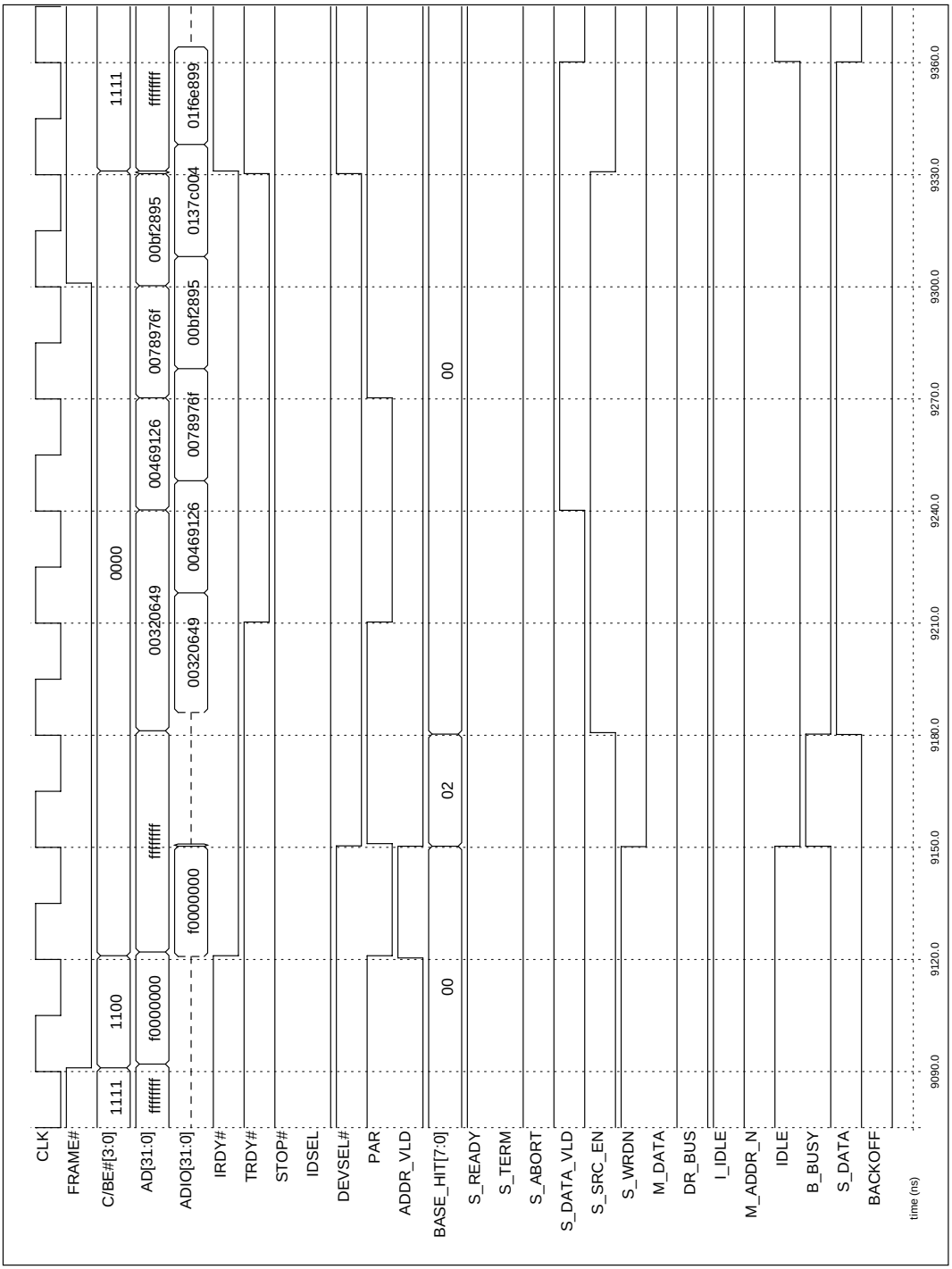
The PCI Interface sees that the bus is no longer idle and asserts Target state machine signals, IDLE and B_BUSY. After B_BUSY, the Target state machine enters the S_DATA state, indicating that the backend will begin transferring data. DEVSEL# is asserted as a medium decode speed. The PCI Interface always adds one cycle of TRDY# latency after DEVSEL# assertion. Since this is a read, TRDY# is not asserted for one extra cycle beyond the usual TRDY# latency due to the Read transaction turnaround cycle where the Initiator stops driving AD and the Target begins driving AD.

BASE_HIT[x] is asserted for one cycle to indicate to the backend logic that it is the target of an access. The signal S_WRDN is asserted low to represent data is being read from the Target.

S_READY is asserted and S_TERM is deasserted, indicating that the backend is able to transfer more than one Dword.

S_SRC_EN is asserted for five cycles and S_DATA_VLD is asserted for four cycles. Since there is a difference between the number of data transfers anticipated (five), and the number that occurred (four), the user design may have to perform a backup of the FIFO. Details on how to do this are documented in the Xilinx *PCI Design Guide*.

Figure 10: Target 32-bit Burst Memory Read Multiple



Target 32-bit Burst Memory Write

Figure 11 represents a Target Memory Write transaction. In this transaction data is captured from the ADIO bus to a FIFO in the user design.

Address Phase

The Initiator drives FRAME# and the address onto the AD lines, and drives a Memory Write command (0111) onto C/BE#. The Initiator signals that it wants to transfer multiple Dwords by keeping FRAME# asserted and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture a copy of this address into the address counter.

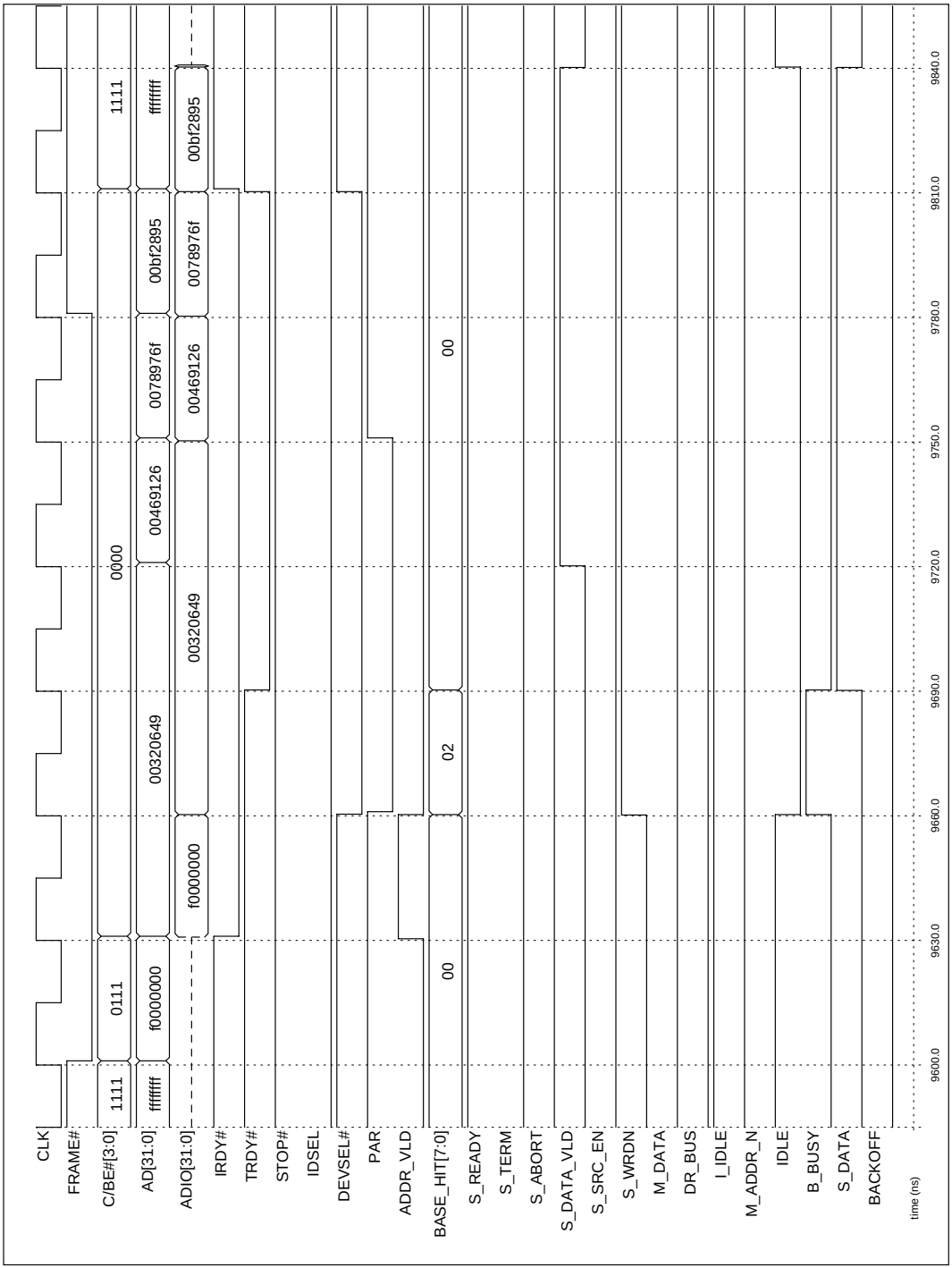
Data Phase

The PCI Interface sees that the bus is no longer idle and asserts Target state machine signals, IDLE and B_BUSY. After B_BUSY, the Target state machine enters the S_DATA state, indicating that the backend will begin transferring data. DEVSEL# is asserted as a medium decode speed. The PCI Interface always adds one cycle of TRDY# latency after DEVSEL# assertion.

BASE_HIT[x] is asserted for one cycle to indicate to the backend logic that it is the target of an access. The signal S_WRDN is asserted high to represent that data is being written to the target.

S_READY is asserted and S_TERM is deasserted, indicating that the backend is able to transfer more than one Dword.

Figure 11: Target 32-bit Burst Memory Write



Target 32-bit Burst Memory Write with Disconnect

Figure 12 represents a Target Memory Write transaction where the backend causes a disconnect with data termination. In this transaction data is captured from the ADIO bus to a FIFO in the user design. Disconnect with Data occurs when both S_READY and S_TERM are asserted.

Address Phase

The Initiator drives FRAME# and the address onto the AD lines, and drives a Memory Write command (0111) onto C/BE#. The Initiator signals that it wants to transfer multiple Dwords by keeping FRAME# asserted and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture a copy of this address into the address counter.

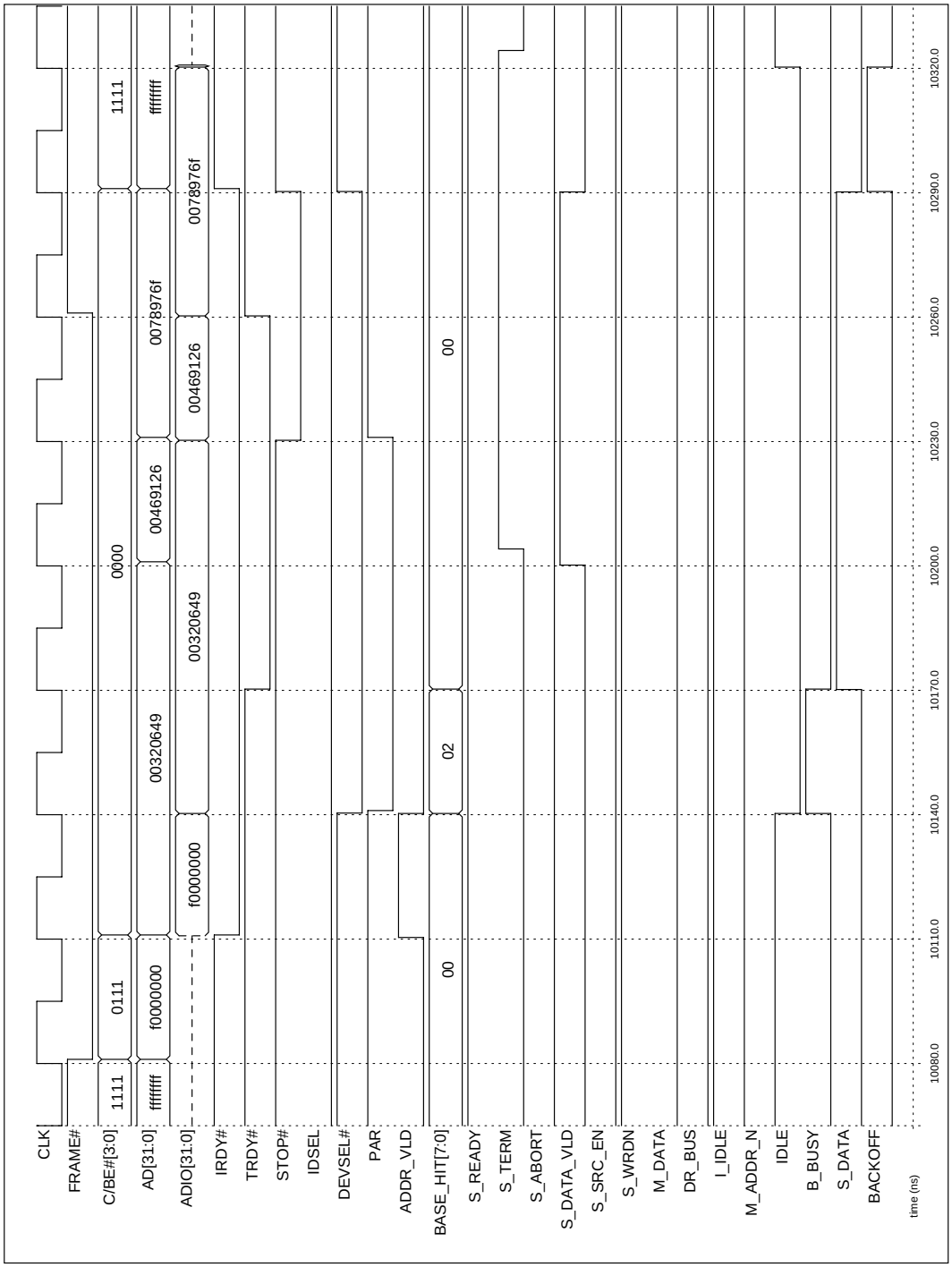
Data Phase

The PCI Interface sees that the bus is no longer idle and asserts Target state machine signals, IDLE and B_BUSY. After B_BUSY, the Target state machine enters the S_DATA state, indicating that the backend will begin transferring data. DEVSEL# is asserted as a medium decode speed. The PCI Interface always adds one cycle of TRDY# latency after DEVSEL# assertion.

BASE_HIT[x] is asserted for one cycle to indicate to the backend logic that it is the target of an access. The signal S_WRDN is asserted high to represent that data is being written to the target.

Initially, S_READY is asserted and S_TERM is deasserted, indicating that the backend is able to transfer more than one Dword. S_TERM is asserted with two more Dwords to transfer. The equation for S_TERM is usually derived from the Almost Full flag on the FIFO. One cycle later, STOP# is asserted, and the Initiator concludes the transaction by deasserting FRAME# and then IRDY# on the subsequent cycle. Only a single Dword is transferred after STOP# is asserted.

Figure 12: Target 32-bit Burst Memory Write with Disconnect



Target 32-bit Retry

Figure 13 represents a Target Retry. This is the same as a disconnect without data on first data phase. The Retry occurs when S_TERM is asserted while S_READY is low. Users may want to signal retrys when designing with very slow peripherals or if the user design implements delayed reads.

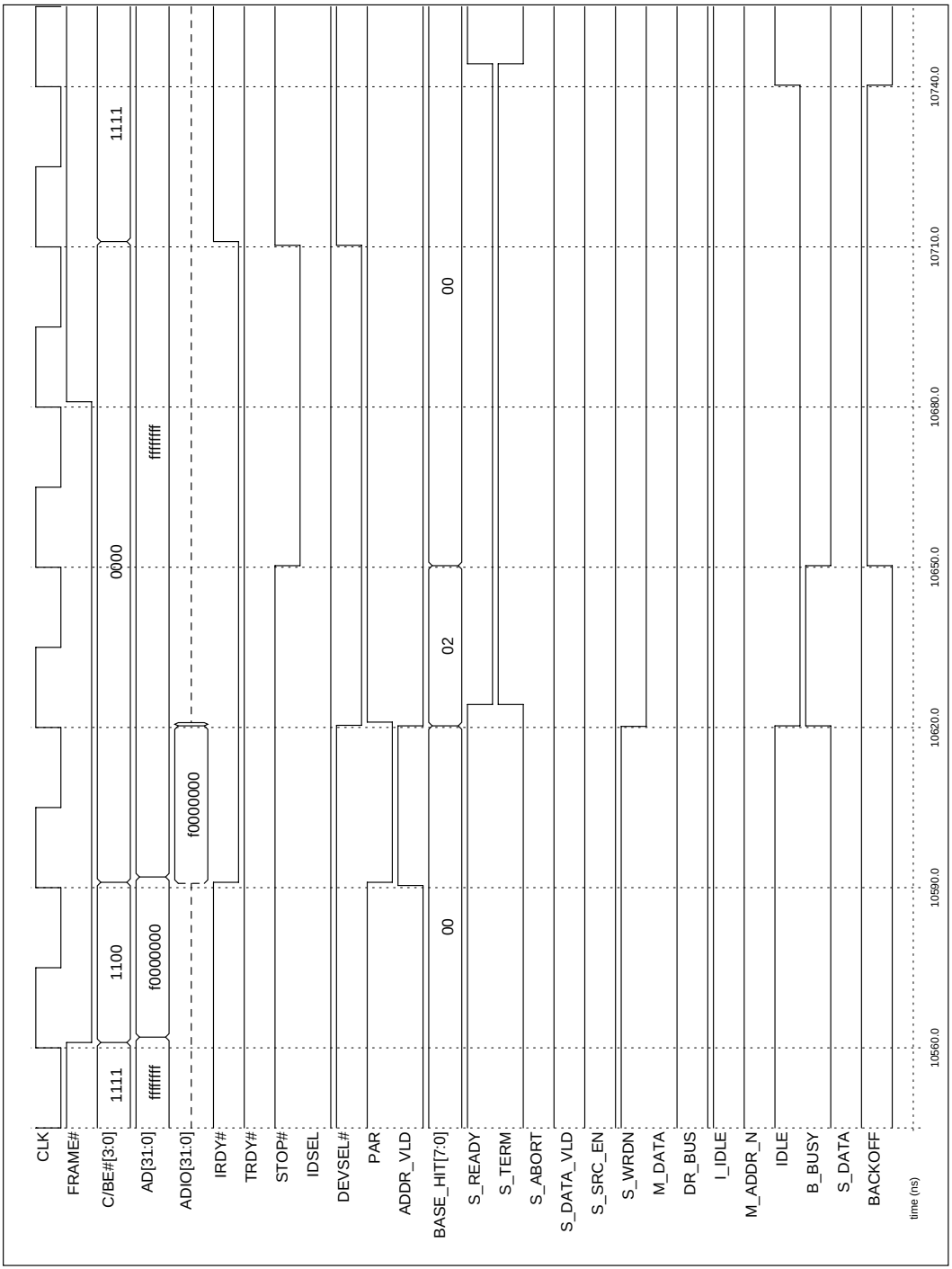
Address Phase

The Initiator drives FRAME# and the address onto the AD lines, and drives a Memory Read Multiple command (1100) onto C/BE#. The Initiator signals that it wants to transfer multiple Dwords by keeping FRAME# asserted and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture a copy of this address into the address counter.

Data Phase

The user design deasserts S_READY and asserts S_TERM, to cause a Retry. The Target state machine never enters the S_DATA state. TRDY# is never asserted and STOP# is asserted. The Initiator deasserts FRAME# and concludes the cycle.

Figure 13: Target 32-bit Retry



Target 32-bit Abort

Figure 14 the signalling of a Target abort. When S_ABORT is asserted, this signals a serious error condition and requires the current transaction to stop. The transaction starts normally as detailed below and end with a Target abort.

Address Phase

The Initiator drives FRAME# and the address onto the AD lines, and drives a Memory Read Multiple command (1100) onto C/BE#. The Initiator signals that it wants to transfer multiple Dwords by keeping FRAME# asserted and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture a copy of this address into the address counter.

Data Phase

The PCI Interface sees that the bus is no longer idle and asserts Target state machine signals, IDLE and B_BUSY. After B_BUSY, the Target state machine enters the S_DATA state, indicating that the backend will begin transferring data. DEVSEL# is asserted as a medium decode speed. The PCI Interface always adds one cycle of TRDY# latency after DEVSEL# assertion. Since this is a read, TRDY# is not asserted for one extra cycle beyond the usual TRDY# latency due to the Read transaction turnaround cycle where the Initiator stops driving AD and the Target begins driving AD.

BASE_HIT[x] is asserted for one cycle to indicate to the backend logic that it is the target of an access. The signal S_WRDN is asserted low to represent data is being read from the Target.

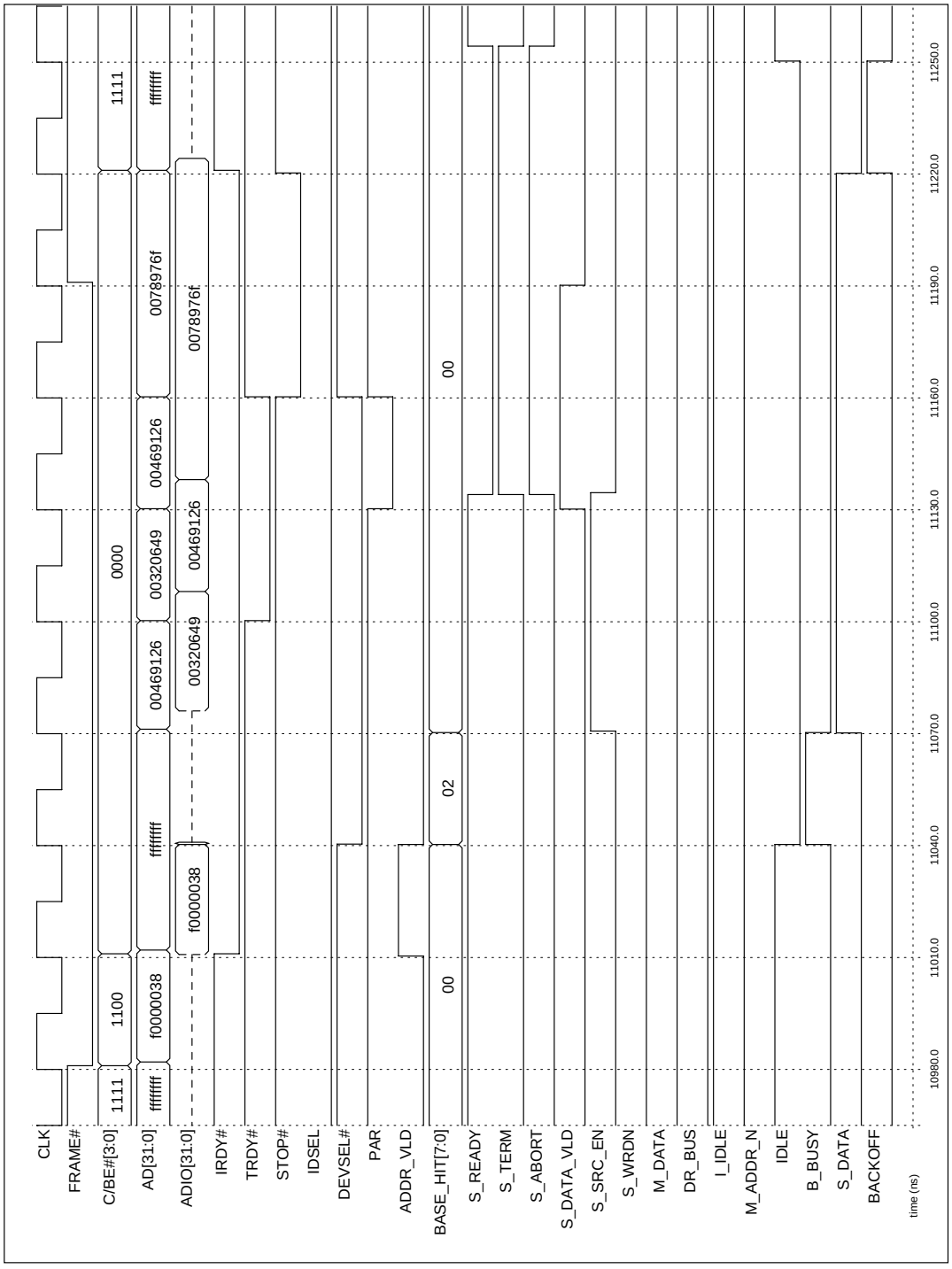
S_READY is asserted and S_TERM is deasserted, indicating that the backend is able to transfer more than one Dword.

The backend design detected a serious error condition, such as attempting to burst past the end of the memory space for that Base address register. When this occurs, a Target abort must be signalled to the Initiator.

On the rising CLK edge after S_ABORT is asserted, STOP# is asserted and both TRDY# and DEVSEL# are deasserted. As a result, FRAME# is deasserted on the next CLK cycle which ends the transaction.

The states of S_READY and S_TERM are not particularly relevant since asserting S_ABORT causes the Target state machine to do a Target abort.

Figure 14: Target 32-bit Abort



Initiator 64-bit Burst Memory Read Multiple

Figure 15 represents a 64-bit burst cycle Memory Read Multiple transaction. This consists of a single address phase followed by two or more data phases.

Requesting the PCI Bus

REQUEST64 is asserted for one clock cycle to generate a REQ# signal to the PCI Arbiter and to signal to the PCI Interface that a 64-bit transaction will be performed. The PCI Interface asserts FRAME# and REQ64# only if GNT# is asserted for more than one clock cycle.

Address Phase

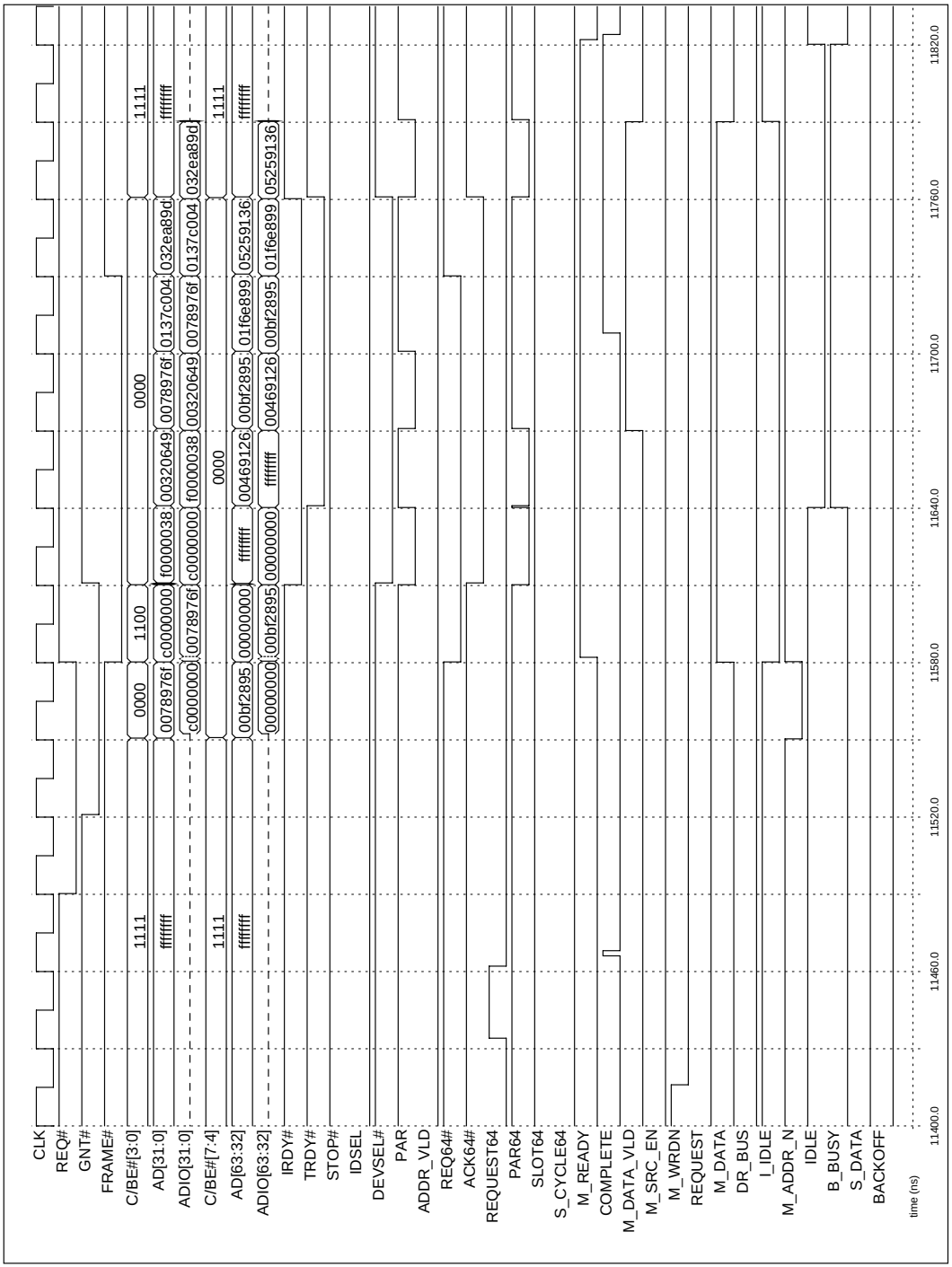
The M_ADDR_N signal is used to drive the output-enable of the 3-state buffers (BUFTs) which enable the valid address onto the internal ADIO bus one CLK cycle before the assertion of FRAME#. Once the Initiator asserts FRAME#, it drives the AD lines with the address put on the ADIO bus during the previous CLK cycle, and drives a Memory Read Multiple command (1100), supplied by the user to the backend (M_CBE[3:0], not shown), onto the C/BE# lines.

Data Phase

The M_DATA phase begins on the CLK cycle immediately following the M_ADDR_N phase. During M_DATA, the byte enables are presented on M_CBE. One clock cycle after FRAME# and REQ64# are asserted, the Initiator asserts IRDY#, and the fast decode, 64-bit Target asserts DEVSEL# and ACK64#. Since this is a Read, there is a one cycle bus turnaround after the address phase; TRDY# can not be asserted until after the turnaround cycle. The Target is now suppling data onto the PCI bus, while the Initiator is supplying the Byte enables. Since this is a multi-Qword transfer, COMPLETE is initially deasserted and M_READY is asserted at the beginning of the transaction, and FRAME# and REQ64# stay asserted until the next to last data phase.

M_DATA_VLD indicates the data is present on the AD bus and is used to advance the Initiator address pointer. With both IRDY# and TRDY# asserted, the 64-bit data transfer occurs without inserted wait states from either agent. COMPLETE is asserted on the next to last data phase, since the Initiator has to deassert FRAME# and REQ64# at the beginning of the last data phase. The transfer completes with the deassertion of IRDY#, TRDY#, DEVSEL#, and ACK64# and the bus has a turnaround cycle. The signal M_WRDN is asserted low to represent data is being read.

Figure 15: Initiator 64-bit Burst Memory Read Multiple



Initiator 64-bit Burst Memory Write

Figure 16 represents a 64-bit burst cycle Memory Write transaction. This consists of a single address phase followed by four data phases.

Requesting the PCI Bus

REQUEST64 is asserted for one clock cycle to generate a REQ# signal to the PCI Arbiter and to signal to the PCI Interface that a 64-bit transaction will be performed. The PCI Interface asserts FRAME# and REQ64# only if GNT# is asserted for more than one clock cycle.

Address Phase

The M_ADDR_N signal is used to drive the output-enable of the 3-state buffers (BUFTs) which enable the valid address onto the internal ADIO bus one CLK cycle before the assertion of FRAME#. Once the Initiator asserts FRAME#, it drives the AD lines with the address put on the ADIO bus during the previous CLK cycle, and drives a Memory Write command (0111), supplied by the user to the backend (M_CBE[3:0], not shown), onto the C/BE# lines.

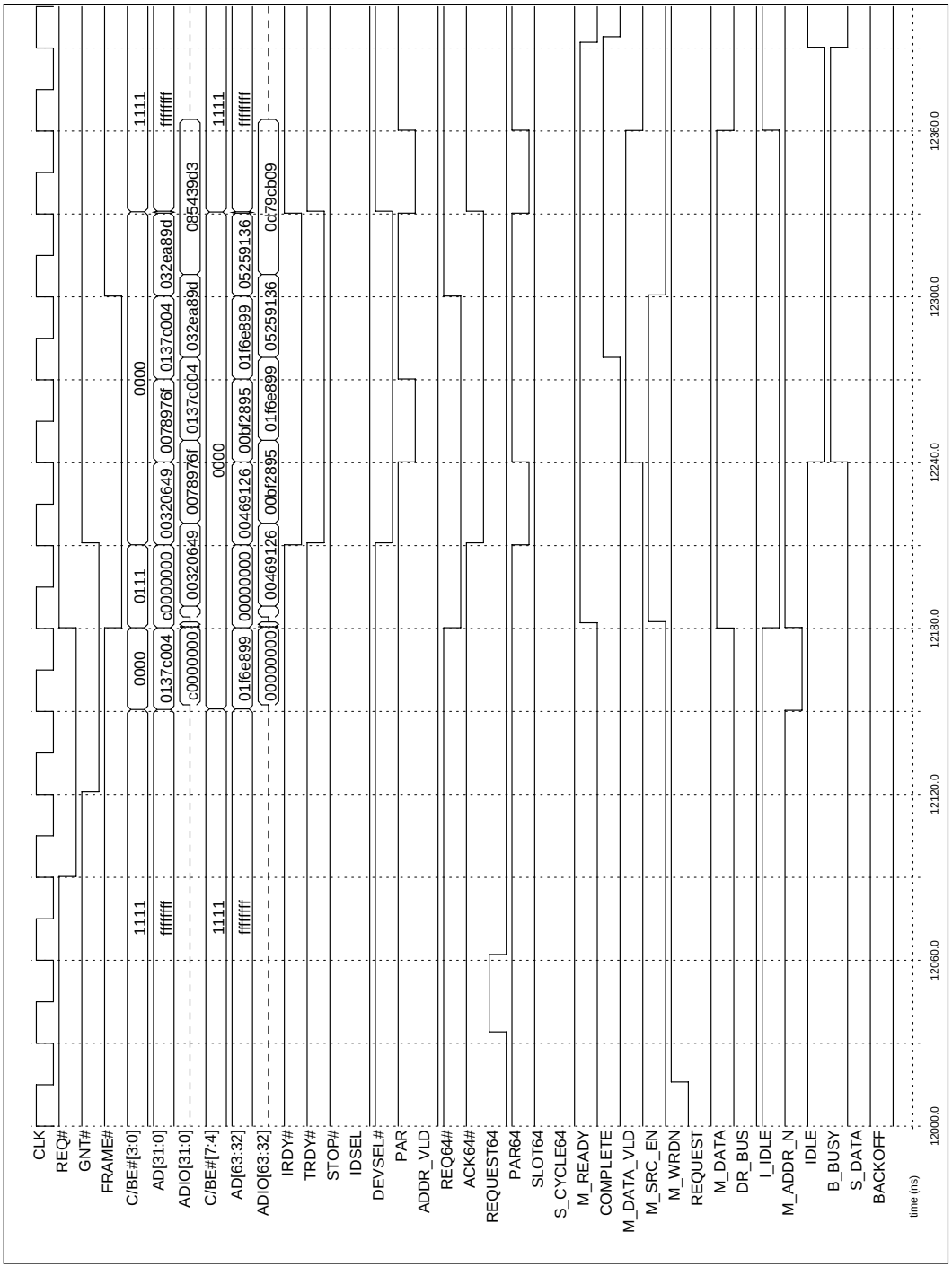
Data Phase

The M_DATA phase begins on the CLK cycle immediately following the M_ADDR_N phase. During M_DATA, the byte enables are presented on M_CBE. Data is placed on the ADIO bus from a FIFO in the user design. The output enable of the FIFO is controlled by M_DATA (conditional upon being the write state!).

One clock cycle after FRAME# is asserted, the Initiator asserts IRDY#, and the fast decode, 64-bit Target asserts DEVSEL#, ACK64#, and TRDY#. Since this is a multi-Qword transfer, COMPLETE is initially deasserted and M_READY is asserted at the beginning of the transaction. COMPLETE is asserted during the next to last data phase and FRAME# and REQ64# are deasserted at the beginning of the last data phase.

M_DATA_VLD indicates the data was taken by the Target and M_SRC_EN is used to advance the Initiator address pointer and read the next piece of data from the FIFO. With both IRDY# and TRDY# asserted, the data transfer occurs without inserted wait states from either agent. The transfer completes with the deassertion of IRDY#, TRDY#, DEVSEL#, and ACK64# and the bus has a turnaround cycle. The signal M_WRDN is asserted high to represent data is being written.

Figure 16: Initiator 64-bit Burst Memory Write



Initiator 64-bit Burst Memory Write with Disconnect

Figure 17 represents a 64-bit burst cycle Memory Write transaction with a Target termination. This consists of a single address phase followed by four data phases but only two Qwords were transferred because of the disconnect with data.

Requesting the PCI Bus

REQUEST64 is asserted for one clock cycle to generate a REQ# signal to the PCI Arbiter and to signal to the PCI Interface that a 64-bit transaction will be performed. The PCI Interface asserts FRAME# and REQ64# only if GNT# is asserted for more than one clock cycle.

Address Phase

The M_ADDR_N signal is used to drive the output-enable of the 3-state buffers (BUFTs) which enable the valid address onto the internal ADIO bus one CLK cycle before the assertion of FRAME#. Once the Initiator asserts FRAME#, it drives the AD lines with the address put on the ADIO bus during the previous CLK cycle, and drives a Memory Write command (0111), supplied by the user to the backend (M_CBE[3:0], not shown), onto the C/BE# lines.

Data Phase

The M_DATA phase begins on the CLK cycle immediately following the M_ADDR_N phase. During M_DATA, the byte enables are presented on M_CBE. Data is placed on the ADIO bus from a FIFO in the user design. The output enable of the FIFO is controlled by M_DATA (conditional upon being the write state!).

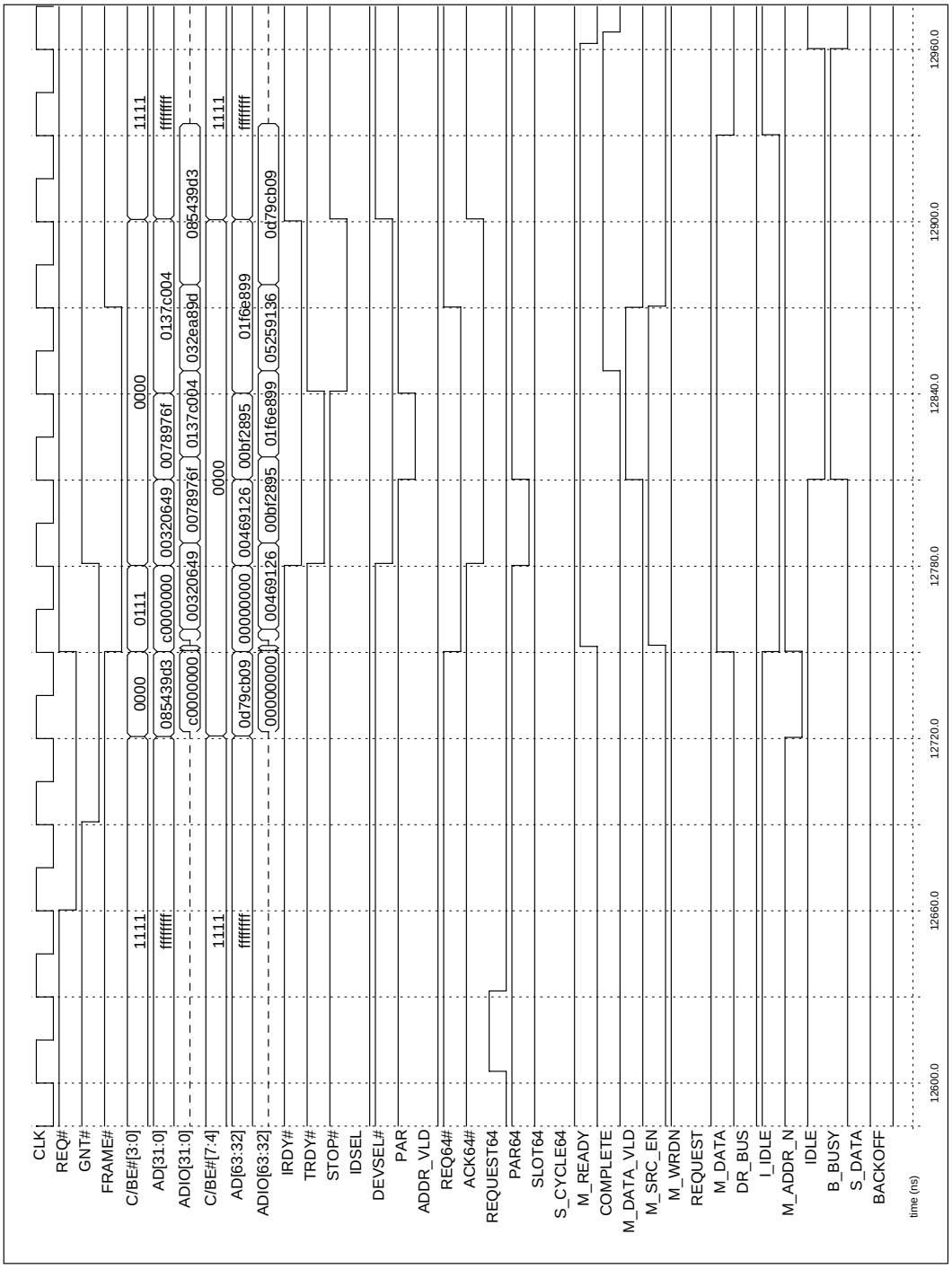
One clock cycle after FRAME# and REQ64# are asserted, the Initiator asserts IRDY#, and the fast decode Target asserts DEVSEL#, TRDY#, and ACK64#. Since this is a multi-Qword transfer, COMPLETE is initially deasserted and M_READY is asserted at the beginning of the transaction. COMPLETE is asserted during the next to last data phase and FRAME# and REQ64# are deasserted at the beginning of the last data phase.

M_DATA_VLD indicates the data was taken by the Target and M_SRC_EN is used to advance the Initiator address pointer and read the next piece of data from the FIFO. With both IRDY# and TRDY# asserted, the data transfer occurs without inserted wait states from either agent.

The Target signalled a disconnect by asserting STOP#, and deasserted TRDY# so that the third Qword was not transferred. Likewise, the last data phase does not have a data transfer because TRDY# is not asserted. M_SRC_EN is asserted for four cycles and M_DATA_VLD is asserted for only two cycles. Since there is a difference between the number of data transfers anticipated (four), and the number that occurred (two), the user design may have to perform a backup of the FIFO. Details on how to do this are documented in the Xilinx *PCI Design Guide*.

The transfer completes with the deassertion of IRDY#, STOP#, DEVSEL#, and ACK64# and the bus has a turnaround cycle. The signal M_WRDN is asserted high to represent data is being written.

Figure 17: Initiator 64-bit Burst Memory Write with Disconnect



Initiator 64-bit Memory Read of a 32-bit Target

Figure 18 represents a 64-bit burst cycle Memory Read Multiple transaction of a 32-bit Target. This consists of a single address phase followed by two or more data phases. The Xilinx PCI core handles this situation by transferring one Qword as two Dwords (assuming the Target will take both Dwords) and then ending the transaction.

Requesting the PCI Bus

REQUEST64 is asserted for one clock cycle to generate a REQ# signal to the PCI Arbiter and to signal to the PCI Interface that a 64-bit transaction will be performed. The PCI Interface asserts FRAME# and REQ64# only if GNT# is asserted for more than one clock cycle.

Address Phase

The M_ADDR_N signal is used to drive the output-enable of the 3-state buffers (BUFTs) which enable the valid address onto the internal ADIO bus one CLK cycle before the assertion of FRAME#. Once the Initiator asserts FRAME#, it drives the AD lines with the address put on the ADIO bus during the previous CLK cycle, and drives a Memory Read Multiple command (1100), supplied by the user to the backend (M_CBE[3:0], not shown), onto the C/BE# lines.

Data Phase

The M_DATA phase begins on the CLK cycle immediately following the M_ADDR_N phase. During M_DATA, the byte enables are presented on M_CBE. One clock cycle after FRAME# and REQ64# are asserted, the Initiator asserts IRDY#, and the fast decode, 32-bit Target asserts DEVSEL#, leaving ACK64# deasserted.

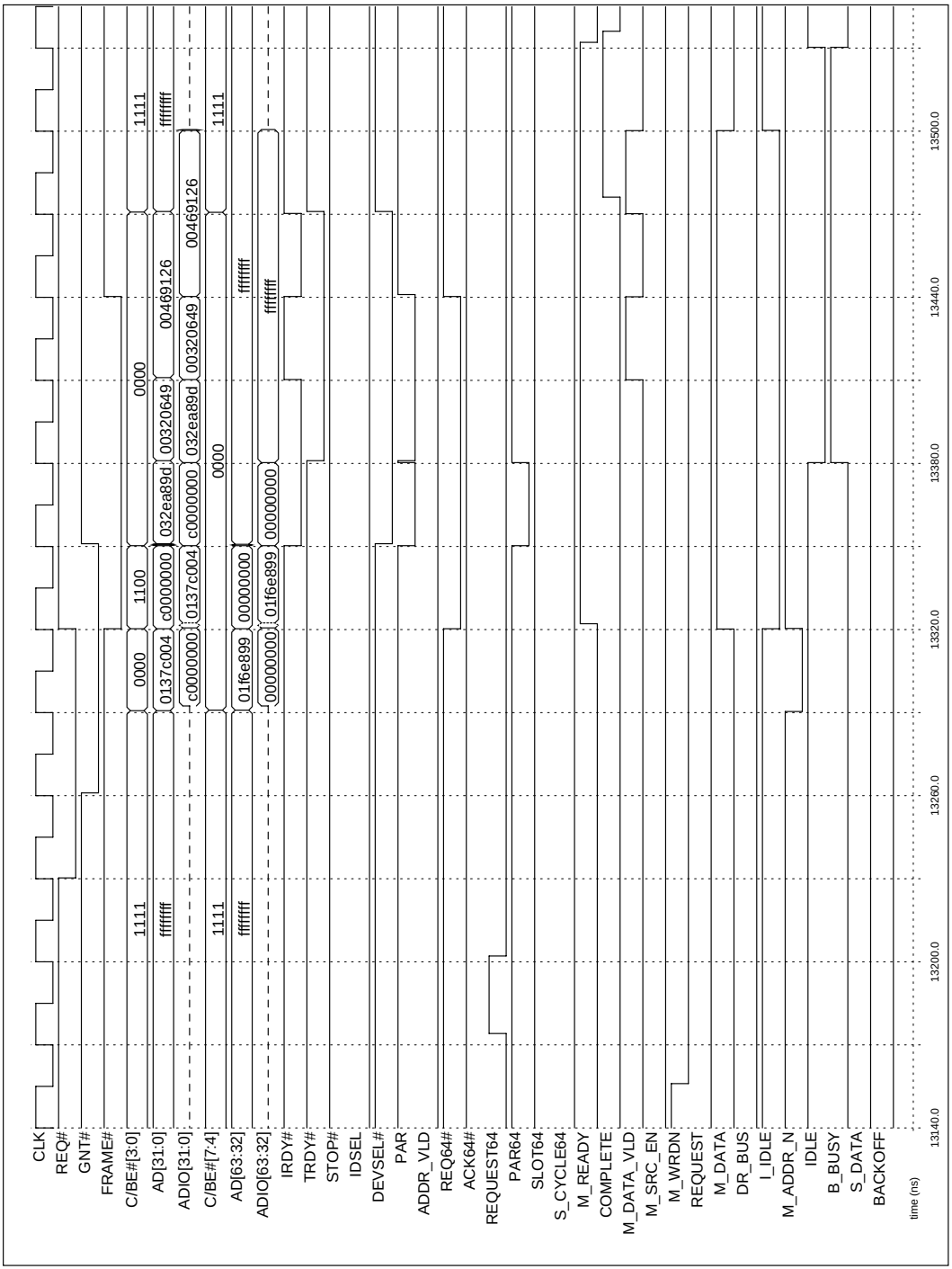
The Xilinx PCI Interface sees that the Target did not assert ACK64#, and after transferring the first Dword, the Initiator state machine automatically deasserts IRDY#, even though M_READY is asserted. The data on the upper ADIO and M_CBE bus is transferred to the lower AD and M_CBE bus inside the PCI core. No extra muxes or back-end inputs are needed. When the PCI core is being used in zero wait state mode, a 64-bit transfer encountering a 32-bit Target is the only instance where a wait state is ever inserted after IRDY# is asserted.

The PCI Interface also asserts M_FAIL64 (not shown) to the user backend. M_FAIL64 is used to indicate that the 64-bit Target request was claimed as a 32-bit transfer by the Target.

M_DATA_VLD indicates the data is present on the AD bus. Since a automatic wait state was inserted, M_DATA_VLD is toggled. The behavior of M_DATA_VLD combined with the state of M_FAIL64 indicates two Dwords are transferred. M_DATA_VLD and M_FAIL64 are used to advance the Initiator address pointer.

When an event like this occurs, the Initiator should re-initiate as a 32-bit transfer and move the data as Dwords. The XIlinx PCI core will not allow you to burst more than two Dwords in this type of event. The user backend design must keep track of the number of Dwords transferred.

Figure 18: Initiator 64-bit Memory Read of a 32-bit Target



Initiator 64-bit Memory Write of a 32-bit Target

Figure 19 represents a single cycle Memory Write transaction from a 64-bit master to a 32-bit Target. This consists of a single address phase followed by two or more data phases. The Xilinx PCI core handles this situation by transferring one Qword as two Dwords (assuming the Target will take both Dwords) and then ending the transaction.

Requesting the PCI Bus

REQUEST64 is asserted for one clock cycle to generate a REQ# signal to the PCI Arbiter and to signal to the PCI Interface that a 64-bit transaction will be performed. The PCI Interface asserts FRAME# and REQ64# only if GNT# is asserted for more than one clock cycle.

Address Phase

The M_ADDR_N signal is used to drive the output-enable of the 3-state buffers (BUFTs) which enable the valid address onto the internal ADIO bus one CLK cycle before the assertion of FRAME#. Once the Initiator asserts FRAME#, it drives the AD lines with the address put on the ADIO bus during the previous CLK cycle, and drives a Memory Write command (0111), supplied by the user to the backend (M_CBE[3:0], not shown), onto the C/BE# lines.

Data Phase

The M_DATA phase begins on the CLK cycle immediately following the M_ADDR_N phase. During M_DATA, the byte enables are presented on M_CBE. One clock cycle after FRAME# and REQ64# are asserted, the Initiator asserts IRDY#, and the fast decode, 32-bit Target asserts DEVSEL#, leaving ACK64# deasserted.

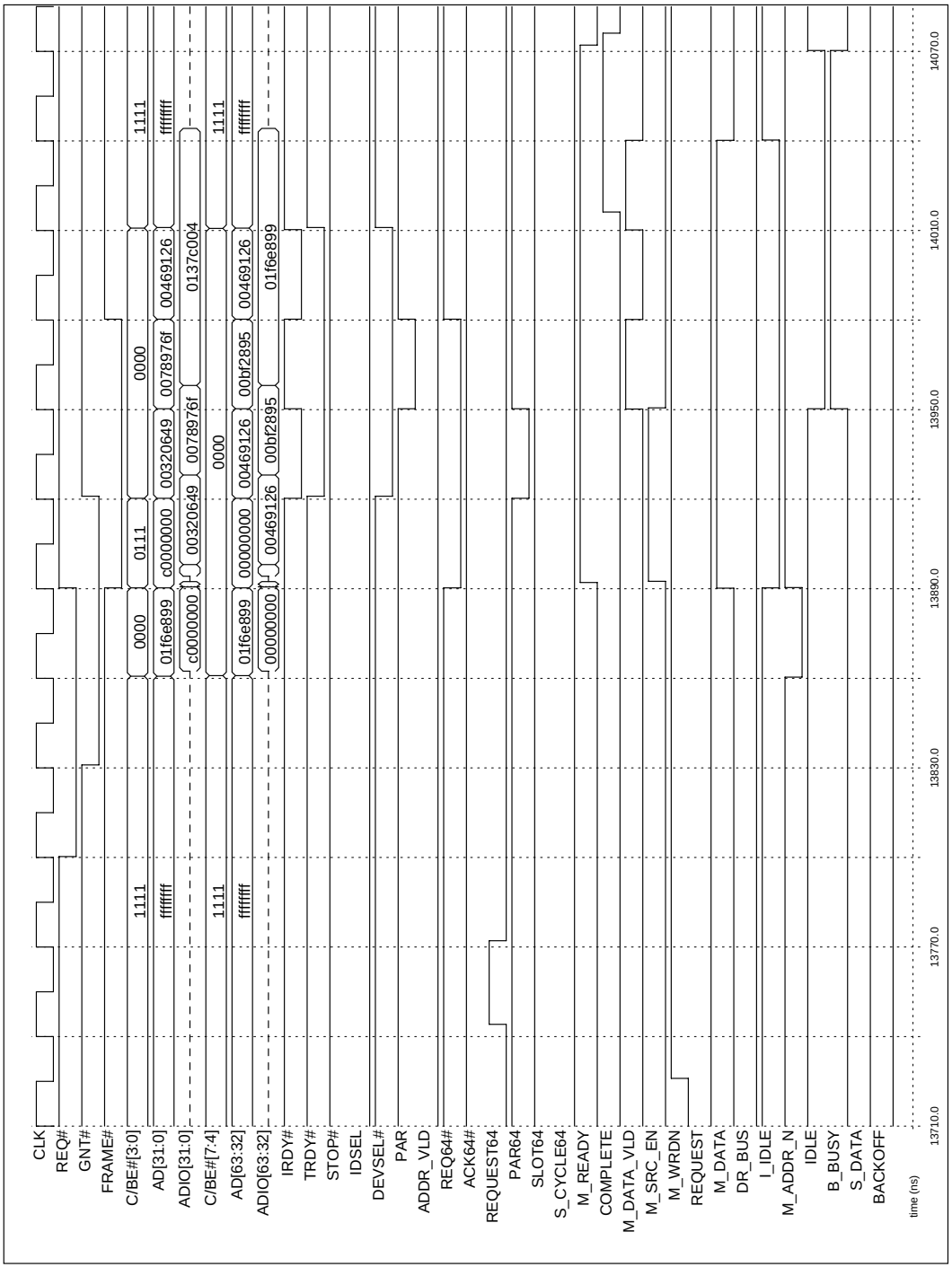
The Xilinx PCI Interface sees that the Target did not assert ACK64#, and after transferring the first Dword, the Initiator state machine automatically deasserts IRDY#, even though M_READY is asserted. The data on the upper ADIO and M_CBE bus is transferred to the lower AD and M_CBE bus inside the PCI core. No extra muxes or backend inputs are needed. When the PCI core is being used in zero wait state mode, a 64-bit transfer encountering a 32-bit Target is the only instance where a wait state is ever inserted after IRDY# is asserted.

The PCI Interface also asserts M_FAIL64 (not shown) to the user backend. M_FAIL64 is used to indicate that the 64-bit Target request was claimed as a 32-bit transfer by the Target.

M_DATA_VLD indicates that data was taken by the Target. M_SRC_EN combined with M_FAIL64 are used to advance the Initiator address pointer and read the next piece of data from the FIFO. In this case, since the PCI Interface asserts M_FAIL and M_SRC_EN, the user FIFO should backup the FIFO data to prevent data loss. This is because two Dwords were transferred, but the FIFO was advanced two Qwords.

When an event like this occurs, the Initiator should re-initiate as a 32-bit transfer and move the data as Dwords. The XIlinx PCI core will not allow you to burst more than two Dwords in this type of event. The user backend design must keep track of the number of Dwords transferred.

Figure 19: Initiator 64-bit Memory Write of a 32-bit Target



Target 64-bit Burst Memory Read Multiple

Figure 20 represents a Target Memory Read Multiple transaction. In this transaction, data is placed on the ADIO bus from a FIFO in the user design.

Address Phase

The Initiator drives FRAME#, REQ64#, and the address onto the AD lines, and drives a Memory Read Multiple command (1100) onto C/BE#. The Initiator signals that it wants to transfer multiple Dwords by keeping FRAME# asserted and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture a copy of this address into the address counter.

Data Phase

The PCI Interface sees that the bus is no longer idle and asserts Target state machine signals, IDLE and B_BUSY. After B_BUSY, the Target state machine enters the S_DATA state, indicating that the backend will begin transferring data. DEVSEL# is asserted as a medium decode speed, along with ACK64#, claiming the 64-bit transaction. The PCI Interface always adds one cycle of TRDY# latency after DEVSEL# assertion. Since this is a read, TRDY# is not asserted for one extra cycle beyond the usual TRDY# latency. This is due to the Read transaction turnaround cycle where the Initiator stops driving the AD bus and the Target begins driving the AD bus.

BASE_HIT[x] is asserted for one cycle when DEVSEL# is asserted to indicate to the backend logic that it is the target of an access. The signal S_WRDN is asserted low to represent data is being read from the Target.

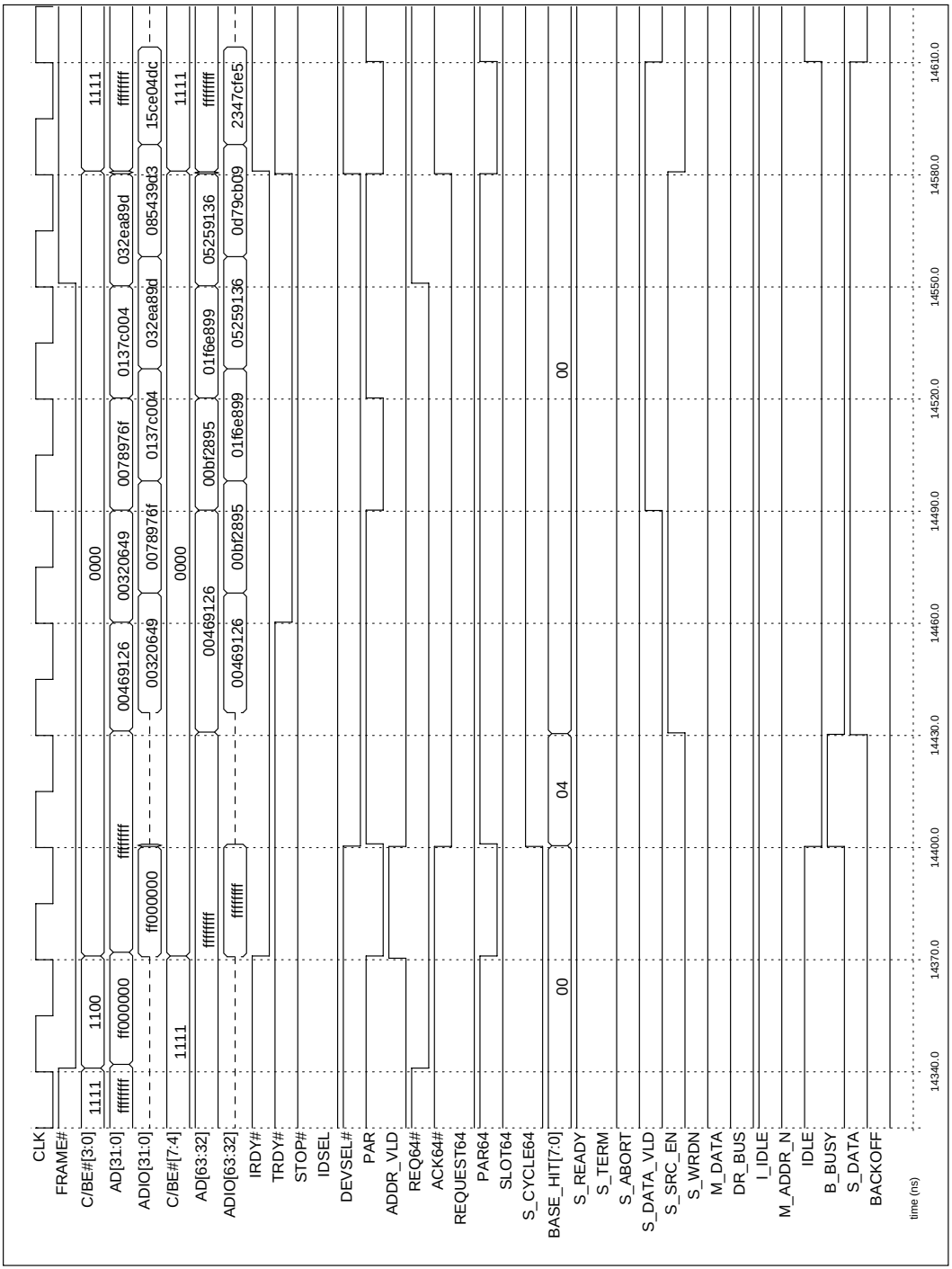
The PCI Interface also asserts S_CYCLE64 starting at the assertion of DEVSEL# and it remains asserted throughout the transfer. This indicates to the user backend that this is a 64-bit transfer.

S_READY is asserted and S_TERM is deasserted, indicating that the backend is able to transfer more than one Dword.

The PCI interface asserts S_SRC_EN to indicate to the user backend that it must supply the next piece of data. The PCI interface also asserts S_DATA_VLD to indicate to the user backend that on the previous clock, data was accepted by the Initiator.

In this example, S_SRC_EN is asserted for five cycles and S_DATA_VLD is asserted for four cycles. Since there is a difference between the number of data transfers anticipated (five), and the number that occurred (four), the user design may have to perform a backup of the FIFO. Details on how to do this are documented in the Xilinx *PCI Design Guide*.

Figure 20: Target 64-bit Burst Memory Read Multiple



Target 64-bit Burst Memory Write

Figure 21 represents a Target Memory Write transaction. In this transaction data is captured from the ADIO bus to a FIFO in the user design.

Address Phase

The Initiator drives FRAME#, REQ64#, and the address onto the AD lines, and drives a Memory Write command (0111) onto C/BE#. The Initiator signals that it wants to transfer multiple Qwords by keeping FRAME# asserted and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture a copy of this address into the address counter.

Data Phase

The PCI Interface sees that the bus is no longer idle and asserts Target state machine signals, IDLE and B_BUSY. After B_BUSY, the Target state machine enters the S_DATA state, indicating that the backend will begin transferring data. DEVSEL# is asserted as a medium decode speed, along with ACK64#, claiming the 64-bit transaction. The PCI Interface always adds one cycle of TRDY# latency after DEVSEL# assertion.

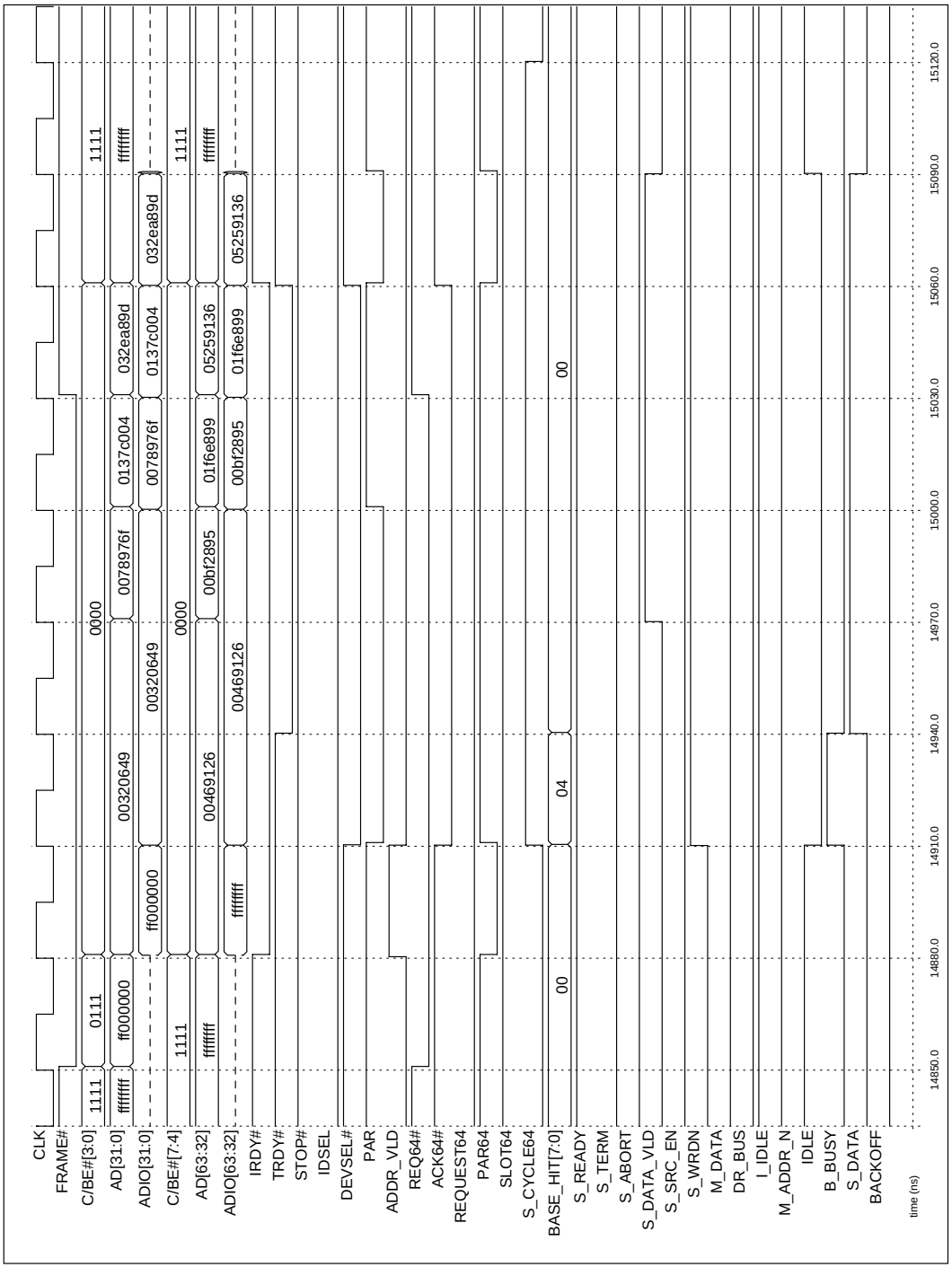
BASE_HIT[x] is asserted for one cycle when DEVSEL# is asserted to indicate to the backend logic that it is the target of an access. The signal S_WRDN is asserted high to represent that data is being written to the target.

The PCI Interface also asserts S_CYCLE64 starting at the assertion of DEVSEL# and it remains asserted throughout the transfer. This indicates to the user backend that this is a 64-bit transfer.

S_READY is asserted and S_TERM is deasserted, indicating that the backend is able to transfer more than one Qword.

The PCI Interface also asserts S_DATA_VLD to indicate to the user backend that valid data is present on the internal ADIO bus. This signal is used by the backend to capture the data into the FIFO.

Figure 21: Target 64-bit Burst Memory Write



Target 64-bit Burst Memory Write with Disconnect

Figure 22 represents a Target Memory Write transaction where the backend causes a disconnect with data termination. In this transaction data is captured from the ADIO bus to a FIFO in the user design.

Address Phase

The Initiator drives FRAME#, REQ64#, and the address onto the AD lines, and drives a Memory Write command (0111) onto C/BE#. The Initiator signals that it wants to transfer multiple Qwords by keeping FRAME# asserted and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture a copy of this address into the address counter.

Data Phase

The PCI Interface sees that the bus is no longer idle and asserts Target state machine signals, IDLE and B_BUSY. After B_BUSY, the Target state machine enters the S_DATA state, indicating that the backend will begin transferring data. DEVSEL# is asserted as a medium decode speed, along with ACK64#, claiming the 64-bit transaction.

The PCI Interface always adds one cycle of TRDY# latency after DEVSEL# assertion.

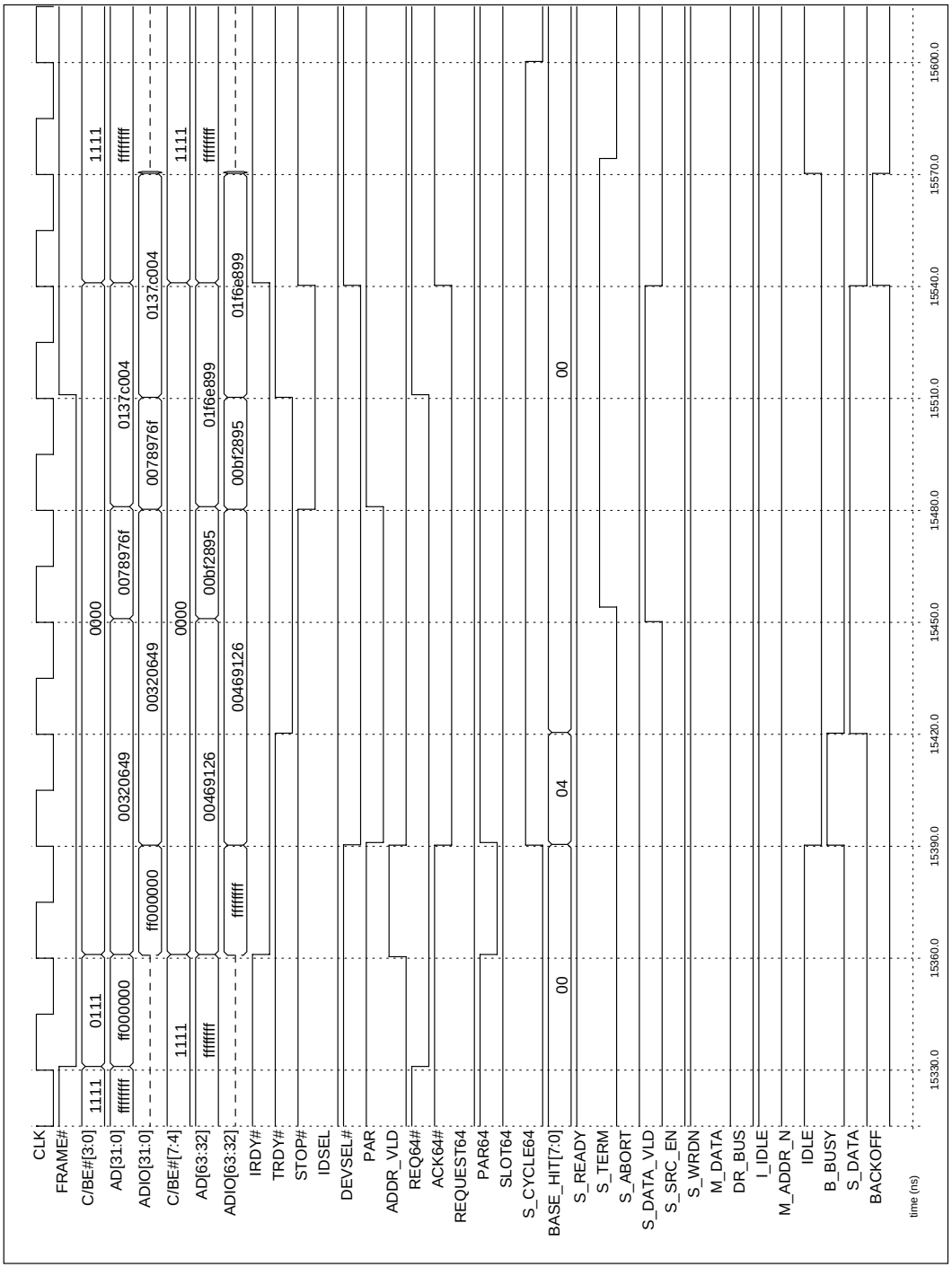
BASE_HIT[x] is asserted for one cycle when DEVSEL# is asserted to indicate to the backend logic that it is the target of an access. The signal S_WRDN is asserted high to represent that data is being written to the target.

The PCI Interface also asserts S_CYCLE64 starting at the assertion of DEVSEL# and it remains asserted throughout the transfer. This indicates to the user backend that this is a 64-bit transfer.

The PCI Interface asserts S_DATA_VLD to indicate to the user backend that valid data is present on the internal ADIO bus. This signal is used by the backend to capture the data into the FIFO.

Initially, S_READY is asserted and S_TERM is deasserted, indicating that the backend is able to transfer more than one Qword. S_TERM is asserted with two more Qwords to transfer. The equation for S_TERM is usually derived from the Almost Full flag on the FIFO. One cycle later, STOP# is asserted, and the Initiator concludes the transaction by deasserting FRAME# and then IRDY# on the subsequent cycle.

Figure 22: Target 64-bit Burst Memory Write with Disconnect



Target 64-bit Retry

Figure 23 represents a 64-bit Target Retry. This is the same as a disconnect without data on first data phase. The Retry occurs when S_TERM is asserted while S_READY is low. Users may want to signal retrys when designing with very slow peripherals or if the user design implements delayed reads.

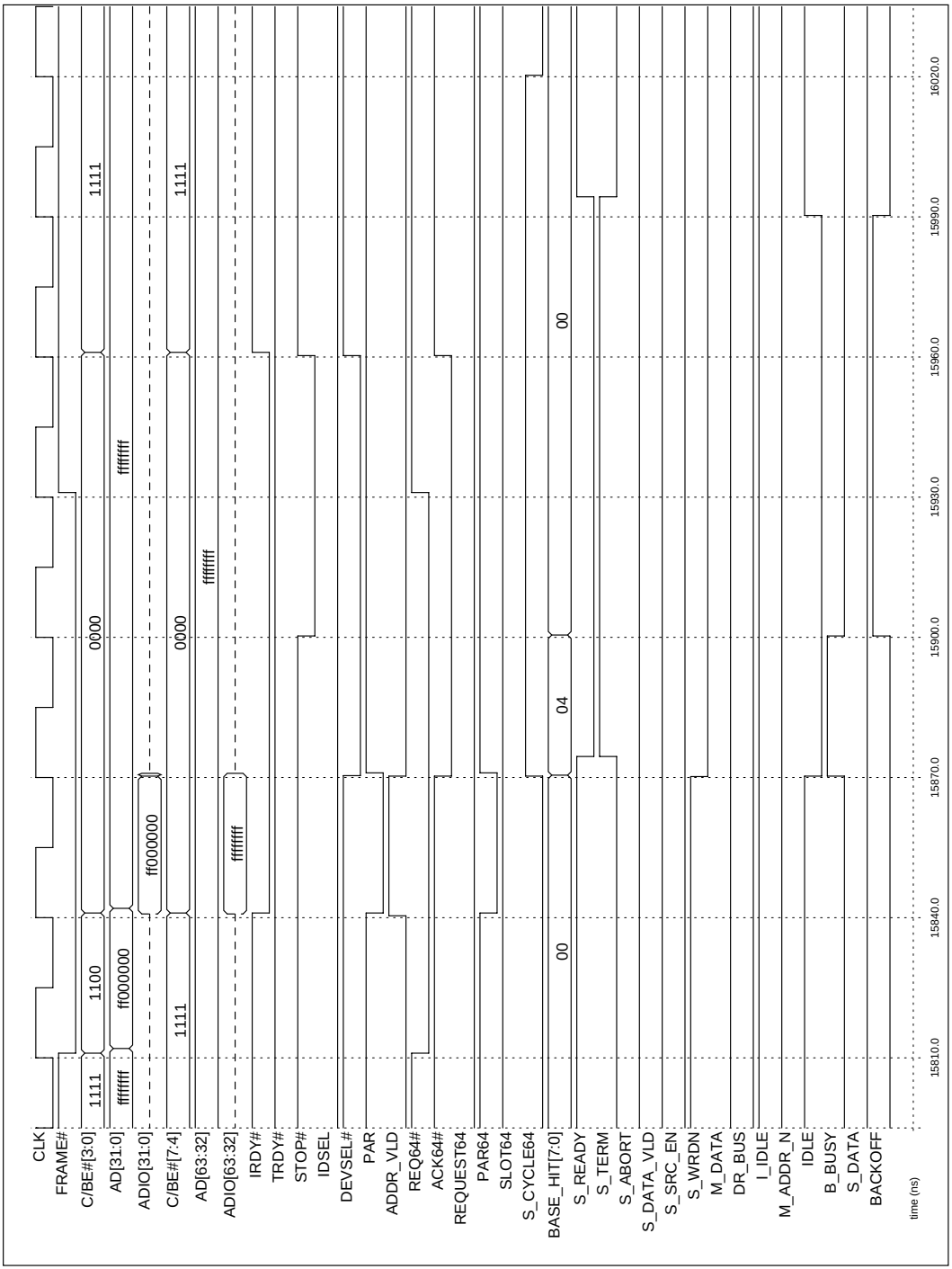
Address Phase

The Initiator drives FRAME#, REQ64#, and the address onto the AD lines, and drives a Memory Read Multiple command (1100) onto C/BE#. The Initiator signals that it wants to transfer multiple Qwords by keeping FRAME# asserted and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture a copy of this address into the address counter.

Data Phase

Based on a decode of the BASE_HIT[x] signal, the user design deasserts S_READY and asserts S_TERM, which will cause a Retry. The Target state machine never enters the S_DATA state. TRDY# is never asserted and STOP# is asserted. The Initiator deasserts FRAME# and concludes the cycle.

Figure 23: Target 64-bit Retry



Target 64-bit Abort

Figure 24 the signalling of a Target abort. When S_ABORT is asserted, this signals a serious error condition and requires the current transaction to stop. The transaction starts normally as detailed below and end with a Target abort.

Address Phase

The Initiator drives FRAME#, REQ64#, and the address onto the AD lines, and drives a Memory Read Multiple command (1100) onto C/BE#. The Initiator signals that it wants to transfer multiple Qwords by keeping FRAME# asserted and asserting IRDY# on the next cycle. ADDR_VLD is asserted one cycle after the address phase, indicating that a valid address is present on ADIO. The ADDR_VLD is used by the user design to capture a copy of this address into the address counter.

Data Phase

The PCI Interface sees that the bus is no longer idle and asserts Target state machine signals, IDLE and B_BUSY. After B_BUSY, the Target state machine enters the S_DATA state, indicating that the backend will begin transferring data. DEVSEL# is asserted as a medium decode speed. The PCI Interface always adds one cycle of TRDY# latency after DEVSEL# assertion. Since this is a read, TRDY# is not asserted for one extra cycle beyond the usual TRDY# latency due to the Read transaction turnaround cycle where the Initiator stops driving the AD bus and the Target begins driving the AD bus.

BASE_HIT[x] is asserted for one cycle to indicate to the backend logic that it is the target of an access. The signal S_WRDN is asserted low to represent data is being read from the Target.

The PCI Interface also asserts S_CYCLE64 starting at the assertion of DEVSEL# and it remains asserted throughout the transfer. This indicates to the user backend that this is a 64-bit transfer.

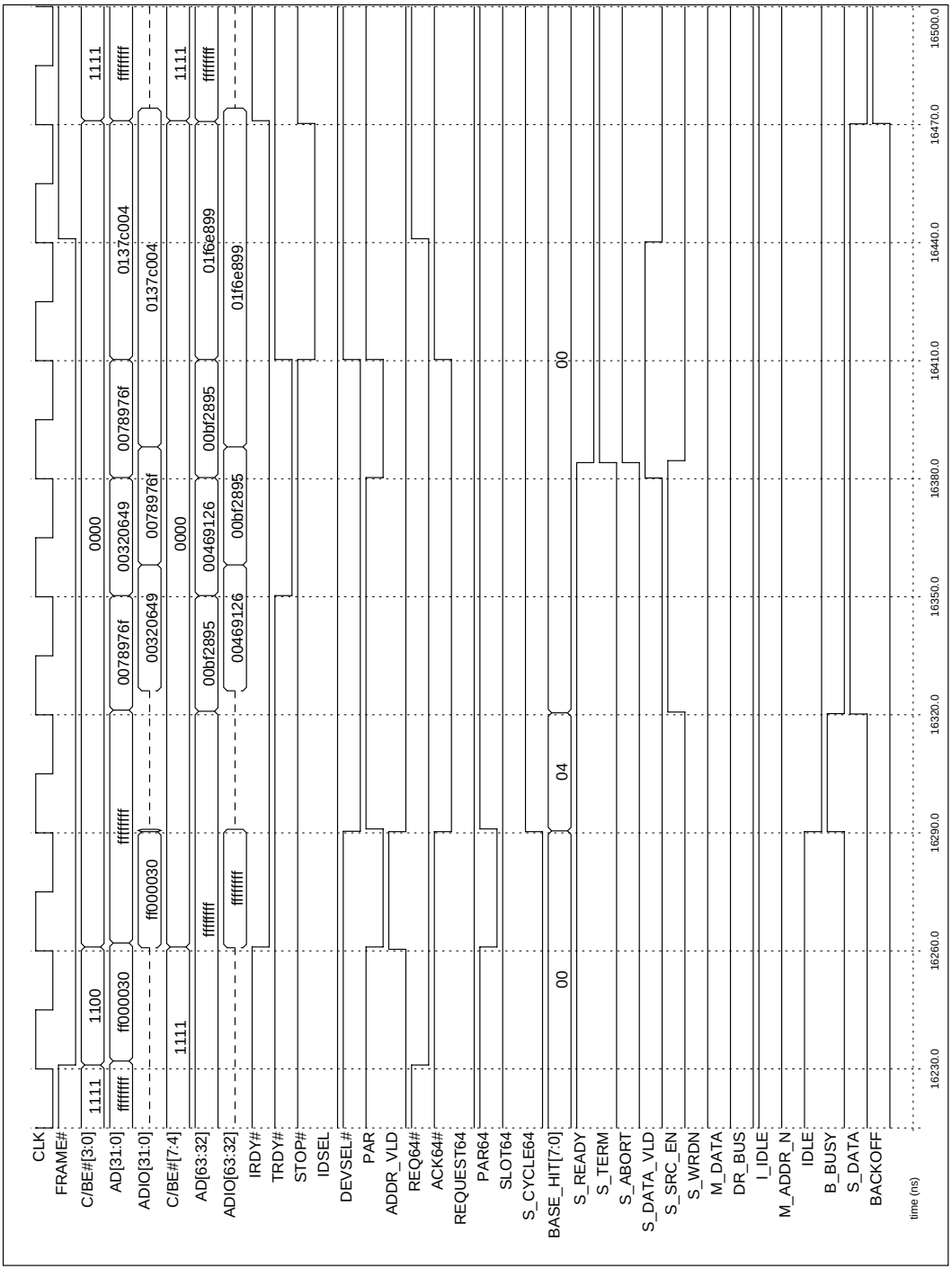
S_READY is asserted and S_TERM is deasserted, indicating that the backend is able to transfer more than one Qword.

The backend design detected a serious error condition, such as attempting to burst past the end of the memory space for that Base address register. When this occurs, a Target abort must be signalled to the Initiator.

On the rising CLK edge after S_ABORT is asserted, STOP# is asserted and TRDY# with DEVSEL# are deasserted. As a result, FRAME# is deasserted on the next CLK cycle which ends the transaction.

The states of S_READY and S_TERM are not particularly relevant since asserting S_ABORT causes the Target state machine to do a Target abort.

Figure 24: Target 64-bit Abort



Ordering Information and License Agreement

Summary

Xilinx provides a complete solution for development of fully compliant PCI and *CompactPCI* products. This chapter describes the various product packages. See Table 1 for a quick overview. Please refer to the individual data sheets for details. Call your local sales office for the latest availability and pricing information.

Xilinx PCI64 Design Kit

Part no: DO-DI-PCI64-DK

Overview

The LogiCORE PCI64 Virtex interface allows the user to rapidly implement 64-bit PCI Interfaces. By forming a partnership with Nallatech Limited, and Compuware NuMega, formerly Vireo Software, our customers will always have access to the leading industry expertise.

Package Includes

See individual data sheets for details

- LogiCORE PCI64 Virtex Interface
  - Master and Slave (Initiator/Target) functionality
  - Configurable and downloadable from Xilinx web
  - CD available upon request only
- LogiCORE PCI32 4000 Interface
  - Master and Slave (Initiator/Target) functionality
  - Configurable and downloadable from Xilinx web
  - CD available upon request only
- LogiCORE PCI32 Spartan Interface
  - Master and Slave (Initiator/Target) functionality
  - Configurable and downloadable from Xilinx web
  - CD available upon request only
- Synthesizable PCI Bridge Design Examples
  - VHDL and Verilog source files
  - On-line Design Examples User's Guide
- LogiCORE PCI Design Guide
- LogiCORE PCI Implementation Guide
- PCI System Architecture (Reference Book from MindShare, Inc.)
- Nallatech PCI64 Prototyping System
  - 64-bit, 66 MHz PCI Virtex prototyping board
  - Example drivers (NuMega)
  - PCI64 User's Guide
  - Demo software
- NuMega DriverWorks Development tools for Windows NT/98 drivers
  - Fully functional development tools
  - Development and prototyping license
  - Unrestricted license required for production and is available from NuMega
  - Technical support provided by NuMega only
- NuMega VtoolsD Development tools for Windows 95/98

Table 1: Xilinx PCI Product Line

Feature See individual data sheets for details	PCI64 Design Kit DO-DI-PCI64-DK	PCI64 Virtex DO-DI-PCI64	PCI32 Design Kit DO-DI-PCI32-DK	PCI32 Spartan DO-DI-PCI32-S
64-bit, 66 MHz PCI Initiator & Target	✔	✔		
64-bit, 33 MHz PCI Initiator & Target	✔	✔		
32-bit, 66 MHz PCI Initiator & Target	✔	✔		
32-bit, 33MHz PCI Initiator & Target	✔	✔	✔	✔
Support for Xilinx Virtex FPGAs	✔	✔		
Support for Xilinx Spartan FPGAs	✔	✔	✔	✔
Support for Xilinx XC4000 FPGAs	✔	✔	✔	
PCI Bridge Design Examples	All	All	32 bit only	SB03
Configuration and Download from web	✔	✔	✔	✔
LogiCORE User's Guide	✔	✔	✔	On-line
PCI System Architecture Book	✔	✔	✔	
Nallatech PCI64 Prototyping Board	✔		✔	
VCC HotPCI Prototyping System			✔	
Example Reference Drivers	✔		✔	
NuMega DriverWorks	✔		✔	
NuMega VtoolsD	✔		✔	
12 month Maintenance Contract	✔	✔	✔	
Free updates	12 months	12 months	12 months	3 months

and 3.x drivers
- Fully functional development tools
- Development and prototyping license
- Unrestricted license required for production and is available from NuMega
- Technical supported provided by NuMega only

Xilinx PCI64 Virtex

Part no: DO-DI-PCI64

Overview

The LogiCORE PCI64 Virtex interface allows the user to rapidly implement 64 bit bits interfaces. Based on the proven LogiCORE PCI32 Interface, the designer will be able to implement custom 64 bit, 66 MHz PCI Interfaces.

Package Includes

See individual data sheets for details

- LogiCORE PCI64 Virtex Interface
  - Master and Slave (Initiator/Target) functionality
  - Configurable and downloadable from Xilinx web
  - CD available upon request only
- LogiCORE PCI32 4000 Interface
  - Master and Slave (Initiator/Target) functionality
  - Configurable and downloadable from Xilinx web
  - CD available upon request only
- LogiCORE PCI32 Spartan Interface
  - Master and Slave (Initiator/Target) functionality
  - Configurable and downloadable from Xilinx web
  - CD available upon request only
- Synthesizable PCI Bridge Design Examples
  - VHDL and Verilog source files
  - On-line Design Examples User's Guide
- LogiCORE PCI User's Guide
- LogiCORE PCI Implementation Guide
- PCI System Architecture (Reference Book from MindShare, Inc.)

Xilinx PCI32 Design Kit

Part no: DO-DI-PCI32-DK

Overview

To minimize the learning curve for PCI and to enable rapid development and prototyping, Xilinx provides a complete PCI Design Kit including cores, a prototyping board and driver development tools. By forming a partnership with Virtual Computer Corporation, leading provider of rapid prototyping boards and reconfigurable computing systems, and Compuware Numega, formerly *Vireo Software*, leading provider of device driver tools, our customers will always have access to the leading industry expertise.

Package Includes

See individual data sheets for details.

- LogiCORE 32-bit 33 MHz PCI designs
  - Master and Slave (Initiator/Target) functionality
  - zero wait-state
  - Configurable and downloadable from Xilinx web
  - CD available upon request only
- Synthesizable PCI Bridge Design Examples
  - VHDL and Verilog source files
  - On-line Design Examples User's Guide
- Xilinx PCI Design Guide
- Xilinx PCI Implementation Guide
- PCI System Architecture (Reference Book from MindShare, Inc.)
- VCC HotPCI Prototyping System
  - PCI32 Spartan prototyping board
  - Example drivers (NuMega)
  - HotPCI User's Guide
  - Demo software CD for Windows NT, 98/95
- NuMega DriverWorks Development tools for Windows NT/98 drivers
  - Fully functional development tools
  - Development and prototyping license
  - Unrestricted license required for production and is available from NuMega
  - Technical support provided by NuMega only
- NuMega VtoolsD Development tools for Windows 95/98 and 3.x drivers
  - Fully functional development tools
  - Development and prototyping license
  - Unrestricted license required for production and is available from NuMega
  - Technical supported provided by NuMega only

LogiCORE PCI32 Spartan

Part no: DO-DI-PCI32-S

Overview

The LogiCORE PCI32 Spartan interface is included in Xilinx PCI32 Design Kit, but may be purchased separately as a web-only, release.

Package Includes

See individual data sheets for details

- LogiCORE PCI32 Spartan and SpartanXL Interface
  - Configurable and downloadable from Xilinx web
  - Support for Xilinx Spartan and SpartanXL families (see data sheet for specific parts)
  - Master and Slave (Initiator/Target) functionality
  - On-line LogiCORE PCI32 User's Guide

Support, Updates, and Licensing

All Xilinx PCI products include a twelve-month maintenance contract including free updates is included with purchase. After expiring, the maintenance contract may be renewed annually. Included in this contract are the following items:

- Access to Xilinx LogiCORE PCI Lounges
  `www.xilinx.com/pci/`
  - LogiCORE PCI design files and updates
  - Reference Designs
  - Extensive Application Notes
  - Known issues and design tips
- Answer Database on Xilinx web-site
  `support.xilinx.com`
- Hotline telephone support
- Apps fax and email
- Technical email Newsletter

Technical support for NuMega DriverWorks, and VtoolsD driver development tools are provided by NuMega only. The NuMega DriverWorks and VtoolsD driver development tools are licensed for development and prototyping only. An unrestricted license can be purchased from NuMega Software.

Licensing

Xilinx LogiCORE PCI64 Interfaces are licensed under the standard LogiCORE license agreement, at the end of this chapter. Additional licenses are available for evaluation or for design with non-Xilinx technology. Contact your local Xilinx representative for more details.

Product Upgrades

Existing Xilinx PCI customers with valid maintenance agreements may be able to upgrade a PCI product to another. See Figure 3 for available upgrade paths.

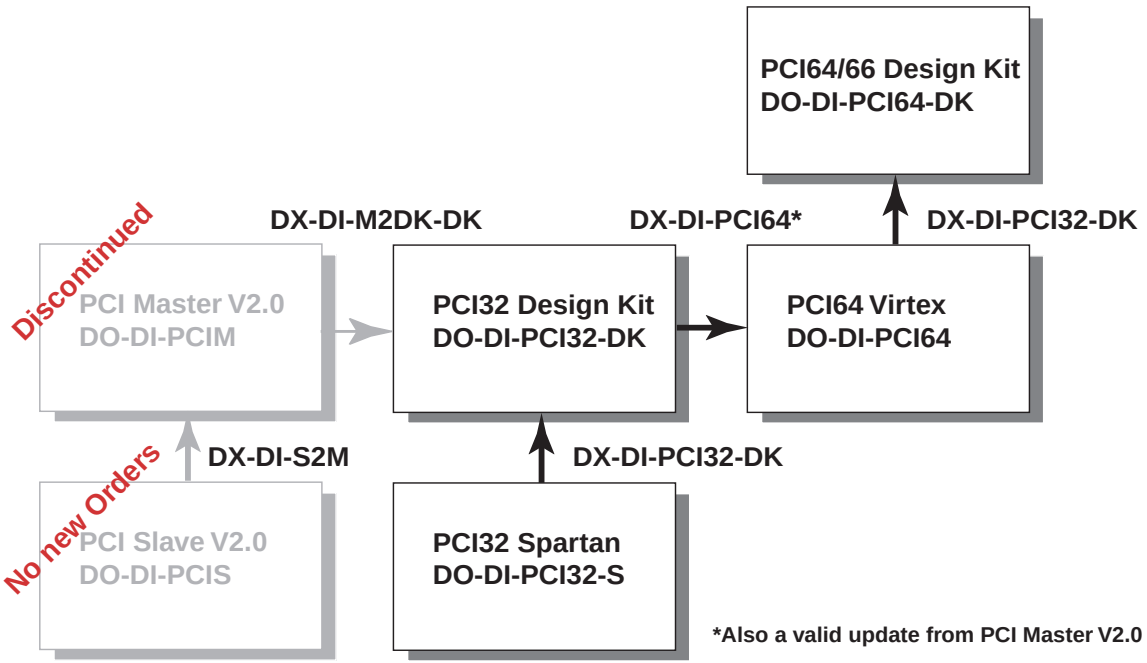


Figure 3: Upgrade paths for various Xilinx PCI products

Additional PCI Products

Additional products are available from Xilinx PCI partners Nallatech, VCC and Compuware Numega. For pricing and availability, please contact the partners directly.

10-14 Market Street
Kilsyth, Glasgow
G65 0BD
Scotland
Phone: +1 44 7020 986532
Fax: +1 44 7020 986534
E-mail: info@nallatech.com
Website: www.nallatech.com

6925 Canby Ave. #103
Reseda, CA 91335 USA
Phone: +1 818-342-8294
Fax: +1 818-342-0240
E-mail: info@vcc.com
Website: www.vcc.com

9 Townsend West
Nashua, NH 03063
Phone: 1 800-4NUMEGA (1 800 468-6342)
+1 603 578-8400
Fax: +1 603 578-8401
E-mail: customer_service@numega.com
Technical support:
www.numega.com/support/support.shtml
Website: www.numega.com

Obsolete products

- LogiCORE PCI Master V2.0 (Part no: DO-DI-PCIM) is no longer available for new purchases. Existing customers with valid maintenance will still receive core design file updates. Additionally, an upgrade package to the complete PCI32 Design Kit is available for customers under valid maintenance agreement. Contact Xilinx for price information.
- LogiCORE PCI Slave V2.0 (Part no: DO-DI-PCIS) is no longer available for new purchases. Existing customers with valid maintenance will still receive core design file updates.

XILINX LOGICORE™ PCI INTEFACE LICENSE AGREEMENT

PLEASE READ THIS DOCUMENT CAREFULLY BEFORE USING THE XILINX LOGICORE PCI INTERFACE DESIGN. UNLESS YOU HAVE A SEPARATE WRITTEN LICENSE EXECUTED BY XILINX COVERING YOUR USE OF THE DESIGN, BY USING THE DESIGN, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE.

License: XILINX, INC. ("XILINX") hereby grants you a nonexclusive, non-transferable license to use the LOGICORE PCI INTERFACE design (the "Design"), solely for your use in developing designs for XILINX programmable logic devices or XILINX HardWire™ devices. Use of the Design in non-XILINX devices or technologies is prohibited unless you have entered into a separate written agreement with XILINX for such use. XILINX retains title to the Design and to any patents, copyrights, trade secrets and other intellectual property rights therein. To protect such intellectual property rights, you may not decompile, reverse engineer, disassemble, or otherwise reduce the Design to a human - perceivable form. You may not modify or prepare derivative works of the Design in whole or in part, except with respect to any source code for the Design supplied by XILINX. This License allows you to make an unlimited number of copies of any source code, schematics and other documentation supplied by XILINX for the Design for internal use only, including without limitation modified versions thereof, provided you reproduce on each such copy the copyright and any other proprietary legends that were on the original copy.

Termination: This License is effective until terminated. You may terminate this License at any time by destroying the Design and all copies thereof. This License will terminate immediately without notice from XILINX if you fail to comply with any provision of this License. Upon termination you must destroy the Design and all copies thereof.

Governmental Use: The Design is commercial computer software developed exclusively at Xilinx's expense. Accordingly, pursuant to the Federal Acquisition Regulations (FAR) Section 12.212 and Defense FAR Supplement Section 227.7202, use, duplication and disclosure of the Design by or for the Government is subject to the restrictions set forth in this License Agreement. Manufacturer is XILINX, INC., 2100 Logic Drive, San Jose, California 95124

Limited Warranty and Disclaimer: THE DESIGN IS PROVIDED TO YOU "AS IS". XILINX AND ITS LICENSORS MAKE AND YOU RECEIVE NO WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. XILINX does not warrant that the functions contained in the Design will meet your requirements, or that the operation of the Design will be uninterrupted or error free, or that defects in the Design will be corrected. Furthermore, XILINX does not warrant or make any representations regarding use or the results of the use of the Design in terms of correctness, accuracy, reliability or otherwise.

Limitation of Liability: IN NO EVENT WILL XILINX OR ITS LICENSORS BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, COST OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES ARISING FROM THE USE OR OPERATION OF THE DESIGN OR ACCOMPANYING DOCUMENTATION, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY. THIS LIMITATION WILL APPLY EVEN IF XILINX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THIS LIMITATION SHALL APPLY NOTWITHSTANDING THE FAILURE OF THE ESSENTIAL PURPOSE OF ANY LIMITED REMEDIES HEREIN.

Export Restriction: You agree that you will not export or re-export the Design, reference images or accompanying documentation in any form without the appropriate United States and foreign government licenses. Your failure to comply with this provision is a material breach of this Agreement.

Third Party Beneficiary: You understand that portions of the Design and related documentation may have been licensed to XILINX from third parties and that such third parties are intended third party beneficiaries of the provisions of this Agreement.

Non-Transferable: You may not provide design source information including, but not limited to, schematics, hardware description language source code, or netlist files, to a third party without prior written approval from XILINX. You may provide device programming files—XILINX bit-stream files or PROM files—or the resulting HardWire gate array to third-parties without prior approval.

Interoperability: If you acquired the Design in the European Union (EU), even if you believe you require information related to the interoperability of the Design with other programs, you shall not decompile or disassemble the Design to obtain such information, and you agree to request such information from Xilinx at the address listed above. Upon receiving such a request, Xilinx shall determine whether you require such information for a legitimate purpose and, if so, Xilinx will provide such information to you within a reasonable time and on reasonable conditions.

Governing Law: This License shall be governed by the laws of the State of California, without reference to conflict of laws principles, provided that if the Design is acquired in the EU, this License shall be governed by the laws of the Republic of Ireland. The local language version of this License shall apply to any Design acquired in the EU. Irish law provides that certain conditions and warranties may be implied in contracts for the sale of goods and in contracts for the supply of services. Such conditions and warranties are hereby excluded, to the extent such exclusion, in the context of this transaction, is lawful under Irish law. Conversely, such conditions and warranties, insofar as they may not be lawfully excluded, shall apply. Accordingly nothing in this License shall prejudice any rights that you may enjoy by virtue of Sections 12, 13, 14 or 15 of the Irish Sale of Goods Act 1893 (as amended).

General: If for any reason a court of competent jurisdiction finds any provision of this License, or portion thereof, to be unenforceable, that provision of the License shall be replaced to the maximum extent permissible so as to effectuate the intent of the parties, and the remainder of this License shall continue in full force and effect. This License constitutes the entire agreement between the parties with respect to the use of this Design and related documentation, and supersedes all prior or contemporaneous understandings or agreements, written or oral, regarding such subject matter.

XILINX REFERENCE DESIGN LICENSE AGREEMENT

By using the accompanying Xilinx, Inc. Reference Designs (the "Designs"), you agree to the following terms and conditions. You may use the Designs solely in support of your use in developing designs for Xilinx programmable logic devices or Xilinx HardWire devices. Access to the Designs is provided only to purchasers of Xilinx programmable logic devices or Xilinx HardWire devices for the purposes set forth herein.

The Designs are provided by Xilinx solely for your reference, for use as-is or as a template to make your own working designs. The Designs may be incomplete, and Xilinx does not warrant that the Designs are completed, tested, or will work on their own without revisions. The success of any designs you complete using the Designs as a starting point is wholly dependent on your design efforts. As provided, Xilinx does not warrant that the Designs will provide any given functionality, and all verification must be completed by the customer.

Xilinx specifically disclaims any obligations for technical support and bug fixes, as well as any liability with respect to the Designs, and no contractual obligations are formed either directly or indirectly by use of the Designs. XILINX SHALL NOT BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION DIRECT, INDIRECT, INCIDENTAL, SPECIAL, RELIANCE OR CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF THE DESIGNS, EVEN IF XILINX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Xilinx makes no representation that the Designs will provide the functionality you are looking for, or that they are appropriate for any given use. Xilinx does not warrant that the Designs are error-free, nor does Xilinx make any other representations or warranties, whether express or implied, including without limitation implied warranties of merchantability or fitness for a particular purpose. The Designs are not covered by any other license or agreement you may have with Xilinx.

The Designs are the copyrighted, confidential and proprietary information of Xilinx. You may not disclose, reproduce, transmit or otherwise copy the Designs by any means for any purpose not set forth in this license, without the prior written permission of Xilinx.

You agree that you will comply with all applicable governmental export rules and regulations, and that you will not export or reexport the Designs in any form without the appropriate government licenses.

Sales Offices, Sales Representatives, and Distributors

Sales Offices, Sales Representatives, and Distributors

May, 1999

Headquarters

XILINX, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: (408) 559-7778
TWX: (510) 600-8750
Fax: (408) 559-7114

Xilinx Sales Offices

NORTH AMERICA

XILINX, Inc.
4825 University Square
Suite 12
Huntsville, AL 35816
Tel: (256) 721-3370
Fax: 256-721-3371

XILINX, Inc.
10235 South 51st St.
Suite 160
Phoenix, AZ 85044
Tel: (602) 753-4503
Fax: 602-753-4504

XILINX, Inc.
1281 Oakmead Pkwy.
Suite 202
Sunnyvale, CA 94086
Tel: (408) 245-9850
Fax: (408) 245-9865

XILINX, Inc.
1227 Platte Ave.
Ventura, CA 93004
Tel: (805) 647-9221
Fax: 805-647-9221

XILINX, Inc.
5690 DTC Blvd.
Suite 490W
Englewood, CO 80111
Tel: (303) 220-7541
Fax: (303) 220-8641

XILINX, Inc.
1500 Kansas Ave.
Suite 1B
Longmont, CO 80501
Tel: (303) 774-1175
Fax: 303-774-1198

XILINX, Inc.
1025 S. Semoran Blvd.
Suite 1093
Winter Park, FL 32792
Tel: (407) 673-8661
Fax: 407-673-8663

XILINX, Inc.
3280 Pointe Parkway
Suite 1600
Norcross, GA 30092
Tel: (770) 448-4733
Fax: 770-448-4857

XILINX, Inc.
15615 Alton Parkway
Suite 280
Irvine, CA 92618
Tel: (949) 727-0780
Fax: (949) 727-3128

XILINX, Inc.
6494 Weathers Place
Suite 100
San Diego, CA 92121
Tel: (619) 597-9855
Fax: 619-597-6418

XILINX, Inc.
61 Spit Brook Rd.
Suite 403
Nashua, NH 03060
Tel: (603) 891-1098
Fax: (603) 891-0890

XILINX, Inc.
39 Surrey Dr.
Belle Mead, NJ 08502
Tel: (908) 359-3136
Fax: 908-359-1253

XILINX, Inc.
30 Two Bridges Rd.
Suite 330
Fairfield, NJ 07004
Tel: (973) 808-2780
Fax: 973-808-2738

XILINX, Inc.
14 Mitchell Terrace
West Long Branch, NJ 07764
Tel: (732) 870-1126
Fax: 732-870-1785

XILINX, Inc.
905 Airport Rd.
Suite 200
West Chester, PA 19380
Tel: (610) 430-3300
Fax: (610) 430-0470

XILINX, Inc.
939 North Plum Grove Road
Suite H
Schaumburg, IL 60173
Tel: (847) 605-1972
Fax: (847) 605-1976

XILINX, Inc.
18283 Minnetonka Blvd.
Suite C
Deephaven, MN 55391
Tel: (612) 473-4816
Fax: 612-473-5060

XILINX, Inc.
12922 Kentbury Dr.
Clarksville, MD 21029
Tel: (301) 924-1300
Fax: 301-924-1301

XILINX, Inc.
6010-C Six Forks Road
Raleigh, NC 27609
Tel: (919) 846-3922
Fax: (919) 846-8316

XILINX, Inc.
4100 McEwen, Suite 237
Dallas, TX 75244
Tel: (972) 960-1043
Fax: (972) 960-0927

XILINX, Inc.
4765 S. Quail Point Road
Salt Lake City, UT 84124
Tel: (801) 273-7338

XILINX, Inc.
14575 Bel-Red Road
Suite 102
Bellevue, WA 98007
Tel: (425) 603-0102
Fax: 425-603-0197

XILINX, Inc.
3554 Brecksville Rd
Richfield, OH 44286
Tel: (330) 659-3131
Fax: (330) 659-9254

XILINX, Inc.
9600 S. W. Oak St.
Suite 320
Portland, OR 97223
Tel: (425) 293-9016
Fax: 425-293-3858

XILINX, Inc.
2910 South Sheridan Way,
Suite 203
Oakville, Ontario
Canada L6J7L9
Tel: (905) 829-9095
Fax: (905) 829-3045

XILINX, Inc.
34 Hampel Crescent
Stittsville, Ontario
Canada K2S 1E4
Tel: (613) 836-5255
Fax: 613-836-5393

EUROPE

XILINX, Ltd.
Benchmark House
203 Brooklands Road
Weybridge, Surrey
KT13 ORH
United Kingdom
Tel: (44) 1932-349401
Fax: (44) 1932-349499

XILINX, Ltd.
(Northern European Sales)
Suite 1B Cobb House
Oyster Lane
Byfleet, Surrey
KT14 7DU
United Kingdom
Tel: (44) 1932-349403
Fax: (44) 1932-345519

XILINX Sarl
Espace Jouy Technology
21, rue Albert Calmette, Bât. C
78353 Jouy en Josas, Cedex
France
Tel: (33) 1 34 63 01 01
Fax: (33) 1 34 63 01 09

XILINX, Sarl
417, Chemin du Cassan
06140 Tourrettes sur Loup
France
Tel: (33) 4-9324-1175
Fax: (33) 4-9324-1007

XILINX GmbH
Süskindstr. 4
D-81929 München
Germany
Tel: (49) 89-93088-0
Tech Support Tel:
(49) 89-93088-130
Fax: (49) 89-93088-188

XILINX AB
Box 1230
Torshamnsgatan 35
S-164 28 Kista
Sweden
Tel: (46) 8-752-2470
Fax: (46) 8-750-6260
E-mail: xilinx-nordic@xilinx.com

XILINX Italia
Via Zamagna 19 - Scala A
20148 Milano
Italy
Tel: (39) 02 487 12 101
Fax: (39) 02 400 94 700

XILINX Benelux bvba
Oude Wichelsesteenweg 27
9340 Lede
Belgium
Tel: (32) 3 205 56 65
Fax: (32) 5381 0472

JAPAN

XILINX K. K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo 163-1118
Japan
Tel: (81) 3-5321-7711
Fax: (81) 3-5321-7765

ASIA PACIFIC

XILINX Asia Pacific
Unit 4312, Tower II
Metroplaza
Hing Fong Road
Kwai Fong, N.T.
Hong Kong
Tel: (852) 2-424-5200
Fax: (852) 2-494-7159
E-mail: hongkong@xilinx.com

XILINX Korea
Room #901
Sambo-Hojung Bldg.,
14-24, Yoido-Dong
Youngdeungpo-Ku
Seoul, South Korea
Tel: (82) 2-761-4277
Fax: (82) 2-761-4278

XILINX Taiwan
Rm. 1006, 10F, No. 2, Lane 150
Sec. 5, Hsin Yin Rd.
Taipei, 105 Taiwan, R.O.C.
Tel: (886) 2-2758-8373
Fax: (886) 2-2758-8367

North American Distributors

Hamilton Hallmark
(Locations throughout
the U.S. and Canada)
Tel: (800) 332-8638
Fax: (800) 257-0568

Insight Electronics
(Locations throughout the U.S.)
Tel: (800) 677-7716
Fax: (619) 587-1380

Nu Horizons
Electronics Corp.
Locations throughout the U.S.
Tel: (516) 396-5000
Fax: (516) 396-7576

U.S. Sales Representatives

ALABAMA

Electro Source, Southeast
4825 University Sq., Ste.12
Huntsville, AL 35816
Tel: (256) 830-2533
Fax: (256) 830-5567

ARIZONA

Quatra Associates
10235 S. 51st St. Suite #160
Phoenix, AZ 85044
Tel: (602) 753-5544
Fax: (602) 753-0640
E-mail: quatra@earthlink.net

ARKANSAS

Bonser-Philhower Sales
689 W. Renner Road
Suite 101
Richardson, TX 75080
Tel: (972) 234-8438
Fax: (972) 437-0897

CALIFORNIA

Norcomp
1267 Oakmead Pkwy
Sunnyvale, CA 94086
Tel: (408) 733-7707
Fax: (408) 774-1947

Norcomp
8880 Wagon Way
Granite Bay, CA 95746
Tel: (916) 791-7776
Fax: (916) 791-2223

Norcomp
30101 Agoura Ct. #234
Agoura, CA 91301
Tel: (818) 865-8330
Fax: (818) 865-2167

Norcomp
30 Corporate Park #200
Irvine, CA 92714
Tel: (949) 260-9868
Fax: (949) 260-9659

Quest-Rep Inc.
6494 Weathers Pl, Suite 200
San Diego, CA 92121
Tel: (619) 622-5040
Fax: (619) 622-9007
E-mail: questrep@questrep.com

COLORADO

Luscombe Engineering, Inc.
1500 Kansas Ave. Suite 1B
Longmont, CO 80501
Tel: (303) 772-3342
Fax: (303) 772-8783

CONNECTICUT

John E. Boeing, Co., Inc.
123 South Main Street
Wallingford, CT 06492
Tel: (203) 265-1318
Fax: 203-265-0235

DELAWARE

Delta Technical Sales, Inc.
122 N. York Rd., Suite 9
Hatboro, PA 19040
Tel: (215) 957-0600
Fax: (215) 957-0920

FLORIDA

Semtronic Assoc., Inc.
(Disti Office)
600 S. North Lake Blvd.
Suite 270
Altamonte Springs, FL 32701
Tel: (407) 831-0451
Fax: (407) 831-6055

Semtronic Assoc., Inc.
(OEM Sales)
600 S. North Lake Blvd.
Suite 220
Altamonte, Springs, FL 32701
Tel: (407) 831-8233
Fax: (407) 831-2844

Semtronic Assoc., Inc.
3471 NW 55th Street
Ft. Lauderdale, FL 33309
Tel: (954) 731-2384
Fax: (954) 731-1019

Semtronic Assoc., Inc.
14004 Roosevelt Blvd.
Suite 604
Clearwater, FL 33762
Tel: (727) 507-0504
Fax: (727) 539-0601

GEORGIA

Electro Source, Southeast
3280 Pointe Parkway,
Suite 1500
Norcross, GA 30092
Tel: (770) 734-9898
Fax: (770) 734-9977

IDAHO (Southwest)

Luscombe Engineering, Inc.
6901 Emerald, Suite 206
Boise, ID 83704
Tel: (208) 377-1444
Fax: (208) 377-0282

Thorson Pacific, Inc.
14575 Bel-Red Road #102
Bellevue, WA 98007
Tel: (425) 603-9393
Fax: (425) 603-9380

ILLINOIS

Advanced Technical Sales
13755 St. Charles Rock Rd.
Bridgeton, MO 63044
Tel: (314) 291-5003
Fax: (314) 291-7958

Beta Technology Sales, Inc.
1009 Hawthorn Drive
Itasca, IL 60143
Tel: (708) 250-9586
Fax: (708) 250-9592

INDIANA

Gen II Marketing, Inc.
31 E. Main St.
Carmel, IN 46032
Tel: (317) 848-3083
Fax: (317-848-1264

Gen II Marketing, Inc.
1415 Magnavox Way
Suite 130
Ft. Wayne, IN 46804
Tel: (219) 436-4485
Fax: (219) 436-1977

IOWA

Advanced Technical Sales
375 Collins Road NE
Cedar Rapids, IA 52402
Tel: (319) 393-8280
Fax: (319) 393-7258

KANSAS

Advanced Technical Sales
2012 Prairie Cir. Suite A
Olathe, KS 66062
Tel: (913) 782-8702
Fax: (913) 782-8641

KENTUCKY

Gen II Marketing, Inc.
861 Corporate Dr. #210
Lexington, KY 40503
Tel: (606) 223-9181
Fax: (606) 223-2864

LOUISIANA (Northern)

Bonser-Philhower Sales
689 W. Renner Rd., Suite 101
Richardson, TX 75080
Tel: (972) 234-8438
Fax: (972) 437-0897

LOUISIANA (Southern)

Bonser-Philhower Sales
10700 Richmond, Suite 150
Houston, TX 77042
Tel: (713) 782-4144
Fax: (713) 789-3072

MAINE

Genesis Associates
128 Wheeler Road
Burlington, MA 01803
Tel: (781) 270-9540
Fax: (781) 229-8913

MARYLAND

Micro Comp, Inc.
1421 S. Caton Avenue
Baltimore, MD 21227-1082
Tel: (410) 644-5700
Fax: (410) 644-5707

MASSACHUSETTS

Genesis Associates
128 Wheeler Road
Burlington, MA 01803
Tel: (781) 270-9540
Fax: (781) 229-8913

MICHIGAN

Miltimore Sales Inc.
22765 Heslip Drive
Novi, MI 48375
Tel: (248) 349-0260
Fax: (248) 349-0756

Miltimore Sales Inc.
3680 44th St., Suite 100-J
Kentwood, MI 49512
Tel: (616) 554-9292
Fax: (616) 554-9210

MINNESOTA

Beta Technology
18283 Minnetonka Blvd.
Suite C
Deephaven, MN 55391
Tel: (612) 473-2680
Fax: (612) 473-2690

MISSISSIPPI

Electro Source, Southeast
4825 University Sq., Ste.12
Huntsville, AL 35816
Tel: (256) 830-2533
Fax: (256) 830-5567

MISSOURI

Advanced Technical Sales
2012 Prairie Cir. Suite A
Olathe, KS 66062
Tel: (913) 782-8702
Fax: (913) 782-8641

Advanced Technical Sales
13755 St. Charles Rock Rd.
Bridgeton, MO 63044
Tel: (314) 291-5003
Fax: (314) 291-7958

MONTANA

Luscombe Engineering, Inc.
670 East 3900 South #103
Salt Lake City, UT 84107
Tel: (801) 268-3434
Fax: (801) 266-9021

NEBRASKA

Advanced Technical Sales
375 Collins Road NE
Cedar Rapids, IA 52402
Tel: (319) 393-8280
Fax: (319) 393-7258

NEVADA

Norcomp
8880 Wagon Way
Granite Bay, CA 95748
Tel: (916) 393-8280
Fax: (916) 393-7258

Quatra Associates
(Las Vegas)
4645 S. Lakeshore Dr., Suite 1
Tempe, AZ 85282
Tel: (602) 820-7050
Fax: (602) 820-7054

NEW HAMPSHIRE

Genesis Associates
128 Wheeler Road
Burlington, MA 01803
Tel: (781) 270-9540
Fax: (781) 229-8913

NEW JERSEY (Northern)

Parallax
734 Walt Whitman Road
Melville, NY 11747
Tel: (516) 351-1000
Fax: (516) 351-1606

NEW JERSEY (Southern)

Delta Technical Sales, Inc.
122 N. York Road, Suite 9
Hatboro, PA 19040
Tel: (215) 957-0600
Fax: (215) 957-0920

NEW MEXICO

Quatra Associates
600 Autumnwood Place, SE
Albuquerque, NM 87123
Tel: (505) 296-6781
Fax: (505) 292-2092

NEW YORK (Metro)

Parallax
734 Walt Whitman Road
Melville, NY 11747
Tel: (516) 351-1000
Fax: (516) 351-1606

NEW YORK

Electra Sales Corp.
333 Metro Park
Rochester, NY 14623
Tel: (716) 427-7860
Fax: (716) 427-0614

Electra Sales Corp.
6057 Corporate Drive
E. Syracuse, NY 13057
Tel: (315) 463-1248
Fax: (315) 463-1717

NORTH CAROLINA

Electro Source, Southeast
5964-A Six Forks Rd.
Raleigh, NC 27609
Tel: (919) 846-5888
Fax: (919) 846-0408

Electro Source
12411 Angle Oak Drive
Huntersville, NC 28078
Tel: (704) 948-8905
Fax: (704) 948-5829

NORTH DAKOTA

Beta Technology
18283 Minnetonka Blvd.
Suite C
Deephaven, MN 55391
Tel: (612) 473-2680
Fax: (612) 473-2690

OHIO

Bear Marketing, Inc.
3554 Brecksville Road
PO Box 427
Richfield, OH 44286-0427
Tel: (216) 659-3131
Fax: (216) 659-4823

Bear Marketing, Inc.
270 Regency Ridge Drive
Suite 115
Dayton, OH 45459
Tel: (513) 436-2061
Fax: (513) 436-9137

OKLAHOMA

Bonser-Philhower Sales
689 W. Renner Rd., Suite 101
Richardson, TX 75080
Tel: (972) 234-8438
Fax: (972) 437-0897

OREGON

Thorson Pacific, Inc.
9600 SW Oak Street,
Suite 320
Portland, OR 97223
Tel: (503) 293-9001
Fax: (503) 293-9007

PENNSYLVANIA

Bear Marketing, Inc.
4284 Rt. 8, Suite 211
Allison Park, PA 15101
Tel: (412) 492-1150
Fax: (412) 492-1155

Delta Technical Sales, Inc.
122 N. York Rd., Suite 9
Hatboro, PA 19040
Tel: (215) 957-0600
Fax: (215) 957-0920

PUERTO RICO

Semtronic Assoc., Inc.
Crown Hills
125 Carite St.
Esq. Avenue Parana
Rio Piedras, P.R. 00926
Tel: (787) 766-0700/0701
Fax: (787) 763-8071

RHODE ISLAND

Genesis Associates
128 Wheeler Road
Burlington, MA 01803
Tel: (781) 270-9540
Fax: (781) 229-8913

SOUTH CAROLINA

Electro Source, Southeast
5964-A Six Forks Rd.
Raleigh NC 27609
Tel: (919) 846-5888
Fax: (919) 846-0408

SOUTH DAKOTA

Beta Technology
18283 Minnetonka Blvd.
Suite C
Deephaven, MN 55391
Tel: (612) 473-2680
Fax: (612) 473-2690

TENNESSEE

Electro Source, Southeast
4825 University Square,
Suite 12
Huntsville, AL 35816
Tel: (205) 830-2533
Fax: (205)-830-5567

TEXAS

Bonser-Philhower Sales
8240 MoPac Expwy.
Suite 295
Austin, TX 78759
Tel: (512) 346-9186
Fax: (512) 346-2393

Bonser-Philhower Sales
10700 Richmond, Suite 150
Houston, TX 77042
Tel: (713) 782-4144
Fax: (713) 789-3072

Bonser-Philhower Sales
689 W. Renner Rd., Suite 101
Richardson, TX 75080
Tel: (972) 234-8438
Fax: (972) 437-0897

TEXAS (El Paso County)

Quatra Associates
600 Autumnwood Place SE
Albuquerque, NM 87123
Tel: (505) 296-6781
Fax: (505) 292-2092

UTAH

Luscombe Engineering Co.
670 East 3900 South #103
Salt Lake City, UT 84107
Tel: (801) 268-3434
Fax: (801) 266-9021

VERMONT

Genesis Associates
128 Wheeler Road
Burlington, MA 01803
Tel: (781) 270-9540
Fax: (781) 229-8913

VIRGINIA

Microcomp, Inc.
1421 S. Caton Avenue
Baltimore, MD 21227
Tel: (410) 644-5700
Fax: (410) 644-5707

WASHINGTON

Thorson Pacific, Inc.
14575 Bel-Red Rd.
Suite 102
Bellevue, WA 98007
Tel: (206) 603-9393
Fax: (206) 603-9380

WASHINGTON (Vancouver, WA only)

Thorson Pacific, Inc.
9600 SW Oak Street
Suite 320
Portland, OR 97223
Tel: (503) 293-9001
Fax: (503) 993-9007

WASHINGTON D.C.

Micro Comp, Inc.
1421 S. Caton Avenue
Baltimore, MD 21227-1082
Tel: (410) 644-5700
Fax: (410) 644-5707

WEST VIRGINIA

Bear Marketing, Inc.
4284 Rt. 8 Suite 211
Allison Park, PA 15101
Tel: (412) 492-1150
Fax: (412) 492-1155

WISCONSIN (Western)

Beta Technology
18283 Minnetonka Blvd.
Suite C
Deephaven, MN 55391
Tel: (612) 473-2680
Fax: (612) 473-2690

WISCONSIN (Eastern)

Beta Technology Sales, Inc.
9401 N. Beloit, Suite 409
Milwaukee, WI 53227
Tel: (414) 543-6609
Fax: (414) 543-9288

WYOMING

Luscombe Engineering, Inc.
1500 Kansas Ave. Suite 1B
Longmont, CO 80501
Tel: (303) 772-3342
Fax: (303) 772-8783

International Sales Representatives

ALGERIA

Development Centre of
Advanced Technologies
128 Chemin Mohamed GACEM
16075 El-Madania
Algiers
Algeria
Tel: (213) 2-67-73-25
Fax: (213) 2-66-26-89

ARGENTINA

Reycom Electronica S.A.
Bdo. de Irigoyen
972 Piso 2do “B”
1304 Buenos Aires
Argentina
Tel: (54) 1-304-2018
Fax: (54) 1-304-2010

Insight Argentina
Av. Adolfo Davila 550, 2nd Floor
1107 Buenos Aires
Argentina
Tel: (54) 1-310-0052
Fax: (54) 1-310-0053

AUSTRALIA

Advanced Component Dist.
Suite 5, Level 1, "Metro Centre"
124 Forest Rd.
Hurstville 2220
Australia
Tel: (61) 2-9585-5533
Fax: (61) 2-9585-5534

Advanced Component Dist.
Unit 2, 17-19 Melrich Road
Bayswater VIC 3153
Melbourne, Australia
Tel: (61) 3-9760-4250
Fax: (61) 3-9760-4255

Advanced Component Dist.
20D William Street
Norwood SA 5067
Australia
Tel: (61) 8-8364-2844
Fax: (61) 8-8364-2811

Advanced Component Dist.
Ste. 1, 1048 Beaudesert Rd.
Cooper Plains
Queensland 4108
Australia
Tel: (61) 7-3246-5214
Fax: (61) 7-3275-3662

EDA Solutions Pty. Ltd.
Level 3, South Tower
1-5 Railway Street
Chatswood NSW 2067
Australia
Tel: (61) 02-9413-4611
Fax: (61) 02-9413-4622

EDA Solutions Pty. Ltd.
Level 2, 854 Glenferrie Road
Hawthorn VIC 3122
Australia
Tel: (61) 03-9819-0000
Fax: (61) 03-9818-8870

AUSTRIA

Metronik GmbH
Diefenbachgasse 35
A-1150 Wien
Austria
Tel: (43) 1-89-5762652
Fax: (43) 1-89-5762650

BELGIUM & LUXEMBURG

SEI Rodelco NV
Limburg Stirum 243
1780 Wemmel
Belgium
Tel: (32) 2-456-0757
Fax: (32) 2-460-0271

BRASIL

Hitech
Rua Branco de Moraes 489
Ch. Santo Antonio
Sao Paulo
04718-010 - SP - Brazil
Tel: (55) 11-882-4000
Fax: (55) 11-882-4100

Insight Brasil
Rua Alcides Ricardini Neves
12 - 13o andar
Conjunto 1306 - Brooklin
04575-050 Sao Paulo SP
Tel: (55) 11-5505-6501/2
Fax: (55) 11-5505-6702
E-Mail: insbrz@uol.com.br

BULGARIA

Petrex 92 Ltd.
Philip Kutev Str. 1
BG-1407 Sofia
Bulgaria
Tel: (359) 2-626987
Fax: (359) 2-627099

CANADA (ALBERTA)

Electro Source
2635 37th Ave NE #245
Calgary, Alberta T1Y 5Z6
Canada
Tel: (403) 735-6230
Fax: (403) 735-0599

CANADA (BRITISH COLUMBIA)

Thorson Pacific, Inc.
4170 Still Creek Dr. #200
Burnaby BC V5C 6C6
Canada
Tel: (604) 294-3999
Fax: (604) 473-7755

CANADA (OTTAWA)

Electro Source, Inc.
50 Hines Road, Suite 220
Kanata, Ontario K2K 2M5
Canada
Tel: (613) 592-3214
Fax: (613) 592-4256

CANADA (QUEBEC)

Electro Source
6600 TransCanada Hwy
Suite 420
Pointe Claire, Quebec H9R 4S2
Canada
Tel: (514) 630-7486
Fax: (514) 630-7421

CANADA (TORONTO)

Electro Source, Inc.
230 Galaxy Blvd.
Rexdale Ontario M9W 5R8
Canada
Tel: (416) 675-4490
Fax: (416)-675-6871

CHILE

DTS Ltda.
Rosas 1444
Santiago
Chile
Tel: (56) 2-6970991
Fax: (56) 2-6993316

CHINA PEOPLE’S REPUBLIC

Insight
Rm. 692, Pana Tower
No. 128 Zhichun Rd.
Haidian District
Beijing 10086
P.R. China
Tel: (86) 10-6262-8985
Fax: (86) 10-6262-0393

Insight
Rm. 2015-2018,
Tong Mei Mansion
No. 76, Section 1,
Jianshe North Rd.,
Chengdu, Sichuan 610051
P.R. China
Tel: (86) 28-3399-629
Fax: (86) 28-3398-829

Insight
Rm. 705,
Wuhan Computer City
39 Luo Yu Road,
Hongshan District
Wuhan, 430079
PR China
Tel: (86) 27-8787-4319
Fax: (86) 27-8786-3102
E-mail:
memecnh@public.wh.hb.cn

Insight
Rm. 715, Bao Hua Bldg.,
#1016, Hua Qiang North Rd.,
Shenzhen 518031
P.R. China
Tel: (86) 755-377-9548
Fax: (86) 755-377-9026

Insight
Room. 1407-1409
China Venturetech Plaza
819 Nanjing Road (W)
Shanghai 200041
P.R. China
Tel: (86) 21-6215-9935
Fax: (86) 21-6215-9938

Insight
Rm. 805, Genius Xin Jie Kou
Commercial Centre
#219 Zhongshan South Road.
Nanjing, 210005, P.R. China
Tel: (86) 25-4549-807 /
4513-183
Fax: (86) 25-4549-585

Insight
Rm. 703, Chongqing (Yuzhou)
Computer City
3 Ke Yuan Yi Rd., Shiqiaopu
Chongqing, 400039, P.R.
China
Tel: (86) 23-6879-0845
Fax: (86) 23-6879-0845

Insight
Rm 35G, Tower 3, Xiangjiang
Great Garden
Xiamen, 361009, P.R. China
Tel: (86) 592-513-7850
Fax: (86) 592-513-7850

CZECH REPUBLIC

MES Praha s.r.o.
Platonova 3287/26
CZ 14300 Praha
Tel: (420) 2-90059190
Fax: (420) 2-90059190
E-mail: mespraha@ms.anet.cz

DENMARK

Micronor A/S
P.O. Box 929
Torvet 1
DK-8600 Silkeborg
Denmark
Tel: (45) 8681-6522
Fax: (45) 8681-2827
E-Mail: e-mail@micronor.dk
WWW: http://www.micronor.dk

Avnet Nortec
Transformervej 17
Dek-2730 Herlev
Denmark
Tel: (45) 44-88-08-00
Fax: (45) 44-88-08-88

EGYPT

Guide Systems Integrators
27 Mokhles Al-Alfi St.
First - Zone Nasr City
Cairo
Egypt
Tel: (20) 2-401-4085
Fax: (20) 2-401-4997
E-Mail: obadr@idsc.gov.eg

FINLAND

Memec Finland OY
Kauppakaarre 1
00700 Helsinki
Finland
Tel: (358) 9-350-8880
Fax: (358) 9-350-88828

FRANCE

REP'TRONIC
1 Bis, rue Marcel Paul
Z.I. La Bonde
91742 Massy Cedex
France
Tel: (33) 1 69 53 67 20
Fax: (33) 1 60 13 91 98
E-Mail:
100745.605@compuserve.com

AVNET EMG.
79 Rue Pierre Sémard
92320 Châtillon
France
Tel: (33) 1 49 65 27 00
Fax: (33) 1 49 65 27 39

AVNET Composants
Sud-Ouest
Technoparc
Bât.4, Voie 5, BP 404
31314 Labège Cedex
France
Tel: (33) 5 61 39 21 12
Fax: (33) 5 61 39 21 40

AVNET Composants
Rhône-Auvergne
Parc Club du Moulin à Vent
Bât 32-33, rue du Dr. G. Levy
69693 Venissieux Cedex
France
Tel: (33) 4 78 00 1280
Fax: (33) 4 78 75 95 97

AVNET EMG
Parc Club du Moulin à Vent
Bât. 40
33, av. du Dr. Georges Levy
69693 Venissieux Cedex
France
Tel: (33) 4 78 77 13 60
Fax: (33) 4 78 77 13 99

AVNET Composants
Ouest
Technoparc-Bât. E
4 Av. des Peupliers, BP 43
35511 Cesson Sévigné Cedex
France
Tel: (33) 2 99 83 84 85
Fax: (33) 2 99 83 80 83

Compress
47, rue de l'Estérel
Silic 539
94633 Rungis Cedex
France
Tel: (33) 1 41 80 29 00
Fax: (33) 1 46 86 67 63

Compress
Rhône-Alpesl
19 Chemin du Goyet
38300 Bourgoin-Jallieu
France
Tel: (33) 4 74 43 53 33
Fax: (33) 4 74 43 53 30

Compress
Bretagne
19 rue de Kerjean
22700 Louannec
France
Tel: (33) 2 96 49 09 52
Fax: (33) 2 96 49 09 56

Compress
Ouest
5 Impasse Guenot
31100 Toulouse
France
Tel: (33) 5 62 87 75 08
Fax: (33) 5 62 87 75 09

GERMANY

Avnet EMG GmbH
Stahlgruberring 12
D-81829 München
Germany
Tel: (49) 89-45110-01
Fax: (49) 89-45110-129

Avnet EMG GmbH
Kurfürstenstr. 130
D-10785 Berlin
Germany
Tel: (49) 30-214882-0
Fax: (49) 30-2141728

Avnet EMG GmbH
Wolfenbüttler Str. 33
D-38102 Braunschweig
Germany
Tel: (49) 531-22073-0
Fax: (49) 531-22073-35

Avnet EMG GmbH
Friedrich-Ebert-Damm 145
D-22047 Hamburg
Germany
Tel: (49) 40-696952-0
Fax: (49) 40-6962787

Avnet EMG GmbH
Benzstr. 1
D-70839 Gerlingen
Stuttgart, Germany
Tel: (49) 7156-4390
Fax: (49) 7156-439-111

Avnet EMG GmbH
Max-Planck-Str. 15b
D-40699 Erkrath
Düsseldorf, Germany
Tel: (49) 211-92003-0
Fax: (49) 211-9200399

Avnet EMG GmbH
Schmidtstr. 49
D-60326 Frankfurt/M.
Germany
Tel: (49) 69-973804-0
Fax: (49) 69-7380712

Avnet EMG GmbH
Fürther Str. 212
D-90429 Nürnberg
Germany
Tel: (49) 911-93149-0
Fax: (49) 911-320821

Intercomp
Am Hochwald 42
D-82319 Starnberg
Germany
Tel: (49) 8151-16044
Fax: (49) 8151-79270
E-Mail: intercomp.tiefenthaler
@t-online.de

Intercomp
Heerstr. 167
D-78628 Rottweil
Germany
Tel: (49) 741-14845
Fax: (49) 741-15220
E-Mail:
intercomp.klink@t-online.de

Intercomp
Schustergasse 25
D-55278 Köngernheim
Germany
Tel: (49) 6737-9881
Fax: (49) 6737-9882
E-Mail:
intercomp.harkam@t-online.de

Intercomp
Heidbergstr 30
D-22846 Norderstedt
Germany
Tel: (49) 405-25-50377
Fax: (49) 405-25-50378
E-Mail:
intercomp.robben@t-online.de

Intercomp
Abdeck 3B
D-84095 Furth bei Landshut
Germany
Tel: (49) 870-48-593
Fax: (49) 870-48-594
E-mail: intercomp.neugebauer
@t-online.de

Metronik GmbH
Leonhardsweg 2
D-82008 Unterhaching
München, Germany
Tel: (49) 89-611-080
Fax: (49) 89-611-08110

Metronik GmbH
Zum Lonnenhohl 40
D-44319 Dortmund
Germany
Tel: (49) 231-9271-10-0
Fax: (49) 231-9271-10-99

Metronik GmbH
Carl-Zeiss Str.6
D-25451 Quickborn
Hamburg Germany
Tel: (49) 4106-77-30-50
Fax: (49) 4106-77-30-52

Metronik GmbH
Osmiastrasse 9
D-69221 Dossenheim
Mannheim, Germany
Tel: (49) 6221-8-70-44
Fax: (49) 6221-8-70-46

Metronik GmbH
Äussere Grossweiden-
mühlstr. 45
D-90419 Nürnberg
Germany
Tel: (49) 911-338802
Fax: (49) 911-338827

Metronik GmbH
Löwenstrasse 37
D-70597 Stuttgart
Germany
Tel: (49) 711-769641-0
Fax: (49) 711-765-5181

Metronik GmbH
Franz-Schubert-Str.41
D-16548 Glienicke
Berlin, Germany
Tel: (49) 33056-62510
Fax: (49) 33056-62550

Metronik GmbH
Schönauer Strasse 113
D-04207 Leipzig
Germany
Tel: (49) 341-4240027
Fax: (49) 341-4240029

Metronik GmbH
Bahnstrasse 9
D-65205 Wiesbaden
Germany
Tel: (49) 611-973-84-0
Fax: (49) 611-973-8418

GREECE

Semicon
104 Aeolou Str.
10564 Athens
Greece
Tel: (30) 1-32-536-26
Fax: (30) 1-32-160-63
E-Mail: semicon@hellas.eu.net

HONG KONG

Insight
Units 3601-02 & 07-25, Tower I,
Metroplaza, Hing Fong Road,
Kwai Fong, N.T.
Hong Kong
Tel: (852) 2410-2780
Fax: (852) 2401-2518

HUNGARY

ChipCAD Kft
Dolmany u. 12
H-1131 Budapest
Tel: (36) 1-270-7680
Fax: (36) 1-270-7699

INDIA

CG-CoreEl Logic System Ltd.
First Floor, Surya Bhavan
1181 Fergusson College Rd.
Pune 411 005
India
Tel: (91) 212-323982, 328074
Fax: (91) 212-323985
Email: xsupport@cromp.ernet.in

CG-CoreEL Logic System Ltd.
961 Main 12th
HAL 2nd Stage
Bangalore 560 008
India
Tel: (91) 80-527-9726
Fax: (91) 80-527-3073
E-Mail:
vishwa@giasbg01.vsni.net.in

CG-CoreEl Logic Systems
ABC Business Centre
N-52 Connaught Place
New Delhi, 110001
India
Tel: (91) 11-3738-973
Fax: (91) 11-3730-404

Core El Micro Systems
45131 Manzanita Ct.
Fremont, CA 94539
Tel: (510) 770-2277
Fax: (510) 770-2288

IRELAND

Memec Ireland Ltd.
Gardner House
Bank Place
Limerick
Ireland
Tel: (353) 61-411842
Fax: (353) 61-411888
E-Mail: memec@iol.ie

ISRAEL

E.I.M. International Ltd.
9 Hashiloach Street
P.O. Box 7025
Petach Tikva 49130
Israel
Tel: (972) 3-923-3257
Fax: (972) 3-922 3577

E.I.M. International Elec
2000 Arctic Ave
Bohemia
New York, NY 11716
Tel: (516) 567-0500
Fax: (516) 567-0011

ITALY

Acsis Srl
Via Alberto Mario, 26
20149 Milano, Italy
Tel: (39) 02-480-22522
Fax: (39) 02-480-12289
WWW: http://www.acsis.it

Avnet EMG
Centro Direzionale
Via Novara, 570
C.A.P 20153 Milan
Italy
Tel: (39) 02-381-901
Fax: (39) 02-380-02988

Avnet EMG
Ancona
Via Adriatica, 13
60022 Castelfidardo
Italy
Tel: (39) 071-781-9644
Fax: (39) 071-781-9699

Avnet EMG
Firenze
Via Panciatichi, 40
50127 Firenze
Italy
Tel: (39) 055-436-0392
Fax: (39) 055-431-035

Avnet EMG
Modena
Via Scaglia Est, 144
41100 Modena
Italy
Tel: (39) 059-351-300
Fax: (39) 059-344993

Avnet EMG
Napoli
Via Ferrante Imparato, 27
80146 Napoli
Italy
Tel: (39) 081-55-91477
Fax: (39) 081-55-91580

Avnet EMG
Roma
Via Zoe Fontana, 220
Tecnocitta
00131 Roma
Italy
Tel: (39) 064-13-1151
Fax: (39) 064-13-1161

Avnet EMG
Torino
Corso Orbassano, 336
10137 Torino
Italy
Tel: (39) 011-311-2347
Fax: (39) 011-308-2138

Avnet EMG
Treviso
Via Delle Querce, 7
31033 Castelfranco Veneto
Italy
Tel: (39) 0423-722-675
Fax: (39) 0423-722-671

Silverstar-Celdis
Viale Fulvio Testi, 280
20126 Milano
Italy
Tel: (39) 02-661-251
Fax: (39) 02-661-01-359

Silverstar-Celdis
Via Collamarini, 22
40138 Bologna
Italy
Tel: (39) 051-53-8500
Fax: (39) 051-53-8831

Silverstar-Celdis
Via G. Antonio Resti, 63
00143 Roma
Italy
Tel: (39) 06-519-57527
Fax: (39) 06-504-3330

Silverstar-Celdis
Centro Piero Della Francesca
Corso Svizzera, 185 Bis
10149 Torino
Italy
Tel: (39) 011-77-10082
Fax: (39) 011-77-64921

Silverstar-Celdis
Centro Direzionale Benelli
Via Degli Abeti, 346
61100 Pesaro
Italy
Tel: (39) 0721-26-560
Fax: (39) 0721-400896

Silverstar-Celdis
Via A.da Noli, 6
50127 Firenze
Italy
Tel: (39) 055-43-5125
Fax: (39) 055-43-77184

Silverstar-Celdis
Via. delle Industrie, 13
35010 Limena
Padova
Italy
Tel: (39) 049-88-40044
Fax: (39) 049-88-41079

Silverstar-Celdis
Via Famagosta, 1/5
17100 Savona
Italy
Tel: (39) 019-81-5090
Fax: (39) 019-81-5091

JAPAN

Kaga Electronics Co., Ltd.
1-26-1, Otowa
Bunkyo-ku, Tokyo 112-8657
Japan
Tel: (81) 3-3942-6744
Fax: (81) 3-3942-6256

Kaga Electronics Co., Ltd.
3-13-20, Nishi-tenman
Kita-ku, Osaka 530-0047
Japan
Tel: (81) 6-364-3911
Fax: (81) 6-364-4191

Marubun Corporation
Marubun Daiya Bldg.
8-1, Nihonbashi, Odenmacho
Chuo-ku, Tokyo 103-8577
Japan
Tel: (81) 3-3639-5120
Fax: (81) 3-3639-9925

Marubun Corporation
5-5-15, Nishi-nakashima
Yodogawa-ku, Osaka 532-0011
Japan
Tel: (81) 6-301-1551
Fax: (81) 6-301-1991

OEL K.K.
Bunkyo Green Court Center Office, 19F
2-28-8 Hon-Komagome
Bunkyo-ku
Tokyo 113-6591
Japan
Tel: (81) 3-5978-8204
Fax: (81) 3-5978-1818

OEL K.K.
4-4-2 Kitakyuhoji-cho
Chuo-ku
Osaka 541-0057
Japan
Tel: (81) 6-282-4810
Fax: (81) 6-282-4160

Tokyo Electron Device Ltd.
No. 1, Higashikata-machi
Tsuzuki-ku,
Yokohama
Kanagawa 224-0003
Japan
Tel: (81) 45-474-5096
Fax: (81) 45-474-5583

Tokyo Electron Device Ltd.
4-1-14, Miyahara, Yodogawa-ku
Osaka 532-0003
Japan
Tel: (81) 6-399-0234
Fax: (81) 6-399-0283

THE NETHERLANDS

Rodelco BV
P.O. Box 6824
Takkebijsters 2
4802 HV Breda
The Netherlands
Tel: (31) 76-5722700
Fax: (31) 76-5710029

NEW ZEALAND

MEMEC EBV (NZ) Ltd.
Suite 5a, Level 4,
North City Plaza,
Titahi Bay Rd., Porirua,
Wellington, New Zealand
Tel: (64) 4-237-9711
Fax: (64) 4-237-9718

MEMEC EBV (NZ) Ltd.
P.O. Box 3700
Christchurch
New Zealand
Tel: (64) 03-379-3889
Fax: (64) 03-379-3072

MEMEC EBV (NZ) Ltd.
Unit 7, 110 Mays Road
Penrose
Auckland
New Zealand
Tel: (64) 09-636-5984
Fax: (64) 09-636-5985

NORWAY

BIT Elektronikk AS
Smedsvingen 4
P.O. Box 194
1360 Nesbru
Norway
Tel: (47) 66-77-65-00
Fax: (47) 66-77-65-01

POLAND

P.T.H. Atest s.c.
ul. Sowinskiego 5
PL-44-100 Gliwice
Poland
Tel: (48) 32-380341
Fax: (48) 32-380692

PORTUGAL

ADM Electronica SA
en 107, No 743
Aguas Santas
4445 Ermesinde
Portugal
Tel: (35) 1-92-973-6957
Fax: (35) 1-92-973-6958

RUSSIA

Scan Ltd.
10/32 “B” Druzhby St.
117330 Moscow
Russia
Tel: (7) 095-232-2343
Fax: (7) 095-938-2247

Scan Ltd
42 Ordjonikidze Street
196143 St. Petersburg
Russia
Tel: (7) 812-299-7028
Fax: (7) 812-264-6000

Scan Engineering
14 Plekhanova St.
Voronezh 394000
Russia
Tel: (7) 0732-521006
Fax: same

Scan Pulsar
9 Rogaliova St.
Dniepropetrovsk 320030
Ukraine
Tel: (7) 0562-472870
Fax: (7) 0562-451115

Scan-West
217, 32 Asanbaljieva St.
Minsk 220024
Belorussia
Tel: (7) 0172-756 261
Fax: (7) 0172-756 750

SINGAPORE

MEMEC (Asia Pacific) Penang
6L-2 Jalan Rumbia
11900 Penang
Malaysia
Tel: (60) 4-646-9986
Fax: (60) 4-646-9946

MEMEC Asia Pacific Ltd.
Singapore Representative Office
10 Anson Road #23-08
International Plaza
Singapore 079903
Tel: (65)-222-4962
Fax: (65)-222-4939

SLOVAK REPUBLIC

Elbatex SK S.R.O.
Kasmirska 7
SK-821 04 Bratislava
Tel: (421) 7-43414173
Fax: (421) 7-43420600
Email: ek.elbatex@net.lask

SLOVENIA/CROATIA

IC Elektronika d.o.o.
Vodovodna 100
SL-1000 Ljubljana
Tel: (386) 61165316010
Fax: (386) 61165317020

SOUTH AFRICA

Avnet - ASD
Avnet Kopp (Pty) Ltd.
PO Box 3853
Rivonia 2128
South Africa
Tel: (27) 11 444 2333
Fax: (27) 11 444 7778

SOUTH KOREA

Hyunmyung Electronics Co. Ltd.
3 Fl, Dukwha Bldg.,
444-17, Seokyo-Dong,
Mapo-Ku, Seoul
South Korea
Tel: (82) 2-3141-0147
Fax: (82) 2-3141-0149

Insight
Rm. 501, Daeha Bldg.
14-11 Yoido-Dong
Youngdeungpo-Ku
Seoul, 150-715
South Korea
Tel: (82) 2-786-8180
Fax: (82) 2-761-4121

Seodu Inchip
4 Fl, Myoungi Bldg.,
142-21, Samsung-Dong,
Kangnam-Ku, Seoul
South Korea
Tel: (82) 2-563-8008
Fax: (82) 2- 563-8411

SPAIN

ADM Electronica SA
Calle Tomas Breton,
No 50, 3-2
28045 Madrid
Spain
Tel: (34) 9-1-530-4121
Fax: (34) 9-1-530-164

ADM Electronica SA
Calle Mallorca 1
08014 Barcelona
Spain
Tel: (349) 3-426-6892
Fax: (349) 3-425-0544

ADM Electronica, SA
Sasi Koa, 26, 3°C
48200 Durango (Vizcaya)
Spain
Tel: (349) 4-620-1572
Fax: (349) 4- 620-2331

SWEDEN

DipCom Electronics AB
Torshamnsgatan 35
P.O. Box 1230
S-164 28 Kista
Sweden
Tel: (46) 8 752 24 80
Fax: (46) 8 751 3649

Avnet EMG AB
Boc 1410
Englundavagen 7
S-171 27 Solna
Sweden
Tel: (46) 8-629-14-00
Fax: (46) 8-29--26-95

SWITZERLAND

Memotec AG
Gaswerkstr. 32
CH-4901 Langenthal
Switzerland
Tel: (41) 629195555
Fax: (41) 629195500

TAIWAN

Insight
Rm. 1005, 10F, No.2, Lane 150
Sec.5, Hsin Yin Rd.
Taipei, Taiwan R.O.C.
Tel: (886) 2-8780-1216
Fax: (886) 2-8780-1220

Avnet - Mercuries Co. Ltd.
14th Floor, No. 145, Sec.
2, Chien-Kuo N. Rd.
Taipei,
Taiwan R.O.C.
Tel: (886) 2-2503-1111
Fax: (886) 2-2505-1449

THAILAND

MEMEC Thailand
240/10, 12th Floor,
Ayodhaya Tower
Ratchadapisek Road
Huey-Kuang
Bangkok, 10310
Thailand
Tel: 662-2741644-5
Fax: 662-2741673

TURKEY

Empa Elektronik ve Bilgi
Teknolojilei AS
Besyol Mah. Florya Kavsagi
Eski Havaalani Cad. No: 26/6
34630 Florya
Istanbul
Turkey
Tel: (90) 212 592 7401
Fax: (90) 212 599 3059

Eltronik Inc.
8427 Kennedy Boulevard
North Bergen
New Jersey 07047
USA
Tel: (201) 453-0410
Fax: (201) 453-0959

UNITED ARAB EMIRATES

Culato
P.O. Box 7820
Dubai
United Arab Emirates
Tel: (971)-4-284031
Fax: (971)-4-284934

Rezwan Trading Est.
PO Box 51973
Dubai
United Arab Emirates
Tel: (97) 1-4279420
Fax: (97) 1-4275741

UNITED KINGDOM

Avnet EMG Ltd
Avnet House
Rutherford Close, Meadway
Stevenage
Hertfordshire, SG1 2EF
England
Tel: (44) 1438-788500
Fax: (44) 1438-788 250

Cedar Technologies
Unit One Old Barns
Rycote Lane Farm
Milton Common
Oxfordshire
OX9 2NZ
England
Tel: (44) 1844-278278
Fax: (44) 1844-278378

Cedar Technologies
32 Enterprise House
Sprinkerse Business Park
Striling FK7 7UF
England
Tel: (44) 1786-446220
Fax: (44) 1786-446223

Memec Plc
17, Thame Park Road
Thame
Oxfordshire
OX9 3XD
England
Tel: (44) 1844-261919
Fax: (44) 1844-1683

Memec Ireland Ltd.
Garden House
Bank Place
Limerick
Eire
Tel: (353) 61-411842
Fax: (353) 61-411888
Email: memec@iol.ie

Microcall Ltd.
The Gate House
Alton House Business Park
Gatehouse Way
Aylesbury, Bucks
HP19 3DL
England
Tel: (44) 1296-330061
Fax: (44) 1296-330065