Advance Data Sheet
November 1999

microelectronics group

Lucent Technologies
Bell Labs Innovations

ORCA® OR3LP26B Field-Programmable System Chip (FPSC) Embedded Master/Target PCI Interface

Introduction

Lucent Technologies Microelectronics Group has developed a solution for designers who need the many advantages of an FPGA-based design implementation, coupled with the high bandwidth of an industry-standard PCI interface. The *ORCA* OR3LP26B, a member of the Series 3+ FPSC family, provides a full-featured 33/50/66 MHz, 32-/64-bit PCI interface, fully designed and tested, in hardware, plus FPGA logic for user-programmable functions.

PCI Bus Core Highlights

- Implemented in an *ORCA* Series 3 OR3L125B base array, displacing the bottom ten rows of 28 columns.
- Core is a well-tested ASIC model.
- Fully compliant to Revision 2.2 of PCI Local Bus Specification.
- Operates at PCI bus speeds up to 66 MHz on a 32-/64-bit wide bus.
- Comprises two independent controllers for Master and Target.
- Meets/exceeds all requirements for *PICMG** Hot Swap Friendly silicon, Full Hot Swap model, per the *CompactPCI** Hot Swap Specification, *PICMG* 2.1 R1.0.
- PCI SIG Hot Plug (R1.0) compliant.
- Four internal FIFOs individually buffer both directions of both the Master and Target interfaces:
  — Both Master FIFOs are 64 bits wide by 32 bits deep.
  — Both Target FIFOs are 64 bits wide by 16 bits deep.
- Capable of no-wait-state, full-burst PCI transfers in either direction, on either the Master or Target interface. The dual 64-bit data paths extend into the FPGA logic, permitting full-bandwidth, simultaneous bidirectional data transfers of up to 528 Mbytes/s to be sustained indefinitely.
- Can be configured to provide either two 64-bit buses (one in each direction) to be multiplexed between Master and Target, or four independent 32-bit buses.
- Provides many hardware options in the PCI core that are set during FPGA logic configuration.
- Operates within the requirements of the PCI 5 V and 3.3 V signaling environments and 3.3 V commercial or industrial environmental conditions, allowing the same device to be used in 5 V or 3.3 V PCI systems.
- FPGA is reconfigurable via the PCI interface's configuration space (as well as conventionally), allowing the FPGA to be field-updated to meet late-breaking requirements of emerging protocols.

* *PICMG* and *CompactPCI* are registered trademarks of the PCI Industrial Computer Manufacturers Group.

Table 1. *ORCA* OR3LP26B PCI FPSC Solution—Available FPGA Logic

Device	Usable Gates†	Number of LUTs	Number of Registers	Max User RAM	Max User I/Os	Array Size	Number of PFUs
OR3LP26B	60K—120K	4032	5304	64K	259	18 x 28	504

† The embedded core and interface comprise approximately 85K standard-cell ASIC gates in addition to these usable gates. The usable gate counts range from a logic-only gate count to a gate count assuming 30% of the PFUs/SLICs being used as RAMs. The logic-only gate count includes each PFU/SLIC (counted as 108 gates per PFU/SLIC), including 12 gates per LUT/FF pair (eight per PFU), and 12 gates per SLIC/FF pair (one per PFU). Each of the four PIOs per PIC is counted as 16 gates (two FFs, fast-capture latch, output logic, CLK drivers, and I/O buffers). PFUs used as RAM are counted at four gates per bit, with each PFU capable of implementing a 32 x 4 RAM (or 512 gates) per PFU.

Table of Contents

List of Figures

Figures **Page**

List of Tables

PCI Bus Core Highlights (continued)

- Master:
  - — Generates all defined command codes except interrupt acknowledge and special cycle.
  - — Capable of accessing its own local Target.
  - — Capable of acting as the system's configuration agent by booting up with the Master logic enabled.
  - — Supports multiple options for Master bus requests, to increase PCI bus bandwidth.
  - — Supports single-cycle I/O space accesses.
  - — Provides option to delay PCI access until FIFO is full on Master writes to increase PCI bandwidth.
  - — Supports programmable latency timer control.
- Target:
  - — Responds legally to all command codes: interrupt acknowledge, special cycle, and reserved commands ignored; memory read multiple and line handled as memory read; memory write and invalidate handled as memory write.
  - — Implements Target abort, disconnect, retry, and wait cycles.
  - — Handles delayed transactions.
  - — Handles fast back-to-back transactions.
  - — Method of handling retries is programmable at FPGA configuration to allow tailoring to different Target data access latencies.
  - — Decodes at medium speed.
  - — Provides option to delay PCI access until FIFO is full on Target reads to increase PCI bandwidth.
- Supports dual-address cycles (both as Master and Target).
- Supports all six base address registers (BARs), as either memory (32-bit or 64-bit) or I/O. Any legal page size can be independently specified for each BAR during FPGA configuration.
- Independent Master and Target clocks can be supplied to the PCI FIFO interface from the FPGA-based logic.
- Provides versatile clocking capabilities with FPGA clocks sourced from PCI bus clock or elsewhere. FIFO interface buffers asynchronous clock domains between the PCI interface and FPGA-based logic.
- PCI interface timing: meets or exceeds 33 MHz, 50 MHz, and 66 MHz PCI requirements.

Parameter	33 MHz	50 MHz	66 MHz
Device Clock = > Out	11.0 ns	7.5 ns	6.0 ns
Device Setup Time	7.0 ns	4.5 ns	3.0 ns
Board Prop. Delay	10.0 ns	6.5 ns	5.0 ns
Board Clock Skew	2.0 ns	1.5 ns	1.0 ns
Total Budget	30.0 ns	20.0 ns	15.0 ns
Load Capacitance	50 pF	50 pF	10 pF

- Configuration options:
  - — Class code, revision ID.
  - — Latency timer.
  - — Cache line size.
  - — Subsystem ID.
  - — Subsystem vendor ID.
  - — Maximum latency, minimum grant.
  - — Interrupt line.
  - — Hot plug/hot swap capability.
- Generates interrupts on INTA# as directed by the FPGA.
- PCI I/O output drivers can be programmed for fast or slew-limited operation.
- Automatically detects 5 V or 3.3 V PCI bus signaling environment and provides appropriate I/O signaling, under 3.3 V commercial or industrial conditions.
- Ideally suited for such applications as:
  - — PCI-based graphics/video/multimedia.
  - — Bridges to ISA/EISA/MCA, LAN, SCSI, Ethernet, ATM, or other bus architectures.
  - — High-bandwidth data transfer in proprietary systems.

FPSC Highlights

- Implemented as an embedded core into the advanced Series 3+ *ORCA* FPSC architecture.
- Allows the user to integrate the core with up to 120K gates of programmable logic, all in one device, and provides up to 259 user I/O pins in addition to the PCI interface pins.
- FPGA portion retains all of the features of the *ORCA* 3 FPGA architecture:
  - — High-performance, cost-effective, 0.25 µm 5-level metal technology.
  - — Twin-quad programmable function unit (PFU) architecture with eight 16-bit look-up tables (LUTs) per PFU, organized in two nibbles for use in nibble- or byte-wide functions. Allows for mixed arithmetic and logic functions in a single PFU.
  - — Softwired LUTs (SWL) allow fast cascading of up to three levels of LUT logic in a single PFU.
  - — Supplemental logic and interconnect cell (SLIC) provides 3-statable buffers, up to 10-bit decoder, and *PAL**-like AND-OR-INVERT (AOI) in each programmable logic cell (PLC).

* *PAL* is a trademark of Advanced Micro Devices, Inc.

FPSC Highlights (continued)

— Up to three ExpressCLK inputs allow extremely fast clocking of signals on- and off-chip plus access to internal general clock routing.
— Dual-use microprocessor interface (MPI) can be used for configuration, readback, device control, and device status, as well as for a general-purpose interface to the FPGA. Glueless interface to *i960** and *PowerPC* † processors with user-configurable address space provided.
— Programmable clock manager (PCM) adjusts clock phase and duty cycle for input clock rates from 15 MHz to 150 MHz. The PCM may be combined with FPGA logic to create complex functions, such as digital phase-locked loops (DPLL), frequency counters, and frequency synthesizers or clock doublers. Two PCMs are provided per device.
— True internal 3-state, bidirectional buses with simple control provided by the SLIC.
— 32 x 4 RAM per PFU, configurable as single or dual port. Create large, fast RAM/ROM blocks (128 x 8 in only eight PFUs) using the SLIC decoders as bank drivers.
— Built-in boundary scan (*IEEE* ‡1149.1 JTAG) and TS_ALL testability function to 3-state all I/O pins.

- High-speed on-chip interface provided between FPGA logic and embedded core to reduce bottlenecks typically found when interfacing off-chip.
- Supported in two packages: 352-pin PBGA and 680-pin PBGAM.

Note: This document will conform to the nomenclature of the PCI Local Bus Specification, as follows:

Term	Meaning
byte	8 bits
word	16 bits
DWORD	32 bits
Quadword	64 bits

Software Support

- Supported by *ORCA* Foundry software and third-party CAE tools for implementing *ORCA* Series 3+ devices and simulation/timing analysis with embedded PCI bus core.
- PCI core configuration options and simulation netlists generated by FPSC Configuration Manager utility in *ORCA* Foundry software.
- Preference files provided for timing interface between PCI bus core and FPGA logic.

* *i960* is a registered trademark of Intel Corporation.
† *PowerPC* is a registered trademark of International Business Machines Corporation.
‡ *IEEE* is a registered trademark of The Institute of Electrical and Electronics Engineers, Inc.

Description

What Is an FPSC?

FPSCs, or field-programmable system chips, are devices that combine field-programmable logic with ASIC or mask-programmed logic on a single device. FPSCs provide the time to market and flexibility of FPGAs, the design effort savings of using soft intellectual property (IP) cores, and the speed, design density, and economy of ASICs.

FPSC Overview

Lucent's Series 3+ FPSCs are created from Series 3 *ORCA* FPGAs. To create a Series 3+ FPSC, several rows of programmable logic cells (see FPGA Logic Overview section for FPGA logic details) are removed from a Series 3 *ORCA* FPGA, and the area is replaced with an embedded logic core. Other than replacing some FPGA gates with ASIC gates, at greater than 10:1 efficiency, none of the FPGA functionality is changed—all of the Series 3 FPGA capability is retained: MPI, PCMs, boundary scan, etc. The rows of programmable logic are replaced at the bottom of the device, allowing pins on the bottom and sides of the replaced rows to be used as I/O pins for the embedded core. The remainder of the device pins retain their FPGA functionality as do special function FPGA pins within the embedded core area.

The embedded cores can take many forms and generally come from Lucent Technologies ASIC libraries. Future offerings will allow customers to supply their own core functions for the creation of custom FPSCs.

FPSC Gate Counting

The total gate count for an FPSC is the sum of its embedded core (standard cell/ASIC gates) and its FPGA gates. Because FPGA gates are generally expressed as a usable range with a nominal value, the total FPSC gate count is sometimes expressed in the same manner. Standard-cell/ASIC gates are, however, 10 to 25 times more silicon area efficient than FPGA gates. Therefore, an FPSC with an embedded function is gate equivalent to an FPGA with a much larger gate count.

FPGA/Embedded Core Interface

The interface between the FPGA logic and the embedded core is designed to look like FPGA I/Os from the FPGA side, simplifying interface signal routing and providing a unified approach with general FPGA design. Effectively, the FPGA is designed as if signals were going off of the device to the embedded core, but the on-chip interface is much faster than going off-chip and requires less power. All of the delays for the interface are precharacterized and accounted for in the *ORCA* Foundry Development System.

Clock spines also can pass across the FPGA/embedded core boundary. This allows for fast, low-skew clocking between the FPGA and the embedded core. Many of the special signals from the FPGA, such as DONE and global set/reset, are also available to the embedded core, making it possible to fully integrate the embedded core with the FPGA as a system.

For even greater system flexibility, FPGA configuration RAMs are available for use by the embedded core. This allows for user-programmable options in the embedded core, in turn allowing for greater flexibility. Multiple embedded core configurations may be designed into a single device with user-programmable control over which configurations are implemented, as well as the capability to change core functionality simply by reconfiguring the device.

ORCA Foundry Development System

The *ORCA* Foundry Development System is used to process a design from a netlist to a configured FPSC. This system is used to map a design onto the *ORCA* architecture and then place and route it using *ORCA* Foundry's timing-driven tools. The development system also includes interfaces to, and libraries for, other popular CAE tools for design entry, synthesis, simulation, and timing analysis.

The *ORCA* Foundry Development System interfaces to front-end design entry tools and provides the tools to produce a configured FPSC. In the design flow, the user defines the functionality of the FPGA portion of the FPSC and embedded core settings at two points in the design flow: at design entry and at the bit stream generation stage.

Description (continued)

Following design entry, the development system's map, place, and route tools translate the netlist into a routed FPSC. A static timing analysis tool is provided to determine device speed and a back-annotated netlist can be created to allow simulation. Timing and simulation output files from *ORCA* Foundry are also compatible with many third-party analysis tools. Its bit stream generator is then used to generate the configuration data which is loaded into the FPSC's internal configuration RAM. When using the bit stream generator, the user selects options that affect the functionality of the FPSC. Combined with the front-end tools, *ORCA* Foundry produces configuration data that implements the various logic and routing options discussed in this data sheet.

FPSC Design Kit

Development is facilitated by an FPSC Design Kit which, together with *ORCA* Foundry and third-party synthesis and simulation engines, provides all software and documentation required to design and verify an FPSC implementation. Included in the kit are the FPSC Configuration Manager, *Verilog* * and *VHDL** gate-level structural netlists, all necessary synthesis libraries, and complete online documentation. The kit's software couples with *ORCA* Foundry under the control of the *ORCA* Foundry Control Center (OFCC), providing a seamless FPSC design environment. More information can be obtained by visiting the *ORCA* website or contacting a local sales office, both listed on the last page of this document.

FPGA Logic Overview

ORCA Series 3 FPGA logic is a new generation of SRAM-based FPGA logic built on the successful Series 2 FPGA line from Lucent Technologies Microelectronics Group, with enhancements and innovations geared toward today's high-speed designs and tomorrow's systems on a single chip. Designed from the start to be synthesis friendly and to reduce place and route times while maintaining the complete routability of the *ORCA* Series 2 devices, the Series 3 more than doubles the logic available in each logic block and incorporates system-level features that can further reduce logic requirements and increase system speed. *ORCA* Series 3 devices contain many new patented enhancements and are offered in a variety of packages, speed grades, and temperature ranges.

ORCA Series 3 FPGA logic consists of three basic elements: programmable logic cells (PLCs), programmable input/output cells (PICs), and system-level features. An array of PLCs is surrounded by PICs. Each PLC contains a programmable function unit (PFU), a supplemental logic and interconnect cell (SLIC), local routing resources, and configuration RAM. Most of the FPGA logic is performed in the PFU, but decoders, *PAL*-like functions, and 3-state buffering can be performed in the SLIC. The PICs provide device inputs and outputs and can be used to register signals and to perform input demultiplexing, output multiplexing, and other functions on two output signals. Some of the system-level functions include the new microprocessor interface (MPI) and the programmable clock manager (PCM).

PLC Logic

Each PFU within a PLC contains eight 4-input (16-bit) look-up tables (LUTs), eight latches/flip-flops (FFs), and one additional flip-flop that may be used independently or with arithmetic functions.

The PFU is organized in a twin-quad fashion: two sets of four LUTs and FFs that can be controlled independently. LUTs may also be combined for use in arithmetic functions using fast-carry chain logic in either 4-bit or 8-bit modes. The carry-out of either mode may be registered in the ninth FF for pipelining. Each PFU may also be configured as a synchronous 32 x 4 single- or dual-port RAM or ROM. The FFs (or latches) may obtain input from LUT outputs or directly from invertible PFU inputs, or they can be tied high or tied low. The FFs also have programmable clock polarity, clock enables, and local set/reset.

The SLIC is connected to PLC routing resources and to the outputs of the PFU. It contains 3-state, bidirectional buffers and logic to perform up to a 10-bit AND function for decoding, or an AND-OR with optional INVERT (AOI) to perform *PAL*-like functions. The 3-state drivers in the SLIC and their direct connections to the PFU outputs make fast, true 3-state buses possible within the FPGA logic, reducing required routing and allowing for real-world system performance.

* *Verilog* and *VHDL* are registered trademarks of Cadance Design Systems, Inc.

Description (continued)

PIC Logic

The Series 3 PIC addresses the demand for ever-increasing system clock speeds. Each PIC contains four programmable inputs/outputs (PIOs) and routing resources. On the input side, each PIO contains a fast-capture latch that is clocked by an ExpressCLK. This latch is followed by a latch/FF that is clocked by a system clock from the internal general clock routing. The combination provides for very low setup requirements and zero hold times for signals coming on-chip. It may also be used to demultiplex an input signal, such as a multiplexed address/data signal, and register the signals without explicitly building a demultiplexer. Two input signals are available to the PLC array from each PIO, and the *ORCA* Series 2 capability to use any input pin as a clock or other global input is maintained.

On the output side of each PIO, two outputs from the PLC array can be routed to each output flip-flop, and logic can be associated with each I/O pad. The output logic associated with each pad allows for multiplexing of output signals and other functions of two output signals.

The output FF, in combination with output signal multiplexing, is particularly useful for registering address signals to be multiplexed with data, allowing a full clock cycle for the data to propagate to the output. The I/O buffer associated with each pad is the same as the *ORCA* Series 3 buffer.

System Features

The Series 3 also provides system-level functionality by means of its dual-use microprocessor interface (MPI) and its innovative programmable clock manager (PCM). These functional blocks allow for easy glueless system interfacing and the capability to adjust to varying conditions in today's high-speed systems. Since these and all other Series 3 features are available in every Series 3+ FPSC, they can also interface to the embedded core providing for easier system integration.

Routing

The abundant routing resources of *ORCA* Series 3 FPGA logic are organized to route signals individually or as buses with related control signals. Clocks are routed on a low-skew, high-speed distribution network and may be sourced from PLC logic, externally from any I/O pad, or from the very fast ExpressCLK pins. ExpressCLKs may be glitchlessly and independently enabled and disabled with a programmable control signal using the new StopCLK feature. The improved PIC routing resources are now similar to the patented intra-PLC routing resources and provide great flexibility in moving signals to and from the PIOs. This flexibility translates into an improved capability to route designs at the required speeds when the I/O signals have been locked to specific pins.

Configuration

The FPGA logic's functionality is determined by internal configuration RAM. The FPGA logic's internal initialization/configuration circuitry loads the configuration data at powerup or under system control. The RAM is loaded by using one of several configuration modes, including serial EEPROM, the microprocessor interface, or the embedded function core.

More Series 3 Information

For more information on Series 3 FPGAs, please refer to the Series 3 FPGA data sheet, available on the *ORCA* worldwide website or by contacting Lucent Technologies as directed on the back of this data sheet.

OR3LP26B Overview

Device Layout

The OR3LP26B FPSC provides a PCI local bus core (with FIFOs) combined with FPGA logic. The device is based on a 2.5 V OR3L125B FPGA. The OR3L125B has a 28 x 28 array of programmable logic cells (PLCs). For the OR3LP26B, the bottom ten rows of PLCs in the array were replaced with the embedded PCI bus core. Table 3 shows a schematic view of the OR3LP26B. The upper portion of the device is an 18 x 28 array of PLCs surrounded on the left, top, and right by programmable input/output cells (PICs). At the bottom of the PLC array are the core interface cells (CICs) connecting to the embedded core region. The embedded core region contains the PCI bus functionality of the device. It is surrounded on the left, bottom, and right by PCI bus dedicated I/Os as well as power and special function FPGA pins. Also shown are the interquad routing blocks (hIQ, vIQ) present in the Series 3 FPGA devices. System-level functions (located in the corners of the PLC array), routing resources, and configuration RAM are not shown in Figure 1.

PCI Local Bus

PCI local bus, or simply, PCI bus, has become an industry-standard interface protocol for use in applications ranging from desktop PC busing to high-bandwidth backplanes in networking and communications equipment. The PCI bus specification* provides for both 5 V and 3.3 V signaling environments. The interface clock speed is specified in the range from dc to 66 MHz with detailed specifications at 33 MHz and 66 MHz as well as recommendations for 50 MHz operation. Data paths are defined as either 32-bit or 64-bit. These data path and frequency combinations allow for the peak data transfer rates described in Table 2.

* PCI Local Bus Specification Rev. 2.2, PCI SIG, December 18, 1998.

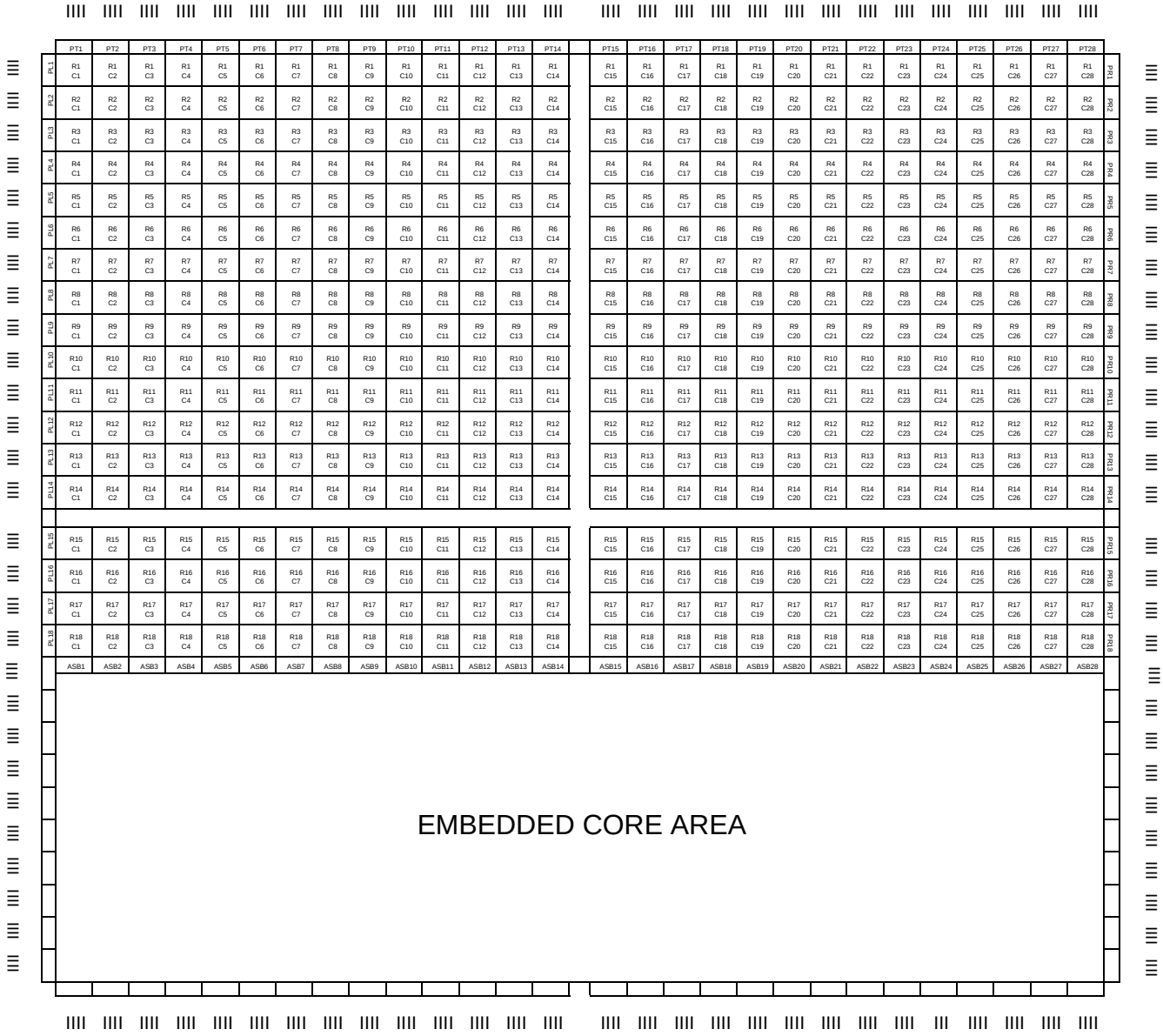
Table 2. PCI Local Bus Data Rates

Clock Frequency (MHz)	Data Path Width (bits)	Peak Data Rate (Mbytes)
33	32	132
33	64	264
66	32	264
66	64	528

The PCI bus is electrically specified so that no glue logic is required to interface to the bus—PCI devices interface directly to the PCI bus. Other features include registers for device and subsystem identification and autoconfiguration, support for 64-bit addressing, and multi-Master capability that allows any PCI bus Master access to any PCI bus Target.

OR3LP26B Overview (continued)

Table 3. OR3LP26B Array



OR3LP26B Overview (continued)

OR3LP26B PCI Bus Core Overview

The OR3LP26B embedded core comprises a PCI bus interface with independent Master and Target controllers, FIFO memories and control logic for data buffering, a dual-/quad-port interface to the FPGA logic which performs data packing and multiplexing, and logic to support embedded core and FPGA configuration. Each of these areas is briefly described in the following paragraphs. A detailed description of all of the features and functionality of the OR3LP26B embedded core is provided in the next section.

PCI Bus Interface

The OR3LP26B PCI bus interface is compliant to Revision 2.2 of the PCI Local Bus specification. It is capable of no-wait-state, full-burst operation at all of the rate/data width combinations described in Table 2 as well as at a 50 MHz specification that provides a speed increase over the 33 MHz specification and a larger bus loading capability than the 66 MHz specification. The OR3LP26B operates in either the 3.3 V or 5 V PCI signaling environment and is automatically configured for the appropriate environment by a PCI bus VIO pin.

Independent Master and Target controllers are provided for use in systems requiring Master/Target or Target only operation. Six 32-bit base address registers (BARs) are provided for choosing the address space of the PCI device, and these six registers can be combined in pairs to produce 64-bit BARs. Dual address cycles are supported in both 32-bit and 64-bit addressing modes. The BARs work in either the I/O or the memory space of the device, and can be configured as prefetchable or nonprefetchable.

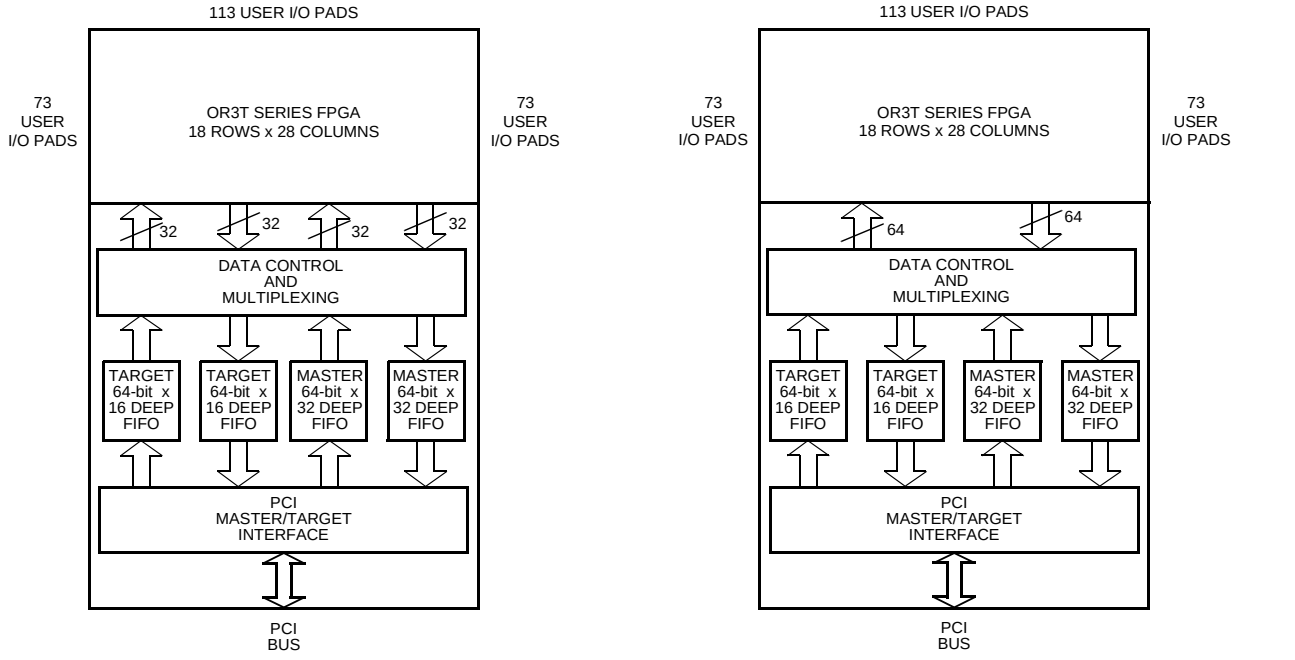
OR3LP26B Overview (continued)

Independent data paths exist for the Master and Target controllers. This allows for separate operation of Master and Target functions, and the capability for a Master to talk to a Target on the same device.

In dual-port mode, the Master and Target controllers share two 64-bit data paths, one in each direction, between the FIFOs and the FPGA logic. This provides for full-rate transfers in both 32- and 64-bit PCI bus operation.

Quad-port mode provides two 32-bit data paths for each controller: one in each direction. This mode allows for simultaneous reads and writes on either the Master or Target controller.

Diagrams for dual-port and quad-port operation are shown in Figure 1.



User I/O pin count includes three ExpressCLK pins.

Figure 1. *ORCA* OR3LP26B PCI FPSC Block Diagram

Embedded Core Options/FPGA Configuration

In addition to the Series 3 FPGA configuration modes (less Master parallel), the OR3LP26B can also be configured via the PCI bus. Configuration as discussed here has two meanings. There is configuration of the FPGA logic, and there is configuration of the options available in the embedded core. Both are accomplished through the FPGA configuration process (some PCI configuration options may also be set via registers within the PCI bus core). Readback of FPGA and PCI core options is also possible using the PCI bus or Series 3 FPGA readback modes. The PCI bus core will be functional in the default PCI bus configuration space, as defined in the PCI bus 2.2 specification, prior to an initial configuration of the FPGA logic or the embedded core options.

PCI Bus Core Detailed Description

The following sections describe the operation of the embedded core PCI bus interface.

PCI Bus Commands

The PCI core supports all commands required by the PCI specification. The following table describes each command. Subsequent sections will describe the protocols in which the commands are used.

Table 4. PCI Bus Command Descriptions

Command Code (Binary)	Command	Master Generates	Target Accepts	Description
0000	Interrupt Acknowledge	—	—	Only implemented as Master by agents that interface to the system CPU and as Target by agents that incorporate the system interrupt controller.
0001	Special Cycle	—	—	Target ignores, per PCI Specification section 3.6.2.
0010	I/O Read	√	√	Fully implemented. **Target:** Bursting is prevented by disconnecting with data on the first data phase. If signal DelTrN is asserted low, I/O (and memory) reads are handled as delayed transactions (on page 68); no wait-states are generated. If signal DelTrN is deasserted high, the unit waits for the data from the FPGA application, inserting wait states (up to the maximum allowed, after which a retry is issued). **Master:** Bursting is allowed, and no wait-states are generated.
0011	I/O Write	√	√	Fully implemented. **Target:** Bursting is prevented by disconnecting with data on the first data phase. If signal DelTrN is asserted low, I/O writes are handled as delayed transactions (on page 57); no wait-states are generated. **Master:** Bursting is allowed, and no wait-states are generated.
0100	(reserved)	—	—	Target ignores, per PCI Specification section 3.1.1.
0101	(reserved)	—	—	Target ignores, per PCI Specification section 3.1.1.
0110	Memory Read	√	√	Fully implemented. **Target:** Bursting is allowed. If signal DelTrN is asserted low, memory (and I/O) reads are handled as delayed transactions (on page 68). If signal DelTrN is deasserted high, the unit waits for the data from the FPGA application, inserting wait-states (up to the maximum allowed, after which a retry is issued). If signal TRBurstPendN is asserted low and the Target Read FIFO is empty, wait states are inserted (up to the maximum allowed, after which a retry is issued). **Master:** Bursting is allowed, and no wait-states are generated.

PCI Bus Core Detailed Description (continued)

Table 4. PCI Bus Command Descriptions (continued)

Command Code (Binary)	Command	Master Generates	Target Accepts	Description
0111	Memory Write	√	√	Fully implemented. **Target:** Writes are posted, bursting is allowed, and no wait-states are generated. **Master:** Bursting is allowed, and no wait-states are generated.
1000	(reserved)	—	—	Target ignores, per PCI Specification section 3.1.1.
1001	(reserved)	—	—	Target ignores, per PCI Specification section 3.1.1.
1010	Configura-tion Read	√	√	Fully implemented. **Target:** Bursting is disallowed, and no wait-states are generated. Target disconnects with data on first data word. The FPGA portion of the device is not involved in Target configuration transactions. **Master:** Bursting is allowed, and no wait-states are generated.
1011	Configura-tion Write	√	√	Fully implemented. **Target:** Bursting is disallowed, and no wait-states are generated. Target disconnects with data on first data word. The FPGA portion of the device is not involved in Target configuration transactions. **Master:** Bursting is allowed, and no wait-states are generated.
1100	Memory Read Multiple	√	√	Fully implemented. Both the Master and the Target treat this instruction the same as a memory read (0110); the user's FPGA logic is responsible for ensuring that the Master operation meets the special requirement that the read request ends on a cacheline boundary.
1101	Dual Access Cycle	√	√	Fully implemented. Per PCI Specification section 3.9, the PCI core will automatically convert a 64-bit address to a 32-bit address if the upper 32 bits are all zeros.
1110	Memory Read Line	√	√	Fully implemented. Both the Master and the Target treat this instruction the same as a memory read (0110); the user's FPGA logic is responsible for ensuring that the Master operation meets the special requirement that the read request continues to the next cacheline boundary.
1111	Memory Write and Invalidate	√	√	Fully implemented. Both the Master and the Target treat this instruction the same as a memory write (0111); the user's FPGA logic is responsible for ensuring that the Master operation meets the special requirement that writes of complete cachelines, with all byte enables, are performed.

PCI Bus Core Detailed Description

(continued)

PCI Protocol Fundamentals

Basic Transfer Control

The following paragraphs describe various aspects of the PCI protocol and the way they are handled by the PCI core.

Addressing. The PCI Specification defines three types of address spaces. The first, configuration address space, is a physical address of space and is intended as a means for powerup software to identify agents and configure them before other address spaces have allocated. The second, I/O address space, is intended for mapping control functions. Control function page sizes in configuration space should be no more than 256 bytes. The third, memory address space, is intended for bulk data transfer. It has features to facilitate this, such as special commands for cache implementation, large page sizes, and mechanisms for prefetching. The PCI core handles all three address space types as both a Master and a Target.

Byte Alignment. On all write operations (configuration, I/O, and memory space, and including the memory write and invalidate instruction), for both the PCI core's Master and Target functions, byte enables are fully implemented from/to the FPGA interface. Note, however, that even though the PCI core implements the ability to control byte enables for the memory write and invalidate instruction, the PCI Specification requires that this instruction assert all byte enables, and this is the FPGA application's responsibility. On read operations, the utility of byte enables is more dubious since the data must be enroute from the PCI bus from Target to Master, at the time that the corresponding byte enables are enroute on the PCI bus Master to Target (unless wait-states are inserted). The PCI core, therefore, does not implement byte enable control for Master or Target reads. Byte enables on master read operations are always asserted, and target ignores the byte enables that are sent, in accordance with PCI Specification requirements.

Device Selection (DEVSEL#)

The target is responsible for responding to a master's request by asserting the PCI bus signal DEVSEL#. DEVSEL# may be asserted one, two, or three clocks after the address phrase of a transaction, corresponding to fast, medium, or slow decode, respectively. The PCI core's target is capable of preforming a medium-speed decode response. The decode response speed has a significant impact on the overall latency and bandwidth of nonburst PCI transactions, but its impact decreases greatly for burst transactions, particularly for burst lengths of the size of the PCI core's FIFOs.

Address/Data Stepping

Stepping is an optional feature added to the PCI Specification to accommodate agents whose bus drive capability is insufficient to handle large groups of signals changing state in one clock cycle. Continuous stepping allows weak drivers multiple cycles for signal transition. Discrete stepping partitions the bus into two or more groups of bits that transition on successive clock cycles. However, stepping exacts a heavy toll on performance, cutting maximum bandwidth by at least 50% and increasing latency. The PCI core is designed for maximum throughput with high-performance buffers, so stepping is unnecessary and not implemented. The wait cycle control, bit 7 of the command register, is therefore hardwired to a zero.

PCI Bus Core Detailed Description

(continued)

Interrupt Acknowledge

The interrupt acknowledge command is a read by the system CPU implicitly addressed to the system interrupt controller. Other agents, including the PCI core, are not required to implement this instruction; the PCI core's Master does not generate it and its Target ignores it.

Arbitration Parking

The PCI Specification requires that all master agents properly handle bus parking, which means that when that agent receives an asserted GNT# without the agent having asserted REQ#, the agent still must drive signal PAR and buses AD and C/BE#. The PCI core meets this requirement.

Parity

The PCI core implements all required and optional features, including the following:

- Master generates parity on all addresses placed on the bus;
- Sending agent generates parity on all data placed on the bus;
- Target calculates parity on all addresses received from the bus;
- Receiving agent calculates parity on all data received from the bus;
- The detected parity error bit in the status register is set whenever an agent calculates corrupted parity;
- The signal PERR# is generated whenever an agent calculates corrupted parity and the parity error response bit is set in the command register.

66 MHz Operation

The PCI core is fully compliant to PCI Specification requirements at all clock rates up to 66 MHz. All 33 MHz requirements are also met.

Timing Budget

The PCI core's timing budget is summarized in Table 5. Note that the 66 MHz timing requirements only allow 5 ns for signal propagation (TPROP), as compared to 10 ns at 33 MHz. The effect of the reduction is to reduce also the number of agents that the bus can support, although the actual number is not specified in the PCI Specification and is dependent on the design of the hardware components. The four components of the timing budget are TVAL (valid output delay), TPROP (propagation time), TSU (input setup time), and TSKEW (clock skew); of these, only TVAL and TSU are controlled by the PCI component, and TPROP and TSKEW are system parameters. Table 5 includes a third column (also shown in the PCI Specification); this column indicates the performance attainable if all 66 MHz requirements are met except TPROP = 10 ns, which is the 33 MHz value. In this case, the total budget increases from 15 ns (66 MHz) to 20 ns (50 MHz).

Table 5. Timing Budgets

Timing Element	33 MHz	50 MHz	66 MHz	Unit
Cycle Time	30.0	20.0	15.0	ns
Valid Output Delay	11.0	7.5	6.0	ns
Propagation Time	10.0	6.5	5.0	ns
Input Setup Time	7.0	4.5	3.0	ns
Clock Skew	2.0	1.5	1.0	ns

64-Bit Addressing

The PCI core fully supports 64-bit addressing, whether or not the PCI core is configured to utilize the 64-bit data extension. When the PCI core is a 64-bit target being addressed by 64-bit master, the PCI core will decode the address one cycle faster so that dual-address operation will have no performance impact; see PCI Specification section 3.9 for details.

Section 3.9 of the PCI Specification also states that a Master that supports 64-bit addressing must nevertheless generate requests utilizing a single address instead of a dual address when the upper 32 bits are all zeros. This shortens the request time by one cycle when communicating with 32-bit Targets. It is the FPGA application's responsibility to ensure that this requirement is met.

FIFO Memories and Control

The OR3LP26B embedded core contains four FIFO memories and supporting control logic. Two FIFOs are for the master interface data and two for the target interface data. These FIFOs are automatically configured to operate in 32-bit and 64-bit modes and also carry byte enable bits on a per-byte basis (e.g., the 64-bit FIFO actually carries 64 bits of data and 8 byte enable bits for a total of 72 bits). All FIFOs have four flags: Full, Almost Full (Full-4), Empty, and Almost Empty (Empty+4). See Table 6. The FPGA application is provided with the Full/Empty signal and Almost Full/Empty signal associated with the FPGA side of the FIFO. In addition, the FPGA application is provided with the PCI side's Full/Empty signal (but not the Almost Full/Empty signal), to enable checking for operation completion. Clocking for the FPGA side of all FIFOs is flexible, with options for different clocks for the Master and Target FIFOs, sourced by the FPGA logic, or by the PCI bus clock.

PCI Bus Core Detailed Description (continued)

Table 6. FIFO Flags Provided to FPGA Application

	Write Operation		Read Operation	
	FPGA Side	**PCI Side**	**FPGA Side**	**PCI Side**
Master Operation	MW_FullN MW_AFullN	MW_EmptyN	MR_EmptyN MR_AEmptyN	MR_FullN
Target Operation	TW_EmptyN TW_AEmptyN	TW_FullN	TR_FullN TR_AFullN	TR_EmptyN

PCI Bus Pin Information

This section describes signals on the PCI bus interface and at the embedded core/FPGA interface. Some signal definitions change name and location based on the mode of operation. Modes of operation are described following the signal descriptions. PCI bus signal package pin locations can be found in Table 35.

Table 7. PCI Bus Pin Descriptions

Symbol	I/O	Description
System Pins		
CLK	I	**Clock.** Provides timing for all transactions on the PCI bus and is an input to the OR3LP26B device. All PCI signals, except RST# and INTA#, are sampled on the rising edge of CLK, and all other PCI bus timing parameters are defined with respect to this edge. CLK operates up to 66 MHz, and the minimum frequency is dc.
RST#	I	**Reset.** An active-low signal used to reset the entire PCI bus. RST# is asynchronous to CLK. During RST#, all PCI output signals are 3-stated.
Address and Data Pins		
AD[31:0]	I/O	**Address and Data.** Multiplexed on the same PCI pins. A PCI bus transaction consists of an address phase followed by one or more data phases. During data phases, AD[7:0] contain the least significant byte and AD[31:24] contain the most significant byte. During memory commands, the AD[31:2] lines specify the address and AD[1:0] specify the type of bursting sequence to use. The table below outlines the bursting sequence based on the values of AD[1:0]. AD[1:0] Bursting sequence. 00 Linear incrementing. 01 Disconnect after first transfer. 10 Disconnect after first transfer. 11 Disconnect after first transfer.
C/BE[3:0]#	I/O	**Bus Command and Byte Enables.** Active-low signals multiplexed on the same PCI pins. During the address phase of a transaction, C/BE[3:0]# define the bus command. During the data phase, C/BE[3:0]# are used as byte enables. The byte enables are valid for the entire data phase and determine which byte lanes carry meaningful data.
PAR	I/O	**Parity.** Specifies even parity across AD[31:0] and C/BE[3:0]#. PAR is stable and valid one clock after the address phase. For data phases, PAR is stable and valid one clock after IRDY# is asserted on a write transaction or TRDY# is asserted on a read transaction. Once PAR is valid, it remains valid until one clock after the completion of the current data phase. The Master drives PAR for address and write data phases; the Target drives PAR for read data phases.

PCI Bus Core Detailed Description (continued)

Table 7. PCI Bus Pin Descriptions (continued)

Symbol	I/O	Description
Interface Control Pins		
FRAME#	I/O	**Cycle Frame.** An active-low signal driven by the current Master to indicate the beginning and duration of an access. FRAME# is asserted to indicate a bus transaction is beginning. While FRAME# is asserted, data transfers continue. When FRAME# is deasserted, the transaction is in the final phase or has completed.
IRDY#	I/O	**Initiator Ready.** An active-low signal indicating the bus Master's ability to complete the current data phase of the transaction. IRDY# is used in conjunction with TRDY#. A data phase is completed on any clock cycle during which both IRDY# and TRDY# are asserted. During a write, IRDY# indicates that valid data is present on AD[31:0]. During a read, it indicates the Master is prepared to accept data. Wait cycles are inserted until both IRDY# and TRDY# are asserted together.
TRDY#	I/O	**Target Ready.** An active-low signal asserted to indicate the readiness of the Target's agent to complete the current data phase of the transaction. TRDY# is used in conjunction with IRDY#. A data phase is completed on any clock where both TRDY# and IRDY# are sampled active. During reads, TRDY# indicates that valid data is present on AD[31:0] lines. During write cycles, TRDY# indicates that the Target is prepared to accept data.
STOP#	I/O	**STOP#.** Indicates that the current Target is requesting the Master to stop the current transaction.
IDSEL	I	**Initialization Device Select.** Used as a chip select during PCI configuration read and write transactions. Generally, the user ties IDSEL to one of the upper 24 address lines, AD[31:8].
DEVSEL#	I/O	**Device Select.** An active-low input indicating that a device on the bus has been selected. As an output, it indicates that the driving device has decoded its address as the Target of the current access.
Arbitration Pins (for Bus Master Only)		
REQ#	O	**Request.** An active-low signal that indicates to the arbiter that the asserting agent desires use of the bus. In the OR3LP26B, this signal is asserted when the OR3LP26B Master controller needs access to the PCI bus.
GNT#	I	**Grant.** An active-low signal that indicates to the OR3LP26B that access to the PCI bus has been granted.
Error Reporting Pins		
PERR#	I/O	**Parity Error.** An active-low signal for the reporting of data parity errors during all PCI transactions except a special cycle. The PERR# pin is a sustained 3-state signal and must be driven active by the agent receiving data two clocks following the data when a data parity error is detected. The minimum duration of PERR# is one clock for each data phase that a data parity error is detected. If sequential data phases each have a data parity error, the PERR# signal will be asserted for more than a single clock. PERR# is driven high for one clock before being 3-stated. PERR# is not asserted until it has claimed the access by asserting DEVSEL# and completed a data phase.

PCI Bus Core Detailed Description (continued)

Table 7. PCI Bus Pin Descriptions (continued)

Symbol	I/O	Description
SERR#	O	**System Error.** An active-low signal pulsed by agents to report errors other than parity. SERR# is sampled every CLK edge, so any agent asserting SERR# must ensure it is valid for at least one clock period. The OR3LP26B asserts SERR# if a Master abort sequence is asserted when the Master controller is accessing the PCI bus.
Interrupt Pins		
INTA#	O	**PCI Interrupt.** The OR3LP26B asserts this active-low signal when it requests an interrupt from the PCI compliant interrupt controller.
64-Bit Bus Extension Pins		
AD[63:32]	I/O	**64-Bit Address and Data.** These signals provide the upper 32 bits of address and data when in PCI 64-bit operation. During an address phase (when using the DAC command and when REQ64# is asserted), these address bits are transferred. During a data phase, the data is valid when REQ64# and ACK64# are both asserted. Otherwise, these bits are 3-stated.
64-Bit Bus Extension Pins		
C/BE[7:4]#	I/O	**Byte Enables.** These are the upper four, active-low, bus command and byte enables when in PCI 64-bit operation. During an address phase (when using the DAC command and when REQ64# is asserted), the bus command is transferred. During a data phase, these bits are the active-low byte enables for data bits 64:32. Otherwise, these bits are 3-stated.
REQ64#	I/O	**Request 64-Bit Transfer.** This active-low signal is asserted by the current bus Master to indicate that it desires to transfer data using 64 bits. REQ64# has the same meaning as FRAME# for 32-bit transfers.
ACK64#	I/O	**Acknowledge 64-Bit Transfer.** The Target drives this signal low to indicate that it has decoded its own address as the Target of the current access and that it can do 64-bit transfers. ACK64# has the same timing as DEVSEL# in 32-bit transfers.
PAR64	I/O	**Upper Double-Word Parity.** The even parity bit that covers AD[63:32] and C/BE[7:4]#. PAR64 is valid one clock after the initial address phase when REQ64# is asserted and the DAC command is indicated on C/BE[3:0]#. It is also valid the clock cycle after the second address phase of a DAC command when REQ64# is asserted.
Hot Swap Function Pins		
ENUM#	O	Active-low signal that notifies the system host that the card has been freshly inserted or is about to be extracted. The system host can then either install (for insertion) or quiesce (for extraction) the card's driver to adjust for the change in system configuration.
LED#	O	Active-low open-drain signal that drives a blue LED, indicating that removal of the card is permitted. This signal is asserted low whenever the LED ON/OFF (LOO) bit in the hot swap control and status register (HSSCR) is asserted high.
EJECTSW	I	Active-high signal that indicates that the card's ejector handle is unseated. This signals that the operator has freshly inserted the card, or will extract the card when the blue LED illuminates. If not used, tie high or low.
VIO	I	**PCI Bus Signaling Environment Voltage.** This input indicates to the PCI core the signaling environment being employed on the PCI bus. The input is tied to the appropriate voltage supply (either 5.0 V or 3.3 V).

PCI Bus Core Detailed Description (continued)

Embedded Core/FPGA Interface Signal Descriptions

In Table 8, an input refers to a signal flowing into the FPGA logic (out of the embedded core) and an output refers to a signal flowing out of the FPGA logic (into the embedded core).

Table 8. Embedded Core/FPGA Interface Signals

Symbol	I/O	Description
Master General Signals		
FPGA_MBusyN	O	**FPGA Master Is Busy.** The FPGA master asserts this active-low signal to indicate to the PCI core that when it gets GNT# from the PCI bus not to release the REQ# signal until FPGA_MBusyN becomes inactive. This is helpful in FPGA applications in which the FPGA Master needs a guaranteed bandwidth on the PCI bus, but is not fast enough to keep the FIFOs full. This signal allows the FPGA master to mix read/write/burst/nonburst transfers easily. Once asserted, this signal needs to remain asserted for a minimum of two CLK cycles.
FPGA_MSysError	I	**FPGA Master Cycle Aborted by PCI Target.** The PCI Master controller in the PCI core asserts this active-high as an indication that the current cycle to the PCI bus has been aborted.
MCfgShiftEnN PCI_MCfg_Stat	O I	MCfgShiftEnN is an active-low signal that determines the data that is output by the PCI core onto signal PCI_MCfg_Stat: MCfgShiftEnN = 1: PCI_MCfg_Stat = wired-OR of all bits below, after being masked by FPGA configuration RAM bits; MCfgShiftEnN = 0: PCI_MCfg_Stat = each bit below, one at a time on successive PciClk rising edges (unmasked), reset when MCfgShiftEnN = 1; Status bits: Data parity error detected, Target abort received, and Master abort received.
Master FIFO Address and Command Register Control Signals		
Symbol	**I/O**	**Description**
MAEnN	O	**Master Command/Address/Burst Length Enable.** This is an active-low signal and is used to enable registering commands, burst length, and start address into the Master address register of the PCI core. This signal must be synchronous to the Master FIFO clock signal. On each rising edge of the clock that this signal is sampled low, command, burst length, and address will be registered.
MA_FullN	I	**Master Address Register Full Flag.** This active-low signal indicates that the Master address register is full and no more addresses can be registered.
MStateCntr[3:0]	I	**Internal State Counter.** Used for Master reads and writes. Details of the Master state machine operation can be found in the PCI Bus Core Detailed Description section of this data sheet.
MFIFOClrN	O	**Master FIFO Clear.** This active-low signal is asynchronously asserted by the FPGA Master to clear all Master FIFOs.
M_Ready	I	**Master Logic Ready.** This active-high signal indicates that the Master logic interfacing to the FPGA logic is ready. This signal will be inactive during PCI bus reset or Master FIFO clears.
MCmd[3:0]	O	**Master Command Code.** Command code for the current Master read/write operation. Refer to Table 10on page 29.
Master Write Data FIFO Signals		
MWDataEnN	O	**Master Write FIFO Data Enable.** This active-low signal enables the registering of bus MWData (quad-port mode) or DataFmFPGA (dual-port mode during Master write operations) into the PCI core Master write data FIFOs on the rising edge of the Master FIFO clock signal. MWDataEnN should not be asserted when the Master write data FIFOs are full, or data may be lost.

PCI Bus Core Detailed Description (continued)

Table 8. Embedded Core/FPGA Interface Signals (continued)

Symbol	I/O	Description
MWPciHold	O	**Master Write PCI Bus Hold.** During burst transfers on the PCI bus, this signal delays the start of the transfer on the PCI bus, allowing the FPGA application to fill the FIFO. The transaction will begin when MWPciHold is deasserted or the FIFO becomes full.
MW_FullN	I	**Master Write Data FIFO Full Flag.** This active-low signal indicates that the Master write data FIFOs are full.
MW_AFullN	I	**Master Write Data FIFO Almost Full Flag.** This active-low signal indicates that only four more empty locations remain in the Master write data FIFOs.
MW_EmptyN	I	**Master Write Data FIFO Empty Flag.** This active-low signal indicates that the Master write data FIFO is empty. Note that this condition is mainly pertinent to the PCI side of the FIFO, but is provided to the FPGA application in order to assist in determining when the Master write operation is complete.
Master Write Data FIFO Signals		
MWData[35:0] (quad-port mode) or DataFmFPGA[63:0], DataFmFPGAx[7:0] (dual-port mode)	O	**Master Command, Master Address, Master Read, Read Burst Count, and Master Write Data.** These buses have different functions, depending on the current operation being performed. The functions are summarized below: Quad-Port Mode / Dual-Port Mode **a. Master Command.** Bus data decoded by the PCI core commands as described in the PCI Bus Core Detailed Description section of this document. (See Table 10) Dual Address Indication: MWData[34] / DataFmFPGAx[2] Repeat Burst Length: MWData[33] / DataFmFPGAx[1] Unused: MWData[15:13] / DataFmFPGA[15:13] Holding Reg. Selector: MWData[12] / DataFmFPGA[12] Master Rd Byte Enables: MWData[31:24] / DataFmFPGA[11:4] Master Command Code: MWData[3:0] / DataFmFPGA[3:0] **b. Master Address (32/64 bits).** Read or Write Address: MWData[31:0] / DataFmFPGA[63:0] Unused: MWData[35:32] / DataFmFPGAx[7:0] **c. Master Read Burst Count (18 bits).** Burst Length[17:16]: MWData[17:16] / DataFmFPGAx[3:2] Burst Length[15:0]: MWData[15:0] / DataFmFPGA[31:16] **d. Master Write Data (32/64 bits with 4/8 byte enables).** Data: MWData[31:0] / DataFmFPGA[63:0] Write Enables: MWData[35:32] / DataFmFPGA[7:0]
MWLastCycN	O	**Master Write Last Data Cycle.** This active-low signal has two functions: **a.** It is asserted low to indicate that the accompanying 32/64 bits of Master read or write address information is the final portion being sent. It can also be asserted prior to any address portion being sent, indicating that the previous address is to be used. **b.** It is asserted low to indicate that the accompanying master write data is the final data for this operation. When more than one cycle is required to transfer a complete data word, this signal is only valid on the last cycle.
Master Read Data FIFO Signals		
MRDataEnN	O	**Master Read FIFO Data Output Enable.** This active-low signal enables the data from the PCI core Master read data FIFOs onto bus MRData (quad-port mode) or DataToF-PGA (dual-port mode during Master read operations) on the rising edge of the Mastrer FIFO clock signal. Valid data will be read from the FIFO whenever it is not empty.
MRData[35:0] (quad-port mode) or DataToFPGA[63:0], DataToFPGAx[7:0] (dual-port mode)	I	**Master Read Data.** Quad-Port Mode / Dual-Port Mode **Master Read Data (32/64 bits)** Data: MRData[31:0] / DataToFPGA[63:0] Unused: MRData[35:32] / DataToFPGAx[7:0]
MR_EmptyN	I	**Master Read Data FIFO Empty.** This active-low signal indicates that the Master read data FIFOs of the PCI core are empty.

PCI Bus Core Detailed Description (continued)

Table 8. Embedded Core/FPGA Interface Signals (continued)

Symbol	I/O	Description
MR_AEmptyN	I	**Master Read Data FIFO Almost Empty.** This active-low signal indicates that only four more data locations are available to be read from the Master read data FIFOs of the PCI core.
MR_FullN	I	**Master Read Data FIFO Full Flag.** This active-low signal indicates that the Master read data FIFO is full. Note that this condition is mainly pertinent to the PCI side of the FIFO, but is provided to the FPGA application in order to assist in determining when the Master read operation is complete.
FPGA_MStopBurstN	O	**Stop Burst Reads.** This active-low signal is used by the FPGA Master to terminate burst reads before completion.
MRLastCycN	I	**Master Read Last Data Cycle.** This active-low signal is asserted to indicate that the accompanying Master read data is the final data for this operation. When more than one cycle is required to transfer a complete data word, this signal is only valid on the last cycle.
Target General Signals		
DiscTimerExpN	I	**Discard Timer Expired.** This active-low signal, when asserted, indicates that the discard timer has expired and the core will now treat the retried delayed transaction as a new transaction. The discard timer is a 15-bit counter which starts its count when a delayed transaction is started.
FPGA_TAbort	O	**Target Abort.** This signal is asserted by the FPGA Target application to abort the current PCI cycle. Once asserted, this signal needs to remain asserted for a minimum of two CLK cycles.
FPGA_TRetryN	O	**Assert Retry.** This active-low signal is asserted by an FPGA Target to the PCI core to send a retry to the PCI bus. Once asserted, this signal needs to remain asserted for a minimum of two CLK cycles.
DelTrN	O	**Target Delayed Transaction.** Used for Target I/O write (page 57) and Target read operations (page 68). Target memory writes are always posted.
TCfgShiftEnN PCI_TCfg_Stat	O I	TCfgShiftEnN is an active-low signal that determines the data that is output by the PCI core onto signal PCI_TCfg_Stat: TCfgShiftEnN = 1: PCI_TCfg_Stat = wired-OR of all bits below, after being masked by FPGA configuration RAM bits; TCfgShiftEnN = 0: PCI_TCfg_Stat = each bit below, one at a time on successive PciClk rising edges (unmasked), reset when TCfgShiftEnN = 1; Status bits: Target abort signaled, system error signaled, and parity error detected.
Target FIFO Address and Command Register Control Signals		
TFIFOClrN	O	**Target FIFO Clear.** This active-low signal is asynchronously asserted by the FPGA Target to clear all Target FIFOs.
TReqN	I	**Target Request from PCI.** This active-low signal is synchronous to the Target FIFO clock signal. The PCI core asserts TReqN as an indication to the Target that a transfer request (either read or write) is pending to the target. As long as there are valid target addresses present in the address FIFO, the TReqN signal will continue to be active.
T_Ready	I	**Target Logic Ready.** This active-high signal indicates that the Target logic interfacing to the FPGA logic is ready. This signal will be inactive during PCI bus reset or Target FIFO clears.

PCI Bus Core Detailed Description (continued)

Table 8. Embedded Core/FPGA Interface Signals (continued)

Symbol	I/O	Description
TAEnN	O	**Target Address and Command Register Output Enable.** This active-low signal enables PCI addresses to be read from the Target address register of the PCI core, and PCI commands to be read from the Target command register. The PCI core will only execute enough address cycles to transfer the address within the matched page (higher-order bits are not stripped).
TCmd[3:0]	I	**Target Command Code.** This bus provides the command code for a new Target operation, and is valid when the FPGA senses TReqN active-low.
BAR[2:0]	I	**Base Address Register Number.** This bus indicates which of the six BARs matched the address for the current Target operation, and is valid when the FPGA senses TReqN active-low. The three 64-bit BARs are designated as numbers 0, 2, and 4.
TStateCntr[3:0]	I	**Internal Target State Counter.** Used for Target reads and writes. Details of the Target state machine operation can be found in the PCI Bus Core Detailed Description of this data sheet.
Target Write Data FIFO Signals		
TWDataEnN	O	**Target Write FIFO Data Enable.** This active-low signal enables data from the PCI core Target write data FIFOs onto bus TWData (quad-port mode) or DataToFPGA (dual-port mode during Target write operations) on the rising edge of the Target FIFO clock signal. Valid data will be read from the FIFO whenever it is not empty.
TWData[35:0] (quad-port mode) or DataToFPGA[63:0], DataToFPGAx[7:0] (dual-port mode)	I	**Target Address and Target Write Data.** These buses have different functions depending on the current operation being performed. The functions are summarized below: Quad-Port Mode / Dual-Port Mode **a. Target Address (32/64 bits).** Read or Write Address: TWData[31:0] / DataToFPGA[63:0] Unused: TWData[35:32] / DataToFPGA[7:0] **b. Target Write Data (32/64 bits with 4/8 byte enables).** Data: TWData[31:0] / DataToFPGA[63:0] Write Enables: TWData[35:32] / DataToFPGA[7:0]
TW_EmptyN	I	**Target Write FIFO Empty.** This signal active indicates that the Target write FIFO is empty.
TW_AEmptyN	I	**Target Write FIFO Almost Empty.** This active-low signal indicates that only four more empty locations are available in the Target write FIFOs.
TW_FullN	I	Target Write Data FIFO Full Flag. This active-low signal indicates that the Target write data FIFO is full. Note that this condition is mainly pertinent to the PCI side of the FIFO, but is provided to the FPGA application in order to assist in determining when the Target write operation is complete.
TWLastCycN	I	**Target Write Last Data Cycle.** This active-low signal has two functions: **a.** It is asserted low to indicate that the accompanying 32/64 bits of Master read or write address information is the final portion being sent. It can also be asserted prior to any address portion being sent, indicating that the previous address is to be used. If MWLastCycN is not asserted on the last possible address portion, a protocol error is declared by asserting ProtocolErr high. **b.** It is asserted low to indicate that the accompanying Target write data is the final data for this operation. When more than one cycle is required to transfer a complete data word, this signal is only valid on the last cycle.
TWBurstPendN	O	**Target Write Burst Data Availability Pending Flag.** This active-low signal directs the PCI core not to immediately disconnect when the Target write FIFO becomes full, but rather to insert PCI bus wait-states (up to the maximum allowed, and then disconnect).

PCI Bus Core Detailed Description (continued)

Table 8. Embedded Core/FPGA Interface Signals (continued)

Symbol	I/O	Description
Target Read Data FIFO Signals		
TRDataEnN	O	**Target Read FIFO Data Enable.** This active-low signal enables the registering of bus TRData (quad-port mode) or DataFmFPGA (dual-port mode during target read operations) into the PCI core Target read data FIFOs on the rising edge of the Target FIFO clock signal. TRDataEnN should not be asserted when the Target read data FIFOs are full, or data may be lost.
TRData[35:0] (quad-port mode) or DataFmFPGA[63:0], DataFmFPGAx[7:0] (dual-port mode)	O	**Target Read Data.** Quad-Port Mode / Dual-Port Mode **Target Read Data (32/64 bits)** Data (32/64 bits): MRData[31:0] / DataFmFPGA[63:0] Unused: MRData[35:32] / DataFmFPGA[7:0]
TR_FullN	I	**Target Read FIFO Full.** This signal is active-low and synchronous to the rising edge of the Target FIFO clock signal. The PCI core asserts this signal to indicate that the Target read FIFOs are full and that no more data can be clocked in.
TR_AFullN	I	**Target Read FIFO Almost Full.** This active-low signal indicates that the Target read FIFO has only four more empty locations available in the FIFOs.
TR_EmptyN	I	**Target Read Data FIFO Empty Flag.** This active-low signal indicates that the Target read data FIFO is empty. Note that this condition is mainly pertinent to the PCI side of the FIFO, but is provided to the FPGA application in order to assist in determining when the Target read operation is complete.
TRPciHold	O	**Target Read PCI Bus Hold**. During burst transfers on the PCI bus, this signal delays the start of the transfer on the PCI bus, allowing the FPGA application to fill the FIFO. The transaction will begin when TRPciHold is deasserted or the FIFO becomes full.
TRLastCycN	I	**Target Read Last Data Cycle.** This active-low signal is asserted to indicate that the accompanying Target read data is the final data for this operation. When more than one cycle is required to transfer a complete data word, this signal is only valid on the last cycle. During a read burst, TRLastCycN may remain inactive for longer than it is required to complete the data transfer. If this occurs, the FPGA Target should continue to write data into the Target read FIFOs unless the incremented address crosses the address decode space of the FPGA Target. The address should be incremented by a double word as long as TRLastCycN is inactive.
TRBurstPendN	O	**Target Read Burst Data Availability Pending Flag.** This active-low signal directs the PCI core not to immediately disconnect when the Target read FIFO becomes empty, but rather to insert PCI bus wait-states (up to the maximum allowed, and then disconnect).
Miscellaneous Signals		
Pci_IntAN	O	**PCI Interrupt Request**. This active-low signal is used to generate a PCI bus interrupt and is forwarded by the PCI core as INTA# onto the PCI bus. Once asserted, this signal needs to remain asserted for a minimum of two CLK cycles.
FClk1 FClk2	O O	**FPGA Clock 1 and 2**. Clocks for use by the PCI core for Master and Target FIFOs. When the PCI clock domain extends into the FPGA, the FPGA may reroute the PCI clock back into FClk1 or FClk2. Both FClk1 and FClk 2 are useable by both the Master FIFOs and Target FIFOs.
PciClk	I	**PCI Clock**. PciClk is synchronous to CLK and may be used by the FPGA logic.
Pci_RstN	I	**PCI Reset for Use by the FPGA Logic**. This active-low signal indicates that a PCI bus reset was received from the PCI bus (RST#).

PCI Bus Core Detailed Description (continued)

Table 8. Embedded Core/FPGA Interface Signals (continued)

Symbol	I/O	Description
FPGA_SysError	O	**System Error**. This pin is used by the FPGA to generate a system error on the PCI bus. This is passed to the PCI bus as SERR#.
PCI_64BIT	I	**PCI Bus in 64-Bit Mode**. This active-high signal indicates that the PCI core detected that it is connected as a 64-bit agent to the PCI bus. This is the result of detecting PCI signal REQ64# as active (low) on the inactive-going (rising) edge of PCI signal RST#. Note that this does not imply that any particular transaction is 64-bit, since each transaction is individually negotiated using PCI signals REQ64# and ACK64#.
FIFO_Sel	O	**FIFO Select.** An active-high signal that is valid in the dual-port modes to select either Master read data (FIFO_Sel = 0) or Target write data (FIFO_Sel = 1).

Embedded Core/FPGA Interface Signal Locations

Table 9 lists the physical locations of all signals on the PCI core/FPGA interface. Separate names are provided for dual-port and quad-port bus signals, since their functionality is port mode dependent.

Table 9. OR3LP26B FPGA/PCI Core Interface Signal Locations

PCI Core/FPGA Interface Site	FPGA Input Signal Name		FPGA Output Signal Name	
	Dual-Port	Quad-Port	Dual-Port	Quad-Port
ASB1A	Pci_RstN		Pci_IntAN	
ASB1B	PCI_64Bit		(unused)	
ASB1C	(unused)		FPGA_SysError	
ASB1D	(unused)		FPGA_MBusyN	
ASB2A	DataToFPGA31	TWData31	DataFmFPGA31	TRData31
ASB2B	DataToFPGA30	TWData30	DataFmFPGA30	TRData30
ASB2C	DataToFPGA29	TWData29	DataFmFPGA29	TRData29
ASB2D	DataToFPGA28	TWData28	DataFmFPGA28	TRData28
ASB3A	DataToFPGA27	TWData27	DataFmFPGA27	TRData27
ASB3B	DataToFPGA26	TWData26	DataFmFPGA26	TRData26
ASB3C	DataToFPGA25	TWData25	DataFmFPGA25	TRData25
ASB3D	DataToFPGA24	TWData24	DataFmFPGA24	TRData24
ASB4A	DataToFPGA23	TWData23	DataFmFPGA23	TRData23
ASB4B	DataToFPGA22	TWData22	DataFmFPGA22	TRData22
ASB4C	DataToFPGA21	TWData21	DataFmFPGA21	TRData21
ASB4D	DataToFPGA20	TWData20	DataFmFPGA20	TRData20
ASB5A	DataToFPGA19	TWData19	DataFmFPGA19	TRData19
ASB5B	DataToFPGA18	TWData18	DataFmFPGA18	TRData18
ASB5C	DataToFPGA17	TWData17	DataFmFPGA17	TRData17
ASB5D	DataToFPGA16	TWData16	DataFmFPGA16	TRData16
ASB6A	DataToFPGAx3	TWData35	DataFmFPGAx3	TRData35
ASB6B	DataToFPGAx2	TWData34	DataFmFPGAx2	TRData34
ASB6C	DataToFPGAx1	TWData33	DataFmFPGAx1	TRData33

PCI Bus Core Detailed Description (continued)

Table 9. OR3LP26B FPGA/PCI Core Interface Signal Locations (continued)

PCI Core/FPGA Interface Site	FPGA Input Signal Name		FPGA Output Signal Name	
	Dual-Port	**Quad-Port**	**Dual-Port**	**Quad-Port**
ASB6D	DataToFPGAx0	TWData32	DataFmFPGAx0	TRData32
ASB7A	DataToFPGA15	TWData15	DataFmFPGA15	TRData15
ASB7B	DataToFPGA14	TWData14	DataFmFPGA14	TRData14
ASB7C	DataToFPGA13	TWData13	DataFmFPGA13	TRData13
ASB7D	DataToFPGA12	TWData12	DataFmFPGA12	TRData12
ASB8A	DataToFPGA11	TWData11	DataFmFPGA11	TRData11
ASB8B	DataToFPGA10	TWData10	DataFmFPGA10	TRData10
ASB8C	DataToFPGA9	TWData9	DataFmFPGA9	TRData9
ASB8D	DataToFPGA8	TWData8	DataFmFPGA8	TRData8
ASB9A	DataToFPGA7	TWData7	DataFmFPGA7	TRData7
ASB9B	DataToFPGA6	TWData6	DataFmFPGA6	TRData6
ASB9C	DataToFPGA5	TWData5	DataFmFPGA5	TRData5
ASB9D	DataToFPGA4	TWData4	DataFmFPGA4	TRData4
CKTOASB9	(unused)		FClk1	
ASB10A	DataToFPGA3	TWData3	DataFmFPGA3	TRData3
ASB10B	DataToFPGA2	TWData2	DataFmFPGA2	TRData2
ASB10C	DataToFPGA1	TWData1	DataFmFPGA1	TRData1
ASB10D	DataToFPGA0	TWData0	DataFmFPGA0	TRData0
ASB11A	TStateCntr0		(unused)	
ASB11B	TStateCntr1		(unused)	
ASB11C	TStateCntr2		(unused)	
ASB11D	PCI_TCfg_Stat		TCfgShiftEnN	
ASB12A	TCmd0		(unused)	
ASB12B	TCmd1		(unused)	
ASB12C	TCmd2		(unused)	
ASB12D	TCmd3		TWBurstPendN	
ASB13A	BAR0		TRBurstPendN	
ASB13B	BAR1		FPGA_TAbort	
ASB13C	BAR2		FPGA_TRetryN	
ASB13D	DiscTimerExpN		DelTrN	
ASB14A	TReqN		TAEnN	
ASB14B	TWLastCycN		TWDataENn	
ASB14C	TW_EmptyN		FIFO_Sel	
ASB14D	TW_AEmptyN		(unused)	
CKFMASB14	PciClk		(unused)	
ASB15A	T_Ready		TFIFOClrN	
ASB15B	TRLastCycN		TRDataEnN	
ASB15C	TR_FullN		(unused)	
ASB15D	TR_AFullN		(unused)	
ASB16A	TW_FullN		TRPciHold	

PCI Bus Core Detailed Description (continued)

Table 9. OR3LP26B FPGA/PCI Core Interface Signal Locations (continued)

PCI Core/FPGA Interface Site	FPGA Input Signal Name		FPGA Output Signal Name	
	Dual-Port	**Quad-Port**	**Dual-Port**	**Quad-Port**
ASB16B	TR_EmptyN		MWPciHold	
ASB16C	MW_EmptyN		FPGA_MStopBurstN	
ASB16D	MR_FullN		(unused)	
ASB17A	MA_FullN		MAEnN	
ASB17B	MW_FullN		MWDataEnN	
ASB17C	MW_AFulln		MWLastCycN	
ASB17D	M_Ready		MRDataEnN	
ASB18A	MRLastCycN		MCmd0	
ASB18B	MR_EmptyN		MCmd1	
ASB18C	MR_AEmptyN		MCmd2	
ASB18D	FPGA_MSysError		MCmd3	
ASB19A	DataToFPGA32	MRData0	DataFmFPGA32	MWData0
ASB19B	DataToFPGA33	MRData1	DataFmFPGA33	MWData1
ASB19C	DataToFPGA34	MRData2	DataFmFPGA34	MWData2
ASB19D	DataToFPGA35	MRData3	DataFmFPGA35	MWData3
CKTOASB19	(unused)		FClk2	
ASB20A	DataToFPGA36	MRData4	DataFmFPGA36	MWData4
ASB20B	DataToFPGA37	MRData5	DataFmFPGA37	MWData5
ASB20C	DataToFPGA38	MRData6	DataFmFPGA38	MWData6
ASB20D	DataToFPGA39	MRData7	DataFmFPGA39	MWData7
ASB21A	DataToFPGA40	MRData8	DataFmFPGA40	MWData8
ASB21B	DataToFPGA41	MRData9	DataFmFPGA41	MWData9
ASB21C	DataToFPGA42	MRData10	DataFmFPGA42	MWData10
ASB21D	DataToFPGA43	MRData11	DataFmFPGA43	MWData11
ASB22A	DataToFPGA44	MRData12	DataFmFPGA44	MWData12
ASB22B	DataToFPGA45	MRData13	DataFmFPGA45	MWData13
ASB22C	DataToFPGA46	MRData14	DataFmFPGA46	MWData14
ASB22D	DataToFPGA47	MRData15	DataFmFPGA47	MWData15
ASB23A	DataToFPGAx4	MRData32	DataFmFPGAx4	MWData32
ASB23B	DataToFPGAx5	MRData33	DataFmFPGAx5	MWData33
ASB23C	DataToFPGAx6	MRData34	DataFmFPGAx6	MWData34
ASB23D	DataToFPGAx7	MRData35	DataFmFPGAx7	MWData35
ASB24A	DataToFPGA48	MRData16	DataFmFPGA48	MWData16
ASB24B	DataToFPGA49	MRData17	DataFmFPGA49	MWData17
ASB24C	DataToFPGA50	MRData18	DataFmFPGA50	MWData18
ASB24D	DataToFPGA51	MRData19	DataFmFPGA51	MWData19
ASB25A	DataToFPGA52	MRData20	DataFmFPGA52	MWData20
ASB25B	DataToFPGA53	MRData21	DataFmFPGA53	MWData21
ASB25C	DataToFPGA54	MRData22	DataFmFPGA54	MWData22
ASB25D	DataToFPGA55	MRData23	DataFmFPGA55	MWData23

PCI Bus Core Detailed Description (continued)

Table 9. OR3LP26B FPGA/PCI Core Interface Signal Locations (continued)

PCI Core/FPGA Interface Site	FPGA Input Signal Name		FPGA Output Signal Name	
	Dual-Port	**Quad-Port**	**Dual-Port**	**Quad-Port**
ASB26A	DataToFPGA56	MRData24	DataFmFPGA56	MWData24
ASB26B	DataToFPGA57	MRData25	DataFmFPGA57	MWData25
ASB26C	DataToFPGA58	MRData26	DataFmFPGA58	MWData26
ASB26D	DataToFPGA59	MRData27	DataFmFPGA59	MWData27
ASB27A	DataToFPGA60	MRData28	DataFmFPGA60	MWData28
ASB27B	DataToFPGA61	MRData29	DataFmFPGA61	MWData29
ASB27C	DataToFPGA62	MRData30	DataFmFPGA62	MWData30
ASB27D	DataToFPGA63	MRData31	DataFmFPGA63	MWData31
ASB28A	MStateCntr0		MFIFOClrN	
ASB28B	MStateCntr1		(unused)	
ASB28C	MStateCntr2		(unused)	
ASB28D	PCI_MCfg_Stat		MCfgShiftEnN	

Table 10. Bit Definitions on FPGA/PCI Core Interface

Bits	Name	Description
A: Dual-Port Master Write		**MStateCntr = 0**
DataFmFPGAx[7:4]	—	Unused
DataFmFPGAx[3]	HR	Holding address register selector: 0 = select HR0 1 = select HR1
DataFmFPGAx[2]	DA	Dual Address Indicator
DataFmFPGAx[1:0]	—	Unused
DataFmFPGA[63:32]	A3 & A2	Address words 3 and 2 (if DA = 1; else unused)
DataFmFPGA[31:0]	A1 & A0	Address words 1 and 0
MCmd[3:0]	MCmd	Master command opcode (Note 1)
B: Dual-Port Master Write, Data		**(MStateCntr = 4)**
DataFmFPGAx[7:0]	BE7—BE0	Byte Enables
DataFmFPGA[63:0]	D7—D0	Data Bytes 7 to 0
C: Quad-Port Master Write (Lower Address Cycle)		**MStateCntr = 0**
MWData[35]	HR	Holding address register selector: 0 = select HR0 1 = select HR1
MWData[34]	DA	Dual Address Indicator
MWData[33:32]	—	Unused
MWData[31:0]	A1 & A0	Address words 1 and 0
MCmd[3:0]	MCmd	Master command opcode (Note 1)

PCI Bus Core Detailed Description (continued)

Table 10. Bit Definitions on FPGA/PCI Core Interface

Bits	Name	Description
D: Quad-Port Master Write (Upper Address Cycle)		**(MStateCntr = 1)**
MWData[35:32]	—	Unused
MWData[31:0]	A3 & A2	Address words 3 and 2
MCmd[3:0]	—	Unused
E: Quad-Port Master Write, Lower Data DWORD		**(MStateCntr = 4)**
MWData[35:32]	BE3—BE0	Byte Enables
MWData[31:0]	D3—D0	Data Bytes 3 to 0
F: Quad-Port Master Write, Upper Data DWORD		**(MStateCntr = 5)**
MWData[35:32]	BE7—BE4	Byte Enables
MWData[31:0]	D7—D4	Data Bytes 7 to 4
G: Dual-Port Master Read (Burst Length Cycle)		**(MStateCntr = 0)**
DataFmFPGAx[7:4]	—	Unused
DataFmFPGAx[3]	HR	Holding address register selector: 0 = select HR0 1 = select HR1
DataFmFPGAx[2]	DA	Dual Address Indicator
DataFmFPGAx[1:0]	—	Unused
DataFmFPGA[63:56]	MRd_BenN	Byte enables
DataFmFPGA[55:50]	—	Unused
DataFmFPGA[49:32]	BL	Burst length
DataFmFPGA[31:0]	A1 & A0	Address words 1 and 0 (unused if 64-bit address required — A1 & A0 supplied in next cycle)
MCmd[3:0]	MCmd	Master command opcode (Note 1)
H: Dual-Port Master Read (64-Bit Address Cycle)		**(MStateCntr = 1)**
DataFmFPGAx[7:0]	—	Unused
DataFmFPGA[63:32]	A3 & A2	Address words 3 and 2
DataFmFPGA[31:0]	A1 & A0	Address words 1 and 0
MCmd[3:0]	—	Unused
J: Dual-Port Master Read, Data		**(MStateCntr = 4)**
DataToFPGAx[7:0]	—	Unused
DataToFPGA[63:0]	D7—D0	Data Bytes 7 to 0

PCI Bus Core Detailed Description (continued)

Table 10. Bit Definitions on FPGA/PCI Core Interface

Bits	Name	Description
K: Quad-Port Master Read (16-Bit Address Cycle)		**(MStateCntr = 0)**
MWData[35]	HR	Holding address register selector: 0 = select HR0 1 = select HR1
MWData[34]	DA	Dual Address Indicator
MWData[33]	SPL = 1	Burst length source 0 = use new burst length 1 = use burst length of previous operation, and only 16-bit address is supplied
MWData[32]	—	Unused
MWData[31:24]	MRd_BenN	Byte enables
MWData[23:16]	—	Unused
MWData[15:0]	A0	Address word 0
MCmd[3:0]	MCmd	Master command opcode (Note 1)
L: Quad-Port Master Read (Burst Length Cycle)		**(MStateCntr = 0)**
MWData[35]	HR	Holding address register selector: 0 = select HR0 1 = select HR1
MWData[34]	DA	Dual Address Indicator
MWData[33]	SPL = 0	Burst length source 0 = use new burst length 1 = use burst length of previous operation
MWData[32]	—	Unused
MWData[31:24]	MRd_BenN	Byte enables
MWData[23:18]	—	Unused
MWData[17:0]	BL	Burst length
MCmd[3:0]	MCmd	Master command opcode (Note 1)
M: Quad-Port Master Read (Lower Address Cycle)		**(MStateCntr = 1)**
MWData[35:32]	—	Unused
MWData[31:0]	A1 & A0	Address words 1 and 0
MCmd[3:0]	—	Unused
N: Quad-Port Master Read (Upper Address Cycle)		**(MStateCntr = 2)**
MWData[35:32]	—	Unused
MWData[31:0]	A3 & A2	Address words 3 and 2
MCmd[3:0]	—	Unused
P: Quad-Port Master Read, Lower Data DWORD		**(MStateCntr = 4)**
MRData[35:32]	—	Unused
MRData[31:0]	D3—D0	Data Bytes 3 to 0

PCI Bus Core Detailed Description (continued)

Table 10. Bit Definitions on FPGA/PCI Core Interface

Bits	Name	Description
Q: Quad-Port Master Read, Upper Data DWORD		**(MStateCntr = 5)**
MRData[35:32]	—	Unused
MRData[31:0]	D7 - D4	Data Bytes 7 to 4
R: Dual-Port Target Write & Read		**(TStateCntr = 0)**
DataToFPGAx[7:4]	—	Unused
DataToFPGAx[3]	Burst_I	Burst Indication
DataToFPGAx[2]	DA	Dual Address Indicator
DataToFPGAx[1:0]	—	Unused
DataToFPGA[63:32]	A3 & A2	Address words 3 and 2
DataToFPGA[31:0]	A1 & A0	Address words 1 and 0
TCmd[3:0]	TCmd	Target command opcode (Note 1)
S: Dual-Port Target Write, Data		**(TStateCntr = 4)**
DataToFPGAx[7:0]	BE7—BE0	Byte Enables
DataToFPGA[63:0]	D7—D0	Data Bytes 7 to 0
T: Dual-Port Target Read, Data		**(TStateCntr = 4)**
DataFmFPGAx[7:0]	—	Unused
DataFmFPGA[63:0]	D7—D0	Data Bytes 7 to 0
U: Quad-Port Target Write & Read (Lower Address Cycle)		**(TStateCntr = 0)**
TWData[35]	Burst_I	Burst Indication
TWData[34]	DA	Dual Address Indicator
TWData[33:32]	—	Unused
TWData[31:0]	A1 & A0	Address words 1 and 0
TCmd[3:0]	TCmd	Target command opcode (Note 1)
V: Quad-Port Target Write & Read (Upper Address Cycle)		**(TStateCntr = 1)**
TWData[35:32]	—	Unused
TWData[31:0]	A3 & A2	Address words 3 and 2
TCmd[3:0]	—	Unused
W: Quad-Port Target Write, Lower Data DWORD		**(TStateCntr = 4)**
TWData[35:32]	BE3—BE0	Byte Enables
TWData[31:0]	D3—D0	Data Bytes 3 to 0
X: Quad-Port Target Write, Upper Data DWORD		**(TStateCntr = 5)**
TWData[35:32]	BE7—BE4	Byte Enables
TWData[31:0]	D7—D4	Data Bytes 7 to 4

PCI Bus Core Detailed Description (continued)

Table 10. Bit Definitions on FPGA/PCI Core Interface

Bits	Name	Description
Y: Quad-Port Target Read, Lower Data DWORD		**(TStateCntr = 4)**
TRData[35:32]	—	Unused
TRData[31:0]	D3—D0	Data Bytes 3 to 0
Z: Quad-Port Target Read, Upper Data DWORD		**(TStateCntr = 5)**
TRData[35:32]	—	Unused
TRData[31:0]	D7—D4	Data Bytes 7 to 4

Note 1: Command Codes (codes correspond to PCI bus command codes):
0000 Not Used (interrupt acknowledge not implemented)
0001 Not Used (special cycle not implemented)
0010 I/O Read
0011 I/O Write
0100 Reserved (per PCI specification)
0101 Reserved (per PCI specification)
0110 Memory Read
0111 Memory Write
1000 Reserved (per PCI specification)
1001 Reserved (per PCI specification)
1010 Configuration Read
1011 Configuration Write
1100 Memory Read Multiple
1101 Not Used (dual address operation is indicated via separate signal)
1110 Memory Read Line
1111 Memory Write and Invalidate

PCI Bus Core Detailed Description (continued)

Table 11. Address Cycle Sequences for Various Operations

Operation	Port Mode	Address Mode	Supplied Address	New Burst Length	Address Cycle Sequence (Once Only)	Data Cycle Sequence (Repeats)
Master Write	Dual	SA	31:0	NA	A	B
		DA	63:0	NA	A	B
	Quad	SA/DA	31:0	NA	C	E, F
		DA	63:0	NA	C, D	E, F
Master Read	Dual	SA/DA	31:0	Yes*	G	J
		DA	63:0	Yes*	G, H	J
	Quad	SA/DA	15:0	No	K	P, Q
		SA/DA	(none)	Yes	L	P, Q
		SA/DA	31:0	Yes	L, M	P, Q
		DA	63:0	Yes	L, M, N	P, Q
Target Write	Dual	SA	31:0	NA	R	S
		DA	63:0	NA	R	S
Target Read	Dual	SA	31:0	NA	R	T
		DA	63:0	NA	R	T
Target Write	Quad	SA/DA	31:0	NA	U	W, X
		DA	63:0	NA	U, V	W, X
Target Read	Quad	SA/DA	31:0	NA	U	Y, Z
		DA	63:0	NA	U, V	Y, Z

* Burst length must be supplied

PCI Bus Core Detailed Description (continued)

Embedded Core Bit Stream Configurable Options

Table 12 lists all optional functionality in the PCI core that can be defined via bits in the FPGA configuration RAM. The table also lists the settings available for each feature. Each of these options is configured using the FPSC Design Kit software.

Table 12. PCI Core Options Settable via FPGA Configuration RAM Bits

	Address in Configuration Space	Optional Settings
Revision ID	08	Any 8-bit value.
Class Code	09—0B	Any 24-bit value.
Bus Master Support	Command register bit 2	Four options. ■ Initially disabled, read-only. ■ Initially disabled, read/write. ■ Initially enabled, read-only.
Report: Data Parity Error Detected	Status register bit 8	Include or exclude in decode for PCI_MCfg_Stat.
Report: Target Abort Signaled	Status register bit 11	Include or exclude in decode for PCI_TCfg_Stat.
Report: Target Abort Received	Status register bit 12	Include or exclude in decode for PCI_MCfg_Stat.
Report: Master Abort Received	Status register bit 13	Include or exclude in decode for PCI_MCfg_Stat.
Report: System Error Signaled	Status register bit 14	Include or exclude in decode for PCI_TCfg_Stat.
Report: Parity Error Detected (nonmaskable)	Status register bit 15	Include or exclude in decode for PCI_TCfg_Stat.
Latency Timer Initial Value	OD	Any 8-bit value divisible by 8.
Base Address Register (BAR) Area 1	10—17	■ One or two 32-bit BARs or one 64-bit BAR, or none (i.e., unprogrammed). ■ If 64-bit BAR, must be memory; page size can be from 2^4 to 2^{64} bytes. ■ 32-bit BARs can be memory or I/O. ■ If 32-bit I/O BAR, page size can be from 2^2 to 2^{32} bytes. ■ If 32-bit memory BAR, address space can be 2^{20} or 2^{32} bytes, page size can be 2^4 to the maximum (2^{20} or 2^{32}) bytes. ■ If memory, can be prefetchable or nonprefetchable.
Base Address Register (BAR) Area 2	18—1F	Same as for BAR area 1.
Base Address Register (BAR) Area 3	20—27	Same as for BAR area 1.
Subsystem Vendor ID	2C—2D	Any 16-bit value.
Subsystem ID	2E—2F	Any 16-bit value.
Minimum Grant (Min_Gnt)	3E	Any 8-bit value.
Maximum Latency (Max_Lat)	3F	Any 8-bit value.
Port Mode		Dual-port or quad-port.
I/O Mode		Fast or slew-limited PCI output buffers.
Master FIFO Interface Clock		FClk1 or FClk2.
Target FIFO Interface Clock		FClk1 or FClk2.
Target Address Comparator		Enabled or disabled; when enabled, PCI core will not transfer most significant byte(s) of Target address if they match previous Target operation's address and require additional bus cycle(s).
Target Maximum Intial Latency		Normal (16) or extended (32); note that only normal latency complies with PCI Specification. Extended latency may be specified in proprietary systems where bandwidth requirements override fairness considerations.

PCI Bus Core Detailed Description (continued)

Embedded Core/FPGA Interface Operation

Dual Master Address Holding Registers

The PCI core utilizes a pair of address holding registers to reduce latency when setting up repeated Master transfers to or from the same address. Every Master operation has associated with it one of the two holding registers, as specified by the holding register selector signal (as described in Table 10). Each address holding register records the full previous address, allowing some, all, or none of that recorded address to be used to build the next address associated with that holding register. This can save up to 2 cycles for quad-port mode. The holding register optionally supplies the most significant portion, or all, or none, of the address. The amount supplied by the holding register is determined by the timing of the signal MWLastCycN, which accompanies the last portion of data, or accompanies the command word when the holding register supplies the entire address. Table 13 below gives examples in quad-port, 64-bit addressing mode, of typical operation using the holding registers, illustrating the above rules.

The two holding registers can be assigned one to read and one to write, thus providing two unrelated areas for the two functions. Another useful application is to dedicate one register to a fixed address such as the beginning of a buffer, the data port of a FIFO or a mailbox register. This especially increases effective bandwidth on shorter bursts.

Table 13. Holding Registers, Examples of Typical Operation

Address on Bus MWData		Last Cycle Valid With:	Holding Register Select	Holding Register 0 Initial Value		Holding Register 1 Initial Value		Master Read/Write Address	
AU	AL			AU	AL	AU	AL	AU	AL
1111-1111	2222-2222	AU	0	xxxx-xxxx	xxxx-xxxx	xxxx-xxxx	xxxx-xxxx	1111-1111	2222-2222
—	3333-3333	AL	0	1111-1111	2222-2222	xxxx-xxxx	xxxx-xxxx	1111-1111	3333-3333
4444-4444	5555-5555	AU	1	1111-1111	3333-3333	xxxx-xxxx	xxxx-xxxx	4444-4444	5555-5555
—	—	Cmd	0	1111-1111	3333-3333	4444-4444	5555-5555	1111-1111	3333-3333
—	6666-6666	AL	0	1111-1111	3333-3333	4444-4444	5555-5555	1111-1111	6666-6666
—	—	Cmd	1	1111-1111	6666-6666	4444-4444	5555-5555	4444-4444	5555-5555
—	7777-7777	AL	1	1111-1111	6666-6666	4444-4444	5555-5555	4444-4444	7777-7777
8888-8888	9999-9999	AU	0	1111-1111	6666-6666	4444-4444	7777-7777	8888-8888	9999-9999

Target Address Holding Register and BAR Number Indicator

The PCI core provides two features that reduce overhead on setup of Target transfers in quad-port 64-bit addressing mode.

First, the PCI core's Target control logic detects the page size of the base address register (BAR) that matched the current PCI address, and only transfers the address bytes necessary to send the page address, and not the virtual address of the page, to the FPGA application.

Second, the PCI core utilizes an optional address holding register so that only the least significant portion of the address that is different from the previous address is sent to the FPGA application. Utilization of this feature usually reduces the amount of address that must be transferred, but may require that the FPGA application build a copy of the holding register in order to reconstruct the address. For this reason, this feature is optional and can be disabled via a bit in the FPGA configuration manager.

PCI Bus Core Detailed Description (continued)

Embedded Core/FPGA Interface Operation Summary

The following sections describe the FIFO bus operation which is the interface between the embedded core and the FPGA logic. Several configurations are possible for the FIFO bus and the signal definitions can change for different modes. Tables are provided to define the modes, the signal definitions, and states of each operation for each mode.

Table 14 is an index to the state tables and timing figures provided for each of the operational modes of the FPGA interface to the PCI core. Each of these operations is detailed on the pages shown in the table.

Table 14. Index to State Sequence Tables

					Dual-Port Mode		Quad-Port Mode	
Master/ Target	**PCI Bus Mode**	**Transaction Type**	**Single/Burst and Delayed/Not Delayed**	**PCI Bus Timing Figure Number**	**State Table**	**FPGA Bus Timing Figure Number**	**State Table**	**FPGA Bus Timing Figure Number**
Master	Write	Config, Memory, I/O	Nonburst	Figure 3	Table 15	Figure 2	Table 16	Figure 6
			Burst	Figure 5		Figure 4		Figure 7
	Read	Config, Memory, I/O	Nonburst	Figure 9	Table 17† Table 18‡	Figure 8	Table 19§ Table 20†	Figure 10
			Burst	Figure 12		Figure 11		Figure 13
Target	Write	Config	Nonburst	Figure 14	Table 21	*	Table 22	*
		I/O	Delayed	Figure 15		Figure 17		Figure 18
		Memory, I/O	Nonburst, Not Delayed	Figure 16		Figure 20		Figure 21
		Memory	Burst	Figure 19				
	Read	Config	Nonburst	Figure 22	Table 23	*	Table 24	*
		I/O	Delayed	Figure 23		Figure 26		Figure 27
			Not Delayed	Figure 24				
		Memory	Nonburst	Figure 28				
			Nonburst Delayed	Figure 25				
			Burst	Figure 32		Figure 30		Figure 31
			Burst Delayed	Figure 29				

* The FPGA interface does not participate in Target configuration operations.
† 64-bit address supplied.
‡ 32-bit address supplied.
§ Duplicate burst length and 16-bit address.

PCI Bus Core Detailed Description (continued)

State Sequence Table Summary

The state sequence tables contain notes that have a common numbering scheme to facilitate readability. The Master list of notes is provided below, and the notes pertinent to each table are listed following that table.

1. MAEnN is deasserted high; the interface is idle.
2. MAEnN is asserted low; a Master command is being initiated and data is being transferred.
3. When TReqN is deasserted high, the interface is idle.
4. When TReqN is asserted low, a Target command is pending.
5. MAEnN must be asserted low for data to transfer and state of MStateCntr to change (Master Address).
6. TAEnN must be asserted low for data to transfer and state of TStateCntr to change (Target Address).
7. MAEnN must be deasserted high and MWDataEnN must be asserted low for data to transfer and state of MStateCntr to change (Master write data).
8. MAEnN must be deasserted high and MRDataEnN must be asserted low for data to transfer and state of MStateCntr to change (Master read data).
9. TAEnN must be deasserted high and TWDataEnN must be asserted low for data to transfer and state of TStateCntr to change (Target write data).
10. TAEnN must be deasserted high and TRDataEnN must be asserted low for data to transfer and state of TStateCntr to change (Target read data).
11. Next state = 0 if MWLastCycN is asserted low (end of Master write data).
12. Next state = 0 if MRLastCycN is asserted low (end of Master read data).
13. Next state = 0 if TWLastCycN is asserted low (end of Target write data).
14. Next state = 0 if TWLastCycN is asserted low (end of Target read data).
15. Next state = 4 if MWLastCycN is asserted low (end of Master address).
16. Next state = 4 if TWLastCycN is asserted low (end of Target address).
17. MWLastCycN must be asserted low (end of Master address).
18. TWLastCycN must be asserted low (end of Target address).

PCI Bus Core Detailed Description

(continued)

Master (FPGA Initiated) Write

Operation Setup

In order to initiate a PCI Master write operation, the FPGA application must supply the required information in the specific order prescribed in Table 15 and Table 16. A master command word and address must be accompanied by assertion of the enable MAEnN. The definition of the Master command word is shown in Table 10. The FPGA application can use the value returned on bus MStateCntr, the Master write counter's present value, to determine the counter's next state, using the state diagram for the particular operation being executed. The counter's next state must be determined because the FPGA application must supply the data to the PCI core that corresponds to the counter value being sent from the core to the FPGA.

Master State Counter

The PCI core provides a state counter, MStateCntr[3:0], that informs the FPGA of the current state of the PCI core's Master state counter. This state counter determines what data is currently being provided by the PCI core or expected from the FPGA application. This state counter transitions from one state to another in a predictable fashion, and thus, it is not strictly necessary to transmit its value to the FPGA. Nonetheless, the value on bus MStateCntr can be used to minimize FPGA logic or verify proper operation.

The data provided by the PCI core to the FPGA application on bus MRData or DataToFPGA is accompanied by a value on bus MStateCntr. This value can be directly used by the FPGA application to determine the proper use of that data. This eliminates the need for logic in the FPGA to duplicate this state counters in this case.

The data required from the FPGA application by the PCI core on bus MWData or DataFmFPGA is also defined by the value on bus MStateCntr. However, the state counter value is being sent to the FPGA in the same cycle that the data must be sent from the FPGA. Therefore, the FPGA application must build its own copy of the state counter value in this case. The value provided by the PCI core can be used as the previous value, or it can be used to verify the proper operation of the FPGA application's logic.

Table 10 lists the values of the state counter MStateCntr and the appropriate accompanying data. Table 15—Table 20 detail the various Master read/write operations, the sequencing of the state counters, and the data transferred between the PCI core and the FPGA application on each cycle.

Data Transfer

The FPGA application begins supplying the write data by deasserting MAEnN and asserting MWDataEnN. On every cycle that MWDataEnN is asserted, the PCI core clocks data and its associated byte enables into the Master write FIFO (64 deep by 36 bits wide in 32-bit PCI mode; 32 deep by 72 bits wide in 64-bit PCI mode) via bus MWData (quad-port mode) or DataFmFPGA (dual-port mode).

FIFO Full/Almost Full

When the Master write FIFO contains four or fewer empty locations, the PCI core asserts MW_AFullN, the almost full indicator. This allows some latency to exist in the FPGA's response without risking overfilling the FIFO. When all locations in the Master write FIFO are full, the PCI core asserts MW_FullN, the FIFO full indicator. Since data can be simultaneously written to and read from the Master write FIFO, both MW_AFullN and MW_FullN can change states in either direction multiple times in the course of a burst transfer.

PCI Bus Core Detailed Description

(continued)

FIFO Empty

In addition to the full and almost full signals that report when the Master write FIFO is currently unable to receive data from the FPGA application, the PCI core also provides the FIFO's empty signal. This signal informs the FPGA application of completion of the current transfer. After all necessary data has been written to the FIFO, the FPGA application monitors MW_EmptyN; when it is asserted low, the transfer is complete on the PCI bus.

Bursting

Instead of using a burst length, the Master write operation relies on the Master write burst signal, MWBurst, to inform the PCI core on a cycle-by-cycle basis when additional burst data is to follow. This allows the FPGA application to maintain control over the length of the Master write burst for as long as possible, but may require the FPGA application to implement a burst length counter if needed. When executing a burst Master write, a deasserted MWLastCycN must accompany every data element except the last element on bus MWData (quad-port mode) or DataFmFPGA (dual-port mode); MWLastCycN must remain asserted throughout a nonburst Master write, since the last data phase is the only data phase.

Termination

Once initiated, Master write operations will repeat on the PCI bus until either: 1) all data is sent; 2) an abort occurs (either Master or Target); 3) the Master FIFO clear signal (MFIFOClrN) is asserted; or 4) the PCI bus's reset signal (RESET#) is asserted. If a PCI transaction is terminated with a retry or disconnect before all data has been written, the PCI core will initiate another Master write operation, continuing from that point.

Reset

The FPGA application can apply the PCI core's reset signal to place the core's logic in a known state. The reset signal, MFIFOClrN, is asynchronous and therefore should be asserted for a minimum of two Master FIFO clock cycles and two concurrent PCI clock cycles. The FIFO must not be utilized until MFIFOClrN is deasserted for a minimum of four concurrent clock cycles.

PCI Bus Core Detailed Description (continued)

Master Write, Nonburst Transaction

Figure 2 (FPGA Bus Dual Port), Figure 6 (FPGA Bus Quad Port), and Figure 3 (PCI Bus) show the timing of a Master write, non-burst transaction. In Figure 2, the transaction is initiated by the FPGA application asserting Master address enable (MAEnN), while providing the command word and the address on bus DataFmFPGA. On the next clock, MAEnN is deasserted and the one Quadword of data is provided on bus DataFmFPGA along with assertion of the Master write data enable (MWDataEnN). Since the protocol for providing start-up data is fixed for a specific operation, the FPGA application can be preprogrammed with the sequence, or can use the value of the Master state counter (MStateCntr) to assist in determination of the next required data word of information. The PCI core knows that this is a nonburst operation because the FPGA application asserts the Master write burst signal (MWLastCycN). This completes the setup for this operation. Execution begins on the PCI bus, as shown in Figure 3.

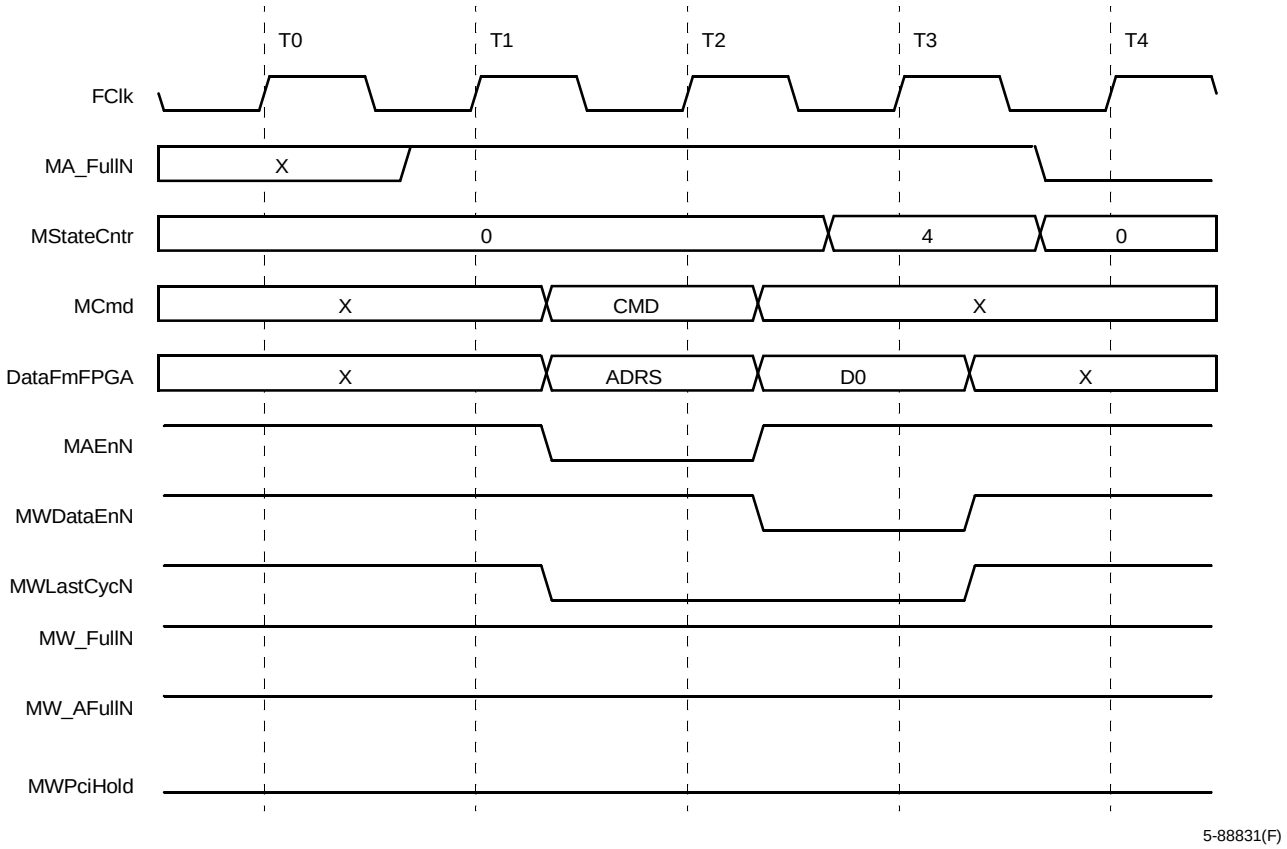


Figure 2. Master Write Single (FPGA Bus, Dual Port)

PCI Bus Core Detailed Description (continued)

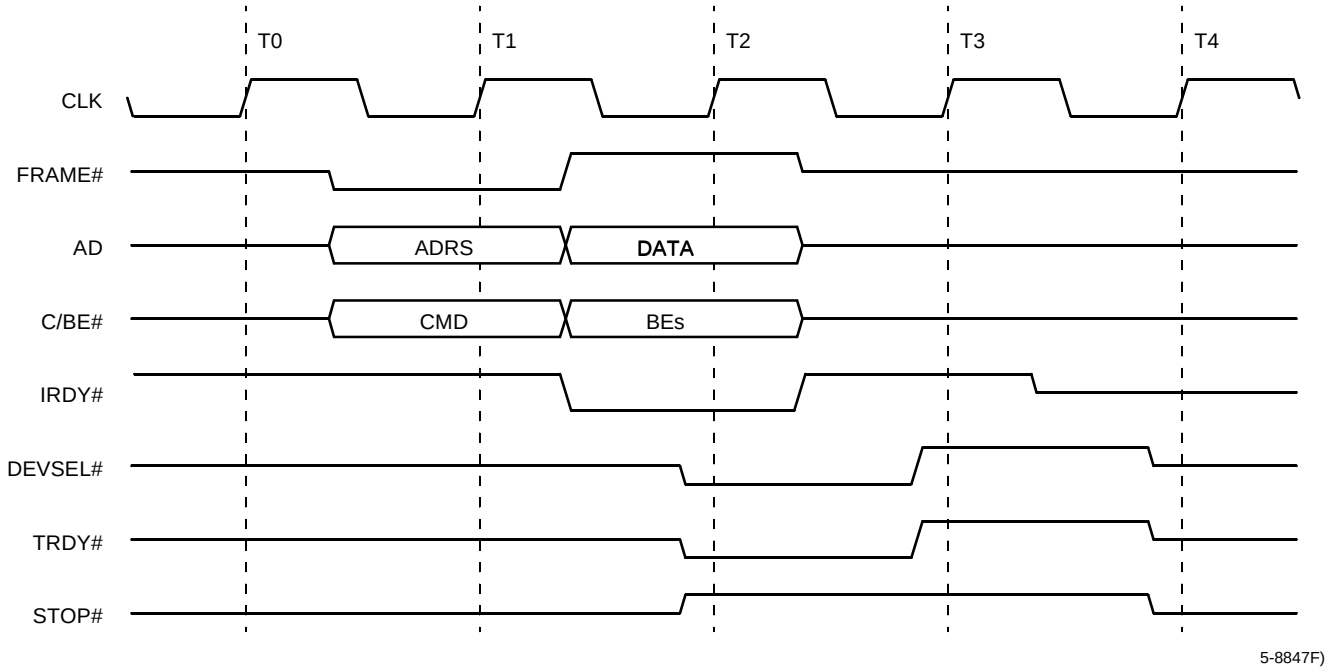


Figure 3. Master Write Single (PCI Bus, 64-Bit)

Master Write, Burst Transaction

Figure 4 (FPGA Bus Dual Port), Figure 7 (FPGA Bus Quad Port), and Figure 5 (PCI Bus) show the timing of a 4-Quadword Master write burst transaction. Operation is similar to that in the previous Master write, non-burst transaction, but extra data is supplied by the FPGA application. In Figure 4, the transaction is initiated by the FPGA application asserting Master address enable (MAEnN), while providing the command word and address on bus DataFmFPGA. On the second through fifth clocks, MAEnN is deasserted, the Master write data enable (MWDataEnN) is asserted, and four Quadwords of data are provided on bus DataFmFPGA. Since the protocol for providing start-up data is fixed for a specific operation, the FPGA application can be preprogrammed with the sequence, or can use the value of the Master state counter (MStateCntr) to assist in determination of the next required Quadword of information. The PCI core knows that this is a burst operation because the FPGA application deasserts the Master write burst signal (MWLastCycN) during all but the final data transfer cycle. Execution begins on the PCI bus, as shown in Figure 5. If the Master write PCI bus hold signal (MWPciHold) is inactive, PCI bus activity will begin when the Master write FIFO goes nonempty; otherwise, the PCI bus activity will wait until all data is loaded, as in this case, or the FIFO goes full. Execution begins on the PCI bus, as shown in Figure 5.

PCI Bus Core Detailed Description (continued)

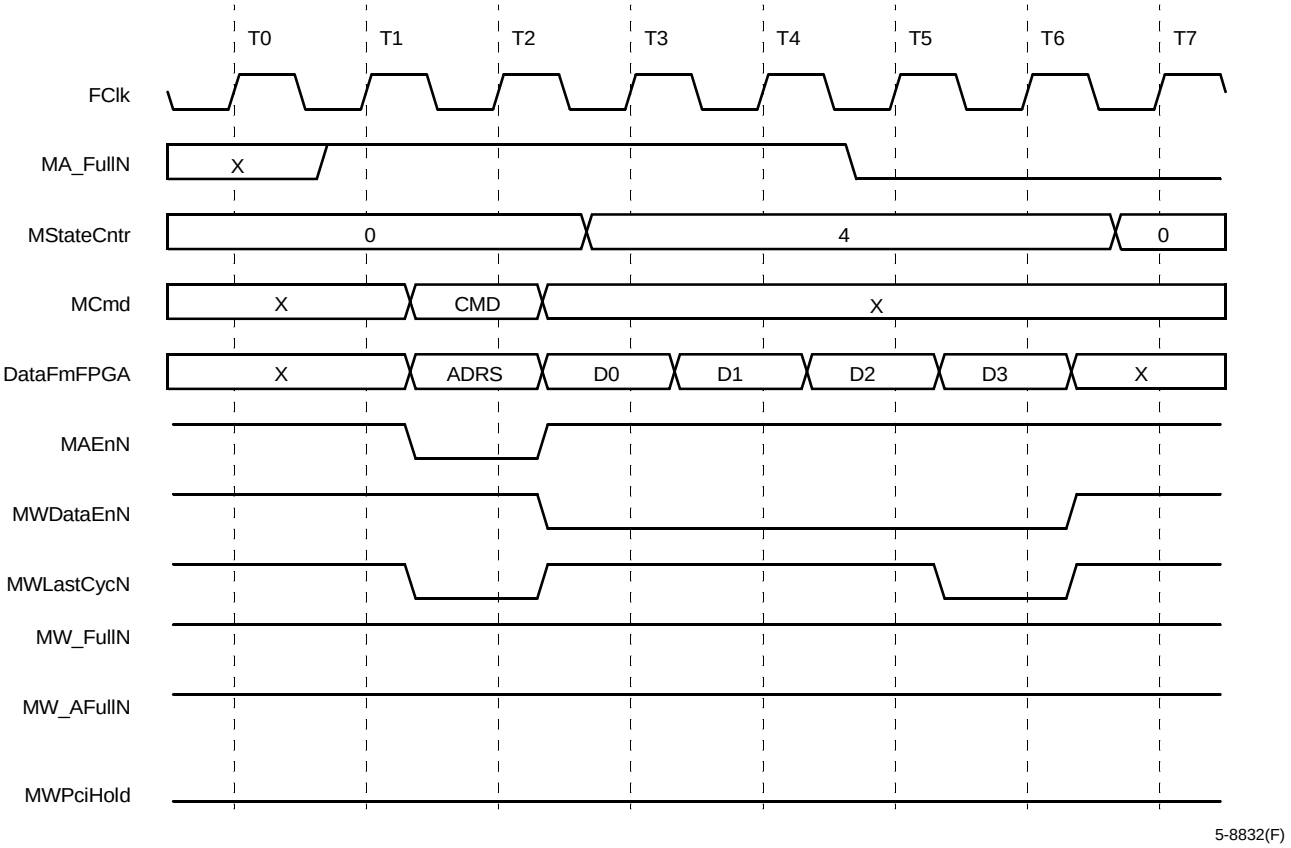


Figure 4. Master Write 32-Byte Burst (FPGA Bus, Dual Port)

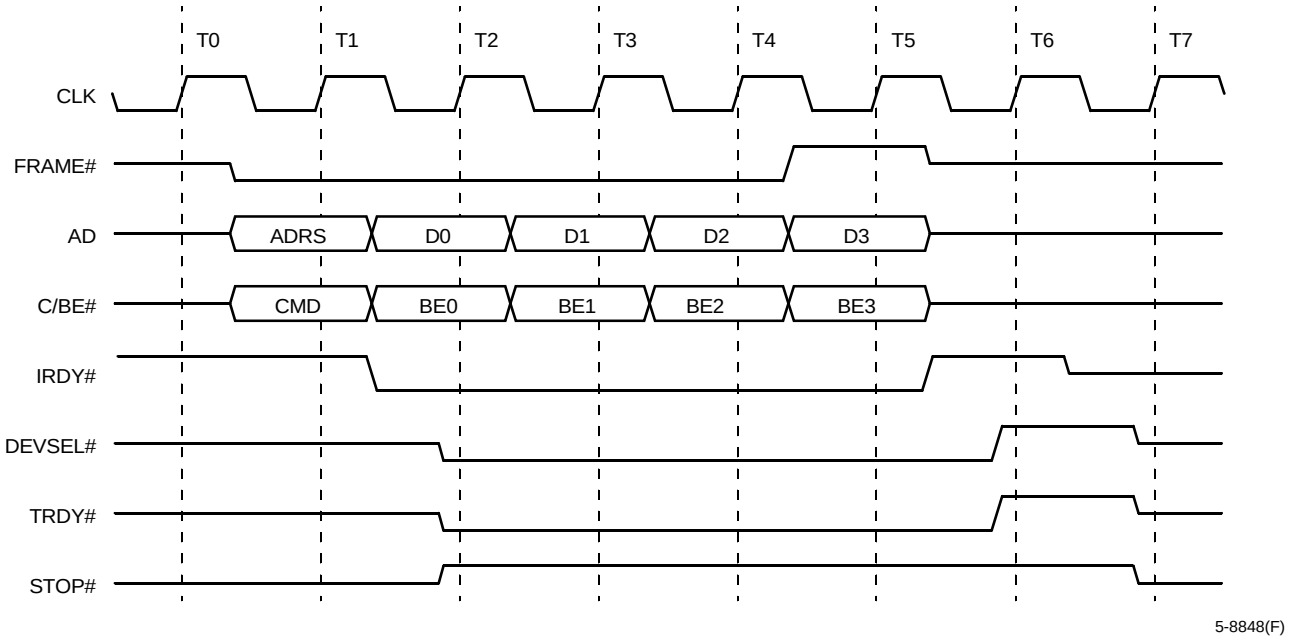


Figure 5. Master Write 32-Byte Burst (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

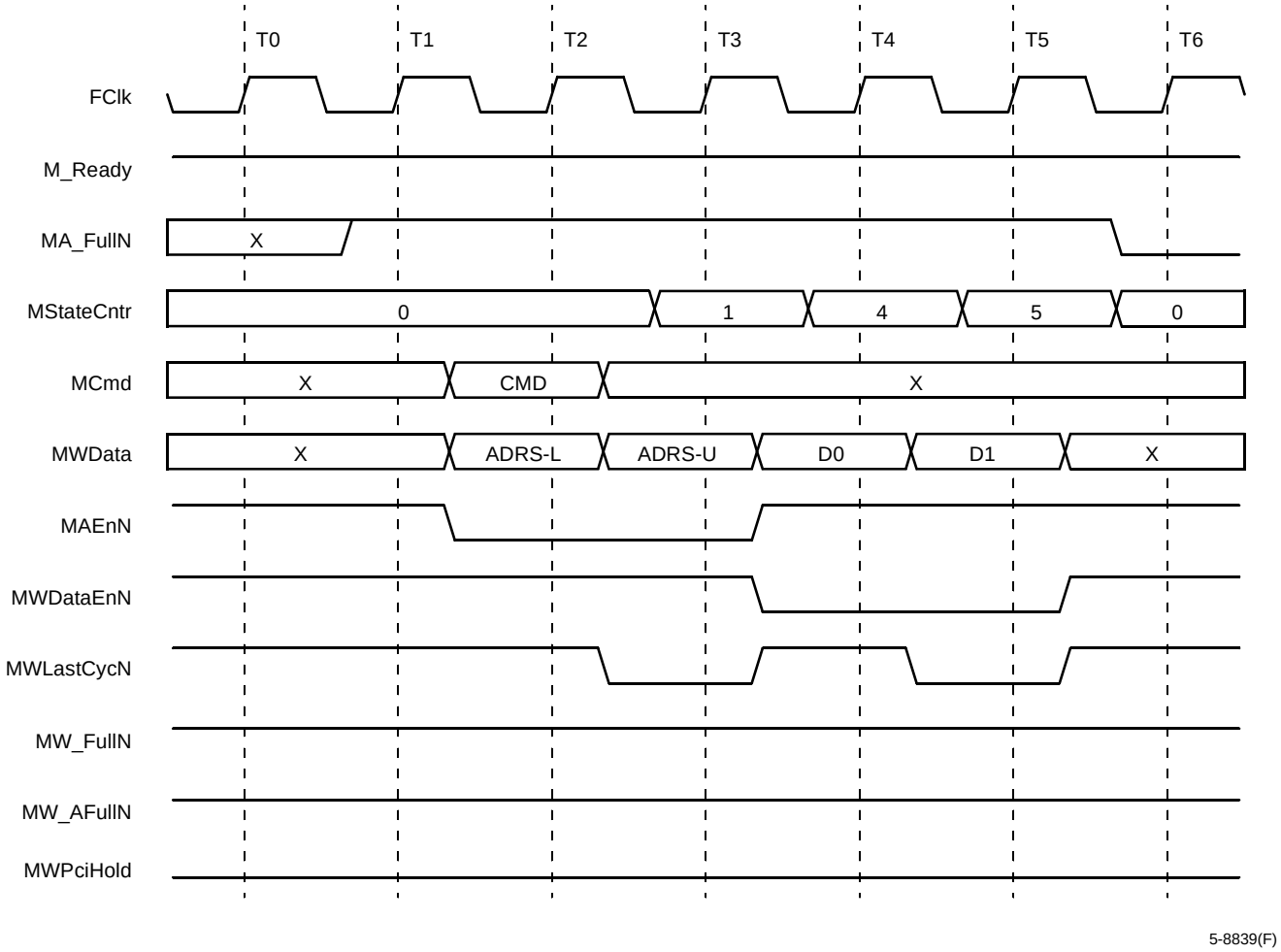


Figure 6. Master Write Single Quadword (FPGA Bus, Quad Port, 64-Bit Address)

PCI Bus Core Detailed Description (continued)

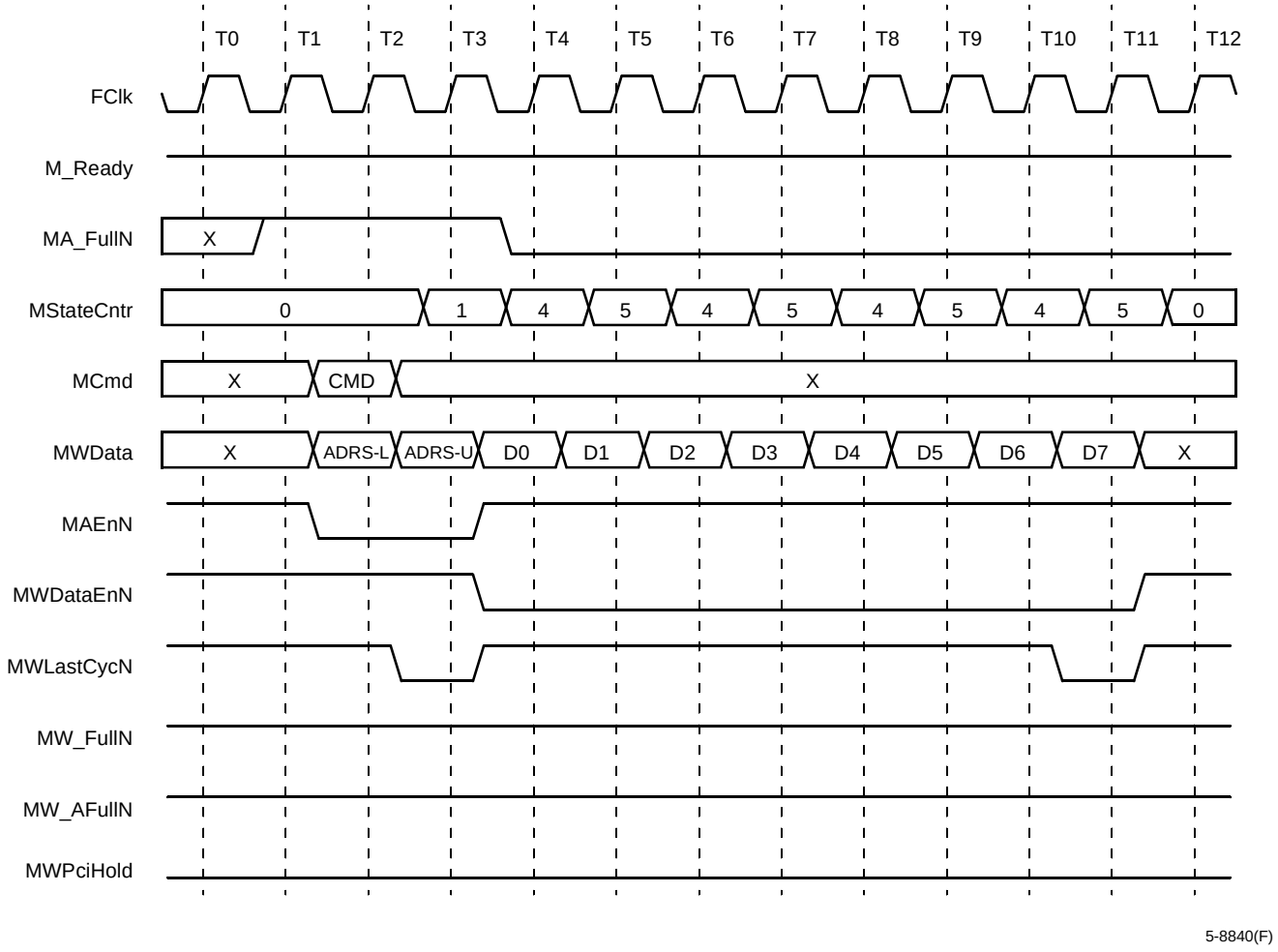


Figure 7. Master Write 32-Byte Burst (FPGA Bus, Quad Port, 64-Bit Address)

PCI Bus Core Detailed Description (continued)

Table 15. Dual-Port Master Write

MStateCntr	Next State of MStateCntr	Description	Bus	Notes
0	0	Idle	—	1
0	4	Address[63:0]	DataFmFPGAx[7:0] DataFmFPGA[63:0]	2, 17
4	4 or 0	Data[63:0], BE[7:0]	DataFmFPGAx[7:0] DataFmFPGA[63:0]	7, 11

1. MAEnN is deasserted high; the interface is idle.
2. MAEnN is asserted low; a Master command is being initiated and data is being transferred.
11. Next state = 0 if MWLastCycN is asserted low (end of Master write data).
17. MWLastCycN must be asserted low (end of Master address).

Table 16. Quad-Port Master Write

MStateCntr	Next State of MStateCntr	Description	Bus	Notes
0	0	Idle	—	1
0	1	Address[31:0]	MWData[35:0]	2
1	4	Address[63:32]	MWData[35:0]	5, 17
4	5 or 0	Data[31:0], BE[3:0]	MWData[35:0]	7, 11
5	4 or 0	Data[63:32], BE[7:4]	MWData[35:0]	7, 11

1. MAEnN is deasserted high; the interface is idle.
2. MAEnN is asserted low; a Master command is being initiated and data is being transferred.
5. MAEnN must be asserted low for data to transfer and state of MStateCntr to change (Master Address).
7. MAEnN must be deasserted high and MWDataEnN must be asserted low for data to transfer and state of MStateCntr to change (Master write data).
11. Next state = 0 if MWLastCycN is asserted low (end of Master write data).
17. MWLastCycN must be asserted low (end of Master address).

PCI Bus Core Detailed Description

(continued)

Master (FPGA Initiated) Read

Operation Setup

In order to initiate a PCI Master read operation, the FPGA application must supply the required information in the specific order prescribed in Table 17 through Table 20. The command word, burst length (if supplied), and address must be accompanied by assertion of the enable MAEnN. The definition of the Master command word was previously described in Table 10. The FPGA application can use the value returned on bus MStateCntr, the Master state counter's present value, to determine the counter's next state, using the state diagram for the particular operation being executed. The counter's next state must be determined because the FPGA application must supply the data to the PCI core that corresponds to the counter value being sent from the core to the FPGA.

Data Transfer

The FPGA application begins receiving the read data by deasserting MAEnN and asserting MRDataEnN. On every cycle that MRDataEnN is asserted, the PCI core clocks data from the Master read FIFO (64 deep by 36 bits wide in 32-bit PCI mode; 32 deep by 72 bits wide in 64-bit PCI mode) to the FPGA application via bus MRData (quad-port mode) or DataToFPGA (dual-port mode).

FIFO Empty/Almost Empty

When the Master read FIFO contains four or fewer data elements, the PCI core asserts MR_AEmptyN, the almost empty indicator. This allows some latency to exist in the FPGA's response without risking overreading the FIFO. When all locations in the Master write FIFO are empty, the PCI core asserts MR_Empty, the FIFO empty indicator. Since data can be simultaneously written to and read from the Master read FIFO, both MR_AEmptyN and MR_EmptyN can change states in either direction multiple times in the course of a burst data transfer.

FIFO Full

In addition to the empty and almost empty signals that report when the Master read FIFO is currently unable to supply data to the FPGA application, the PCI core also provides the FIFO's Full signal. This signal informs the FPGA application of completion of the current transfer. When a block of exactly 32 quad words (one complete FIFO image) of data remains to be read from the FIFO, the FPGA application monitors MR_FullN; when it is asserted low, the transfer is complete on the PCI bus.

Bursting

The PCI core uses the burst count supplied during operation setup to determine the Master read operation's burst length (unlike the Master write, which uses signal MWLastCycN). If the burst length for a Master read operation in Quad-port mode is the same as for the previous Master read, the FPGA application may elect to set MWData[33], in which case no burst length is supplied; instead, the burst length is kept from the previous operation. This saves a clock cycle during operation setup when in quad-port mode. In dual-port made the full burst-length is always available. The burst length of 18-bits allows bursts of up to 2^{18}–1 bytes to be specified.

Master Read Byte Enables

During master reads, byte enables are always supplied by the Master to the Target, even though on reads the data is flowing in the opposite direction. Thus, the byte enables cannot be buffered in a FIFO alongside the corresponding data. Also, the byte enables must be presented on the bus by the Master at the same time that the data is being presented on the bus by the Target (unless the Target uses TRDY# to insert wait-states), and so the data provided by the Target cannot depend on the byte enables (once again, without wait-states).

Termination

Once initiated, Master read operations will repeat on the PCI bus until either: 1) all data is received; 2) an abort occurs (either Master or Target); 3) the Master FIFO clear signal (MFIFOClrN) is asserted; or 4) the PCI bus' reset signal (RESET#) is asserted. If a PCI transaction is terminated with a retry or disconnect before all data has been received, the PCI core will initiate another Master read operation, continuing from that point.

Reset

The FPGA application can apply the PCI core's reset signal to place the core's logic in a known state. The reset signal, MFIFOClrN, is asynchronous and therefore should be asserted for a minimum of two Master FIFO clock cycles and two concurrent PCI clock cycles. The FIFO must not be utilized until MFIFOClrN is deasserted for a minimum of four concurrent clock cycles.

PCI Bus Core Detailed Description (continued)

Master Read, Nonburst Transaction

Figure 8 (FPGA Bus Dual Port), Figure 10 (FPGA Bus Quad Port), and Figure 9 (PCI Bus) show the timing of a single Quadword Master read. In Figure 8, the transaction is initiated by the FPGA application asserting Master address enable (MAEnN), while providing the command, burst length, and lower DWORD address on bus DataFmFPGA. On the next clock, the FPGA application provides the upper DWORD address and asserts MWLastCycN. On the third cycle, both MAEnN and MWLastCycN are deasserted. PCI bus activity now begins as shown in Figure 9. Once data is transfered on the PCI bus and MREmptyN is deasserted high, the FPGA application asserts MRDataEnN and one Quadword of data is transferred on bus DataToFPGA.

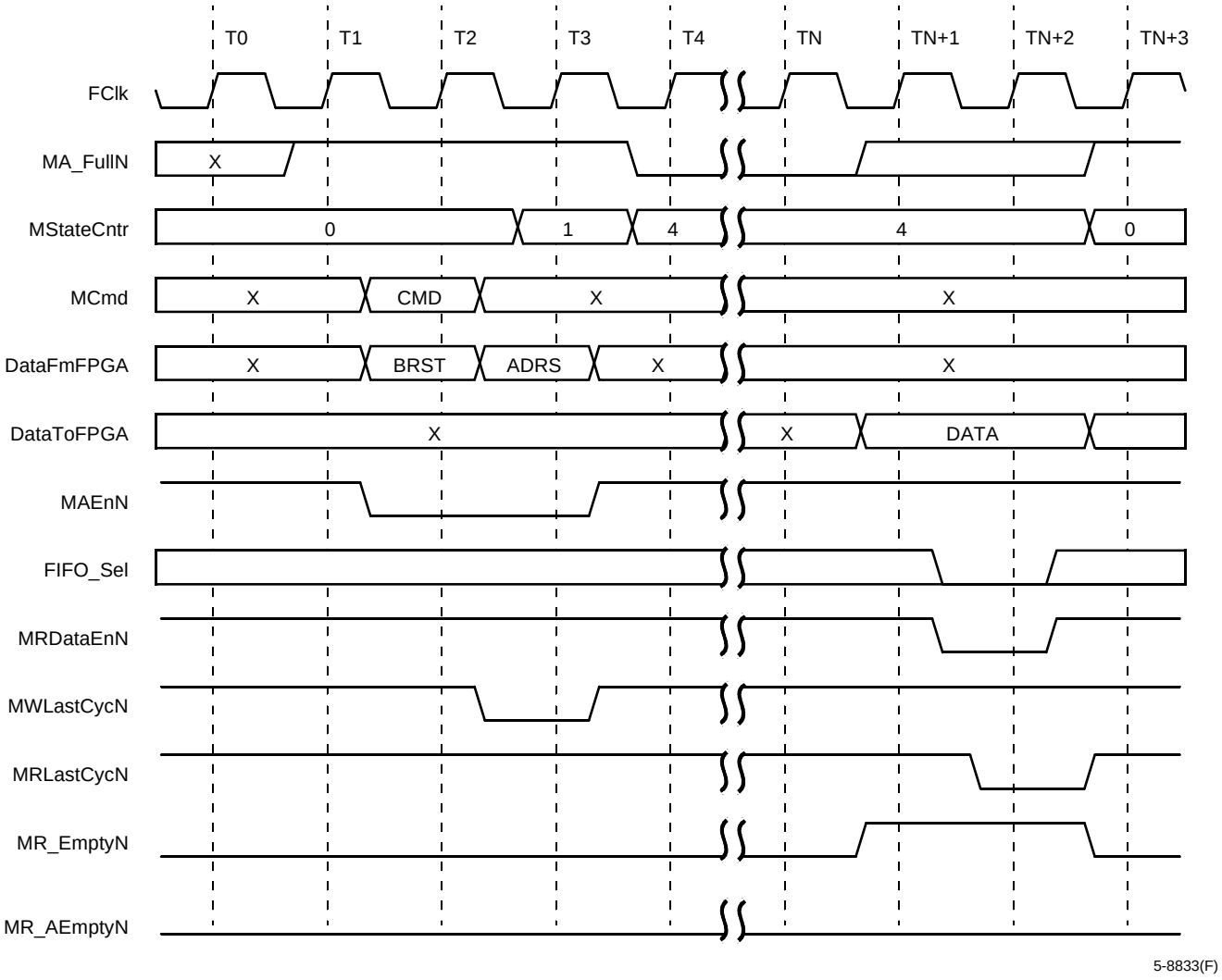


Figure 8. Master Read Single (FPGA Bus, Dual Port, Specified Burst Length, 64-Bit Address)

PCI Bus Core Detailed Description (continued)

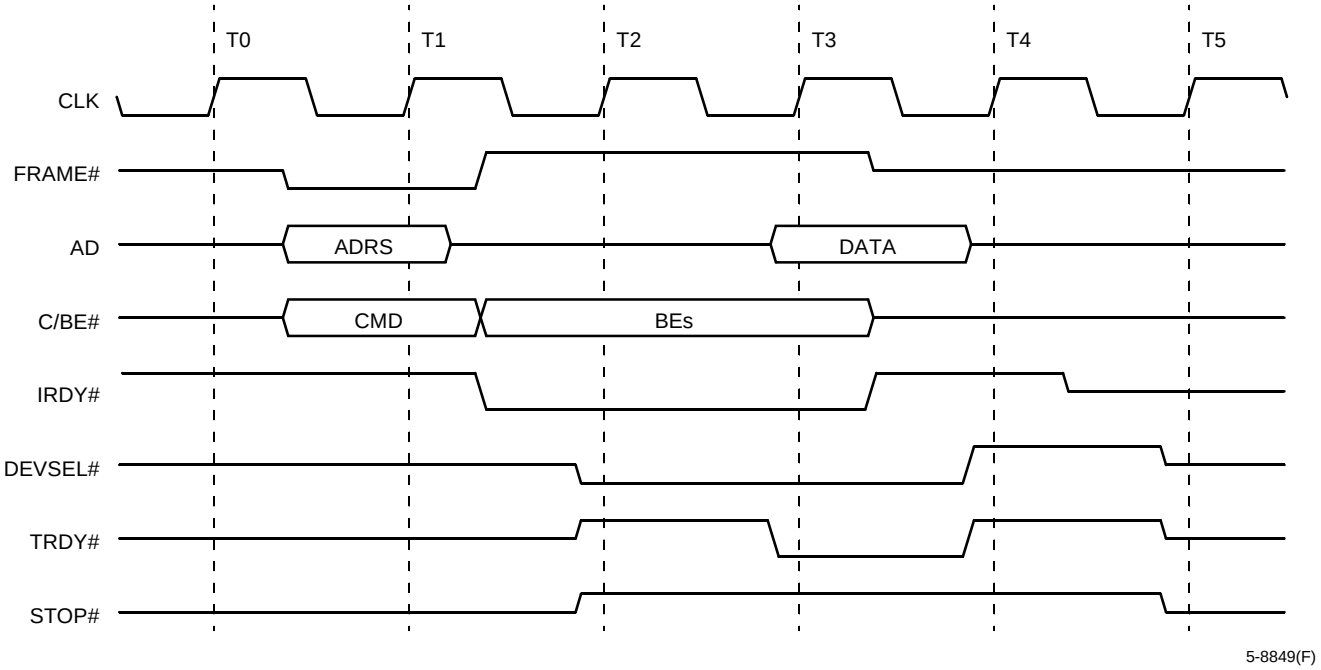


Figure 9. Master Read Single (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

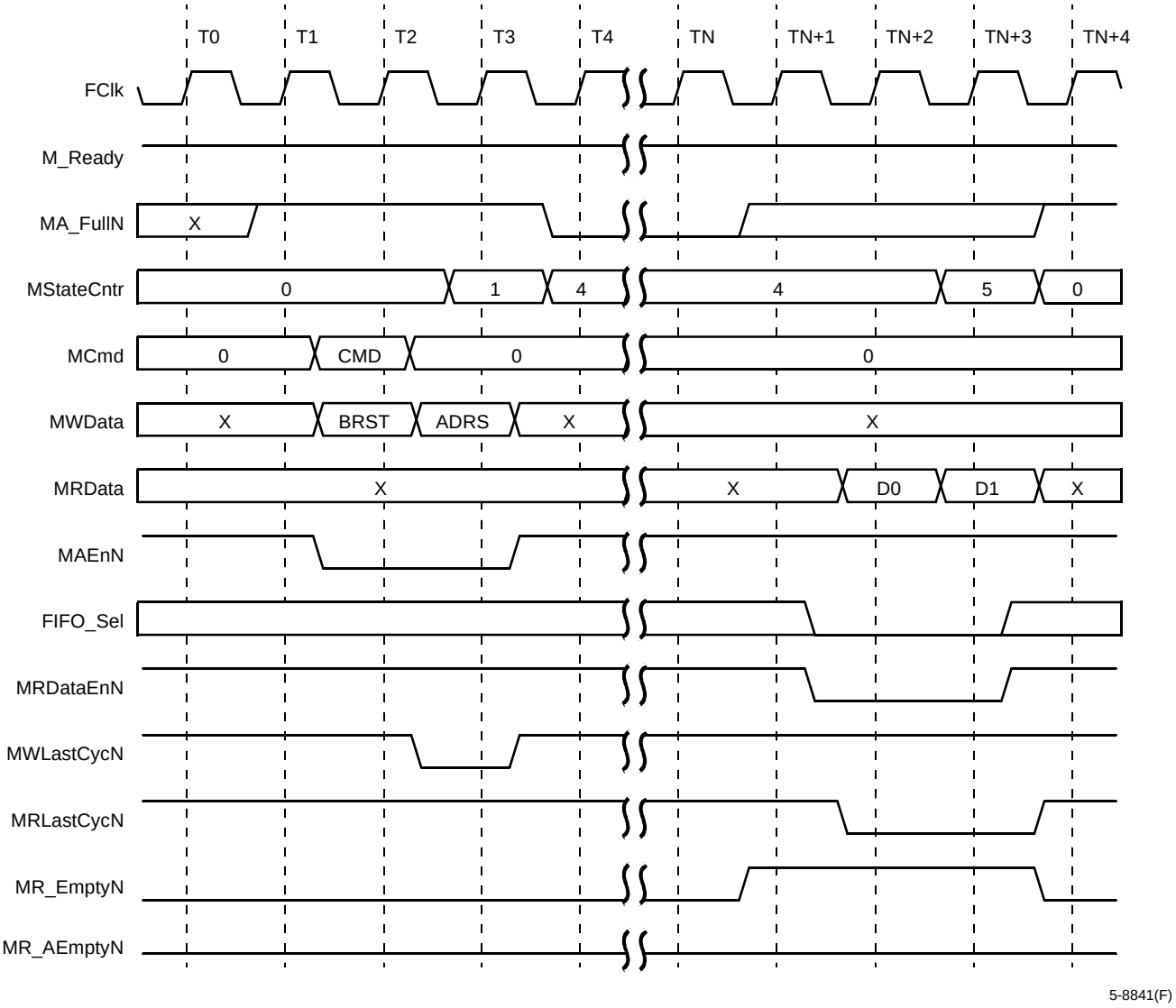
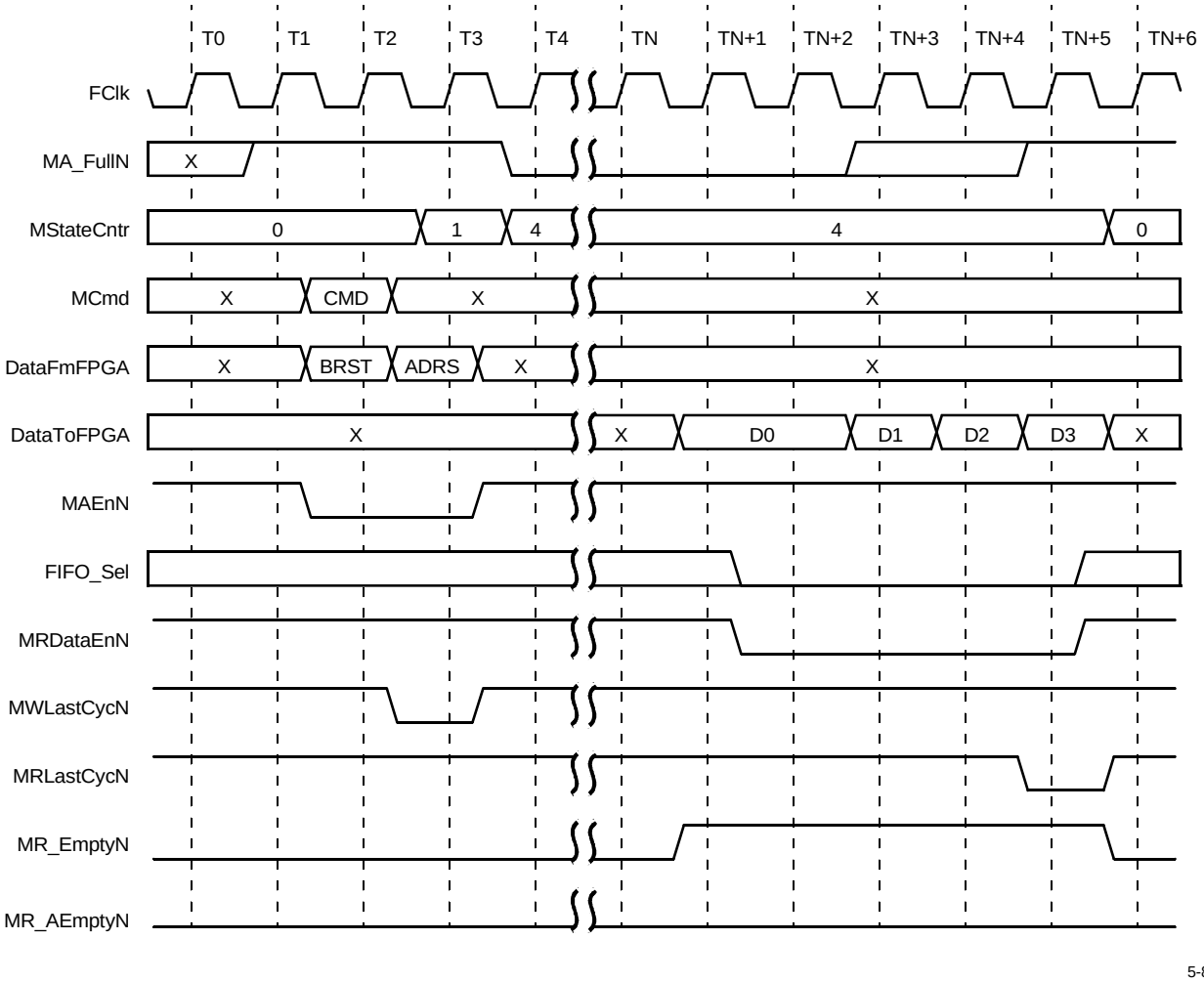


Figure 10. Master Read Single Quadword (FPGA Bus, Quad Port, Specified Burst Length, 64-Bit Address)

PCI Bus Core Detailed Description (continued)

Master Read, Burst Transaction

Figure 11 (FPGA Bus Dual Port), Figure 13 (FPGA Bus Quad Port), and Figure 12 (PCI Bus) show the timing of a four Quadword Master read burst. Operation is similar to that in the Master read, nonburst transaction, but extra data words are supplied by the FPGA application. In Figure 11, the transaction is initiated by the FPGA application asserting Master address enable (MAEnN), while providing the command, burst length, and lower DWORD address on bus DataFmFPGA. On the next clock, the FPGA application provides the upper DWORD address and asserts MWLastCycN. On the third cycle, both MAEnN and MWLastCycN are deasserted. PCI bus activity now begins as shown in Figure 12. Once data is transfered on the PCI bus and MREmptyN is deasserted high, the FPGA application asserts MRDataEnN and four quadwords of data are transferred on bus DataToFPGA.



5-8834(F)

Figure 11. Master Read 32-Byte Burst (FPGA Bus, Dual Port, Burst Length and 64-Bit Address)

PCI Bus Core Detailed Description (continued)

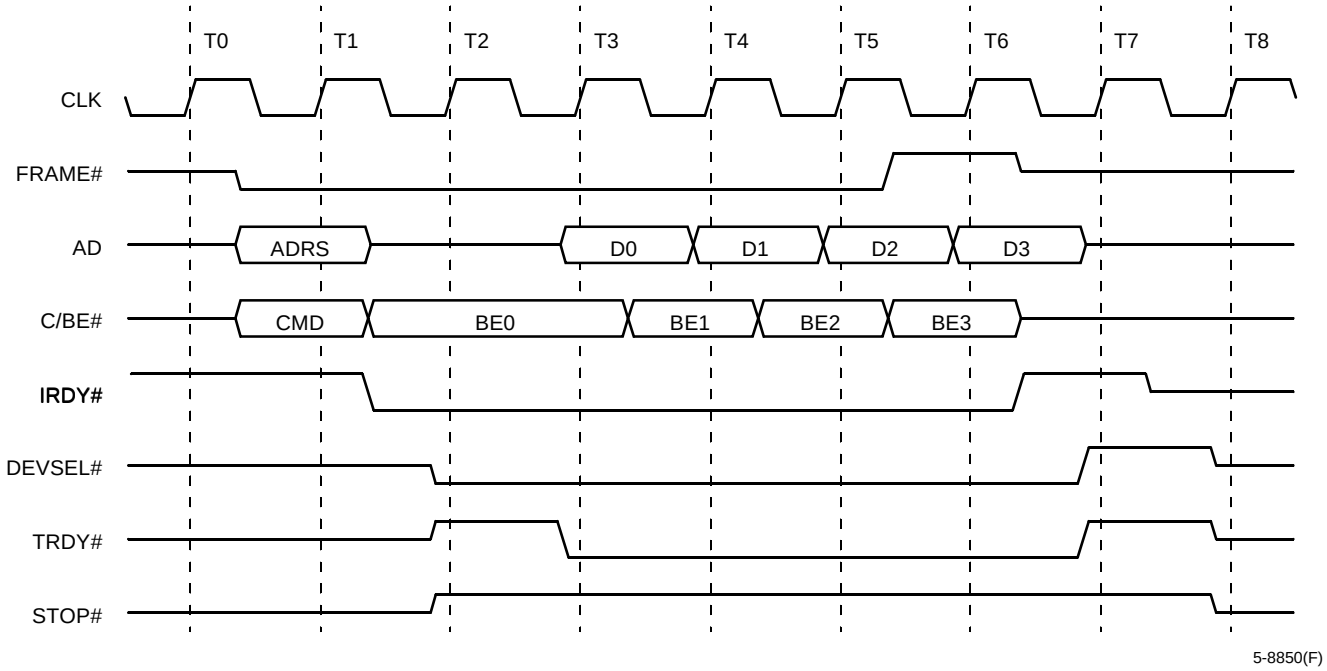


Figure 12. Master Read 32-Byte Burst (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

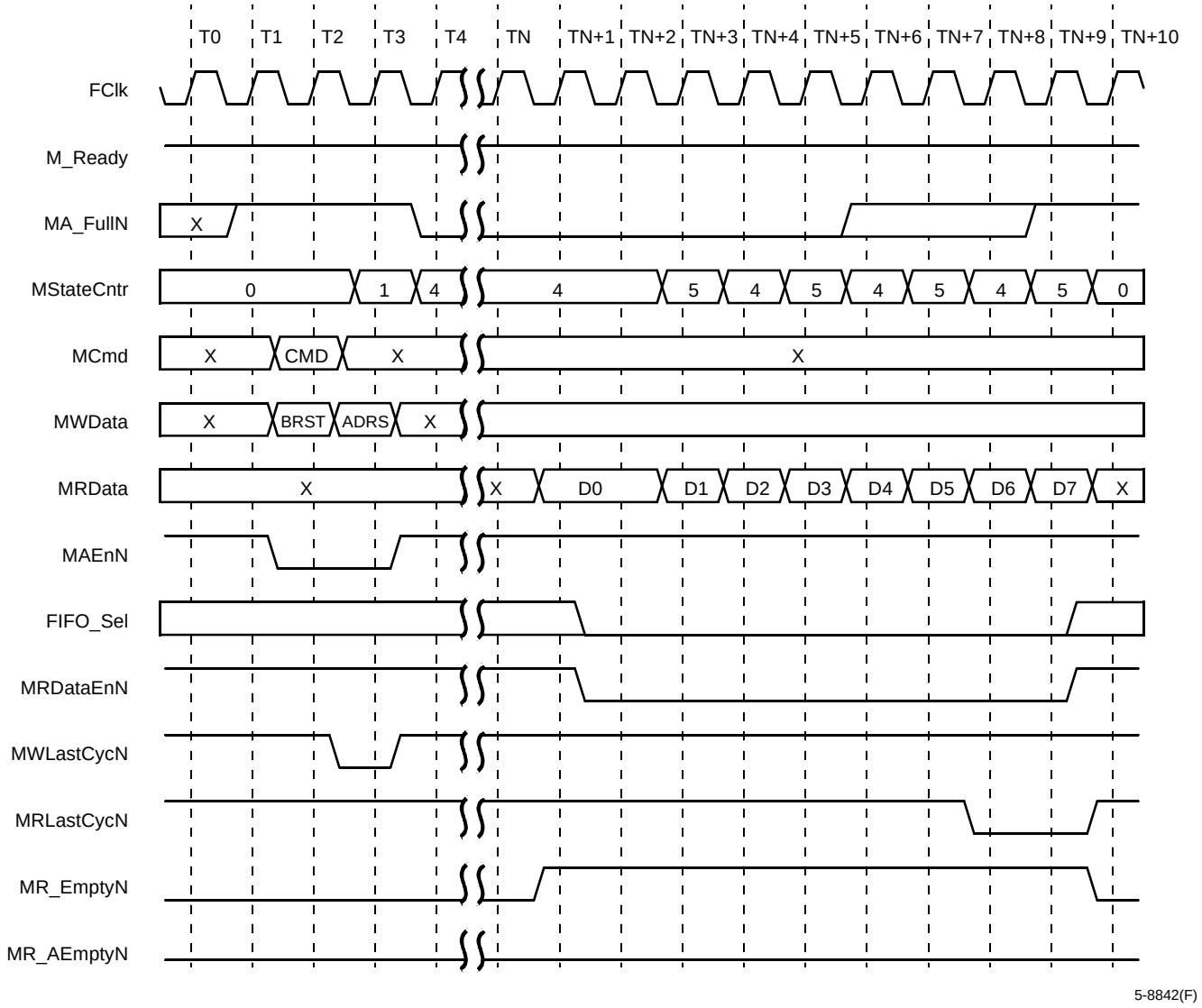


Figure 13. Master Read 32-Byte Burst (FPGA Bus, Quad Port, Specified Burst Length, 64-Bit Address)

PCI Bus Core Detailed Description (continued)

Table 17. Dual-Port Master Read, 64-Bit Address Supplied

MStateCntr	Next State of MStateCntr	Description	Bus	Notes
0	0	Idle	—	1
0	1	BE[7:0], Burst Length	DataFmFPGAx[7:0] DataFmFPGA[63:0]	2
1	4	Address[63:0]	DataFmFPGA[63:0]	5, 17
4	4 or 0	Data[63:0]	DataToFPGA[63:0]	8, 12

1. MAEnN is deasserted high; the interface is idle.
2. MAEnN is asserted low; a Master command is being initiated and data is being transferred.
5. MAEnN must be asserted low for data to transfer and state of MStateCntr to change (Master Address).
8. MAEnN must be deasserted high and MRDataEnN must be asserted low for data to transfer and state of MStateCntr to change (Master read data).
12. Next state = 0 if MRLastCycN is asserted low (end of Master read data).
17. MWLastCycN must be asserted low (end of Master address).

Table 18. Dual-Port Master Read, 32-Bit Address Supplied

MStateCntr	Next State of MStateCntr	Description	Bus	Notes
0	0	Idle	—	1
0	4	BE[7:0], Burst Length, Address[31:0]	DataFmFPGAx[7:0] DataFmFPGA[63:0]	2, 17
4	4 or 0	Data[63:0]	DataToFPGA[63:0]	8, 12

1. MAEnN is deasserted high; the interface is idle.
2. MAEnN is asserted low; a Master command is being initiated and data is being transferred.
8. MAEnN must be deasserted high and MRDataEnN must be asserted low for data to transfer and state of MStateCntr to change(Master read data).
12. Next state = 0 if MRLastCycN is asserted low (end of Master read data).
17. MWLastCycN must be asserted low (end of Master address).

PCI Bus Core Detailed Description (continued)

Table 19. Quad-Port Master Read, Duplicate Burst Length and 16-Bit Address

MStateCntr	Next State of MStateCntr	Description	Bus	Notes
0	0	Idle	—	1
0	4	BE[7:0], Address[15:0]	MWData[35:0]	2, 17
4	5 or 0	Data[31:0]	MRData[31:0]	8, 12
5	4 or 0	Data[63:32]	MRData[31:0]	8, 12

1. MAEnN is deasserted high; the interface is idle.
2. MAEnN is asserted low; a Master command is being initiated and data is being transferred.
8. MAEnN must be deasserted high and MRDataEnN must be asserted low for data to transfer and state of MStateCntr to change (Master read data).
12. Next state = 0 if MRLastCycN is asserted low (end of Master read data).
17. MWLastCycN must be asserted low (end of Master address).

Table 20. Quad-Port Master Read, Specified Burst Length and 64-Bit Address

MStateCntr	Next State of MStateCntr	Description	Bus	Notes
0	0	Idle	—	1
0	1 or 4	BE[7:0], Burst Length	MWData[35:0]	2, 15
1	2 or 4	Address[31:0]	MWData[31:0]	5, 15
2	4	Address[63:32]	MWData[31:0]	5, 17
4	5 or 0	Data[31:0]	MRData[31:0]	8, 12
5	4 or 0	Data[63:32]	MRData[31:0]	8, 12

1. MAEnN is deasserted high; the interface is idle.
2. MAEnN is asserted low; a Master command is being initiated and data is being transferred.
5. MAEnN must be asserted low for data to transfer and state of MStateCntr to change (Master Address).
8. MAEnN must be deasserted high and MRDataEnN must be asserted low for data to transfer and state of MStateCntr to change (Master read data).
12. Next state = 0 if MRLastCycN is asserted low (end of Master read data).
15. Next state = 4 if MWLastCycN is asserted low (end of Master address).
17. MWLastCycN must be asserted low (end of Master address).

PCI Bus Core Detailed Description
(continued)

Target (PCI Bus Initiated) Write

Operation Setup

The FPGA application waits for Target request, TReqN, from the PCI core to become active, indicating a Target operation, either read or write. It then asserts Target address enable, TAEnN, to clock out the command and its address. Table 21 and Table 22 describe the specific order of operation for a Target write transaction.

Bursts can be of any length, but will disconnect when any of the following conditions occur.

- TW_FullN is asserted-low and TWBurstPendN is deasserted-high.
- The maximum number of wait states has been inserted.
- The BAR boundary has been crossed.

Target State Counter

The PCI core provides a state counter, TStateCntr[3:0], that informs the FPGA of the current state of the PCI core's Target state counter. This state counter determines what data is currently being provided by the PCI core or expected from the FPGA application. This state counter transitions from one state to another in a predictable fashion, and thus, it is not strictly necessary to transmit its value to the FPGA. Nonetheless, the value on bus TStateCntr can be used to minimize FPGA logic or verify proper operation.

The data provided by the PCI core to the FPGA application on bus TWData (quad-port mode) or DataToFPGA (dual-port mode) is accompanied by a value on bus TStateCntr. This value can be directly used by the FPGA application to determine the proper use of that data. This eliminates the need for logic in the FPGA to duplicate this state counters in this case.

The data required from the FPGA application by the PCI core on bus TRData or DataFmFPGA is also defined by the value on bus TStateCntr. However, the state counter value is being sent to the FPGA in the same cycle that the data must be sent from the FPGA. Therefore, the FPGA application must build its own copy of the state counter value in this case. The value provided by the PCI core can be used as the previous value, or it can be used to verify the proper operation of the FPGA application's logic.

Table 10 lists the values of the state counter TStateCntr and the appropriate accompanying data. Table 21—Table 24 detail the various Target Read/Write operations, the sequencing of the state counters, and the data transferred between the PCI core and the FPGA application on each cycle.

Data Transfer

For a Target write data transfer, the FPGA application begins receiving the supplied data by deasserting TAEnN and asserting TWDataEnN. On every cycle that TWDataEnN is asserted, the FPGA application clocks data out of the PCI core's Target write FIFO (32 deep by 36 bits wide in 32-bit PCI mode; 16 deep by 72 bits wide in 64-bit PCI mode) via bus TWData (quad-port mode) or bus DataToFPGA (dual-port mode).

FIFO Empty/Almost Empty

Data to be written is buffered in the Target write FIFO (32 deep by 36 bits wide in 32-bit PCI mode; 16 deep by 72 bits wide in 64-bit PCI mode). When this FIFO contains four or fewer data elements, the PCI core asserts TW_AEmpty, the FIFO almost empty indicator. This allows some latency to exist in the FPGA's response without risking overreading the FIFO. When the PCI core has read all data out of the Target write FIFO, the PCI core asserts TW_EmptyN, the FIFO empty indicator. Since data can be simultaneously written to and read from the Target write FIFO, both TW_AEmptyN and TW_EmptyN can change states in either direction multiple times in the course of a burst data transfer.

FIFO Full

In addition to the empty and almost empty signals that report when the Target write FIFO is currently unable to supply data to the FPGA application, the PCI core also provides the FIFO's Full signal. When the FPSC Target and the remote Master have agreed on a predetermined data burst length, this signal can be used in place of, or in conjunction with, signal TWLastCycN to inform the FPGA application of completion of the current transfer. When a block of exactly 16 quadwords (one complete FIFO image) of data remains to be read from the FIFO, the FPGA application monitors TW_FullN; when it is asserted low, the transfer is complete on the PCI bus.

PCI Bus Core Detailed Description

(continued)

Nondelayed Transactions

Target memory and I/O write operations may work in a nondelayed transaction mode. Once the PCI core Target determines that it is the intended recipient, it asserts DEVSEL# and TRDY# and begins loading data into the Target write FIFO. After the core accepts the data element that fills the FIFO, the next data element will cause a disconnect without data. The operation is then complete on the PCI bus; even if the FPGA partially empties the Target write FIFO, no Target write transaction, even a continuation of the previous burst, will be accepted until the FIFO is emptied. The next Target write operation will be considered a new transaction.

Delayed Transactions

Target I/O write operations may also be handled as delayed transactions by asserting DelTrN. In this case, the operation is not accepted on the first request. Instead, on the first request, the PCI core records the command, address, and first data word (32 or 64 bits) along with its byte enables (4 or 8 bits). The first command and address are put in the Target address FIFO, and the data word and byte enables are put in the Target write FIFO. The request is terminated in a retry (the Master does not know that the data was snooped), and the FPGA application is informed as usual that a Target request is pending via the assertion of TReqN. Masters are required to repeat requests terminated in retry until data is moved (see PCI Specification section 3.3.3.2.2). The transaction status at this time is DWR (delayed write request—see PCI Specification section 3.3.3.3.6), and subsequent requests will be terminated in retry. When the FPGA application reads the FIFO and empties it, the transaction status changes to DWC (delayed write completion), and the next Target I/O write that matches the stored command, address, data, and byte enables will be accepted with a disconnect with data, completing the transaction and clearing the Target address and Target write FIFOs.

Termination

Nondelayed write transaction completion occurs when the last item remaining in the Target write FIFO has been read by the FPGA application (although the actual PCI bus transaction may have completed much earlier). Delayed write transaction completion occurs when the I/O write results in a disconnect with data. The PCI core signals end of transaction to the FPGA application by deasserting TReqN for at least one clock (if TReqN subsequently reasserts, this indicates a new, unrelated transaction).

Reset

The FPGA application can apply the PCI core's reset signal to place the core's logic in a known state. The reset signal, TFIFOClrN, is asynchronous and therefore should be asserted for a minimum of two Target FIFO clock cycles and two concurrent PCI clock cycles. The FIFO must not be utilized until TFIFOClrN is deasserted for a minimum of four concurrent clock cycles.

PCI Bus Core Detailed Description (continued)

Target Write to Configuration Space Transaction

Figure 14 shows the timing on the PCI interface for a Target write to configuration space. Accesses of configuration space occur without any involvement of the FPGA interface. All configuration space accesses are disconnected with data on the first data word and are thus restricted from bursting. Address decode speed is medium, and the PCI core signals that it is ready to receive the data by asserting TRDY# one cycle after DEVSEL# is asserted.

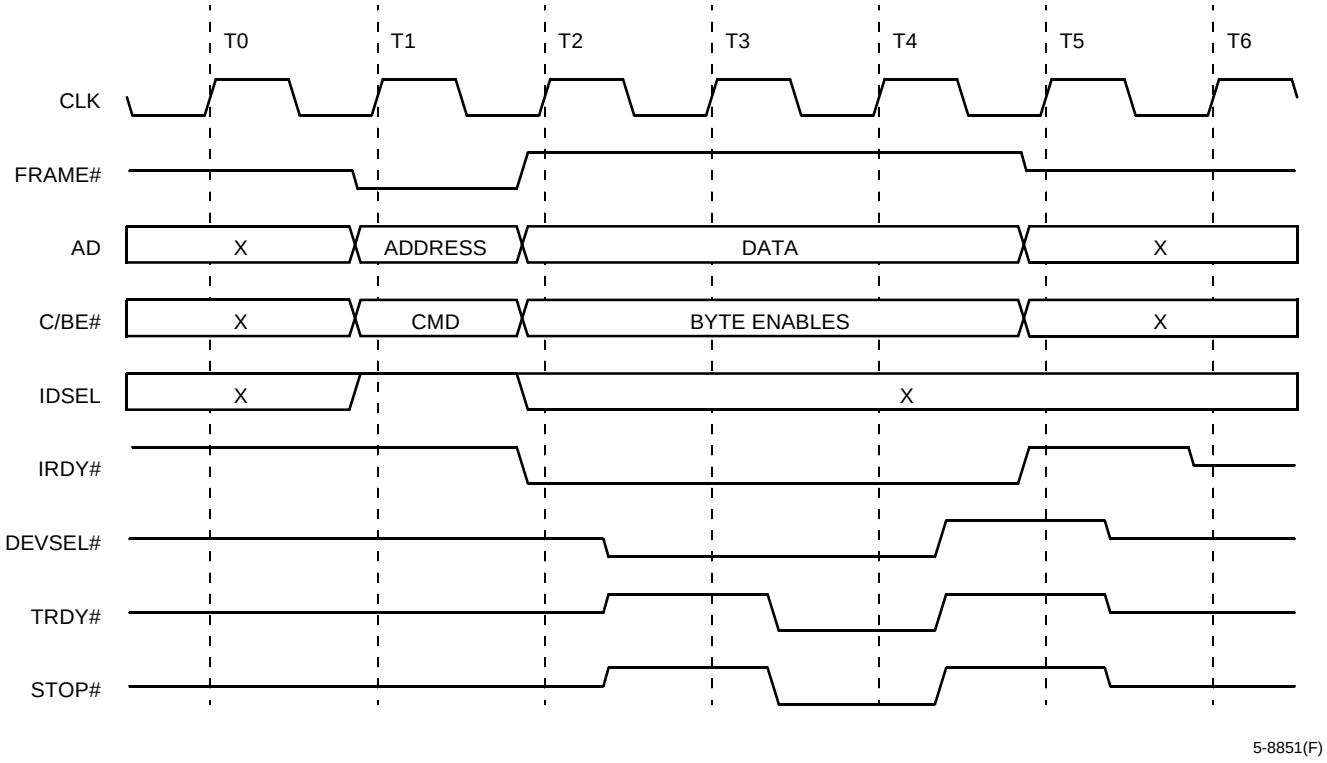


Figure 14. Target Configuration Write (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

Target Write I/O, Delayed Transaction

Figure 15 (PCI Bus), Figure 17 (FPGA Bus Dual Port), and Figure 18 (FPGA Bus Quad Port) for a Target I/O write operation that is handled as a delayed transaction, that is, the operation completes on the local (FPGA) bus before completing on the PCI bus. The FPGA application indicates its desire to do this by asserting signal DelTrN. In Figure 15, three transactions are shown: the first is the initial write that latches the command, address, data, and byte enables in the PCI core. The core's Target logic then issues a retry, obligating the remote Master to continue to issue that identical request until data is moved. Meanwhile, the information is relayed to the FPGA interface via the address and data FIFOs, triggering the FPGA interface exchange discussed below and shown in Figure 18. All subsequent read or write requests to memory, I/O, or configuration space will result in retries, as shown in the second transaction of Figure 15. The third transaction is the final transaction that completes the transfer of data. Although the data was actually latched and forwarded to the FPGA from the first transaction, it is not until the FPGA acknowledges that it has received the data, by emptying the Target write FIFO, that the PCI core acknowledges to the remote Master that it has received the data by performing a disconnect with data. The timing on this third transaction is identical to the timing of the first except that TRDY# accompanies STOP# to indicate the disconnect with data.

The timing on the FPGA interface (Figure 18) shows that the first indication to the FPGA application that a new operation has begun is the assertion of target request (TReqN), together with the new command on bus DataToFPGA. The FPGA application responds by asserting target address enable (TAEnN) and accepting the command and subsequent address on bus DataToFPGA. This is followed by deassertion of TAEnN, assertion of Target write data enable (DataToFPGAEnN), and the receiving of the data on bus DataToFPGA. This is a nonburst transaction; therefore, target write burst (TWLastCycN) is never asserted. Although only 32 bits of data are being transferred, the FPGA application must accept 64 bits of data (two clock cycles) because the FIFOs are operating in 64-bit mode.

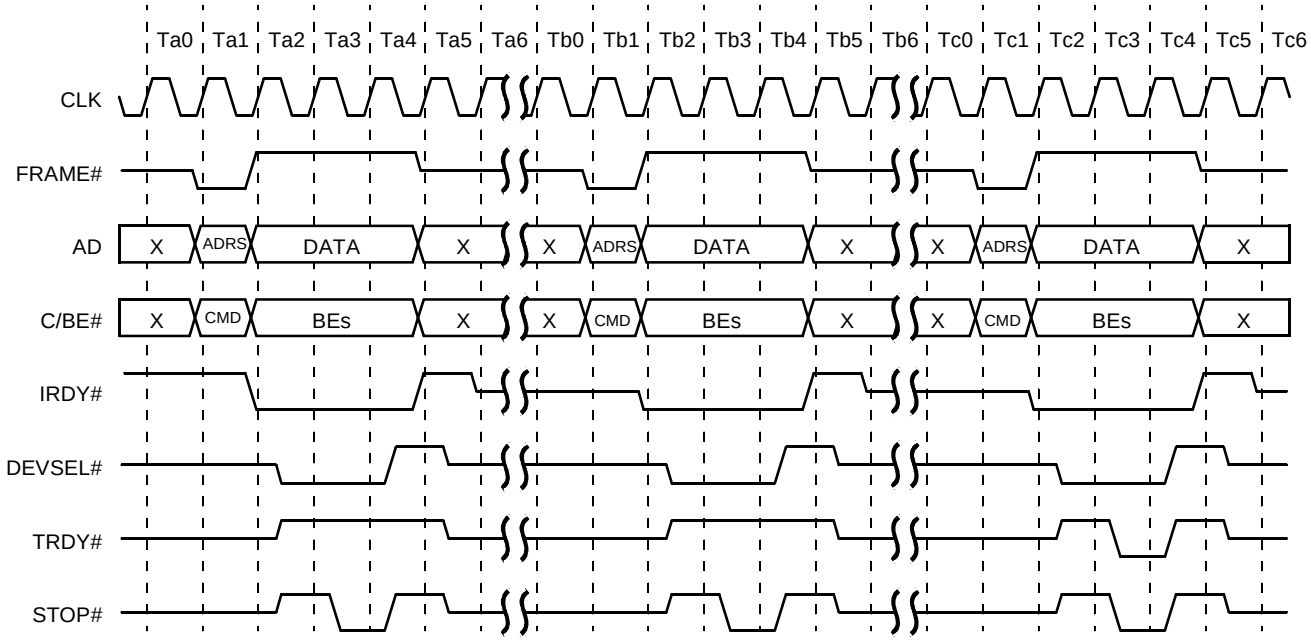


Figure 15. Target I/O Write, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

Target Write Nonburst Transaction

Figure 16 (PCI Bus), Figure 17 (FPGA Bus Dual Port), and Figure 18 (FPGA Bus Quad Port) show the timing on the PCI and FPGA interfaces, respectively, for a Target memory nonburst write transaction. The timing on the PCI interface (Figure 16) is similar to that of an I/O write except that, since bursts to memory space are allowed, the signal STOP# is not asserted. The FPGA interface timing is as shown in Figure 17, and is the same as the timing for memory and I/O write transactions.

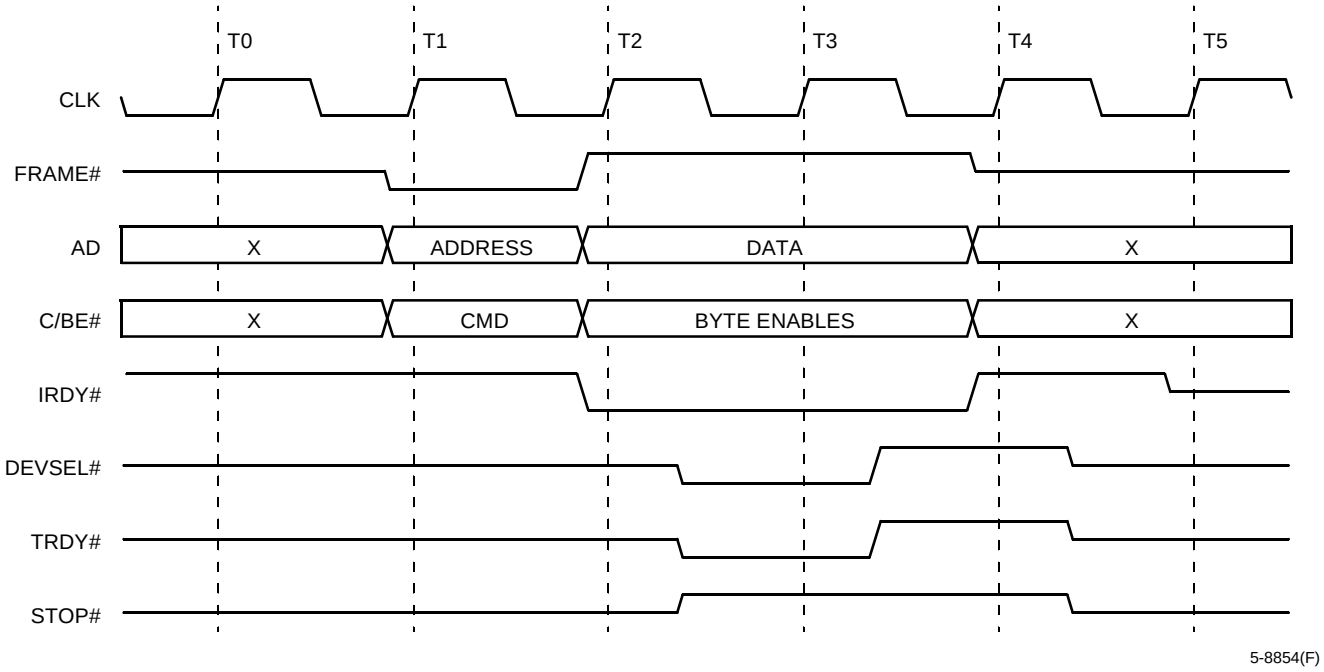


Figure 16. Target Write Memory Single (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

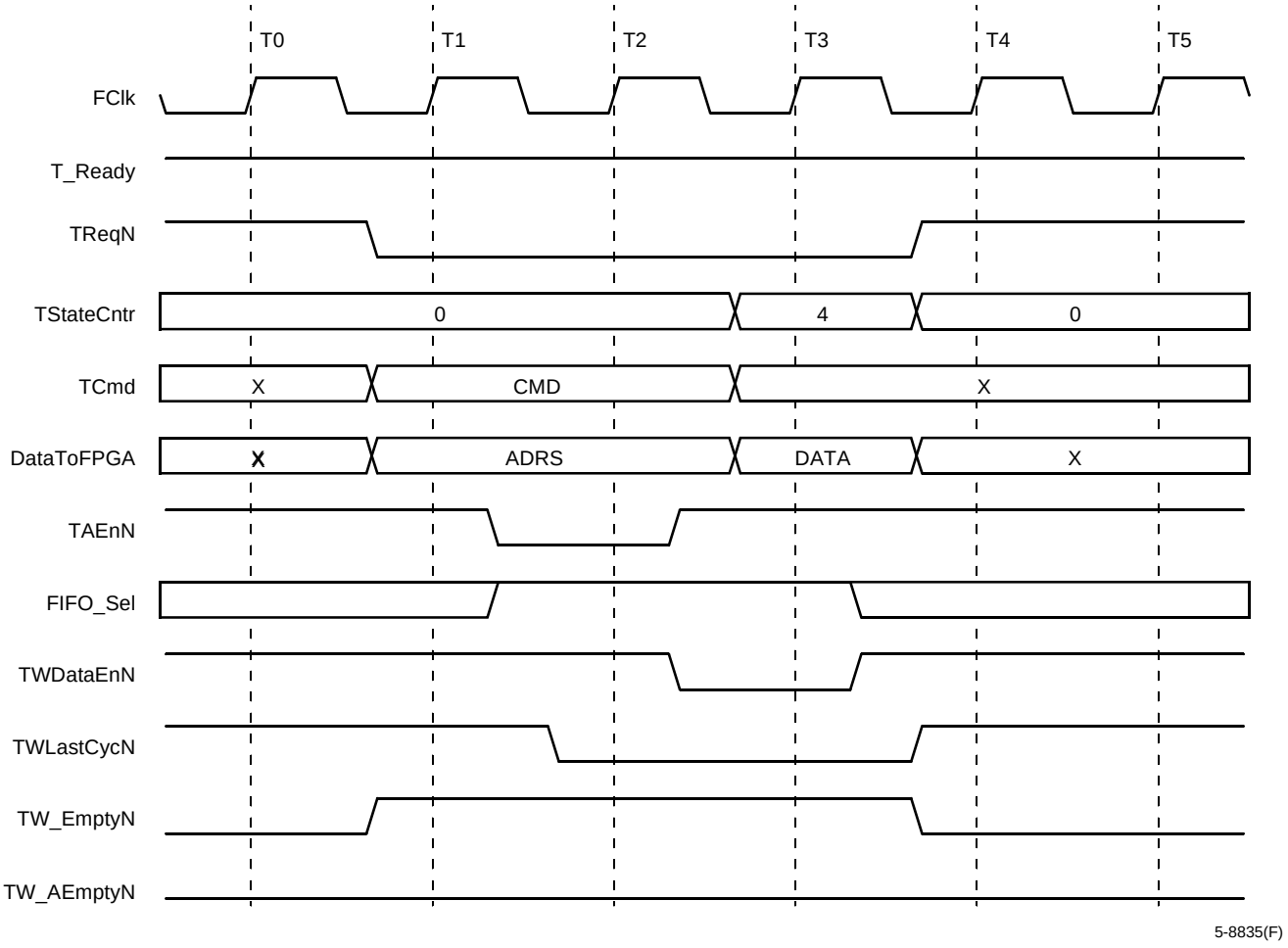


Figure 17. Target Write Single (FPGA Bus, Dual Port)

PCI Bus Core Detailed Description (continued)

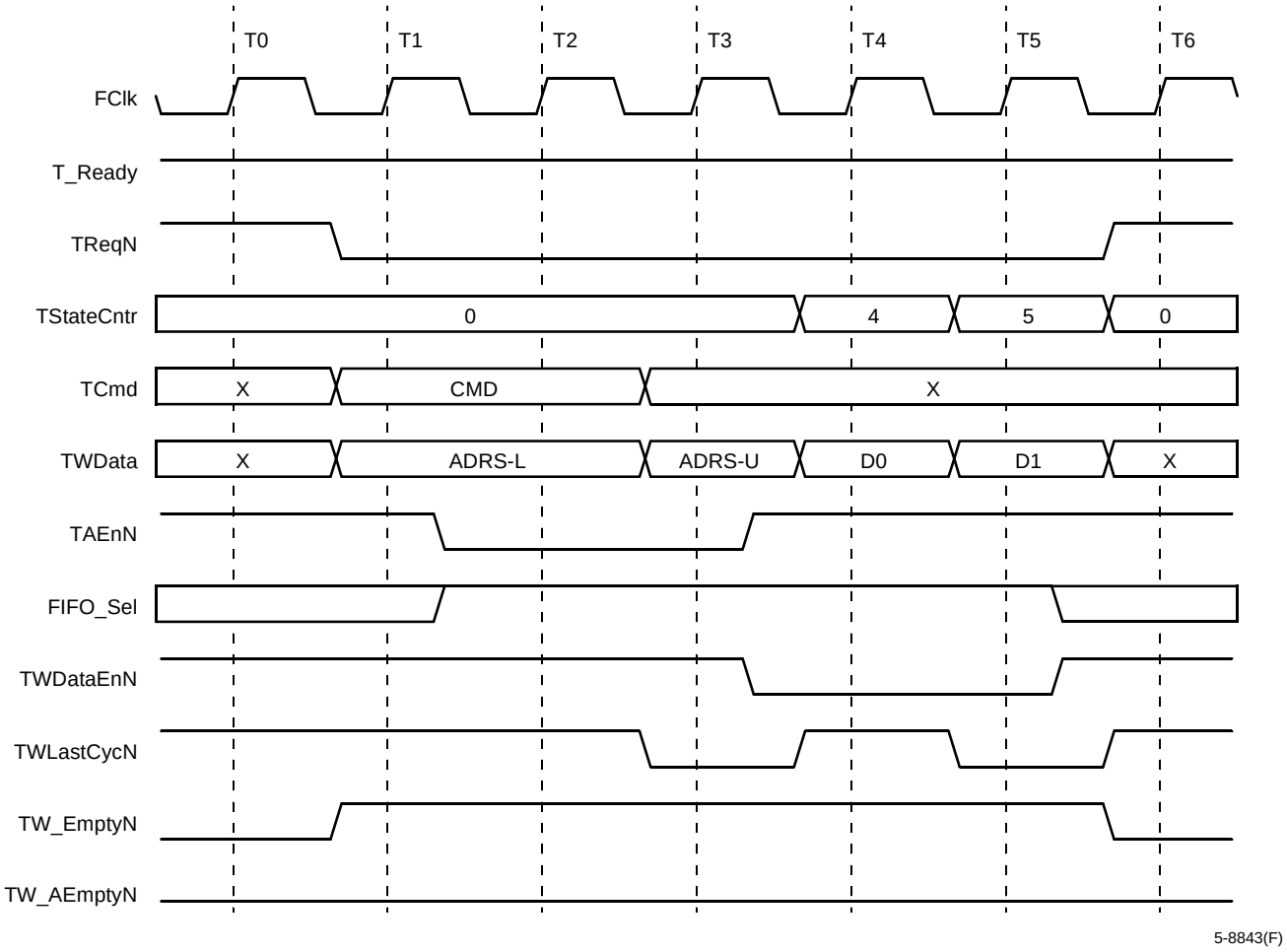


Figure 18. Target Write Single Quadword (FPGA Bus, Quad Port, 64-Bit Address)

PCI Bus Core Detailed Description (continued)

Target Write Memory Burst Transaction

Figure 19 (PCI Bus), Figure 20 (FPGA Bus Dual Port), and Figure 21 (FPGA Bus Quad Port) show the timing for a Target memory write burst of four quadwords. The timing on the PCI interface (Figure 19) is typical for a medium-speed decode Target. Note that TRDY# is asserted at the earliest possible time, which is concurrent with assertion of DEVSEL#. In the example of a four quadword burst, the FIFO is not filled, so execution continues to completion. This would also be the case for a burst of any length when the FPGA application is capable of unloading the FIFO as fast as the PCI interface is loading it. If the Target write FIFO becomes full, the PCI core Target will disconnect without data on the first data word it cannot accept.

The timing on the FPGA interface (Figure 20) shows that the first indication to the FPGA application that a new operation has begun is the assertion of target request (TReqN), together with the new command on bus TCmd. The FPGA application responds by asserting target address enable (TAEnN) and accepting the address on bus DataToFPGA. This is followed by deassertion of TAEnN, assertion of Target write data enable (TWDataEnN), and the receiving of the data on bus DataToFPGA. The FPGA application is informed that the last 64-bit data is being presented when Target write burst (TWLastCycN) is asserted.

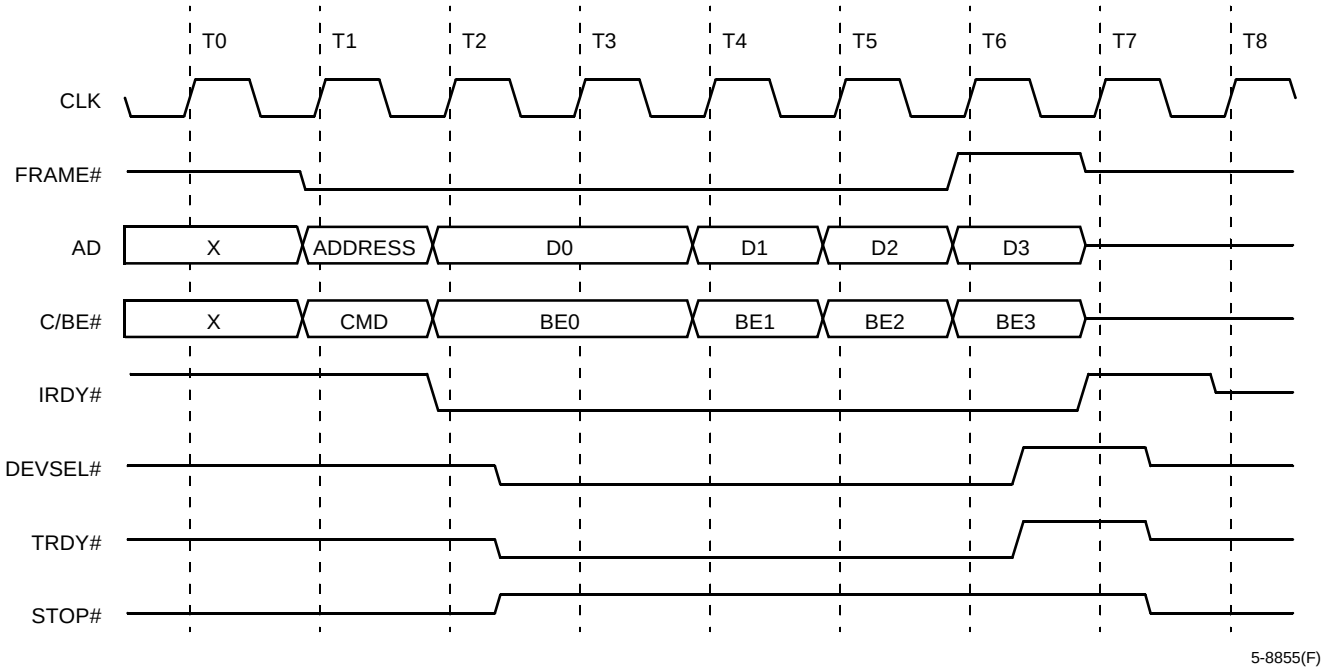


Figure 19. Target Memory Write 32-Byte Burst (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

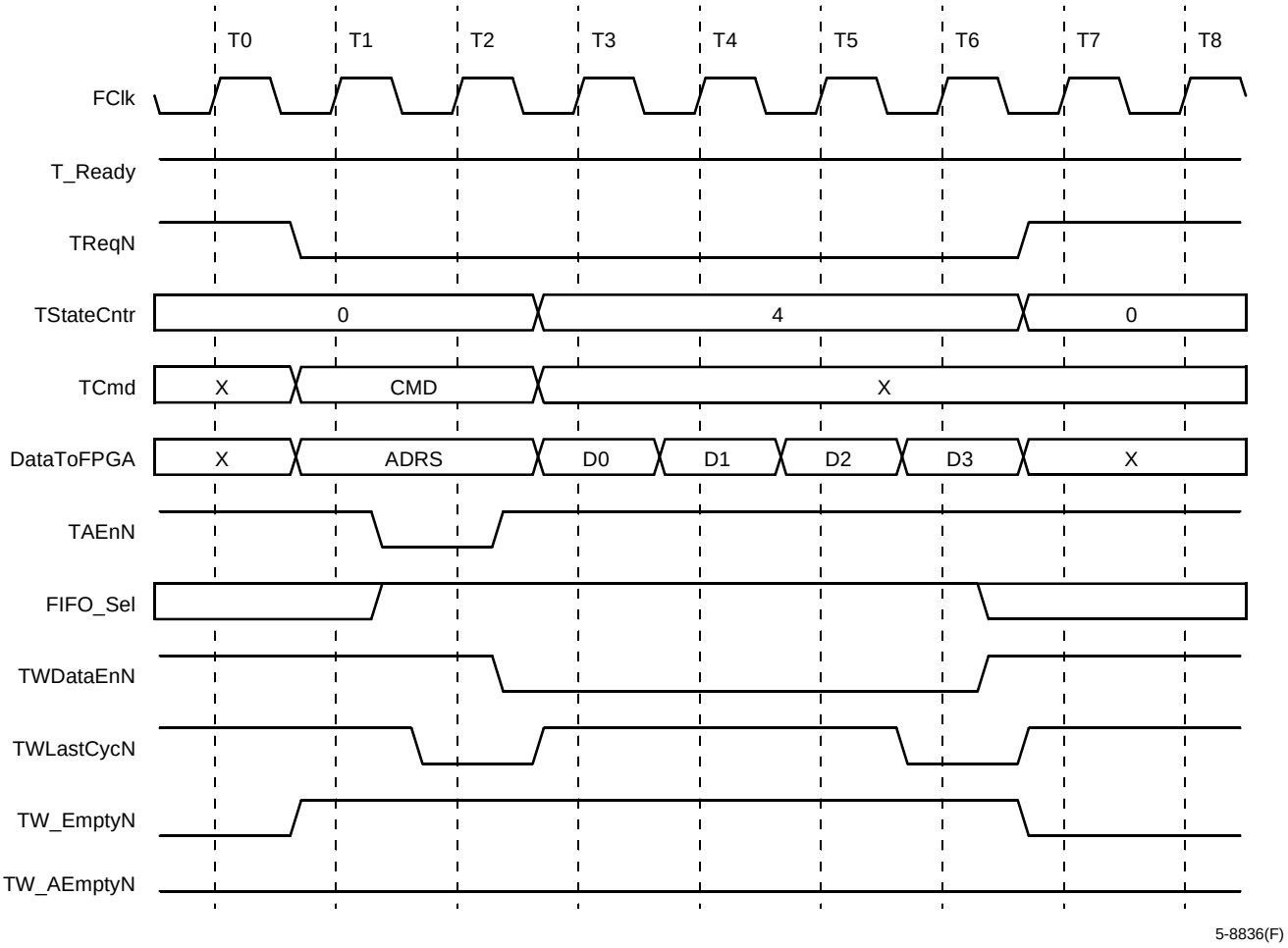


Figure 20. Target Write Memory 32-Byte Burst (FPGA Bus, Dual Port)

PCI Bus Core Detailed Description (continued)

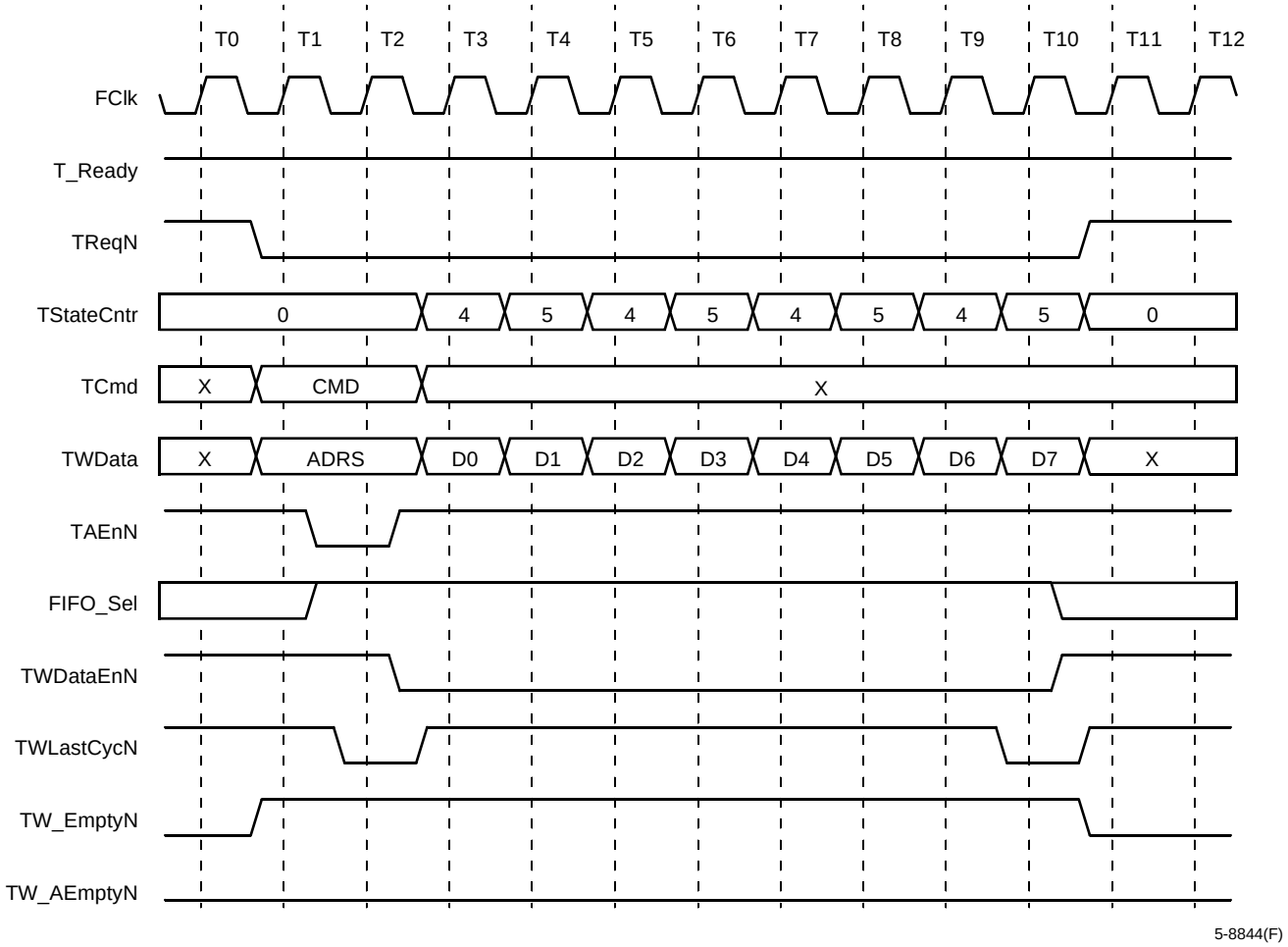


Figure 21. Target Write Memory 32-Byte Burst (FPGA Bus, Quad Port, 32-Bit Address)

PCI Bus Core Detailed Description (continued)

Table 21. Dual-Port Target Write

TStateCntr	Next State of TStateCntr	Description	Bus	Notes
0	0	Idle	—	3
0	4	Address[63:0]	DataToFPGAx[7:0] DataToFPGA[63:0]	4, 18
4	4 or 0	Data[63:0], BE[7:0]	DataToFPGAx[7:0] DataToFPGA[63:0]	9, 13

3. When TReqN is deasserted high, the interface is idle.
4. When TReqN is asserted low, a Target command is pending.
9. TAEnN must be deasserted high and TWDataEnN must be asserted low for data to transfer and state of TStateCntr to change (Target write data).
13. Next state = 0 if TWLastCycN is asserted low (end of Target write data).
18. TWLastCycN must be asserted low (end of Target address).

Table 22. Quad-Port Target Write

TStateCntr	Next State of TStateCntr	Description	Bus	Notes
0	0	Idle	—	3
0	1 or 4	Address[31:0]	TWData[35:0]	4, 16
1	4	Address[63:32]	TWData[32:0]	6, 18
4	5 or 0	Data[31:0], BE[3:0]	TWData[35:0]	9, 13
5	4 or 0	Data[63:32], BE[7:4]	TWData[35:0]	9, 13

3. When TReqN is deasserted high, the interface is idle.
4. When TReqN is asserted low, a Target command is pending.
6. TAEnN must be asserted low for data to transfer and state of TStateCntr to change (Target Address).
9. TAEnN must be deasserted high and TWDataEnN must be asserted low for data to transfer and state of TStateCntr to change (Target write data).
13. Next state = 0 if TWLastCycN is asserted low (end of Target write data).
16. Next state = 4 if TWLastCycN is asserted low (end of Target address).
18. TWLastCycN must be asserted low (end of Target address).

PCI Bus Core Detailed Description

(continued)

Target (PCI Bus Initiated) Read

The Target read operation presents unique demands on the PCI core, because only in the Target read operation does the PCI core request data that is needed to complete the transaction after the PCI transaction has already begun on the PCI bus. Target latency rules require that the data be acquired quickly or that the Target terminate the transaction with a retry/disconnect. Also, once the transfer process is underway, the Target does not know how much more data will be requested, yet the Target must prefetch data so that it will be available if needed. Special signals and protocols are described below to efficiently deal with these unique demands.

Operation Setup

The FPGA application waits for Target request, TReqN, from the PCI core to be active, indicating a Target operation, either read or write. It then asserts address enable, TAEnN, to clock out the command and its address. Table 23 and Table 24 describe the specific order of operation for a Target read transaction.

Bursts can be of any length, but will disconnect when either of the following conditions occur.

- TR_EmptyN is asserted low.
- The BAR boundary has been crossed.

Data Transfer

For a target read data transaction, the FPGA application begins supplying the requested data by deasserting TAEnN and asserting TRDataEnN. On every cycle that TRDataEnN is asserted, the FPGA application clocks data into the PCI core's Target read FIFO (32 deep by 36 bits wide in 32-bit PCI mode; 16 deep by 72 bits wide in 64-bit PCI mode) via bus TRData (quad-port mode) or bus DataFmFPGA (dual-port mode). Since the Target read FIFO will always be empty at the start of a transaction, the first Target read request to a specific address will result in a retry, initiating a delayed transaction (if signal TRBurstPendN is deasserted-high) or PCI bus wait-states (if signal TRBurstPendN is asserted low).

The signal TRPciHold can be asserted to hold off activation of the nonempty condition. While TRPciHold is active, the Target read FIFO empty flag will not change to the nonempty state until it is full, but then will remain in the nonempty state until that FIFO truly becomes empty. Use of this signal can result in more efficient utilization of PCI bus bandwidth by causing a full buffer contents to be burst, without wait-states, whenever the PCI bus is claimed.

FIFO Full/Almost Full

When the Target read FIFO contains four or fewer empty locations, the PCI core asserts TR_AFullN, the almost full indicator. This allows some latency to exist in the FPGA's response without risking overfilling the FIFO. When all locations in the Target read FIFO are full, the PCI core asserts TR_FullN, the full indicator. Since the data can be simultaneously written to and read from the Target read FIFO, both TR_AFullN and TR_FullN can change states in either direction multiple times in the course of a burst data transfer.

FIFO Empty

In addition to the full and almost full signals that report when the Target read FIFO is currently unable to receive data from the FPGA application, the PCI core also provides the FIFO's Empty signal. When the FPSC Target and the remote Master have agreed on a predetermined data burst length, this signal can be used in place of, or in conjunction with, signal TRLastCycN to inform the FPGA application of completion of the current transfer. After all necessary data has been written to the FIFO, the FPGA application monitors TR_EmptyN; when it is asserted low, the transfer is complete on the PCI bus.

Bursting

Signal TRLastCycN tells the FPGA application whether the current read is a burst. One data element must be supplied regardless of this signal's state. The FPGA application continues to supply data elements (contingent on the full bits) as long as TRLastCycN is inactive. Note that this may result in the discarding of unused data elements supplied in excess of the PCI transaction's needs. Burst transfers are done either as continuous data phases if read data continues to be available in the read data FIFO, or as a series of transfers terminated as disconnects without data.

PCI Bus Core Detailed Description

(continued)

Delayed Transactions

A dynamic output signal from the FPGA to the PCI core (DelTrN) influences the behavior of Target read operations. When DelTrN is asserted low, the PCI core Target read logic will issue a retry whenever no Target read operation is already pending. When this signal is inactive-high, it will instead generate wait-states, and continue to do so until either the FIFO becomes not empty, when it will transmit the data, or until the maximum initial latency value (16 or 32 clock cycles) has been reached. This signal should be inactive when minimum latency is desired on the initial data word, at the expense of overall PCI bus efficiency. Whereas disable delayed transactions affects the transaction's behavior on the initial data word, signal TRBurstPendN affects behavior when the Target read FIFO empties. When TRBurstPendN is inactive, a disconnect without data results from an attempt to read from an empty FIFO. With TRBurstPendN active, the PCI core will wait for data from the FIFO by inserting wait-states (up to the maximum subsequent latency value of 8, at which time a disconnect without data will be generated). Asserting TRBurstPendN will minimize latency for this transaction's data at the expense of overall PCI bus efficiency. TRBurstPendN must remain static throughout a Target read transaction.

I/O Reads

I/O reads differ from memory reads in that I/O reads always perform a disconnect with data on the first data element read from the Target read FIFO.

Termination

Normal transaction completion occurs immediately upon completion of the PCI bus transfer, even if extra data remains in the Target read FIFO. When the PCI transaction ends either normally, or as retry, disconnect, or Target abort, the PCI core signals end of transaction to the FPGA application by deasserting TReqN for at least one clock if TReqN subsequently reasserts, this indicates a new, unrelated transaction. When TReqN deasserts, the FPGA application must immediately deassert TRDataEnN.

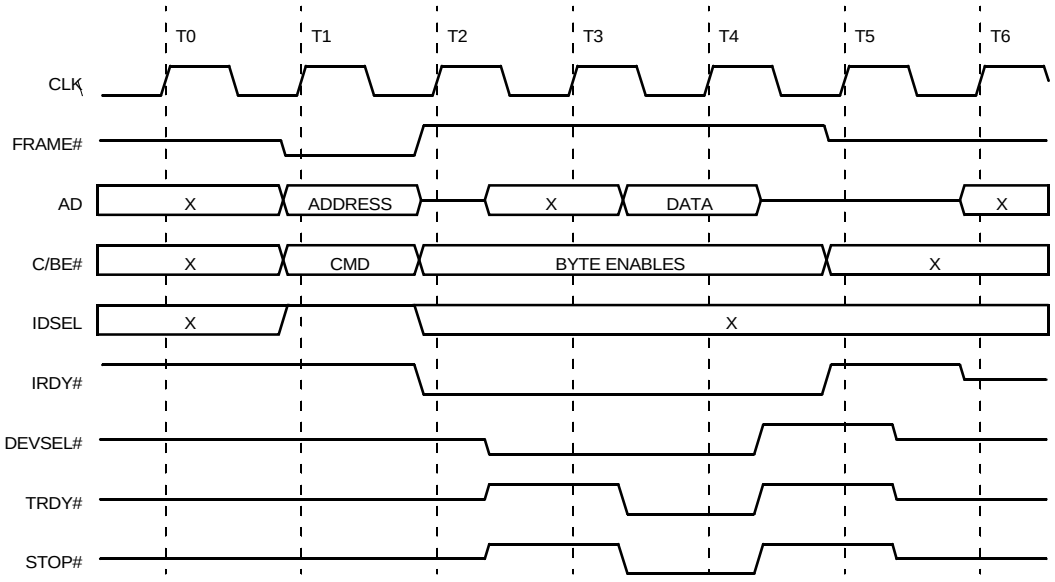
Reset

The FPGA application can apply the PCI core's reset signal to place the core's logic in a known state. The reset signal, TFIFOClrN, is asynchronous and therefore should be asserted for a minimum of two Target FIFO clock cycles and two concurrent PCI clock cycles. The FIFO must not be utilized until TFIFOClrN is deasserted for a minimum of four concurrent clock cycles.

PCI Bus Core Detailed Description (continued)

Target Read from Configuration Space

Figure 22 shows the timing on the PCI interface for a Target read from configuration space. Accesses of configuration space occur without any involvement of the FPGA interface. All configuration space accesses are disconnected with data on the first data word, and are thus restricted from bursting. Address decode speed is medium, and the PCI core signals that it is supplying the word of data by asserting TRDY# one cycle after DEVSEL# is asserted.



5-8856(F)

Figure 22. Target Configuration Read (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

Target Read I/O, Delayed Transaction

Figure 23 (PCI Bus), Figure 26 (FPGA Bus Dual Port), and Figure 27 (FPGA Bus Quad Port) show the timing for a Target I/O read that is handled as a delayed transaction. In other words, the operation completes on the local (FPGA) bus before completing on the PCI bus. The FPGA application indicates its desire to do this by driving the delayed transaction signal DelTrN active-low. In Figure 23, three transactions are shown: the first is the initial read that latches the command, address, and byte enables. The PCI core's Target logic then issues a retry, obligating the remote Master to continue to issue that identical request until data is moved. Meanwhile, the latched information is relayed to the FPGA interface via the address FIFO, triggering the FPGA interface exchange discussed below and in Figure 26. All subsequent read or write requests to memory or I/O space will result in retries, as shown in the second transaction of Figure 23. The third transaction is the final transaction that completes the transfer of data. The timing on this third transaction is identical to the timing of the first except that TRDY# accompanies STOP# to indicate the disconnect with data.

The timing on the FPGA interface (Figure 26) shows that the first indication to the FPGA application that a new operation has begun is the assertion of Target request (TReqN), together with the new command on bus DataToFPGA. The FPGA application responds by asserting Target address enable (TAEnN) and accepting the command and subsequent address on bus DataToFPGA, after which TAEnN is deasserted. The FPGA application then accesses the requested data, asserts Target read data enable (TRDataEnN), and transmits the data on bus DataFmFPGA. This is a non-burst transaction; therefore, Target read burst (TRLastCycN) is kept asserted. Although 32 bits of data are being transferred, the FPGA application must buffer to 64 bits of data (two clock cycles) because the FIFOs are always in 64-bit mode.

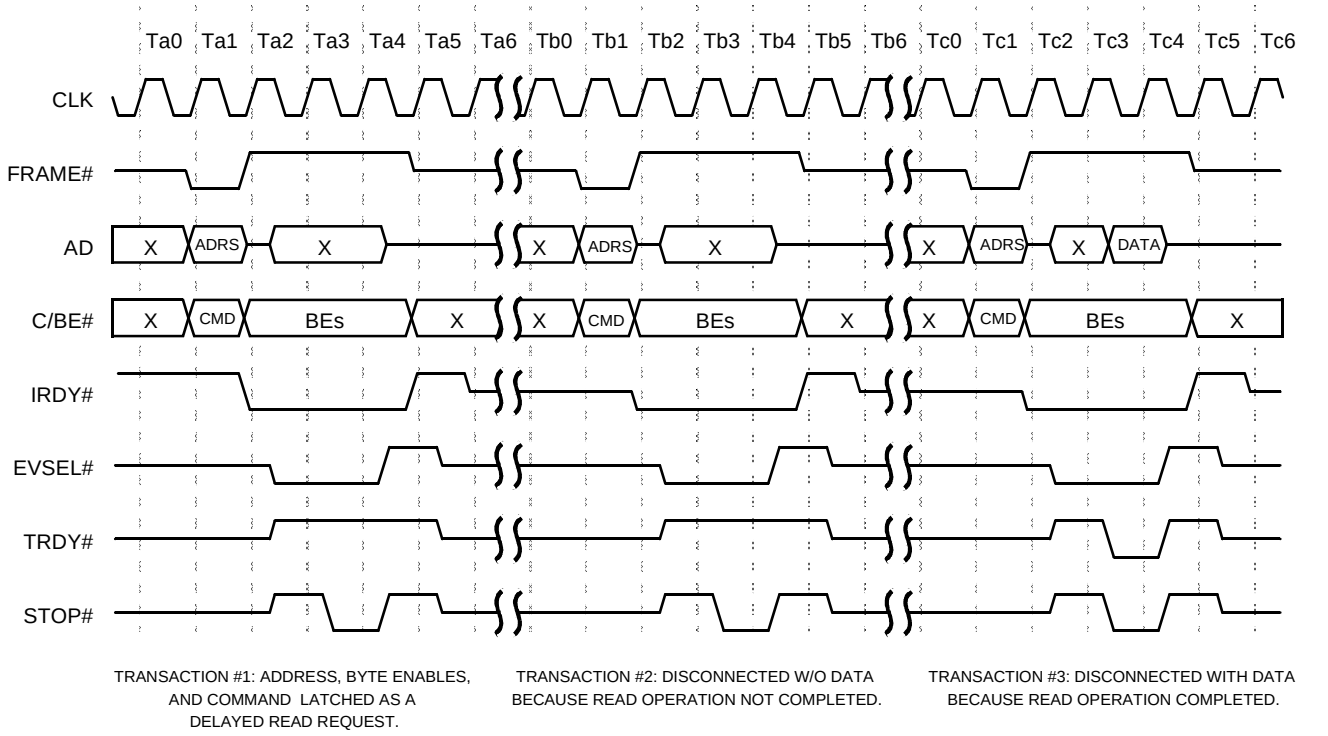


Figure 23. Target I/O Read, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

Target Read I/O, No Delayed Transaction

Figure 24 (PCI Bus), Figure 26 (FPGA Bus Dual Port), and Figure 27 (FPGA Bus Quad Port) show the timing for a Target I/O read that is handled as an immediate execution; that is, the operation completes on the PCI bus immediately and then is presented to the FPGA via the FPGA interface. The FPGA application indicates its desire to do this by deasserting signal DelTrN. The PCI core Target terminates the I/O read request by disconnecting with data on the first data word, thus disallowing bursting. The PCI interface timing shown in Figure 24 is identical to the timing of the third (final) transaction of Target I/O read, delayed transaction (Figure 23), which shows a Target I/O read with delayed transaction. Also, the FPGA interface timing is as shown in Figure 26, regardless of whether delayed transactions are enabled.

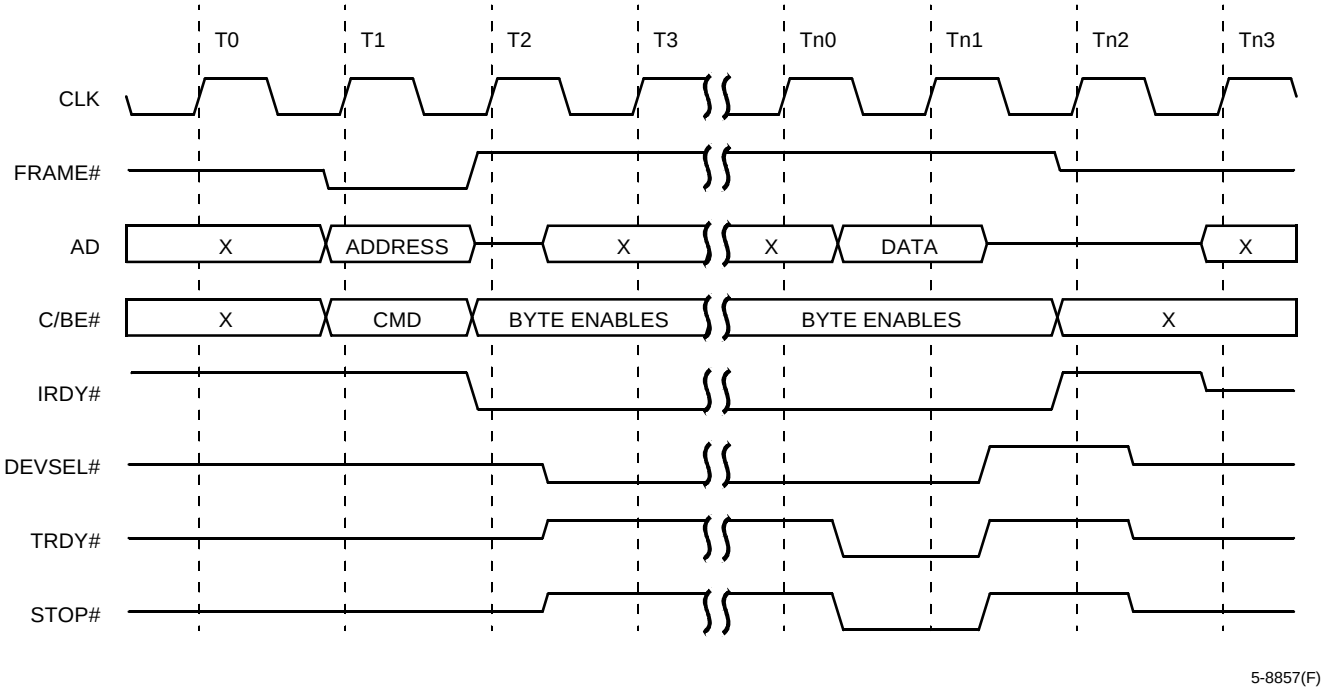


Figure 24. Target I/O Read, Not Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

Target Read Memory, Non-Burst, Delayed Transaction

Figure 25 (PCI Bus), Figure 26 (FPGA Bus Dual Port), and Figure 27 (FPGA Bus Quad Port) show the timing for a Target memory nonburst read handled as a delayed transaction. The FPGA application indicates its desire to do this by asserting signal DelTrN. The timing on the PCI interface (Figure 25) is similar to that of an I/O read (Figure 23) except that stop is not asserted here to cause disconnect with data, but rather the operation is free to continue since it is allowed to complete on the source (PCI) bus before it completes on the destination (FPGA) bus. The FPGA interface timing is as shown in Figure 26 and is the same as the timing in the I/O accesses of Target I/O read, delayed transaction and Target I/O read, no delayed transaction.

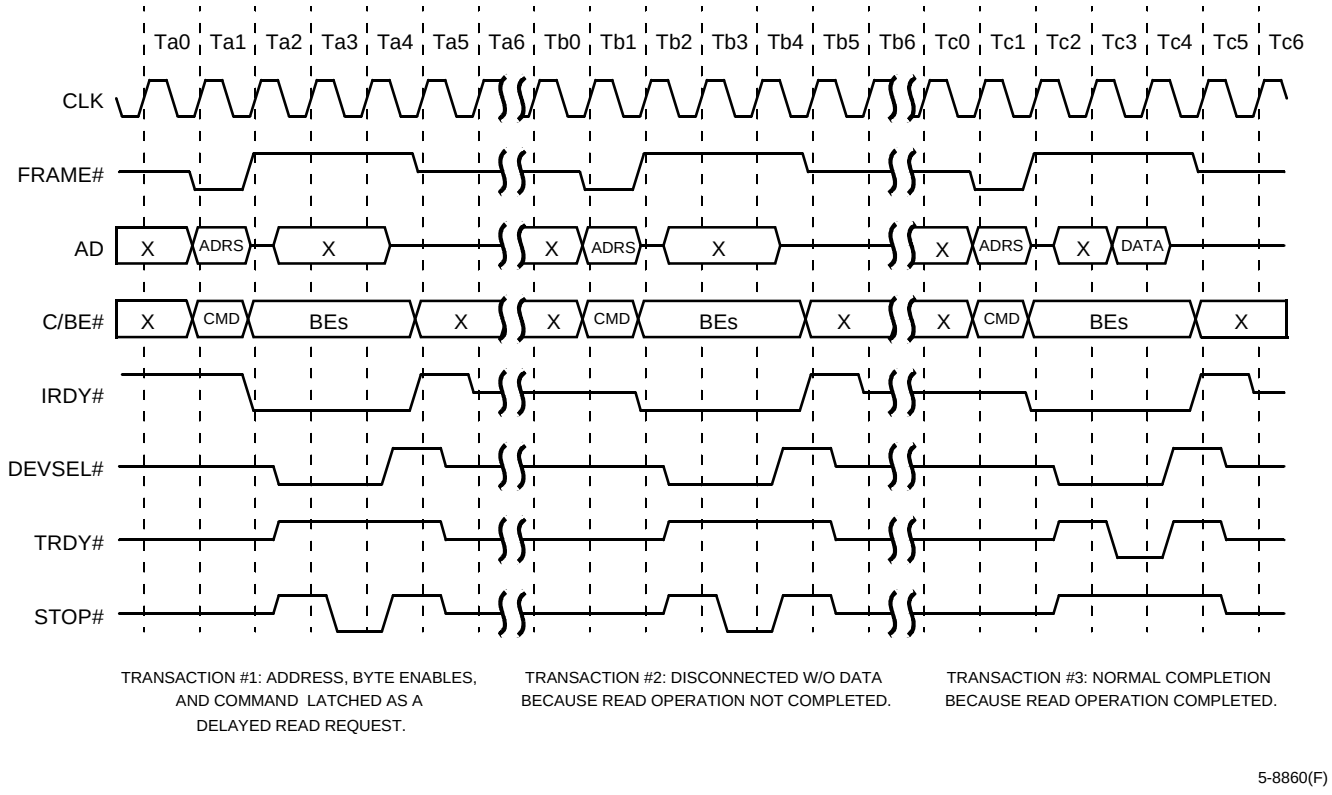


Figure 25. Target Memory Single Read, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

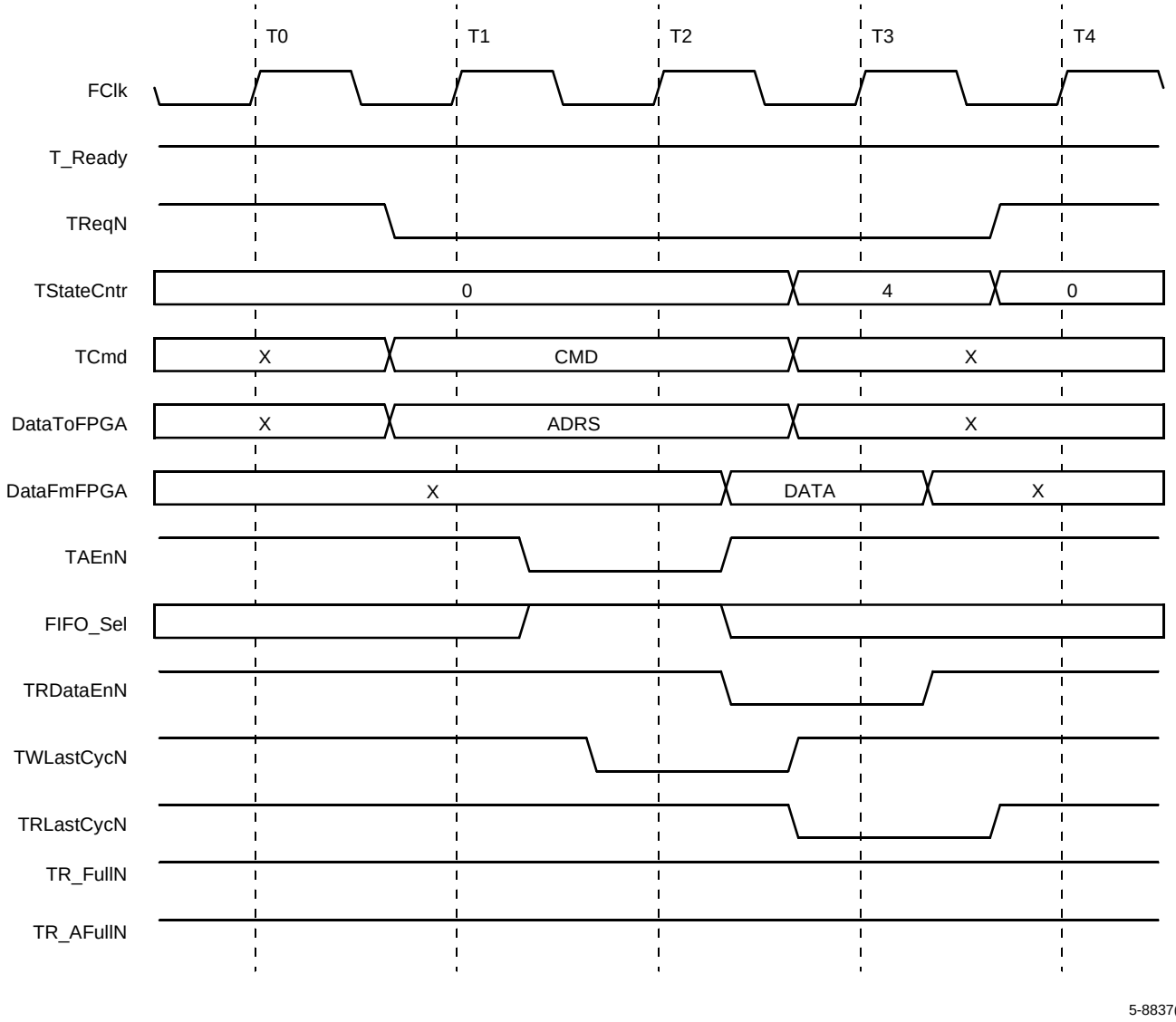


Figure 26. Target Read Single (FPGA Bus, Dual Port)

PCI Bus Core Detailed Description (continued)

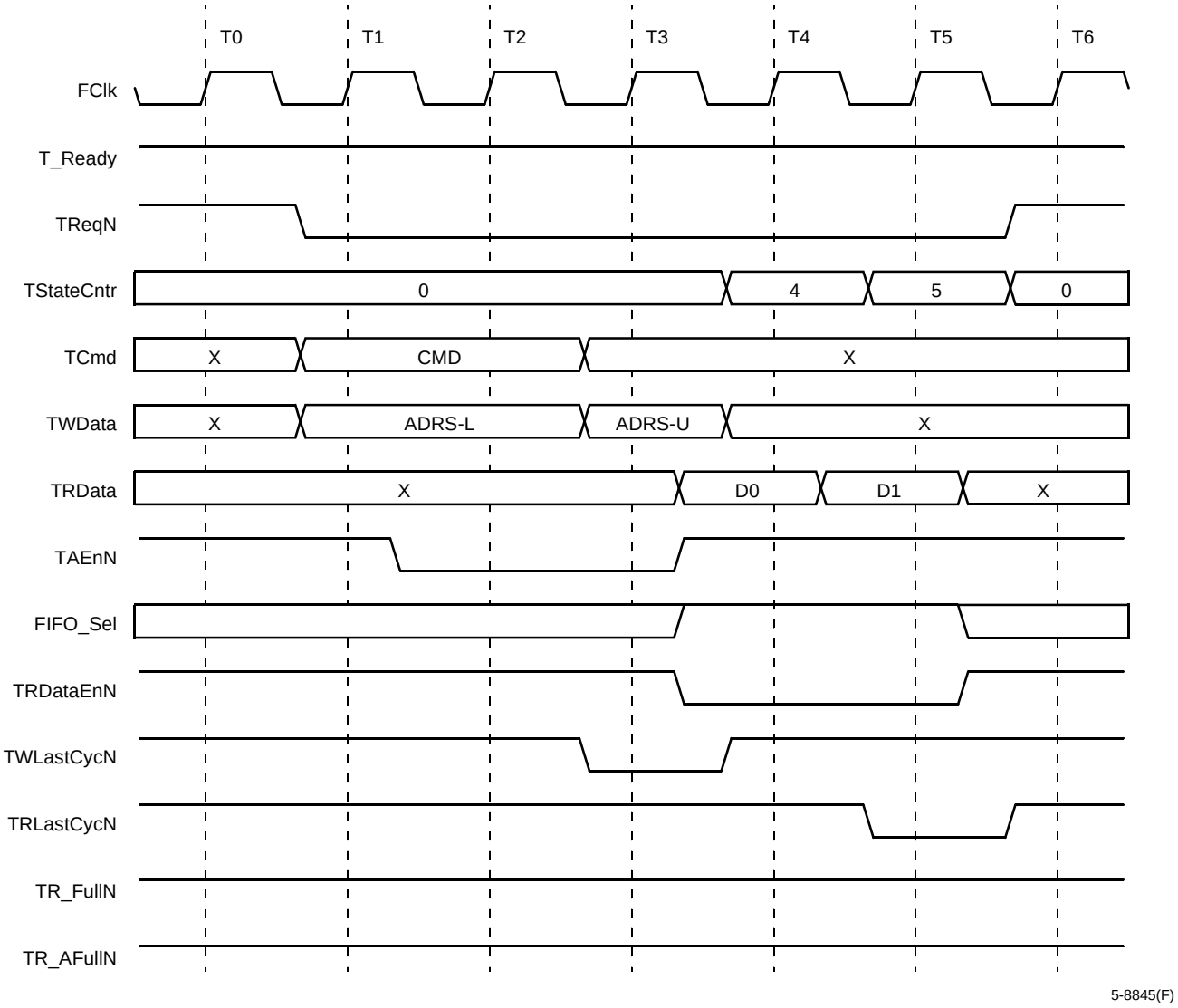


Figure 27. Target Read Single (FPGA Bus, Quad Port, 64-Bit Address)

PCI Bus Core Detailed Description (continued)

Target Read Memory, Nonburst, No Delayed Transaction

Figure 28 (PCI Bus), Figure 26 (FPGA Bus Dual Port), and Figure 27 (FPGA Bus Quad Port) show the timing for a Target memory nonburst read handled as an immediate (nondelayed) transaction. The FPGA application indicates its desire to do this by deasserting signal DelTrN. The timing on the PCI interface is shown in Figure 28. Here the PCI core accepts the transaction without issuing a retry but does not immediately assert TRDY#. Wait-states are inserted until the requested data in placed in the Target read FIFO, at which time TRDY# is asserted and the data is returned. If the FPGA application cannot fetch the data within the initial/subsequent latency time, the PCI core issues a retry or disconnect without data. The FPGA interface timing is as shown in Figure 26, and is the same as the timing in the accesses of Target I/O read, delayed transaction, and Target I/O read, no delayed transaction, and Target read memory non-burst, delayed transaction.

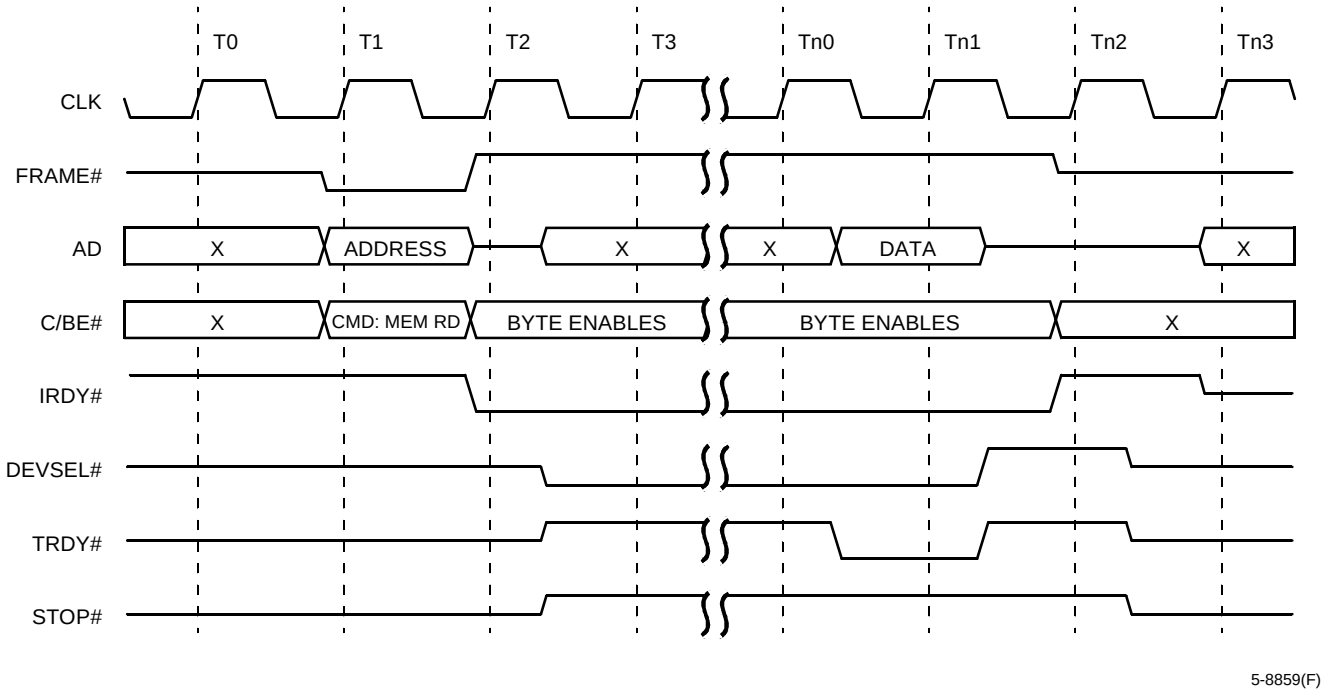


Figure 28. Target Memory Read Single, Not Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

Target Read Memory Burst, Delayed Transaction

Figure 29 (PCI Bus), Figure 30 (FPGA Bus Dual Port), and Figure 31 (FPGA Bus Quad Port) show the timing for a Target memory burst read of four quadwords handled as a delayed transaction. The FPGA application indicates its desire to do this by asserting signal DelTrN. On the PCI interface (Figure 29), three transactions are shown. In the first, the PCI core responds to the request after determining that the address matches one of its BARs by asserting DEVSEL#. However, since delayed transaction has been specified by the FPGA application by asserting signal DelTrN, the PCI core issues a retry. The PCI core now waits for the FPGA application to load the Target read FIFO; until this occurs, all memory and I/O accesses result in retries as exemplified by the second transaction in Figure 29. After the required data is loaded (either the first data word or a complete FIFO contents, depending on whether the Target read PCI bus hold signal TRPciHold is deasserted or asserted, respectively), the actual data transfer will occur as shown in the third transaction in Figure 29. The FPGA interface timing is as shown in Figure 30. This is similar to the timing for a Target nonburst read as shown in Figure 26 except that multiple data cycles are required as long as TRLastCycN is inactive-high.

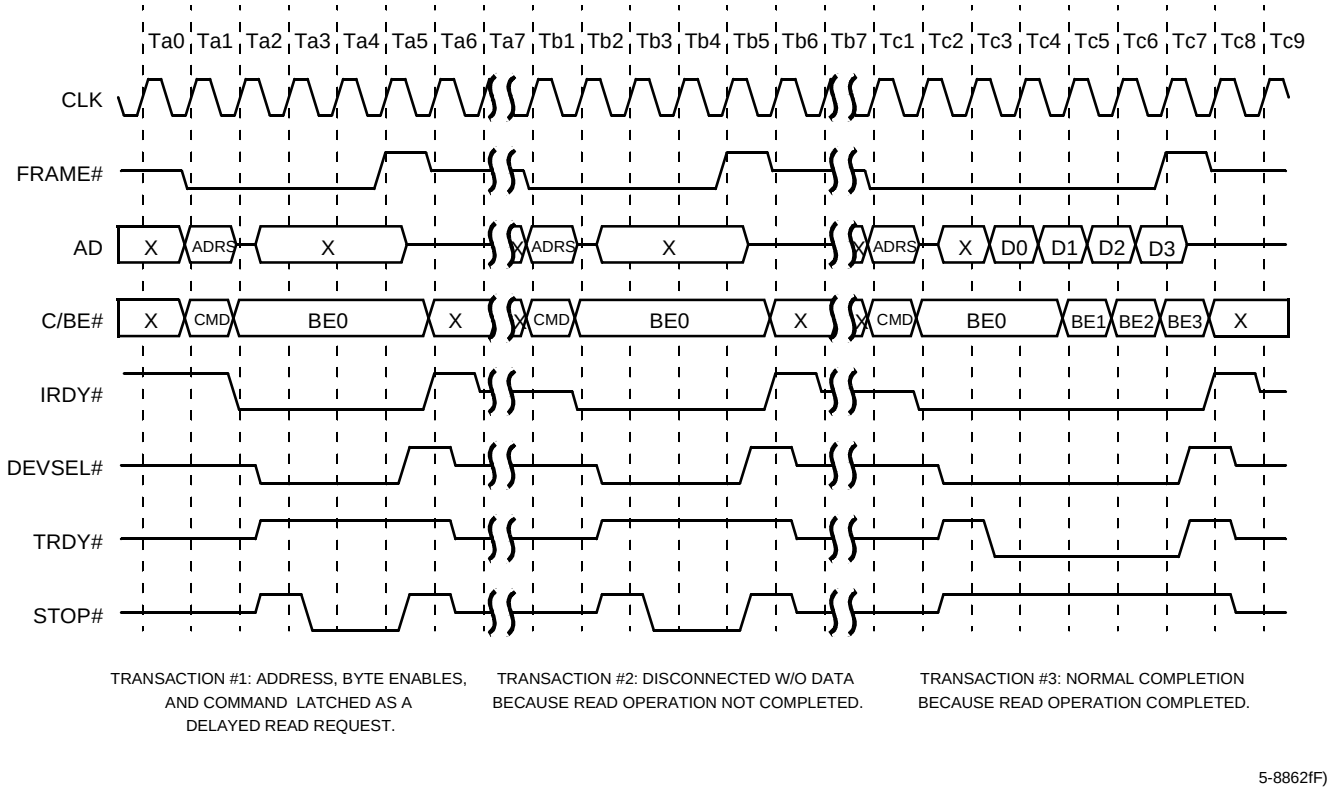


Figure 29. Target Memory Read 32-Byte Burst, Not Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description (continued)

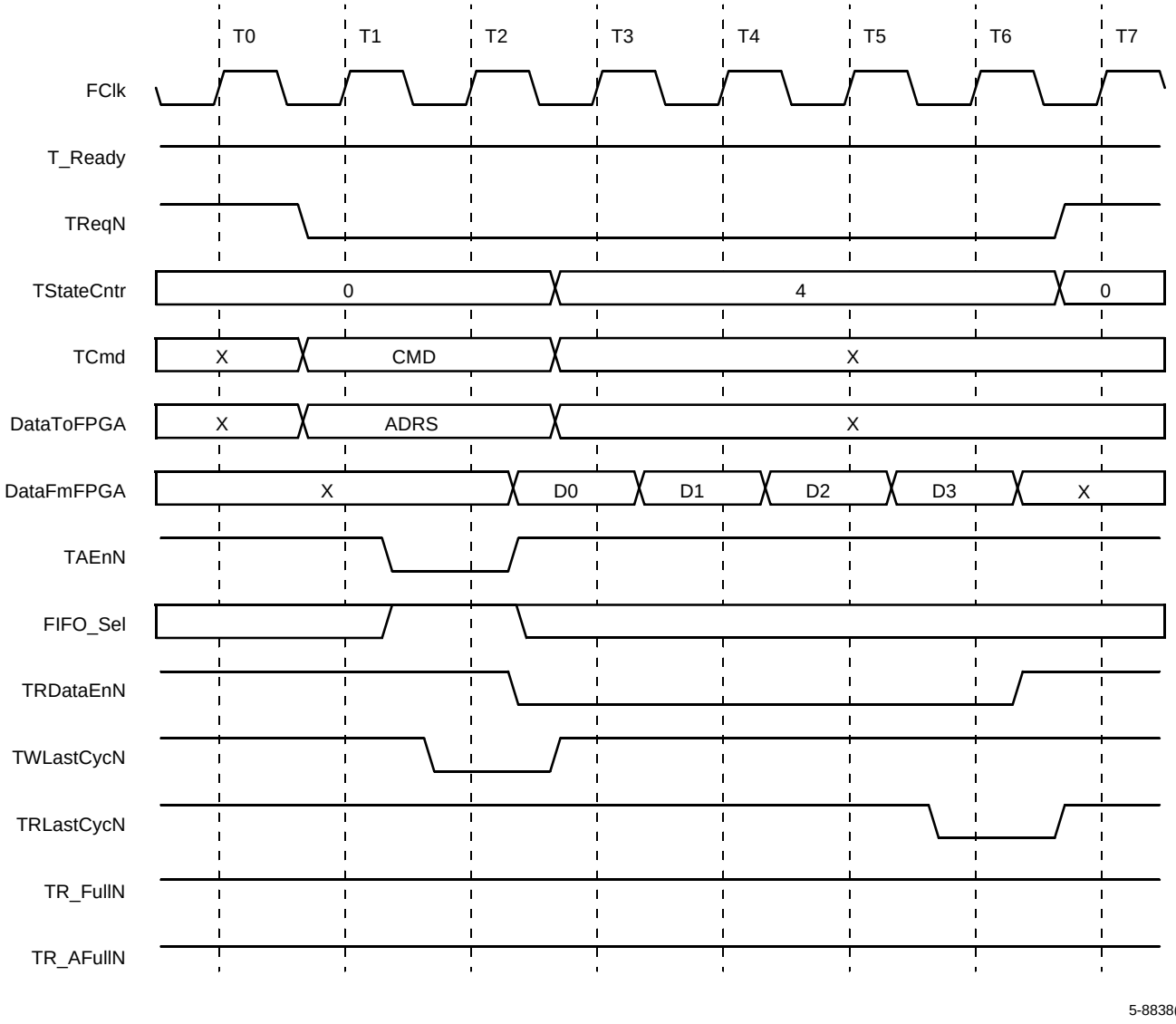


Figure 30. Target Read Memory 32-Byte Burst (FPGA, Dual Port)

PCI Bus Core Detailed Description (continued)

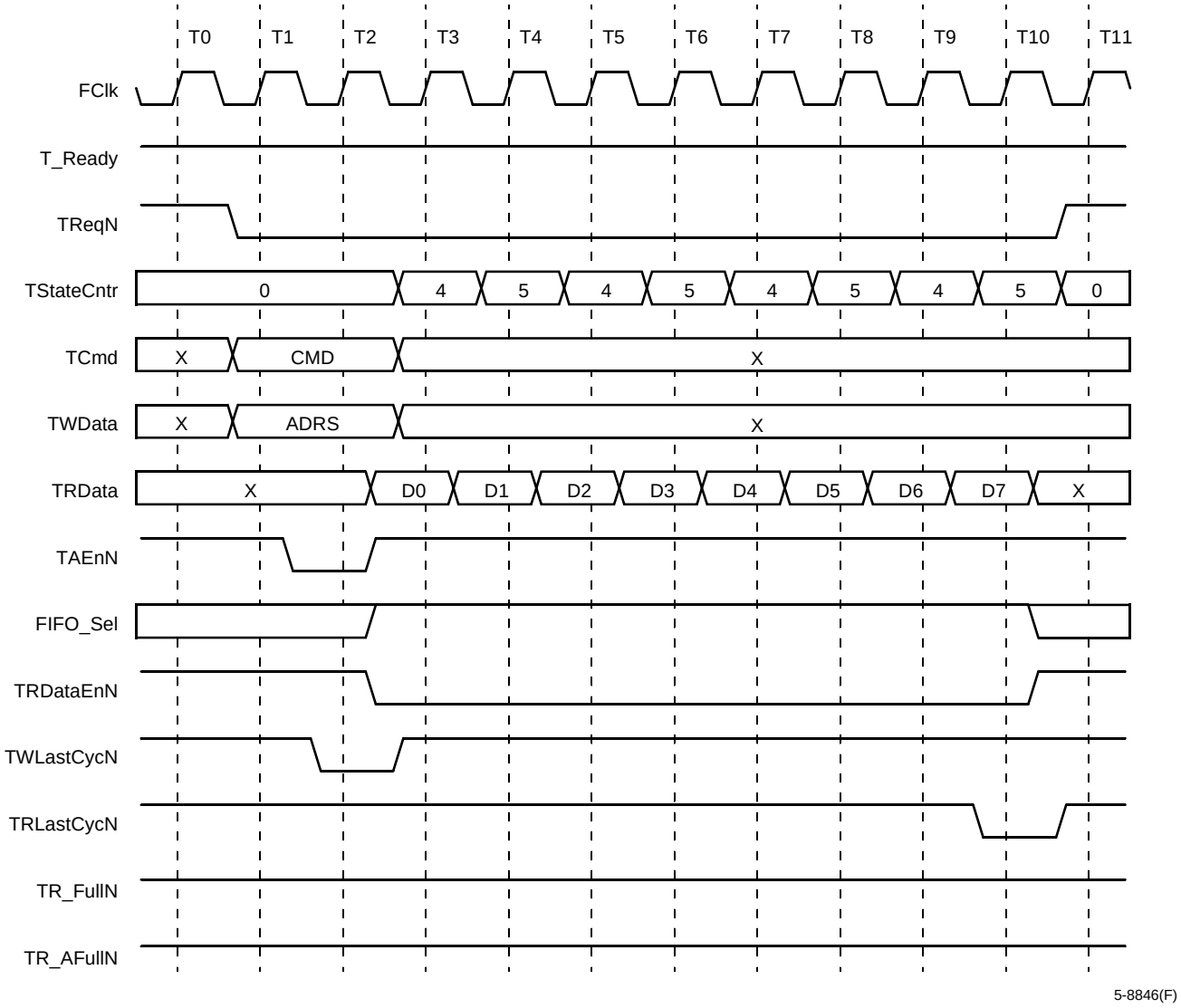


Figure 31. Target Read Memory 32-Byte Burst (FPGA Bus, Quad Port, 32-Bit Address)

PCI Bus Core Detailed Description (continued)

Target Read Memory Burst, No Delayed Transaction

Figure 32 (PCI Bus), Figure 30 (FPGA Bus Dual Port), and Figure 31 (FPGA Bus Quad Port) show the timing for a Target memory burst read of four quadwords handled as a nondelayed transaction. Figure 32 shows the timing on the PCI interface is similar to that of an I/O read (Figure 24) except that stop is not asserted here to cause disconnect with data, but rather the operation is free to continue since it is allowed to complete on the source (PCI) bus before it completes on the destination (FPGA) bus.

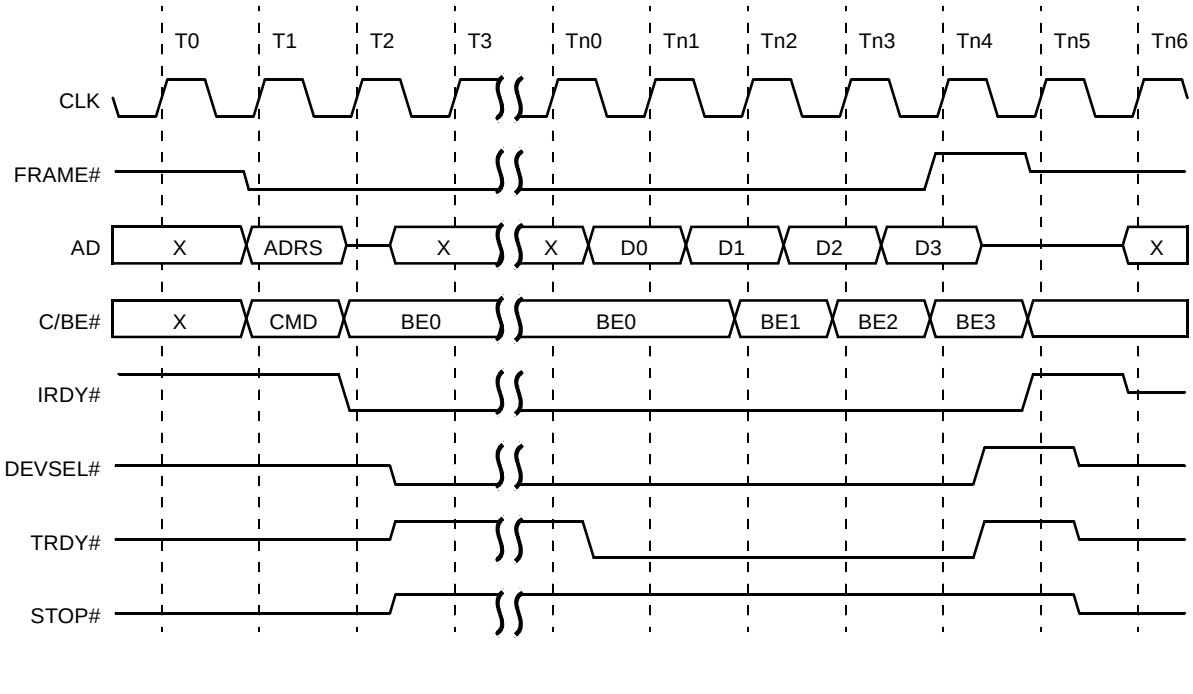


Figure 32. Target Read Memory Burst, No Delayed (PCI Bus, 32-Bit)

PCI Bus Core Detailed Description (continued)

Table 23. Dual-Port Target Read

TStateCntr	Next State of TStateCntr	Description	Bus	Notes
0	0	Idle	—	3
0	4	Address[63:0]	DataToFPGAx[7:0] DataToFPGA[63:0]	4, 18
4	4 or 0	Data[63:0]	DataFmFPGA[63:0]	10, 14

3. When TReqN is deasserted high, the interface is idle.
4. When TReqN is asserted low, a Target command is pending.
10. TAEnN must be deasserted high and TRDataEnN must be asserted low for data to transfer and state of TStateCntr to change (Target read data).
14. Next state = 0 if TWLastCycN is asserted low (end of Target read data).
18. TWLastCycN must be asserted low (end of Target address).

Table 24. Quad-Port Target Read

TStateCntr	Next State of TStateCntr	Description	Bus	Notes
0	0	Idle	—	3
0	1 or 4	Address[31:0]	DataToFPGAx[7:0] DataToFPGA[63:0]	4, 16
1	4	Address[63:32]	DataToFPGA[63:0]	6, 18
4	5 or 0	Data[31:0]	DataFmFPGA[31:0]	10, 14
5	4 or 0	Data[63:32]	DataFmFPGA[31:0]	10, 14

3. When TReqN is deasserted high, the interface is idle.
4. When TReqN is asserted low, a Target command is pending.
6. TAEnN must be asserted low for data to transfer and state of TStateCntr to change (Target Address).
10. TAEnN must be deasserted high and TRDataEnN must be asserted low for data to transfer and state of TStateCntr to change (Target read data).
14. Next state = 0 if TWLastCycN is asserted low (end of Target read data).
16. Next state = 4 if TWLastCycN is asserted low (end of Target address).
18. TWLastCycN must be asserted low (end of Target address).

PCI Bus Core Detailed Description (continued)

Configuration Space of the PCI Core

The following section describes the configuration space of the PCI core. This includes the layout and organization as called out in the PCI Specification as well as details specific to the PCI core's implementation. Note that the term configuration has two meanings: in the FPGA context, it refers to the programming of the FPGA's SRAM to define its functionality, and in the PCI context, it refers to the process of initializing the personality of the PCI agent residing at a specific location or card slot via a data space that is physically addressed. The PCI's configuration space is being discussed here.

PCI Bus Configuration Space Organization

Table 25 shows the layout of the PCI core's configuration space. The Header Type is 00 hex (non-PCI-to-PCI Bridge). All required and many optional features are implemented. Note that the defined space extends beyond 3F hex, and includes provisions for hot swap and FPGA configuration via the PCI bus. Table 26 further details the content and function of each register in the PCI configuration space.

Table 25. Configuration Space Layout

31	24	23	16	15	8	7	0	
Device ID				Vendor ID				00h
Status				Command				04h
Class Code						Revision ID		08h
BIST		Header Type		Latency Timer		Cache Line Size		0Ch
Base Address Registers								10h 14h 18h 1Ch 20h 24h
Cardbus CIS Pointer								28h
Subsystem ID				Subsystem Vendor ID				2Ch
Expansion ROM Base Address								30h
Reserved						Cap_Ptr		34h
Max_Lat		Min_Gnt		Interrupt Pin		Interrupt Line		3Ch
Reserved				FPGA Configuration Command-Status Register				40h
FPGA Configuration Data Register								44h
Scratch Register								48c
Reserved								40c
Reserved		HS_CSR		Next Item		Capability ID		48h
Reserved								54h thru FFh

PCI Bus Core Detailed Description (continued)

Table 26. Configuration Space Assignment

Bytes	Width	Bit	Description	Read/Write	Initial Value
00—01	16	—	Vendor ID	Read Only	11C1h (Lucent)
02—03	16	—	Device ID	Read Only	5400h (OR3LP26B)
04—05	16		Command:		
		0	Enable I/O Space	Read/Write	0
		1	Enable Memory Space	Read/Write	0
		2	Enable Bus Master	Read/Write	Note 1
		3	Enable Special Cycle	Read Only	0
		4	Enable Mem Wr & Inv	Read/Write	0
		5	Enable VGA Palette Snoop	Read Only	0
		6	Enable Par Err Response	Read/Write	0
		7	Enable Stepping	Read Only	0
		8	Enable SERR#	Read/Write	0
		9	Enable Fast Back-to-Back	Read/Write	0
		15—10	Reserved	Read Only	zeros
06—07	16		Status:		
		4—0	Reserved	Read Only	zeros
		5	66 MHz Capable	Read Only	1
		6	UDF Supported	Read Only	0
		7	Fast Back-to-Back	Read Only	1
		8	Data PERR# Detected	Note 2	0
		10—9	DEVSEL# Timing	Read Only	01b (medium)
		11	Target Abort Signaled	Note 2	0
		12	Target Abort Received	Note 2	0
		13	Master Abort Received	Note 2	0
		14	System Error Signaled	Note 2	0
		15	Parity Error Detected	Note 2	0
08	8	—	Revision ID	Read Only	Note 1
09—0B	24	—	Class Code	Read Only	Note 1
0C	8	—	Cache Line Size	Read Only	zeros
0D	8		Latency Timer:		
		7—3	Programmable Portion	Read/Write	zeros
		2—0	Granularity = 8 clks	Read Only	zeros

1. These values are intended to be custom assigned, per the intended application, by assigning constants via the FPGA configuration bit stream.
2. These exhibit special behavior per the PCI Specification:
   — Reads behave normally.
   — Writing a 1 clears the bit to zero.
   — Writing a 0 has no effect on the bit.

PCI Bus Core Detailed Description (continued)

Table 26. Configuration Space Assignment (continued)

Bytes	Width	Bit	Description	Read/Write	Initial Value
0E	8	—	Header Type	Read Only	00h
0F	8	—	BIST	Read Only	zeros
10—27	192		BAR	Note 3	Note 1
28—2B	32	—	Cardbus CIS Pointer	Read Only	zeros
2C—2D	16	—	Subsystem Vendor ID	Read Only	zeros
2E—2F	16	—	Subsystem ID	Read Only	Note 1
30—33	32	—	Expansion ROM Base Address	Read Only	zeros
34	8	—	(Capabilities Pointer)	—	50h
35—37	24	—	(Reserved)	Read Only	zeros
38—3B	32	—	(Reserved)	Read Only	zeros
3C	9	—	Interrupt Line	Read/Write	zeros
3D	8	—	Interrupt Pin	Read Only	01h (INTA#)
3E	8	—	Min_Gnt	Read Only	Note 1
3F	8	—	Max_Lat	Read Only	Note 1
40—41	16		FPGA Config. Command-Status Register:		
		15	Gsr — PCI Core Global Set/Reset	Read/Write	0
		14	ConfigFPGA — Enable FPGA Config.	Read/Write	0
		13	RdCfgN — Enable Readback	Read/Write	1
		12	PrgmN — Reset FPGA Config. Logic	Read/Write	1
		11	FastSlowN — Fast/Slow Config. Clock	Read/Write	0
		10	BitErr_1 — Error Signal from FPGA	Read Only	0
		9	BitErr_0 — Error Signal from FPGA	Read Only	0
		8	CfgBusy — Cfg Not In Idle State	Read Only	0
		7	RdBkNext — Readback Handshake	Read Only	0
		6	PciRegVld — Configuration Handshake	Read Only	0
		5	SRFull — Shift Reg Full	Read Only	0
		4	SREmpty — Shift Reg Empty	Read Only	0
		3	HndShkErr — Handshake Error	Read/Write	0
		2	InitN — FPGA's INITN	Read Only	Note 4
		1	Done — FPGA's DONE	Read Only	Note 4
		0	Mode — PCI Core Mode	Read Only	0

1. These values are intended to be custom assigned, per the intended application, by assigning constants via the FPGA configuration bit stream.
2. These exhibit special behavior per the PCI Specification:
   — Reads behave normally.
   — Writing a 1 clears the bit to zero.
   — Writing a 0 has no effect on the bit.
3. Bytes 10—27 hex contain the base address registers (BARs).
   — Any legal combination of memory and I/O BARs is permitted, as long as 64-bit BARs are naturally aligned, that is, they occupy bytes 10—17, 18—1F, or 20—27 hex.
   — Memory BARs may be marked as prefetchable/nonprefetchable by setting/resetting bit 3; however, the PCI core's behavior is not affected by this setting. In particular, the Target read operation may discard unused FIFO read-ahead data even though the data space is marked as nonprefetchable (this is not a violation, since the nonprefetchable bit only says that data can't be discarded once it has been sent over the PCI bus; nevertheless, caution must be exercised when this bit is reset).
4. These signals are tied to the FPGA signal of the same name and are not initialized.

PCI Bus Core Detailed Description (continued)

Table 26. Configuration Space Assignment (continued)

Bytes	Width	Bit	Description	Read/Write	Initial Value
42—43	16		(Reserved)	Read Only	zeros
44—47	32		FPGA Config. Data Register	Read/Write	zeros
48-4B	32		Scratch Register	Read/Write	zeros
4C	32		Reserved for Manufacturing Testing	Note 6	Note 6
50	8		Capability ID	Read Only	06h (Hot Plug)
51	8		Next Item	Read Only	00h (Last item)
52			Hot Swap Control Status Register:		
	7		INS ENUM# Status - Insertion	Note 5	1
	6		EXT ENUM# Status - Extraction	Note 5	0
	5		Reserved	Read Only	0
	4		Reserved	Read Only	0
	3		LOO	Read/Write	0
	2		Reserved	Read Only	0
	1		EIM ENUM# Signal Mark	Read/Write	0
	0		Reserved	Read Only	0

1. These values are intended to be custom assigned, per the intended application, by assigning constants via the FPGA configuration bit stream.
2. These exhibit special behavior per the PCI Specification:
   — Reads behave normally.
   — Writing a 1 clears the bit to zero.
   — Writing a 0 has no effect on the bit.
3. Bytes 10—27 hex contain the base address registers (BARs).
   — Any legal combination of memory and I/O BARs is permitted, as long as 64-bit BARs are naturally aligned, that is, they occupy bytes 10—17, 18—1F, or 20—27 hex.
   — Memory BARs may be marked as prefetchable/nonprefetchable by setting/resetting bit 3; however, the PCI core's behavior is not affected by this setting. In particular, the Target read operation may discard unused FIFO read-ahead data even though the data space is marked as nonprefetchable (this is not a violation, since the nonprefetchable bit only says that data can't be discarded once it has been sent over the PCI bus; nevertheless, caution must be exercised when this bit is reset).
4. These signals are tied to the FPGA signal of the same name and are not initialized.
5. These exhibit special behavior per the CompactPCI Hot Swap Specification:
   — Read behave normally.
   — Writing a 1 clears the bit to zero.
   — Writing a 0 has no effect on the bit.
6. This 32-bit register is used during manufacturing test. Writes are not allowed; reads are allowed and cause no side effects, but the value returned is undefined.

PCI Bus Core Detailed Description

(continued)

FPSC Configuration

The OR3LP26B FPSC provides the designer many configuration options. In addition to all the configuration options provided in the standard Series 3 architecture (except Master parallel mode) including configuration via the microprocessor and boundary-scan (JTAG) interfaces, the OR3LP26B PCI FPSC also allows configuration via the PCI interface. With this capability, many configuration schemes can be implemented. For example, a generic FPSC configuration can be loaded via a serial configuration PROM and updated via the PCI bus or the microprocessor interface. The FPSC can also be reprogrammed in the field, or the configuration can be dynamically modified to perform different tasks.

When the FPSC is configured via the PCI interface, there is a priority issue that must be resolved. The Subsystem vendor ID and subsystem ID that reside at 2Ch—2Fh in the PCI configuration space can be assigned during FPGA configuration, but these same pieces of information may be needed by system software to determine which FPSC configuration bit stream to use for each FPSC when two or more FPSCs reside on one PCI bus. For this reason, the OR3LP26B FPSC is designed to allow for two different configuration schemes.

The first option is more flexible; in this scheme, the FPSC is first configured without employing the PCI interface (e.g., via serial PROM). The access to the FPSC's configuration registers via the PCI interface occurs after this first configuration completes, so that when the subsystem vendor ID and subsystem ID are finally read, they properly and uniquely identify the card on which the FPSC resides. This initial configuration bit stream is only required to provide correct subsystem vendor ID and subsystem ID values for system software use, but it may in addition be the first version of the FPSC's application code. The PCI system software is then able to invoke the proper procedures that will reconfigure the FPSC using the desired version of the configuration bit stream.

The disadvantage of the first option is that it requires that the FPSC be preconfigured prior to receiving the working bit stream via the PCI interface. In a proprietary system, however, a **second option** may be employed if the configuring software may already know which bit stream to use to configure the FPSC. The system software can simply locate the OR3LP26B by reading the vendor ID and device ID, and then proceed directly to FPSC configuration via the PCI bus. This feature takes advantage of the fact that the PCI interface is functional even before the FPSC has been configured.

PCI Bus Core Detailed Description

(continued)

Configuration via PCI Bus

The OR3LP26B is configured using locations 40 hex through 47 hex. These registers are dedicated to the FPSC configuration and readback functions, as detailed in Tables 36 and 37. The FPGA configuration control-status register (FCCSR) is a 16-bit register at address 40 hex—41 hex, and the FPGA configuration data register (FCDR) is a 32-bit register at address 44 hex—47 hex.

The following is an example sequence which configures the FPSC via the PCI interface:

1. Read the vendor ID and device ID registers. If the vendor ID is 11C1 hex, the vendor, or chip manufacturer, is Lucent. If, in addition, the device ID is 5400 hex, the device is a Lucent OR3LP26B PCI FPSC; go to step 2.
2. At this point, the configuration software may do one of two things. If this is a proprietary system and the configuration software already knows how to configure any Lucent OR3LP26B, the software may skip the next two steps, and the FPSC does not need to be preconfigured. If this is a standard system, the configuration software must perform the next two steps to uniquely identify the application that is utilizing the OR3LP26B.
3. Read the FCCSR until Done goes active-high, signaling that the FPSC preconfiguration operation has completed, typically via a serial configuration PROM (note that the serial configuration PROM needs only to program the registers in the next step, and thus may be very small).
4. Read the class code, revision ID, subsystem vendor ID, and subsystem ID registers. This information is programmed into the FPSC by the preconfiguration step. This information is used by the configuration software to locate the correct FPSC configuration bit stream and driver for the FPSC's application, and is provided by the manufacturer of the adapter card containing the FPSC.
5. Read the FCCSR until Bit 0 goes high. If communication with the FPSC is underway via the boundary-scan hardware, this signal will remain inactive low until it completes.
6. Write to the FCCSR three times, first with PrgmN high, then active-low, then high; then write to the FCCSR with ConfigFPGA active-high. This will initiate an FPSC configuration session via the PCI interface.
7. Write a DWORD of FPSC configuration data to FCDR. This will set PciRegVld in the FCCSR to active-high, indicating that it holds a valid DWORD of data.
8. Read the FCCSR until PciRegVld goes inactive low, indicating that the DWORD it contained has been transferred to the shift register that feeds the serial configuration data to the FPSC.
9. Repeat steps 6 and 7 until all the configuration data has been written.
10. Read the FCCSR and verify that Done went active-high, indicating that the configuration was successful.

Readback via PCI interface

The procedure for performing a readback via the PCI interface is similar to the above procedure for configuring, and also similar to the standard readback procedure. The steps are outlined below:

1. Read the FCCSR until Bit 0 goes high. If communication with the FPSC is underway via the boundary-scan hardware, this signal will remain inactive low until it completes.
2. Write to the FCCSR with RdCfgN active-low. This enables the readback mode.
3. Read the FCCSR until SRegFull goes active-high, indicating that a DWORD of data is available in register FCDR.
4. Read the data from the FCDR.
5. Repeat steps 3 and 4 until all readback data has been accessed.

Interaction Among Configuration Modes

The basic configuration options, including configuration via the microprocessor and boundary-scan interfaces, are performed in a manner identical to that of *ORCA* Series 3 FPGAs. FPSC configuration via the PCI interface is available at any time, either prior to or after the FPSC has been configured and regardless of the value to which the FPGA configuration mode pins (M2, M1, and M0) have been strapped. In addition, a PCI-directed configuration will override any strapped configuration operation already underway, an FPGA configuration via the boundary-scan interface will override one via the PCI interface, and the $\overline{\text{PRGM}}$ pin overrides both.

PCI Bus Core Detailed Description

(continued)

Clocking Options at FPGA/Core Boundary

The OR3LP26B supports a wide variety of integrated FPGA/core clocking schemes which, in conjunction with the asynchronous interface between the PCI bus and the FPGA provided by four FIFOs, gives the designer many flexible options.

The Master and Target FIFOs are independently clocked on the FPGA side by either FClk1 or FClk2. The clocks used for the Master FIFO and Target FIFO interfaces to the FPGA logic are independent when the interface is configured in quad-port mode, but they must be tied to the same clock signal for dual-port mode.

Figure 33 illustrates the special clock paths provided to service the clocking needs of PCI functions. The various clocking options shown in Figure 33 are discussed below.

PCI Clock as System Clock

The clock received from the PCI interface can be brought across the PCI core into the FPGA logic section and used as the clock for the entire FPSC, or even as the clock for the entire board on which the FPSC resides. It is important that this signal be available via the PCI core since PCI rules allow for only one load per agent on the PCI bus clock. The FPSC incorporates special clock lines for the purpose of distributing the PCI clock; these lines are hard-connected to the PCI core's circuitry but can also be passed up onto the FPGA portion's clock grid. From there, in addition to feeding clocks to all PFUs and PIOs, this clock can also drive the clock inputs to the FPGA side of the Master and/or Target FIFOs, and can be made available off-chip.

Local Clock as System Clock

The FIFO-buffered interface between the PCI logic and the FPGA allows other clocks to be utilized in the FPGA as well. The Master and Target interfaces each have independent clock nets and can be connected to the same or separate clocks. Essentially, this means that both the Master and Target logic and FIFOs can be independently set to use the PCI clock or another clock. The clocks for the FPGA side of the PCI core's Master and Target FIFOs are fed from specific vertical clock spines, namely, the spines in PLC columns 9 and 19; consequently, minimum clock net delay is obtained by feeding external clocks from I/O pads in these locations on the top of the array. Nevertheless, clocks can be fed from any I/O pad, from express clock inputs, or from internal logic, and can be fed via the programmable clock manager (PCM).

PCI Bus Core Detailed Description (continued)

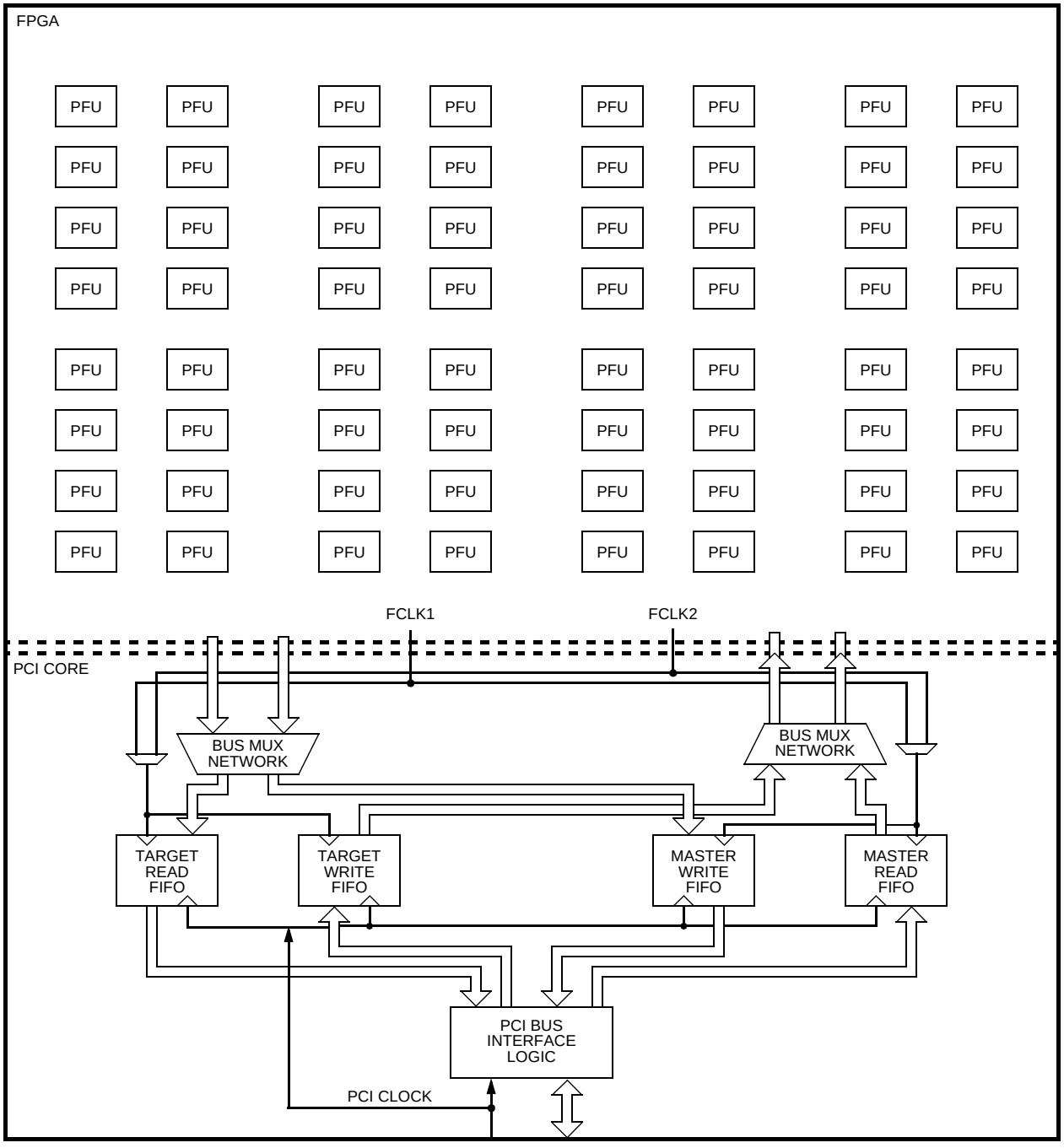


Figure 33. FPSC Block Diagram and Clock Network

FPGA Configuration Data Format

The *ORCA* Foundry Development System interfaces with front-end design entry tools and provides tools to produce a fully configured FPSC. This section discusses using the *ORCA* Foundry Development System to generate configuration RAM data and then provides the details of the configuration frame format.

Using *ORCA* Foundry to Generate Configuration RAM Data

The configuration data bit stream defines the PCI embedded core configuration, the FPGA logic functionality, and the I/O configuration and interconnection. The data bit stream is generated by the *ORCA* Foundry development tools. The bit stream created by the bit stream generation tool is a series of 1s and 0s used to write the FPSC configuration RAM. It can be loaded into the FPSC using one of the configuration modes discussed elsewhere in this data sheet.

For FPSCs, the bit stream is prepared in two separate steps in the design flow. The configuration options of the embedded core are specified using *ORCA* OR3LP26B Design Kit Software at the beginning of the design process. This offers the designer a specific configuration to simulate and design the FPGA logic to. Upon completion of the design, the bit stream generator combines the embedded core options and the FPGA configuration into a single bit stream for download into the FPSC.

FPGA Configuration Data Frame

Configuration data can be presented to the FPSC in two frame formats: autoincrement and explicit. A detailed description of the frame formats is shown in Figure 34, Figure 35, and Table 27. The two modes are similar except that autoincrement mode uses assumed address incrementation to reduce the bit stream size, and explicit mode requires an address for each data frame. In both cases, the header frame begins with a series of 1s and a preamble of 0010, followed by a 24-bit length count field representing the total number of configuration clocks needed to complete the loading of the FPSC.

The mandatory ID frame contains data used to determine if the bit stream is being loaded to the correct type of *ORCA* device (i.e., a bit stream generated for an OR3LP26B is being sent to an OR3LP26B). Error checking is always enabled for Series 3+ devices, through the use of an 8-bit checksum. One bit in the ID frame also selects between the autoincrement and explicit address modes for this load of the configuration data.

A configuration data frame follows the ID frame. A data frame starts with a 01-start bit pair and ends with enough 1-stop bits to reach a byte boundary. If using autoincrement configuration mode, subsequent data frames can follow. If using explicit mode, one or more address frames must follow each data frame, telling the FPSC at what addresses the preceding data frame is to be stored (each data frame can be sent to multiple addresses).

Following all data and address frames is the postamble. The format of the postamble is the same as an address frame with the highest possible address value with the checksum set to all ones.

FPGA Configuration Data Format (continued)

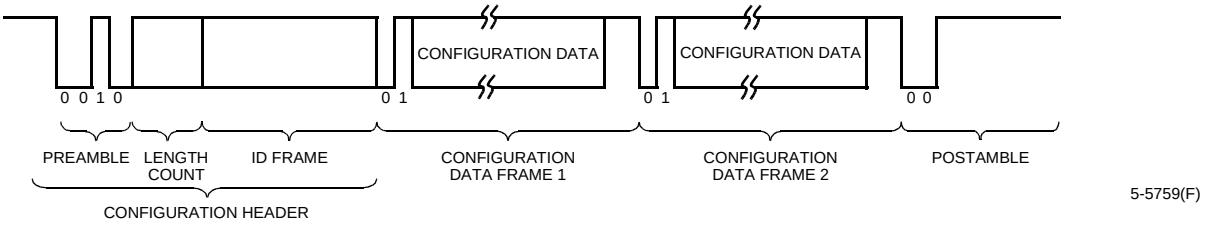


Figure 34. Serial Configuration Data Format—Autoincrement Mode

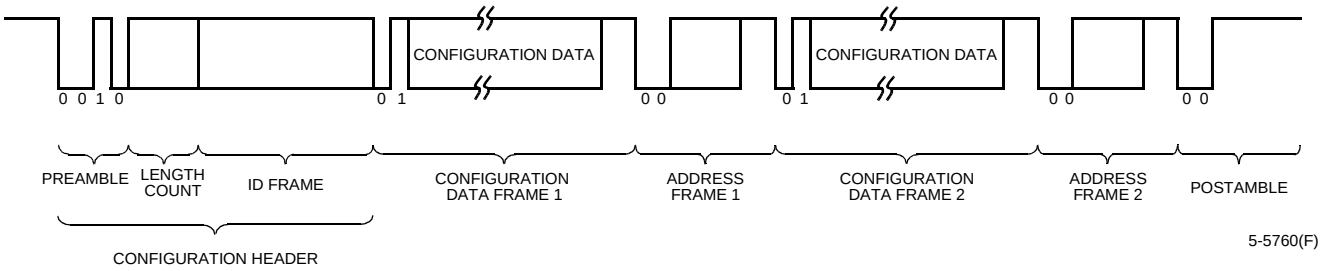


Figure 35. Serial Configuration Data Format—Explicit Mode

Table 27. Configuration Frame Format and Contents

Header	11110010	Preamble.
	24-bit Length Count	Configuration frame length.
	11111111	Trailing header—8 bits.
ID Frame	0101 1111 1111 1111	ID frame header.
	Configuration Mode	00 = autoincrement, 01 = explicit.
	Reserved [41:0]	Reserved bits set to 0.
	ID	20-bit part ID.
	Checksum	8-bit checksum.
	11111111	Eight stop bits (high) to separate frames.
Configuration Data Frame (repeated for each data frame)	01	Data frame header.
	Data Bits	Number of data bits depends upon device.
	Alignment Bits = 0	String of 0 bits added to bit stream to make frame header, plus data bits reach a byte boundary.
	Checksum	8-bit checksum.
	11111111	Eight stop bits (high) to separate frames.
Configuration Address Frame	00	Address frame header.
	14 Address Bits	14-bit address of location to start data storage.
	Checksum	8-bit checksum.
	11111111	Eight stop bits (high) to separate frames.
Postamble	00	Postamble header.
	11111111 111111	Dummy address.
	1111111111111111	16 stop bits.

Note: For slave parallel mode, the byte containing the preamble must be 11110010. The number of leading header dummy bits must be (n * 8) + 4, where n is any nonnegative integer and the number of trailing dummy bits must be (n * 8), where n is any positive integer. The number of stop bits/frame for slave parallel mode must be (x * 8), where x is a positive integer. Note also that the bit stream generator tool supplies a bit stream that is compatible with all configuration modes, including slave parallel mode.

FPGA Configuration Data Format

(continued)

The length and number of data frames and information on the PROM size for the OR3LP26B is given in Table 28.

Table 28. Configuration Frame Size

Devices	OR3LP26B
# of Frames	1880
Data Bits/Frame	292
Configuration Data (# of frames • # of data bits/frame)	548,960
Maximum Total # Bits/Frame (align bits, 01 frame start, 8-bit checksum, 8 stop bits)	312
Maximum Configuration Data (# bits/frame • # of frames)	586,560
Maximum PROM Size (bits) (add configuration header and postamble)	586,728

Bit Stream Error Checking

There are three different types of bit stream error checking performed in the *ORCA* Series 3+ FPSCs: ID frame, frame alignment, and CRC checking.

The ID data frame is sent to a dedicated location in the FPSC. This ID frame contains a unique code for the device for which it was generated. This device code is compared to the internal code of the FPSC. Any differences are flagged as an ID error. This frame is automatically created by the bit stream generation program in *ORCA* Foundry.

Each data and address frame in the FPSC begins with a frame start pair of bits and ends with eight stop bits set to 1. If any of the previous stop bits were a 0 when a frame start pair is encountered, it is flagged as a frame alignment error.

Error checking is also done on the FPSC for each frame by means of a checksum byte. If an error is found on evaluation of the checksum byte, then a checksum/parity error is flagged.

When any of the three possible errors occur, the FPSC is forced into an idle state, forcing $\overline{\text{INIT}}$ low. The FPSC will remain in this state until either the $\overline{\text{RESET}}$ or $\overline{\text{PRGM}}$ pins are asserted.

If using either of the MPI modes or the PCI embedded core to configure the FPSC, the specific type of bit stream error is written to one of the MPI registers or a PCI register, respectively, by the FPGA configuration logic. The $\overline{\text{PGRM}}$ bit of the MPI control register or the PCI embedded core can also be used to reset out of the error condition and restart configuration.

FPGA Configuration Modes

There are eight methods for configuring the FPSC. Six of the configuration modes are selected on the M0, M1, and M2 input and are shown in Table 29. The seventh mode is PCI bus configuration as previously discussed and the eighth configuration mode is accessed through the boundary-scan interface. A fourth input, M3, is used to select the frequency of the internal oscillator, which is the source for CCLK in some configuration modes. The nominal frequencies of the internal oscillator are 1.25 MHz and 10 MHz. The 1.25 MHz frequency is selected when the M3 input is unconnected or driven to a high state.

Note that the Master parallel mode of configuration that is available in the *ORCA* Series 3 FPGAs is not available in the OR3LP26B. This is due to the use of Master parallel configuration pins for the PCI bus interface.

More information on the general FPGA modes of configuration can be found in the *ORCA* Series 3 data sheet.

Table 29. Configuration Modes

M2	M1	M0	CCLK	Configuration Mode	Data
0	0	0	Output	Master Serial	Serial
0	0	1	Input	Slave Parallel	Parallel
0	1	0	Output	Microprocessor: *Motorola* PowerPC*	Parallel
0	1	1	Output	Microprocessor: *Intel† i960*	Parallel
1	0	0	Reserved		
1	0	1	Output	Async Peripheral	Parallel
1	1	0	Reserved		
1	1	1	Input	Slave Serial	Serial

* *Motorola* is a registered trademark of Motorola, Inc.
† *Intel* is a registered trademark of Intel Corporation.

Absolute Maximum Ratings

Stresses in excess of the absolute maximum ratings can cause permanent damage to the device. These are absolute stress ratings only. Functional operation of the device is not implied at these or any other conditions in excess of those given in the operations sections of this data sheet. Exposure to absolute maximum ratings for extended periods can adversely affect device reliability.

The *ORCA* Series 3+ FPSCs include circuitry designed to protect the chips from damaging substrate injection currents and to prevent accumulations of static charge. Nevertheless, conventional precautions should be observed during storage, handling, and use to avoid exposure to excessive electrical stress.

Table 30. Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Storage Temperature	T_{stg}	–65	150	°C
Supply Voltage with Respect to Ground	V_{DD}	–0.5	7.0	V
Input Signal with Respect to Ground	—	–0.5	V_{DD} + 0.3	V
Signal Applied to High-impedance Output	—	–0.5	V_{DD} + 0.3	V
Maximum Package Body Temperature	—	—	220	°C

* For PCI bus signals used for 5 V signaling and FPGA inputs used as 5 V tolerant the maximum value is 5.8 V.

Recommended Operating Conditions

Table 31. Recommended Operating Conditions

Mode	OR3LP26B		
	Temperature Range (Ambient)	I/O Supply Voltage (V_{DD})	Internal Supply Voltage ($V_{DD}2$)
Commercial	0 °C to 70 °C	3.0 V to 3.6 V	2.5 V ± 5%
Industrial	–40 °C to +85 °C	3.0 V to 3.6 V	2.3 V to 2.7 V

Note: The maximum recommended junction temperature (T_J) during operation is 125 °C.

Electrical Characteristics

Table 32. Electrical Characteristics

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, $V_{DD}2$ = 2.3 V to 2.7 V, 0 °C < T_A < 70 °C; Industrial: V_{DD} = 3.0 V to 3.6 V, $V_{DD}2$ = 2.3 V to 2.7 V, –40 °C < T_A < +85 °C.

Parameter	Symbol	Test Conditions	OR3LP26B		Unit
			Min	Max	
Input Voltage: High Low	 V_{IH} V_{IL}	Input configured as CMOS (clamped to $V_{DD}2$)	 50% VDD GND – 0.5	 VDD + 0.3 30% VDD	 V V
Input Voltage: High Low	 V_{IH} V_{IL}	Input configured as TTL (5 V tolerant)	 50% VDD GND – 0.5	 5.8 V 30% VDD	 V V
Output Voltage: High Low	 V_{OH} V_{OL}	 V_{DD} = min, I_{OH} = 6 mA or 3 mA V_{DD} = min, I_{OL} = 12 mA or 6 mA	 2.4 —	 — 0.4	 V V
Input Leakage Current	IL	V_{DD} = max, V_{IN} = V_{SS} or V_{DD}	–10	10	µA
Standby Current	IDDSB	(T_A = 25 °C, V_{DD} = 3.3 V, $V_{DD}2$ = 2.5 V) internal oscillator running, no output loads, inputs at V_{DD} or GND (after configuration)	—	TBD	mA
Standby Current	IDDSB	(T_A = 25 °C, V_{DD} = 3.3 V, $V_{DD}2$ = 2.5 V) internal oscillator stopped, no output loads, inputs at V_{DD} or GND (after configuration)	—	TBD	mA
Data Retention Voltage	VDR	T_A = 25 °C	TBD	—	V
Powerup Current	IPP	Power supply current at approximately 1 V, within a recommended power supply ramp rate of 1 ms—200 ms	TBD	—	mA
Input Capacitance	CIN	T_A = 25 °C, V_{DD} = 3.3 V, $V_{DD}2$ = 2.5 V Test frequency = 1 MHz	—	8	pF
Output Capacitance	COUT	T_A = 25 °C, V_{DD} = 3.3 V, $V_{DD}2$ = 2.5 V Test frequency = 1 MHz	—	8	pF
DONE Pull-up Resistor*	RDONE	—	100	—	kΩ
M[3:0] Pull-up Resistors*	RM	—	100	—	kΩ
I/O Pad Static Pull-up Current*	IPU	V_{DD} = 3.6 V, V_{IN} = V_{SS}, T_A = 0 °C	14.4	50.9	µA
I/O Pad Static Pull-down Current	IPD	V_{DD} = 3.6 V, V_{IN} = V_{SS}, TA = 0 °C	26	103	µA
I/O Pad Pull-up Resistor*	RPU	V_{DD} = all, V_{IN} = V_{SS}, T_A = 0 °C	100	—	kΩ
I/O Pad Pull-down Resistor	RPD	V_{DD} = all, V_{IN} = V_{DD}, T_A = 0 °C	50	—	kΩ

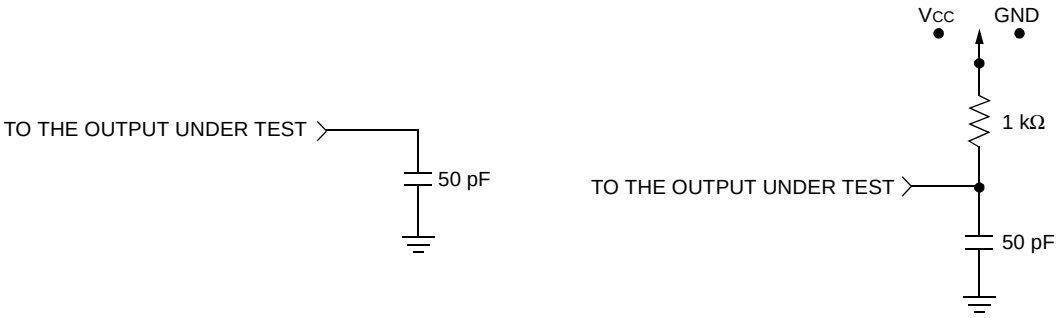
* On the Series 3 devices, the pull-up resistor will externally pull the pin to a level 1.0 V below V_{DD}.

Timing Characteristics

This section will be included in a future release of this data sheet.

General FPGA timing parameters may be found in the *ORCA* Series 3 FPGA data sheet.

Input/Output Buffer Measurement Conditions



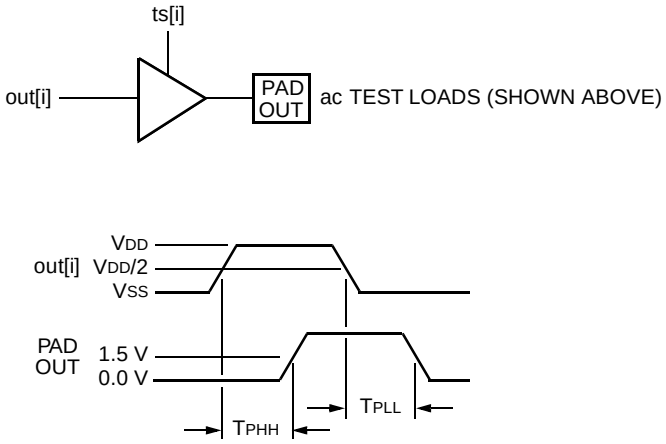
A. Load Used to Measure Propagation Delay

B. Load Used to Measure Rising/Falling Edges

5-3234(F)

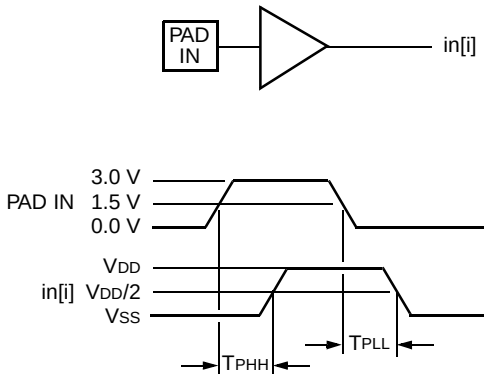
Note: Switch to V_{DD} for T_{PLZ}/T_{PZL}; switch to GND for T_{PHZ}/T_{PZH}.

Figure 36. ac Test Loads



5-3233.a(F)

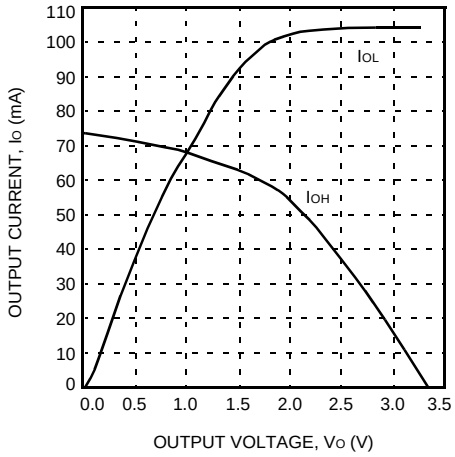
Figure 37. Output Buffer Delays



5-3235(F)

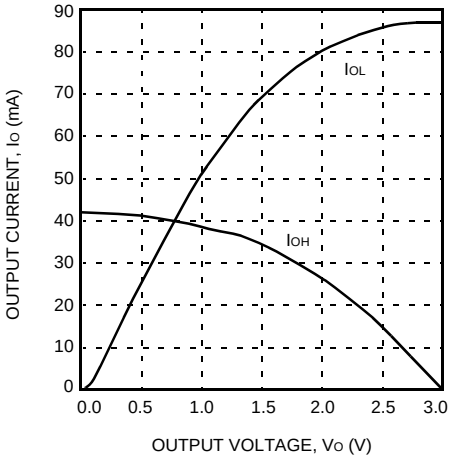
Figure 38. Input Buffer Delays

Output Buffer Characteristics



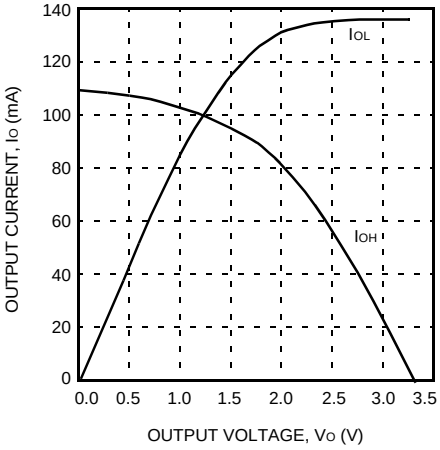
5-6865(F)

Figure 39. Sinklim (T_J = 25 °C, V_{DD} = 3.3 V)



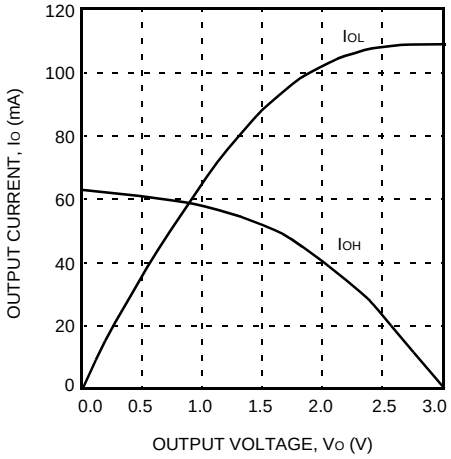
5-6866(F)

Figure 42. Sinklim (T_J = 125 °C, V_{DD} = 3.0 V)



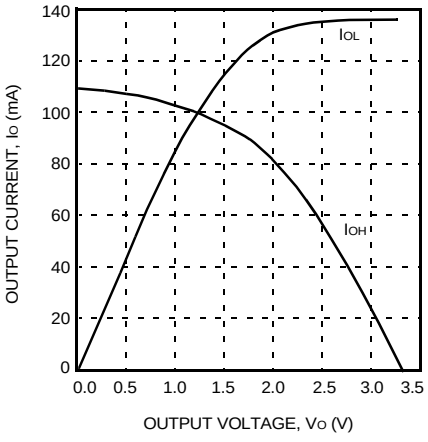
5-6867(F)

Figure 40. Slewlim (T_J = 25 °C, V_{DD} = 3.3 V)



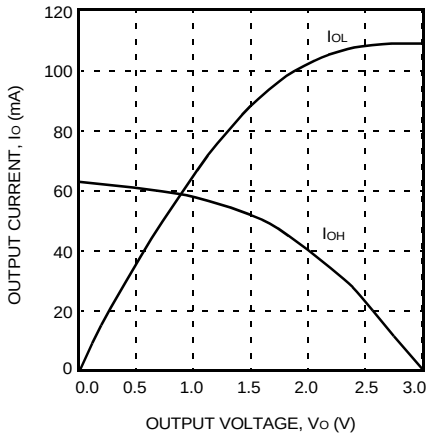
5-6868(F)

Figure 43. Slewlim (T_J = 125 °C, V_{DD} = 3.0 V)



5-6867(F)

Figure 41. Fast (T_J = 25 °C, V_{DD} = 3.3 V)



5-6868(F)

Figure 44. Fast (T_J = 125 °C, V_{DD} = 3.0 V)

Estimating Power Dissipation

This section will be included in a future release of this data sheet.

General FPGA power estimation parameters can be found in the *ORCA* Series 3 data sheet.

Pin Information

This section describes the pins and signals that perform FPGA-related functions. Any pins not described in Table 7 or here in Table 33 are user-programmable I/Os. During configuration, the user-programmable I/Os are 3-stated and pulled-up with an internal resistor. If any FPGA function pin is not used (or not bonded to package pin), it is also 3-stated and pulled-up after configuration.

Table 33. FPGA Common-Function Pin Descriptions

Symbol	I/O	Description
Dedicated Pins		
VDD	—	3.3 V power supply.
VDD2	—	2.5 V power supply.
GND	—	Ground supply.
$\overline{\text{RESET}}$	I	During configuration, $\overline{\text{RESET}}$ forces the restart of configuration and a pull-up is enabled. After configuration, RESET can be used as an FPGA logic direct input, which causes all PLC latches/FFs to be asynchronously set/reset.
CCLK	I	In the Master and asynchronous peripheral modes, CCLK is an output which strobes configuration data in. In the slave or synchronous peripheral mode, CCLK is input synchronous with the data on DIN or D[7:0]. In microprocessor and PCI modes, CCLK is used internally and output for daisy-chain operation.
DONE	I O	As an input, a low level on DONE delays FPGA start-up after configuration.* As an active-high, open-drain output, a high level on this signal indicates that configuration is complete. DONE is also used in the embedded PCI core start-up sequence. DONE has an optional pull-up resistor.
$\overline{\text{PRGM}}$	I	$\overline{\text{PRGM}}$ is an active-low input that forces the restart of configuration and resets the boundary-scan circuitry. This pin always has an active pull-up.
$\overline{\text{RD\_CFG}}$	I	This pin must be held high during device initialization until the $\overline{\text{INIT}}$ pin goes high. This pin always has an active pull-up. During configuration, $\overline{\text{RD\_CFG}}$ is an active-low input that activates the TS_ALL function and 3-states all of the I/O. After configuration, $\overline{\text{RD\_CFG}}$ can be selected (via a bit stream option) to activate the TS_ALL function as described above, or, if readback is enabled via a bit stream option, a high-to-low transition on $\overline{\text{RD\_CFG}}$ will initiate readback of the configuration data, including PFU output states, starting with frame address 0.
RD_DATA/TDO	O	RD_DATA/TDO is a dual-function pin. If used for readback, RD_DATA provides configuration data out. If used in boundary scan, TDO is test data out.
Special-Purpose Pins		
M0, M1, M2	I	During powerup and initialization, M0—M2 are used to select the configuration mode with their values latched on the rising edge of $\overline{\text{INIT}}$; see Table 29 for the configuration modes. During configuration, a pull-up is enabled.
	I/O	After configuration, M2 can be a user-programmable I/O.*
M3	I	During powerup and initialization, M3 is used to select the speed of the internal oscillator during configuration with their values latched on the rising edge of $\overline{\text{INIT}}$. When M3 is low, the oscillator frequency is 10 MHz. When M3 is high, the oscillator is 1.25 MHz. During configuration, a pull-up is enabled.
	I/O	After configuration, M2 can be a user-programmable I/O pin.*

* The *ORCA* Series 3 FPGA data sheet contains more information on how to control these signals during start-up. The timing of DONE release is controlled by one set of bit stream options, and the timing of the simultaneous release of all other configuration pins (and the activation of all user I/Os) is controlled by a second set of options.

Pin Information (continued)

Table 33. FPGA Common-Function Pin Descriptions (continued)

Symbol	I/O	Description
Special-Purpose Pins (continued)		
TDI, TCK, TMS	I	If boundary scan is used, these pins are test data in, test clock, and test mode select inputs. If boundary scan is not selected, all boundary-scan functions are inhibited once configuration is complete. Even if boundary scan is not used, either TCK or TMS must be held at logic 1 during configuration. Each pin has a pull-up enabled during configuration.
	I/O	After configuration, these pins are user-programmable I/O.*
RDY/RCLK/ MPI_ALE	O	During configuration in peripheral mode, RDY/RCLK indicates another byte can be written to the FPGA. If a read operation is done when the device is selected, the same status is also available on D7 in asynchronous peripheral mode.
	O	During the Master parallel configuration mode, RCLK is a read output signal to an external memory. This output is not normally used.
	I	In *i960* microprocessor mode, this pin acts as the address latch enable (ALE) input.
	I/O	After configuration, if the MPI is not used, this pin is a user-programmable I/O pin.*
HDC	O	High During Configuration is output high until configuration is complete. It is used as a control output indicating that configuration is not complete.
$\overline{\text{LDC}}$	O	Low During Configuration is output low until configuration is complete. It is used as a control output indicating that configuration is not complete.
$\overline{\text{INIT}}$	I/O	$\overline{\text{INIT}}$ is a bidirectional signal before and during configuration. During configuration, a pull-up is enabled, but an external pull-up resistor is recommended. As an active-low open-drain output, $\overline{\text{INIT}}$ is held low during power stabilization and internal clearing of memory. As an active-low input, $\overline{\text{INIT}}$ holds the FPGA in the wait-state before the start of configuration.
$\overline{\text{CS0}}$, CS1	I	$\overline{\text{CS0}}$ and CS1 are used in the asynchronous peripheral, slave parallel, and microprocessor configuration modes. The FPGA is selected when $\overline{\text{CS0}}$ is low and CS1 is high. During configuration, a pull-up is enabled.
	I/O	After configuration, these pins are user-programmable I/O pins.*
$\overline{\text{RD}}$/$\overline{\text{MPI\_STRB}}$	I	$\overline{\text{RD}}$ is used in the asynchronous peripheral configuration mode. A low on $\overline{\text{RD}}$ changes D7 into a status output. As a status indication, a high indicates ready, and a low indicates busy. $\overline{\text{WR}}$ and $\overline{\text{RD}}$ should not be used simultaneously. If they are, the write strobe overrides.
	I	This pin is also used as the microprocessor interface (MPI) data transfer strobe. For *PowerPC*, it is the transfer start (TS). For *i960*, it is the address/data strobe ($\overline{\text{ADS}}$).
	I/O	After configuration, if the MPI is not used, this pin is a user-programmable I/O pin.*
$\overline{\text{WR}}$	I	$\overline{\text{WR}}$ is used in the asynchronous peripheral configuration mode. When the FPGA is selected, a low on the write strobe, $\overline{\text{WR}}$, loads the data on D[7:0] inputs into an internal data buffer. $\overline{\text{WR}}$ and $\overline{\text{RD}}$ should not be used simultaneously. If they are, the write strobe overrides.
	I/O	After configuration, this pin is a user-programmable I/O pin.*

* The *ORCA* Series 3 FPGA data sheet contains more information on how to control these signals during start-up. The timing of DONE release is controlled by one set of bit stream options, and the timing of the simultaneous release of all other configuration pins (and the activation of all user I/Os) is controlled by a second set of options.

Pin Information (continued)

Table 33. FPGA Common-Function Pin Descriptions (continued)

Symbol	I/O	Description
Special-Purpose Pins (continued)		
$\overline{\text{MPI\_IRQ}}$	O	MPI active-low interrupt request output.
	I/O	If the MPI is not in use, this is a user-programmable I/O.
$\overline{\text{MPI\_BI}}$	O	*PowerPC* mode MPI burst inhibit output.
	I/O	If the MPI is not in use, this is a user-programmable I/O.
$\overline{\text{MPI\_ACK}}$	O	In *PowerPC* mode MPI operation, this is the active-high transfer acknowledge ($\overline{\text{TA}}$) output. For *i960* MPI operation, it is the active-low ready/record ($\overline{\text{RDYRCV}}$) output. If the MPI is not in use, this is a user-programmable I/O.
MPI_RW	I	In *PowerPC* mode MPI operation, this is the active-low write/ active-high read control signals. For *i960* operation, it is the active-high write/active-low read control signal.
	I/O	If the MPI is not in use, this is a user-programmable I/O.
MPI_CLK	I	This is the clock used for the synchronous MPI interface. For *PowerPC*, it is the CLKOUT signal. For *i960*, it is the system clock that is chosen for the *i960* external bus interface.
	I/O	If the MPI is not in use, this is a user-programmable I/O.
A[4:0]	I	For *PowerPC* operation, these are the *PowerPC* address inputs. The address bit mapping (in *PowerPC*/FPGA notation) is A[31]/A[0], A[30]/A[1], A[29]/A[2], A[28]/A[3], A[27]/A[4]. Note that A[27]/A[4] is the MSB of the address. The A[4:2] inputs are not used in *i960* MPI mode.
	I/O	If the MPI is not in use, this is a user-programmable I/O.
A[1:0]/$\overline{\text{MPI\_BE[1:0]}}$	I	For *i960* operation, $\overline{\text{MPI\_BE[1:0]}}$ provide the *i960* byte enable signals, $\overline{\text{BE[1:0]}}$, that are used as address bits A[1:0] in *i960* byte-wide operation.
D[7:0]	I	During Master parallel, peripheral, and slave parallel configuration modes, D[7:0] receive configuration data, and each pin has a pull-up enabled. During serial configuration modes, D0 is the DIN input. D[7:0] are also the data pins for *PowerPC* microprocessor mode and the address/data pins for *i960* microprocessor mode.
	I/O	After configuration, the pins are user-programmable I/O pins.*
DIN	I	During slave serial or Master serial configuration modes, DIN accepts serial configuration data synchronous with CCLK. During parallel configuration modes, DIN is the D0 input. During configuration, a pull-up is enabled.
	I/O	After configuration, this pin is a user-programmable I/O pin.*
DOUT	O	During configuration, DOUT is the serial data output that can drive the DIN of daisy-chained slave LCA devices. Data out on DOUT changes on the falling edge of CCLK.
	I/O	After configuration, DOUT is a user-programmable I/O pin.*

* The *ORCA* Series 3 FPGA data sheet contains more information on how to control these signals during start-up. The timing of DONE release is controlled by one set of bit stream options, and the timing of the simultaneous release of all other configuration pins (and the activation of all user I/Os) is controlled by a second set of options.

Pin Information (continued)

Package Compatibility

Table 34 lists the number of user I/Os available for the *ORCA* OR3LP26B FPSC for each available package. Each package has six dedicated configuration pins and six dedicated special-purpose pins.

Table 35 provides the package pin and pin function for the *ORCA* OR3LP26B FPSC in each available package. The bond pad name is identified in the PIC nomenclature used in the *ORCA* Foundry design editor.

When the number of FPGA bond pads exceeds the number of package pins, bond pads are unused. When the number of package pins exceeds the number of bond pads, package pins are left unconnected (no connects). When a package pin is to be left as a no connect for a specific die, it is indicated as a note in the device pad column for the FPGA.

Table 34. *ORCA* OR3LP26B I/Os Summary

		352-Pin PBGA	680-Pin PBGAM
User I/Os*		162	242
V_{DD}		16	56
$V_{DD}2$		11	76
V_{SS}		68	100
Configuration/Special-Purpose Pins†		12	12
PCI Interface Pins		93	93
Unused Pins	PCI Core Section	26	90
	FPGA Section	0	11

* User I/O count includes three ExpressCLK inputs.

† Configuration pins: CCLK, DONE, $\overline{\text{RESET}}$, $\overline{\text{PRGM}}$, $\overline{\text{RD\_CFG}}$;
Special-purpose pins: RD_DATA/TDO, HDC, $\overline{\text{LDC}}$, $\overline{\text{INIT}}$, M0, M1, M2

Pin Information (continued)

Table 35. Pinout Information

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
VSS	VSS	VSS[1]	VSS[1]
VDD	VDD	VDD[1]	VDD[1]
VSS	VSS	VSS[1]	VSS[1]
PL1D	I/O	B1	D1
PL1C	I/O	C2	F4
PL1B	I/O	C1	F3
PL1A	I/O	D2	F2
VDD	VDD	VDD[1]	VDD[1]
PL2D	I/O-A0-$\overline{\text{MPI\_BE0}}$	D3	F1
PL2C	I/O	—	G5
PL2B	I/O	—	G4
PL2A	I/O	D1	G2
PL3D	I/O	E2	G1
PL3C	I/O	—	H5
PL3B	I/O	E4	H4
PL3A	I/O	E3	H2
VSS	VSS	VSS[1]	VSS[1]
PL4D	I/O	—	—
VDD2	VDD2	E1	VDD2
PL4C	I/O	F2	H1
PL4B	I/O	G4	J5
PL4A	I/O	—	J4
PL5D	I/O	F3	J3
PL5C	I/O	—	J2
PL5B	I/O	—	J1
PL5A	I/O	—	K5
VDD	VDD	VDD[1]	VDD[1]
PL6D	I/O	F1	K4
PL6C	I/O	G2	K3
PL6B	I/O	G1	K2
PL6A	I/O	—	K1
PL7D	I/O-A1-$\overline{\text{MPI\_BE1}}$	G3	L5
PL7C	I/O	—	L4
PL7B	I/O	—	L2
PL7A	I/O	—	L1
VSS	VSS	VSS[1]	VSS[1]
PL8D	I/O	H2	M5
PL8C	I/O	J4	M4
PL8B	I/O	H1	M2
PL8A	I/O-A2	H3	M1

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PL9D	I/O	J2	N5
PL9C	I/O	J1	N4
PL9B	I/O	K2	N3
PL9A	I/O-A3	J3	N2
V_{DD}	V_{DD}	V_{DD}[1]	V_{DD}[1]
PL10D	I/O	K1	—
V_{DD}2	V_{DD}2	—	V_{DD}2
PL10C	I/O	—	N1
PL10B	I/O	—	P5
PL10A	I/O	K4	P4
PL11D	I/O	L2	P3
PL11C	I/O	—	P2
PL11B	I/O	—	P1
PL11A	I/O-A4	K3	R5
V_{SS}	V_{SS}	V_{SS}[1]	V_{SS}[1]
PL12D	I/O	L1	R4
PL12C	I/O	—	R2
PL12B	I/O	—	R1
PL12A	I/O	M2	—
V_{DD}2	V_{DD}2	—	V_{DD}2
PL13D	I/O	M1	T5
PL13C	I/O	—	T4
PL13B	I/O	—	T2
PL13A	I/O	L3	T1
V_{SS}	V_{SS}	V_{SS}[1]	V_{SS}[1]
PECKL	I-ECKL	N2	U5
PL14D	—	—	—
PL14C	I/O	M4	U3
PL14B	I/O	N1	U2
PL14A	I/O-$\overline{\text{MPI\_CLK}}$	M3	U1
V_{DD}	V_{DD}	V_{DD}[1]	V_{DD}[1]
PL15D	I/O	P2	V1
PL15C	——	—	—
V_{DD}2	V_{DD}2	P4	V_{DD}2
PL15B	I/O	P1	V2
PL15A	I/O-$\overline{\text{MPI\_RW}}$	N3	V3
V_{SS}	V_{SS}	V_{SS}[1]	V_{SS}[1]
PL16D	I/O-$\overline{\text{MPI\_ACK}}$	R2	V4
PL16C	I/O	—	V5
PL16B	I/O	—	W1

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PL16A	I/O	P3	W2
PL17D	I/O	R1	W4
PL17C	I/O	—	W5
PL17B	I/O	—	Y1
PL17A	I/O-$\overline{\text{MPI\_BI}}$	T2	Y2
VSS	VSS	VSS[1]	VSS[1]
PL18D	I/O	R3	Y4
PL18C	I/O	—	Y5
PL18B	I/O	—	AA1
PL18A	I/O-SECKLL	T1	AA2
PL19D	No Connect[2]	R4	AA3
PL19C	No Connect[2]	—	AA4
PL19B	No Connect[2]	—	AA5
PL19A	MPI_IRQ	U2	AB1
VDD	VDD	VDD[1]	VDD[1]
PL20D	No Connect[2]	T3	AB2
PL20C	No Connect[2]	U1	AB3
PL20B	No Connect[2]	U4	—
VDD2	VDD2	—	VDD2
PL20A	No Connect[2]	V2	AB4
PL21D	VDD	U3	AB5
PL21C	VSS	V1	AC1
PL21B	VSS	W2	AC2
PL21A	INTA#	W1	AC4
VSS	VSS	VSS[1]	VSS[1]
PL22D	RST#	V3	AC5
PL22C	GNT#	Y2	AD1
PL22B	No Connect[2]	—	AD2
PL22A	No Connect[2]	—	AD4
PL23D	REQ#	W4	AD5
PL23C	No Connect[2]	—	AE1
PL23B	No Connect[2]	—	AE2
PL23A	No Connect[2]	—	AE3
VDD	VDD	VDD[1]	VDD[1]
PL24D	AD31	Y1	AE4
PL24C	No Connect[2]	—	AE5
PL24B	No Connect[2]	—	AF1
PL24A	AD30	W3	AF2
PL25D	No Connect[2]	—	AF3
PL25C	AD29	AA2	AF4

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PL25B	AD28	Y4	AF5
PL25A	AD27	AA1	AG1
VSS	VSS	VSS[1]	VSS[1]
PL26D	—	—	—
VDD2	VDD2	Y3	VDD2
PL26C	AD26	AB2	AG2
PL26B	No Connect[2]	—	AG4
PL26A	AD25	AB1	AG5
PL27D	AD24	AA3	AH1
PL27C	C/BE3#	AC2	AH2
PL27B	No Connect[2]	—	AH4
PL27A	IDSEL	AB4	AH5
VDD	VDD	VDD[1]	VDD[1]
PL28D	AD23	AC1	AJ3
PL28C	No Connect[2]	AB3	AJ4
PL28B	No Connect[2]	AD2	AK1
PL28A	VIO	AC3	AK2
VSS	VSS	VSS[1]	VSS[1]
PCCLK	CCLK	AD1	AL1
VDD	VDD	VDD[1]	VDD[1]
VSS	VSS	VSS[1]	VSS[1]
VDD	VDD	VDD[1]	VDD[1]
VSS	VSS	VSS[1]	VSS[1]
PB1A	AD22	AF2	AP4
PB1B	No Connect[2]	AE3	AN5
PB1C	AD21	AF3	AM6
PB1D	AD20	AE4	AN6
VDD	VDD	VDD[1]	VDD[1]
PB2A	AD19	AD4	AP6
PB2B	No Connect[2]	—	AK7
PB2C	No Connect[2]	—	AL7
PB2D	—	—	—
VDD2	VDD2	AF4	VDD2
PB3A	AD18	AE5	AN7
PB3B	No Connect[2]	—	AP7
PB3C	AD17	AC5	AK8
PB3D	AD16	AD5	AL8
VSS	VSS	VSS[1]	VSS[1]
PB4A	C/BE2#	AF5	AN8
PB4B	PERR#	AE6	AP8
PB4C	SERR#	AC7	AK9

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PB4D	PAR	AD6	AL9
PB5A	C/BE1#	AF6	AM9
PB5B	AD15	AE7	AN9
PB5C	AD14	AF7	AP9
PB5D	AD13	AD7	AK10
VSS	VSS	VSS[1]	VSS[1]
PB6A	AD12	AE8	AL10
PB6B	No Connect[2]	—	AM10
PB6C	No Connect[2]	—	AN10
PB6D	AD11	AC9	AP10
PB7A	AD10	AF8	AK11
PB7B	No Connect[2]	—	AL11
PB7C	No Connect[2]	—	AN11
PB7D	AD9	AD8	AP11
VSS	VSS	VSS[1]	VSS[1]
PB8A	AD8	AE9	AK12
PB8B	No Connect[2]	—	AL12
PB8C	No Connect[2]	—	AN12
PB8D	C/BE0#	AF9	AP12
PB9A	AD7	AE10	AK13
PB9B	No Connect[2]	—	AL13
PB9C	No Connect[2]	—	AM13
PB9D	AD6	AD9	AN13
VDD	VDD	VDD[1]	VDD[1]
VDD2	VDD2	—	VDD2
PB10A	No Connect[2]	AF10	—
PB10B	No Connect[2]	—	AP13
PB10C	No Connect[2]	—	AK14
PB10D	AD5	AC10	AL14
PB11A	AD4	AE11	AM14
PB11B	No Connect[2]	—	AN14
PB11C	No Connect[2]	—	AP14
PB11D	AD3	AD10	AK15
VDD	VDD	VDD[1]	VDD[1]
PB12A	AD2	AF11	AL15
PB12B	No Connect[2]	—	AN15
PB12C	No Connect[2]	—	AP15
PB12D	AD1	AE12	AK16
PB13A	AD0	AF12	AL16
PB13B	No Connect[2]	—	AN16
PB13C	No Connect[2]	—	AP16

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PB13D	FRAME#	AD11	AK17
VSS	VSS	VSS[1]	VSS[1]
PB14A	No Connect[2]	AE13	—
VDD2	VDD2	—	VDD2
PB14B	IRDY#	AC12	AM17
PB14C	TRDY#	AF13	AP17
PB14D	DEVSEL#	AD12	AP18
VSS	VSS	VSS[1]	VSS[1]
PECKB	CLK	AE14	AN18
PB15A	—	—	—
PB15B	STOP#	AC14	AM18
PB15C	ACK64#	AF14	AL18
PB15D	REQ64#	AD13	AK18
VSS	VSS	VSS[1]	VSS[1]
PB16A	—	—	—
VDD2	VDD2	AE15	VDD2
PB16B	No Connect[2]	—	AP19
PB16C	No Connect[2]	—	AN19
PB16D	C/BE7#	AD14	AL19
PB17A	C/BE6#	AF15	AK19
PB17B	No Connect[2]	—	AP20
PB17C	No Connect[2]	—	AN20
PB17D	C/BE5#	AE16	AL20
VDD	VDD	VDD[1]	VDD[1]
PB18A	HDC	AD15	AK20
PB18B	No Connect[2]	—	AP21
PB18C	No Connect[2]	—	AN21
PB18D	C/BE4#	AF16	AM21
PB19A	AD63	AC15	AL21
PB19B	No Connect[2]	—	AK21
PB19C	No Connect[2]	—	AP22
PB19D	No Connect[2]	AE17	—
VDD2	VDD2	—	VDD2
VDD	VDD	VDD[1]	VDD[1]
PB20A	LDC	AD16	AN22
PB20B	No Connect[2]	—	AM22
PB20C	No Connect[2]	—	AL22
PB20D	AD62	AF17	AK22
PB21A	AD61	AC17	AP23
PB21B	No Connect[2]	—	AN23
PB21C	No Connect[2]	—	AL23

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PB21D	AD60	AE18	AK23
VSS	VSS	VSS[1]	VSS[1]
PB22A	AD59	AD17	AP24
PB22B	No Connect[2]	—	AN24
PB22C	No Connect[2]	—	AL24
PB22D	No Connect[2]	—	AK24
PB23A	AD58	AF18	AP25
PB23B	No Connect[2]	—	AN25
PB23C	AD57	AE19	AM25
PB23D	AD56	AF19	AL25
VSS	VSS	VSS[1]	VSS[1]
PB24A	INIT	AD18	AK25
PB24B	AD55	AE20	AP26
PB24C	AD54	AC19	AN26
PB24D	AD53	AF20	AM26
PB25A	—	—	—
VDD2	VDD2	AD19	VDD2
PB25B	AD52	AE21	AL26
PB25C	AD51	AC20	AK26
PB25D	AD50	AF21	AP27
VSS	VSS	VSS[1]	VSS[1]
PB26A	AD49	AD20	AN27
PB26B	AD48	AE22	AL27
PB26C	No Connect[2]	—	AK27
PB26D	AD47	AF22	AP28
PB27A	AD46	AD21	AN28
PB27B	No Connect[2]	AE23	AL28
PB27C	No Connect[2]	—	AK28
PB27D	AD45	AC22	AP29
VDD	VDD	VDD[1]	VDD[1]
PB28A	AD44	AF23	AN29
PB28B	AD43	AD22	AM29
PB28C	No Connect[2]	AE24	AP30
PB28D	PAR64	AD23	AN30
VSS	VSS	VSS[1]	VSS[1]
PDONE	DONE	AF24	AP31
VDD	VDD	VDD[1]	VDD[1]
VSS	VSS	VSS[1]	VSS[1]
PRESETN	RESET	AE26	AL34
PPRGMN	PRGM	AD25	AK33
PR28A	M0	AD26	AK34

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PR28B	No Connect[2]	—	AJ31
PR28C	AD42	AC25	AJ32
PR28D	AD41	AC24	AJ33
V_{DD}	V_{DD}	V_{DD}[1]	V_{DD}[1]
V_{DD}2	V_{DD}2	—	V_{DD}2
PR27A	No Connect[2]	AC26	—
PR27B	No Connect[2]	—	AJ34
PR27C	No Connect[2]	—	AH30
PR27D	AD40	AB25	AH31
PR26A	AD39	AB23	AH33
PR26B	AD38	AB24	AH34
PR26C	No Connect[2]	—	AG30
PR26D	AD37	AB26	AG31
V_{SS}	V_{SS}	V_{SS}[1]	V_{SS}[1]
PR25A	AD36	AA25	AG33
PR25B	AD35	Y23	AG34
PR25C	AD34	AA24	AF30
PR25D	No Connect[2]	—	AF31
PR24A	AD33	AA26	AF32
PR24B	No Connect[2]	—	AF33
PR24C	No Connect[2]	—	AF34
PR24D	No Connect[2]	—	AE30
V_{DD}	V_{DD}	V_{DD}[1]	V_{DD}[1]
PR23A	AD32	Y25	AE31
PR23B	ENUM#	Y26	AE32
PR23C	No Connect[2]	—	AE33
PR23D	LED#	Y24	AE34
PR22A	No Connect[2]	—	AD30
PR22B	No Connect[2]	—	AD31
PR22C	No Connect[2]	—	AD33
PR22D	M1	W25	AD34
V_{SS}	V_{SS}	V_{SS}[1]	V_{SS}[1]
PR21A	EJECTSW	V23	AC30
PR21B	No Connect[2]	W26	AC31
PR21C	No Connect[2]	W24	AC33
PR21D	No Connect[2]	—	—
V_{DD}2	V_{DD}2	V25	V_{DD}2
PR20A	No Connect[2]	V26	AC34
PR20B	No Connect[2]	U25	AB30
PR20C	No Connect[2]	V24	AB31
PR20D	No Connect[2]	U26	AB32

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
VDD	VDD	VDD[1]	VDD[1]
PR19A	M2	U23	AB33
PR19B	No Connect[2]	—	AB34
PR19C	No Connect[2]	—	AA30
PR19D	No Connect[2]	T25	AA31
PR18A	I/O	U24	AA32
PR18B	I/O	—	AA33
PR18C	I/O	—	AA34
PR18D	I/O	T26	Y30
VSS	VSS	VSS[1]	VSS[1]
PR17A	I/O-M3	R25	Y31
PR17B	I/O	—	Y33
PR17C	I/O	—	Y34
PR17D	I/O	R26	W30
PR16A	I/O	T24	W31
PR16B	I/O	—	W33
PR16C	I/O	—	W34
PR16D	I/O	P25	—
VDD2	VDD2	—	VDD2
VSS	VSS	VSS[1]	VSS[1]
PR15A	I/O	R23	V30
PR15B	I/O	P26	V32
PR15C	I/O	R24	V33
PR15D	I/O	N25	V34
VDD	VDD	VDD[1]	VDD[1]
PECKR	I-ECKR	N23	U34
PR14A	—	—	—
PR14B	I/O	N26	U33
PR14C	I/O	P24	U32
PR14D	I/O	M25	U31
VSS	VSS	VSS[1]	VSS[1]
PR13A	—	—	—
VDD2	VDD2	N24	VDD2
PR13B	I/O	—	U30
PR13C	I/O	—	T34
PR13D	I/O	M26	T33
PR12A	I/O	L25	T31
PR12B	I/O	—	T30
PR12C	I/O	—	R34
PR12D	I/O	M24	R33
VSS	VSS	VSS[1]	VSS[1]

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PR11A	I/O-CS1	L26	R31
PR11B	I/O	—	R30
PR11C	I/O	—	P34
PR11D	I/O	M23	P33
PR10A	I/O	K25	P32
PR10B	I/O	—	P31
PR10C	I/O	—	P30
PR10D	I/O	L24	—
VDD2	VDD2	—	VDD2
VDD	VDD	VDD[1]	VDD[1]
PR9A	I/O-CS0	K26	N34
PR9B	I/O	K23	N33
PR9C	I/O	J25	N32
PR9D	I/O	K24	N31
PR8A	I/O	J26	N30
PR8B	I/O	H25	M34
PR8C	I/O	H26	M33
PR8D	I/O	J24	M31
VSS	VSS	VSS[1]	VSS[1]
PR7A	I/O-RD	G25	M30
PR7B	I/O	—	L34
PR7C	I/O	—	L33
PR7D	I/O	—	L31
PR6A	I/O	H23	L30
PR6B	I/O	—	K34
PR6C	I/O	G26	K33
PR6D	I/O	—	K32
VDD	VDD	VDD[1]	VDD[1]
PR5A	I/O	H24	K31
PR5B	I/O	—	K30
PR5C	I/O	—	J34
PR5D	I/O	—	J33
PR4A	—	—	—
VDD2	VDD2	F25	VDD2
PR4B	I/O	G23	J32
PR4C	I/O	F26	J31
PR4D	I/O	G24	J30
VSS	VSS	VSS[1]	VSS[1]
PR3A	I/O-WR	E25	H34
PR3B	I/O	E26	H33
PR3C	I/O	—	H31

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PR3D	I/O	F24	H30
PR2A	I/O	D25	G34
PR2B	I/O	—	G33
PR2C	I/O	—	G31
PR2D	I/O	E23	G30
V_{DD}	V_{DD}	V_{DD}[1]	V_{DD}[1]
PR1A	I/O	D26	F34
PR1B	I/O	E24	F32
PR1C	I/O	C25	F31
PR1D	I/O	D24	E33
V_{SS}	V_{SS}	V_{SS}[1]	V_{SS}[1]
PRD_CFGN	RD_CFGN	C26	D34
V_{DD}	V_{DD}	V_{DD}[1]	V_{DD}[1]
V_{SS}	V_{SS}	V_{SS}[1]	V_{SS}[1]
V_{DD}	V_{DD}	V_{DD}[1]	V_{DD}[1]
V_{SS}	V_{SS}	V_{SS}[1]	V_{SS}[1]
PT28D	I/O-SECKUR	A25	A31
PT28C	I/O	B24	A30
PT28B	I/O	A24	C29
PT28A	I/O	B23	B29
V_{DD}	V_{DD}	V_{DD}[1]	V_{DD}[1]
PT27D	I/O	C23	A29
PT27C	I/O	—	E28
PT27B	I/O	—	D28
PT27A	I/O-RDY/RCLK	A23	B28
PT26D	I/O	B22	A28
PT26C	I/O	D22	E27
PT26B	I/O	—	D27
PT26A	I/O	C22	B27
V_{SS}	V_{SS}	V_{SS}[1]	V_{SS}[1]
PT25D	I/O	A22	A27
PT25C	I/O	B21	E26
PT25B	I/O	D20	D26
PT25A	I/O	C21	C26
PT24D	I/O-D7	A21	B26
PT24C	I/O	B20	A26
PT24B	I/O	A20	E25
PT24A	I/O	C20	D25
V_{SS}	V_{SS}	V_{SS}[1]	V_{SS}[1]
PT23D	—	—	—
V_{DD}2	V_{DD}2	B19	V_{DD}2

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PT23C	I/O	D18	C25
PT23B	I/O	A19	B25
PT23A	I/O	—	A25
PT22D	I/O	C19	E24
PT22C	I/O	—	D24
PT22B	I/O	—	B24
PT22A	I/O	—	A24
VSS	VSS	VSS[1]	VSS[1]
PT21D	I/O	B18	E23
PT21C	I/O	—	D23
PT21B	I/O	—	B23
PT21A	I/O	A18	A23
PT20D	I/O-D6	B17	E22
PT20C	I/O	—	D22
PT20B	I/O	—	C22
PT20A	I/O	C18	B22
VDD	VDD	VDD[1]	VDD[1]
PT19D	I/O	A17	A22
PT19C	I/O	—	E21
PT19B	I/O	—	D21
PT19A	I/O	D17	C21
PT18D	I/O	B16	—
VDD2	VDD2	—	VDD2
PT18C	I/O	—	B21
PT18B	I/O	—	A21
PT18A	I/O-D5	C17	E20
VDD	VDD	VDD[1]	VDD[1]
PT17D	I/O	A16	D20
PT17C	I/O	—	B20
PT17B	I/O	—	A20
PT17A	I/O	B15	E19
PT16D	I/O	A15	D19
PT16C	I/O	—	B19
PT16B	I/O	—	A19
PT16A	I/O-D4	C16	E18
VSS	VSS	VSS[1]	VSS[1]
PECKT	I-ECKT	B14	D18
PT15D	—	—	—
PT15C	I/O	D15	—
VDD2	VDD2	—	VDD2
PT15B	I/O	A14	C18

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PT15A	I/O-D3	C15	A18
VSS	VSS	VSS[1]	VSS[1]
PT14D	I/O	B13	A17
PT14C	I/O	D13	B17
PT14B	—	—	—
VDD2	VDD2	A13	VDD2
PT14A	I/O-D2	C14	C17
VSS	VSS	VSS[1]	VSS[1]
PT13D	I/O-D1	B12	D17
PT13C	I/O	—	E17
PT13B	I/O	—	A16
PT13A	I/O	C13	B16
PT12D	I/O	A12	D16
PT12C	I/O	—	E16
PT12B	I/O	—	A15
PT12A	I/O-D0-DIN	B11	B15
VDD	VDD	VDD[1]	VDD[1]
PT11D	I/O	C12	D15
PT11C	I/O	—	E15
PT11B	I/O	—	A14
PT11A	I/O	A11	B14
PT10D	I/O	D12	C14
PT10C	I/O	—	D14
PT10B	I/O	—	E14
PT10A	I/O-DOUT	B10	A13
VDD	VDD	VDD[1]	VDD[1]
PT9D	I/O	C11	—
VDD2	VDD2	—	VDD2
PT9C	I/O	—	B13
PT9B	I/O	—	C13
PT9A	I/O	A10	D13
PT8D	I/O	D10	E13
PT8C	I/O	—	A12
PT8B	I/O	—	B12
PT8A	I/O	B9	D12
VSS	VSS	VSS[1]	VSS[1]
PT7D	I/O	C10	E12
PT7C	I/O	—	A11
PT7B	I/O	—	B11
PT7A	I/O	A9	D11
PT6D	I/O	B8	E11

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PT6C	I/O	—	A10
PT6B	I/O	—	B10
PT6A	I/O-TDI	A8	C10
VSS	VSS	VSS[1]	VSS[1]
PT5D	I/O	C9	D10
PT5C	I/O	B7	E10
PT5B	I/O	D8	A9
PT5A	—	—	—
VDD2	VDD2	A7	VDD2
PT4D	I/O	C8	B9
PT4C	I/O	B6	C9
PT4B	I/O	D7	D9
PT4A	I/O-TMS	A6	E9
VSS	VSS	VSS[1]	VSS[1]
PT3D	I/O	C7	A8
PT3C	I/O	—	B8
PT3B	I/O	—	D8
PT3A	I/O	B5	E8
PT2D	I/O	A5	A7
PT2C	I/O	C6	B7
PT2B	I/O	B4	D7
PT2A	I/O	D5	E7
VDD	VDD	VDD[1]	VDD[1]
PT1D	I/O	A4	A6
PT1C	I/O	C5	C6
PT1B	I/O	B3	D6
PT1A	I/O-TCK	C4	B5
VSS	VSS	VSS[1]	VSS[1]
PRD_DATA	RD_DATA/TDO	A3	A4
VDD	VDD	VDD[1]	VDD[1]
VDD2	VDD2	—	VDD2
Note 3	VDD	—	—
Note 3	VDD	—	A3
Note 3	VDD	—	A32
Note 3	VDD	—	—
Note 3	VDD	—	B3
Note 3	VDD	—	B4
Note 3	VDD	—	B31
Note 3	VDD	—	B32
Note 3	VDD	—	C1
Note 3	VDD	—	C2

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
Note 3	VDD	—	C4
Note 3	VDD	—	C7
Note 3	VDD	—	C11
Note 3	VDD	—	C15
Note 3	VDD	—	C20
Note 3	VDD	—	C24
Note 3	VDD	—	C28
Note 3	VDD	—	C31
Note 3	VDD	—	C33
Note 3	VDD	—	C34
Note 3	VDD	—	D2
Note 3	VDD	—	D3
Note 3	VDD	—	—
Note 3	VDD	D6	—
Note 3	VDD	D11	—
Note 3	VDD	D16	—
Note 3	VDD	D21	—
Note 3	VDD	—	—
Note 3	VDD	—	D32
Note 3	VDD	—	D33
Note 3	VDD	—	G3
Note 3	VDD	—	G32
Note 3	VDD	F4	L3
Note 3	VDD	F23	L32
Note 3	VDD	L4	R3
Note 3	VDD	L23	R32
Note 3	VDD	T4	Y3
Note 3	VDD	T23	Y32
Note 3	VDD	AA4	AD3
Note 3	VDD	AA23	AD32
Note 3	VDD	—	AH3
Note 3	VDD	—	AH32
Note 3	VDD	—	AL2
Note 3	VDD	—	AL3
Note 3	VDD	—	—
Note 3	VDD	AC6	—
Note 3	VDD	AC11	—
Note 3	VDD	AC16	—
Note 3	VDD	AC21	—
Note 3	VDD	—	—
Note 3	VDD	—	AL32

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
Note 3	V_{DD}	—	AL33
Note 3	V_{DD}	—	AM1
Note 3	V_{DD}	—	AM2
Note 3	V_{DD}	—	AM4
Note 3	V_{DD}	—	AM7
Note 3	V_{DD}	—	AM11
Note 3	V_{DD}	—	AM15
Note 3	V_{DD}	—	AM20
Note 3	V_{DD}	—	AM24
Note 3	V_{DD}	—	AM28
Note 3	V_{DD}	—	AM31
Note 3	V_{DD}	—	AM33
Note 3	V_{DD}	—	AM34
Note 3	V_{DD}	—	AN3
Note 3	V_{DD}	—	AN4
Note 3	V_{DD}	—	AN31
Note 3	V_{DD}	—	AN32
Note 3	V_{DD}	—	—
Note 3	V_{DD}	—	AP3
Note 3	V_{DD}	—	AP32
Note 3	V_{DD}	—	—
Note 3	$V_{DD}2$	—	C5
Note 3	$V_{DD}2$	—	C30
Note 3	$V_{DD}2$	—	D5
Note 3	$V_{DD}2$	—	D30
Note 3	$V_{DD}2$	—	E3
Note 3	$V_{DD}2$	—	E4
Note 3	$V_{DD}2$	—	E5
Note 3	$V_{DD}2$	—	E6
Note 3	$V_{DD}2$	—	E29
Note 3	$V_{DD}2$	—	E30
Note 3	$V_{DD}2$	—	E31
Note 3	$V_{DD}2$	—	E32
Note 3	$V_{DD}2$	—	F5
Note 3	$V_{DD}2$	—	F30
Note 3	$V_{DD}2$	—	AJ5
Note 3	$V_{DD}2$	—	AJ30
Note 3	$V_{DD}2$	—	AK3
Note 3	$V_{DD}2$	—	AK4
Note 3	$V_{DD}2$	—	AK5
Note 3	$V_{DD}2$	—	AK6

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
Note 3	VDD2	—	AK29
Note 3	VDD2	—	AK30
Note 3	VDD2	—	AK31
Note 3	VDD2	—	AK32
Note 3	VDD2	—	AL5
Note 3	VDD2	—	AL30
Note 3	VDD2	—	AM5
Note 3	VDD2	—	AM30
Note 3	VSS	A1	A1
Note 3	VSS	A2	A2
Note 3	VSS	—	—
Note 3	VSS	—	—
Note 3	VSS	—	—
Note 3	VSS	—	—
Note 3	VSS	—	—
Note 3	VSS	—	—
Note 3	VSS	—	A33
Note 3	VSS	A26	A34
Note 3	VSS	—	B1
Note 3	VSS	B2	B2
Note 3	VSS	B25	B33
Note 3	VSS	B26	B34
Note 3	VSS	—	—
Note 3	VSS	C3	C3
Note 3	VSS	—	C8
Note 3	VSS	—	C12
Note 3	VSS	—	C16
Note 3	VSS	—	C19
Note 3	VSS	—	C23
Note 3	VSS	—	C27
Note 3	VSS	C24	C32
Note 3	VSS	—	—
Note 3	VSS	D4	D4
Note 3	VSS	D9	—
Note 3	VSS	D14	—
Note 3	VSS	D19	—
Note 3	VSS	D23	D31
Note 3	VSS	—	H3
Note 3	VSS	—	H32
Note 3	VSS	H4	M3
Note 3	VSS	J23	M32

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
Note 3	Vss	—	T3
Note 3	Vss	—	T32
Note 3	Vss	N4	—
Note 3	Vss	P23	—
Note 3	Vss	—	W3
Note 3	Vss	—	W32
Note 3	Vss	V4	AC3
Note 3	Vss	W23	AC32
Note 3	Vss	—	AG3
Note 3	Vss	—	AG32
Note 3	Vss	AC4	AL4
Note 3	Vss	AC8	—
Note 3	Vss	AC13	—
Note 3	Vss	AC18	—
Note 3	Vss	AC23	AL31
Note 3	Vss	—	—
Note 3	Vss	AD3	AM3
Note 3	Vss	—	AM8
Note 3	Vss	—	AM12
Note 3	Vss	—	AM16
Note 3	Vss	—	AM19
Note 3	Vss	—	AM23
Note 3	Vss	—	AM27
Note 3	Vss	AD24	AM32
Note 3	Vss		—
Note 3	Vss	AE1	AN1
Note 3	Vss	AE2	AN2
Note 3	Vss	—	—
Note 3	Vss	—	—
Note 3	Vss	AE25	AN33
Note 3	Vss	—	AN34
Note 3	Vss	AF1	AP1
Note 3	Vss	—	AP2
Note 3	Vss	—	—
Note 3	Vss	—	—
Note 3	Vss	—	—
Note 3	Vss	—	—
Note 3	Vss	—	—
Note 3	Vss	—	—
Note 3	Vss	—	—
Note 3	Vss	AF25	AP33

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
Note 3	VSS	AF26	AP34
Note 3	VSS	L11	N13
Note 3	VSS	L12	N14
Note 3	VSS	L13	N15
Note 3	VDD2	—	N16
Note 3	VDD2	—	N17
Note 3	VDD2	—	N18
Note 3	VDD2	—	N19
Note 3	VSS	L14	N20
Note 3	VSS	L15	N21
Note 3	VSS	L16	N22
Note 3	VSS	M11	P13
Note 3	VSS	M12	P14
Note 3	VSS	M13	P15
Note 3	VDD2	—	P16
Note 3	VDD2	—	P17
Note 3	VDD2	—	P18
Note 3	VDD2	—	P19
Note 3	VSS	M14	P20
Note 3	VSS	M15	P21
Note 3	VSS	M16	P22
Note 3	VSS	N11	R13
Note 3	VSS	N12	R14
Note 3	VSS	N13	R15
Note 3	VDD2	—	R16
Note 3	VDD2	—	R17
Note 3	VDD2	—	R18
Note 3	VDD2	—	R19
Note 3	VSS	N14	R20
Note 3	VSS	N15	R21
Note 3	VSS	N16	R22
Note 3	VDD2	—	T13
Note 3	VDD2	—	T14
Note 3	VDD2	—	T15
Note 3	VSS		T16
Note 3	VSS		T17
Note 3	VSS		T18
Note 3	VSS		T19
Note 3	VDD2	—	T20
Note 3	VDD2	—	T21
Note 3	VDD2	—	T22

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
Note 3	$V_{DD}2$	—	U13
Note 3	$V_{DD}2$	—	U14
Note 3	$V_{DD}2$	—	U15
Note 3	V_{SS}	—	U16
Note 3	V_{SS}	—	U17
Note 3	V_{SS}	—	U18
Note 3	V_{SS}	—	U19
Note 3	$V_{DD}2$	—	U20
Note 3	$V_{DD}2$	—	U21
Note 3	$V_{DD}2$	—	U22
Note 3	$V_{DD}2$	—	V13
Note 3	$V_{DD}2$	—	V14
Note 3	$V_{DD}2$	—	V15
Note 3	V_{SS}	—	V16
Note 3	V_{SS}	—	V17
Note 3	V_{SS}	—	V18
Note 3	V_{SS}	—	V19
Note 3	$V_{DD}2$	—	V20
Note 3	$V_{DD}2$	—	V21
Note 3	$V_{DD}2$	—	V22
Note 3	$V_{DD}2$	—	W13
Note 3	$V_{DD}2$	—	W14
Note 3	$V_{DD}2$	—	W15
Note 3	V_{SS}	—	W16
Note 3	V_{SS}	—	W17
Note 3	V_{SS}	—	W18
Note 3	V_{SS}	—	W19
Note 3	$V_{DD}2$	—	W20
Note 3	$V_{DD}2$	—	W21
Note 3	$V_{DD}2$	—	W22
Note 3	V_{SS}	P11	Y13
Note 3	V_{SS}	P12	Y14
Note 3	V_{SS}	P13	Y15
Note 3	$V_{DD}2$	—	Y16
Note 3	$V_{DD}2$	—	Y17
Note 3	$V_{DD}2$	—	Y18
Note 3	$V_{DD}2$	—	Y19
Note 3	V_{SS}	P14	Y20
Note 3	V_{SS}	P15	Y21
Note 3	V_{SS}	P16	Y22
Note 3	V_{SS}	R11	AA13

Pin Information (continued)

Table 35. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
Note 3	VSS	R12	AA14
Note 3	VSS	R13	AA15
Note 3	VDD2	—	AA16
Note 3	VDD2	—	AA17
Note 3	VDD2	—	AA18
Note 3	VDD2	—	AA19
Note 3	VSS	R14	AA20
Note 3	VSS	R15	AA21
Note 3	VSS	R16	AA22
Note 3	VSS	T11	AB13
Note 3	VSS	T12	AB14
Note 3	VSS	T13	AB15
Note 3	VDD2	—	AB16
Note 3	VDD2	—	AB17
Note 3	VDD2	—	AB18
Note 3	VDD2	—	AB19
Note 3	VSS	T14	AB20
Note 3	VSS	T15	AB21
Note 3	VSS	T16	AB22

1. These pads are connected to a power plane in the package rather than to a particular pin. The entry's location in the table indicates the position of the power pad relative to nearby signal pads.
2. Pins marked "No Connect" must be left unconnected.
3. These pins are connected to a power plane in the package rather than to a particular pad.

Package Thermal Characteristics Summary

There are three thermal parameters that are in common use: Θ_{JA}, ψ_{JC}, and Θ_{JC}. It should be noted that all the parameters are affected, to varying degrees, by package design (including paddle size) and choice of materials, the amount of copper in the test board or system board, and system airflow.

ΘJA

This is the thermal resistance from junction to ambient (a.k.a. theta-JA, R-theta, etc.).

$$\Theta_{JA} = \frac{T_J - T_A}{Q}$$

where T_J is the junction temperature, T_A is the ambient air temperature, and Q is the chip power.

Experimentally, Θ_{JA} is determined when a special thermal test die is assembled into the package of interest, and the part is mounted on the thermal test board. The diodes on the test chip are separately calibrated in an oven. The package/board is placed either in a JEDEC natural convection box or in the wind tunnel, the latter for forced convection measurements. A controlled amount of power (Q) is dissipated in the test chip's heater resistor, the chip's temperature (T_J) is determined by the forward drop on the diodes, and the ambient temperature (T_A) is noted. Note that Θ_{JA} is expressed in units of °C/W.

ΨJC

This JEDEC designated parameter correlates the junction temperature to the case temperature. It is generally used to infer the junction temperature while the device is operating in the system. It is not considered a true thermal resistance, and it is defined by:

$$\psi_{JC} = \frac{T_J - T_C}{Q}$$

where T_C is the case temperature at top dead center, T_J is the junction temperature, and Q is the chip power. During the Θ_{JA} measurements described above, besides the other parameters measured, an additional temperature reading, T_C, is made with a thermocouple attached at top-dead-center of the case. ψ_{JC} is also expressed in units of °C/W.

ΘJC

This is the thermal resistance from junction to case. It is most often used when attaching a heat sink to the top of the package. It is defined by:

$$\Theta_{JC} = \frac{T_J - T_C}{Q}$$

The parameters in this equation have been defined above. However, the measurements are performed with the case of the part pressed against a water-cooled heat sink to draw most of the heat generated by the chip out the top of the package. It is this difference in the measurement process that differentiates Θ_{JC} from Ψ_{JC}. Θ_{JC} is a true thermal resistance and is expressed in units of °C/W.

ΘJB

This is the thermal resistance from junction to board (a.k.a. Θ_{JL}). It is defined by:

$$\Theta_{JB} = \frac{T_J - T_B}{Q}$$

where T_B is the temperature of the board adjacent to a lead measured with a thermocouple. The other parameters on the right-hand side have been defined above. This is considered a true thermal resistance, and the measurement is made with a water-cooled heat sink pressed against the board to draw most of the heat out of the leads. Note that Θ_{JB} is expressed in units of °C/W, and that this parameter and the way it is measured are still in JEDEC committee.

FPGA Maximum Junction Temperature

Once the power dissipated by the FPGA has been determined (see the Estimating Power Dissipation section), the maximum junction temperature of the FPGA can be found. This is needed to determine if speed derating of the device from the 85 °C junction temperature used in all of the delay tables is needed. Using the maximum ambient temperature, T_{Amax}, and the power dissipated by the device, Q (expressed in °C), the maximum junction temperature is approximated by:

$$T_{Jmax} = T_{Amax} + (Q \bullet \Theta_{JA})$$

Table 36 lists the thermal characteristics for all packages used with the *ORCA* OR3LP26B Series of FPGAs.

Package Thermal Characteristics Summary (continued)

Table 36. *ORCA* OR3LP26B Plastic Package Thermal Guidelines

Package[1]	ΘJA (°C/W)			TAMB = 70 °C Max TJ = 125 °C Max 0 fpm (W)
	0 fpm	200 fpm	500 fpm	
352-Pin PBGA[2, 3]	19.0	16.0	15.0	2.9
680-Pin PBGAM[2, 3]	14.5	TBD	TBD	3.8

1. Mounted on a 4-layer JEDEC standard test board with two power/ground planes.
2. With thermal balls connected to board ground plane.
3. The value of ψ_{JC} for all packages is <1 °C/W.

Package Coplanarity

The coplanarity limits of the *ORCA* Series 3 packages are as follows.

Table 37. Package Coplanarity

Package Type	Coplanarity Limit (mils)
PBGA	8.0
PBGAM	8.0

Package Parasitics

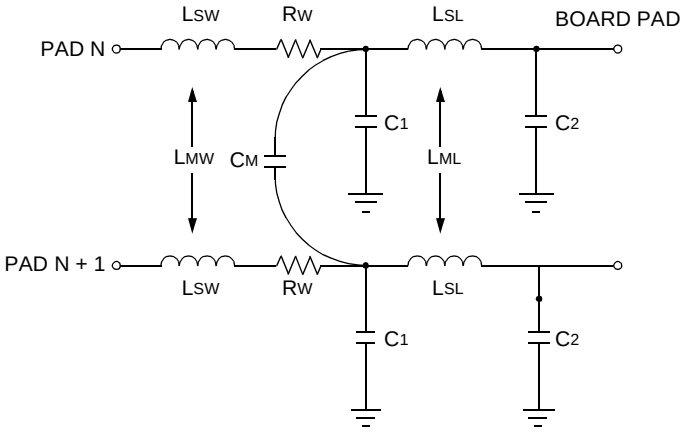
The electrical performance of an IC package, such as signal quality and noise sensitivity, is directly affected by the package parasitics. Table 38 lists eight parasitics associated with the *ORCA* packages. These parasitics represent the contributions of all components of a package, which include the bond wires, all internal package routing, and the external leads.

Four inductances in nH are listed: L_{SW} and L_{SL}, the self-inductance of the lead; and L_{MW} and L_{ML}, the mutual inductance to the nearest neighbor lead. These parameters are important in determining ground bounce noise and inductive crosstalk noise. Three capacitances in pF are listed: C_M, the mutual capacitance of the lead to the nearest neighbor lead; and C_1 and C_2, the total capacitance of the lead to all other leads (all other leads are assumed to be grounded). These parameters are important in determining capacitive crosstalk and the capacitive loading effect of the lead. The lead resistance value, R_W, is in mΩ.

The parasitic values in Table 38 are for the circuit model of bond wire and package lead parasitics. If the mutual capacitance value is not used in the designer's model, then the value listed as mutual capacitance should be added to each of the C_1 and C_2 capacitors.

Table 38. Package Parasitics

Package Type	L_{SW} (nH)	L_{MW} (nH)	R_W (mΩ)	C_1 (pF)	C_2 (pF)	C_M (pF)	L_{SL} (nH)	L_{ML} (nH)
352-Pin PBGA	5	2	220	1.5	1.5	1.5	7—12	3—6
680-Pin EBGA	3.8	1.3	250	1.0	1.0	0.3	2.8—5.0	0.5—1.0



5-3862(F).a

Figure 45. Package Parasitics

Package Outline Diagrams

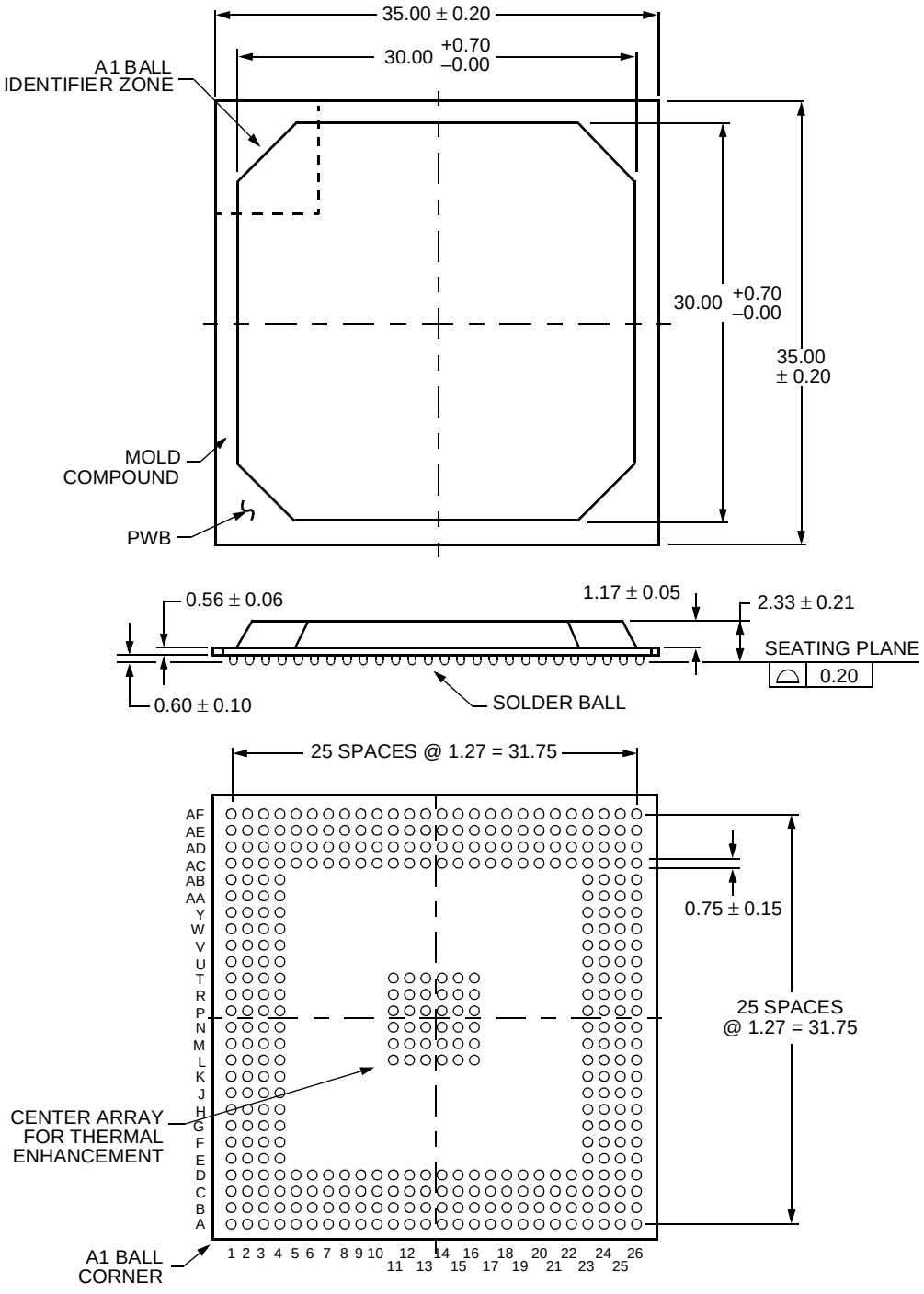
Terms and Definitions

Basic Size (BSC):	The basic size of a dimension is the size from which the limits for that dimension are derived by the application of the allowance and the tolerance.
Design Size:	The design size of a dimension is the actual size of the design, including an allowance for fit and tolerance.
Typical (TYP):	When specified after a dimension, this indicates the repeated design size if a tolerance is specified or repeated basic size if a tolerance is not specified.
Reference (REF):	The reference dimension is an untoleranced dimension used for informational purposes only. It is a repeated dimension or one that can be derived from other values in the drawing.
Minimum (MIN) or Maximum (MAX):	Indicates the minimum or maximum allowable size of a dimension.

Package Outline Diagrams (continued)

352-Pin PBGA

Dimensions are in millimeters.

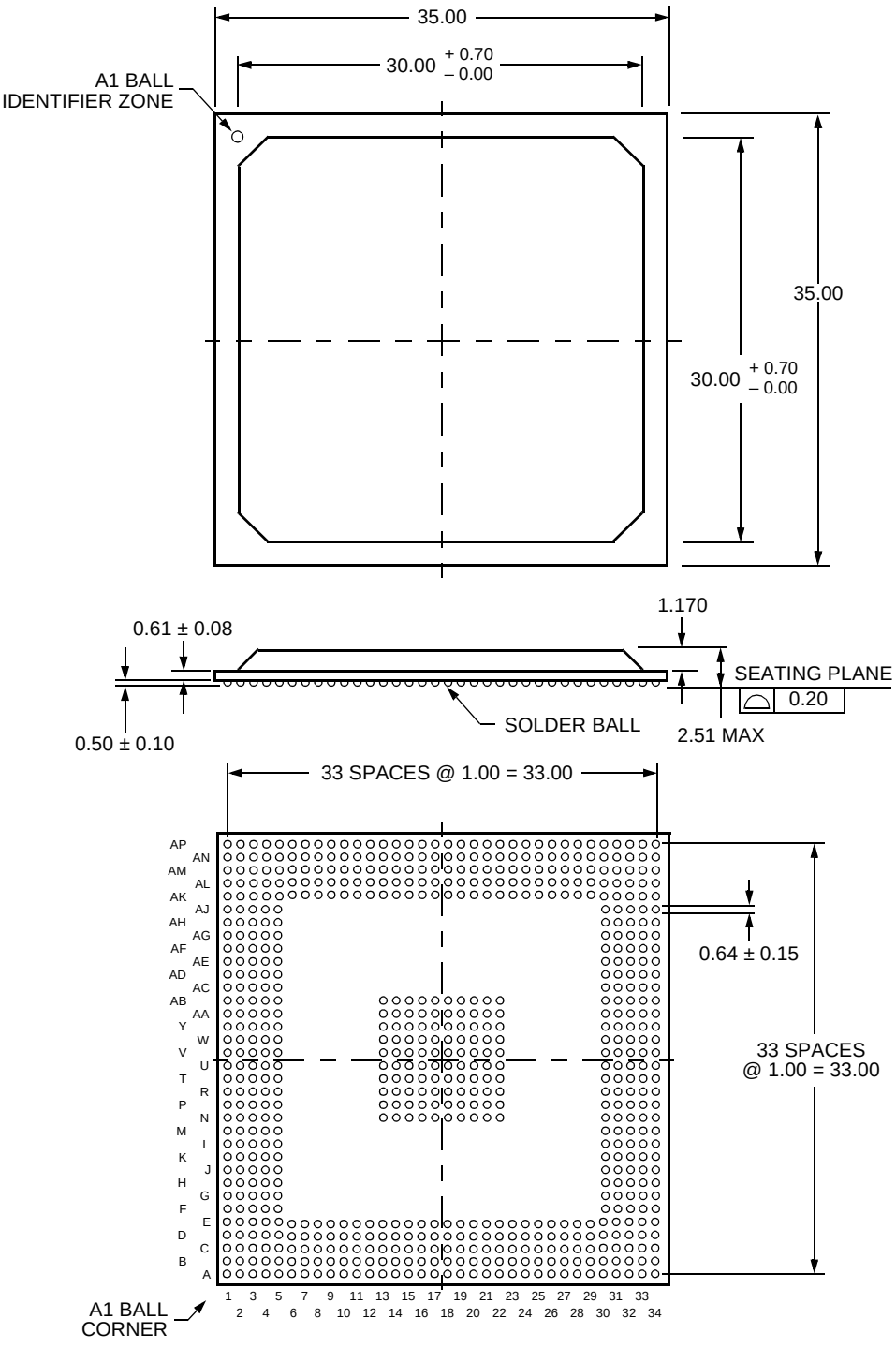


5-4407(F)

Package Outline Diagrams (continued)

680-Pin PBGA

Dimensions are in millimeters.



Ordering Information

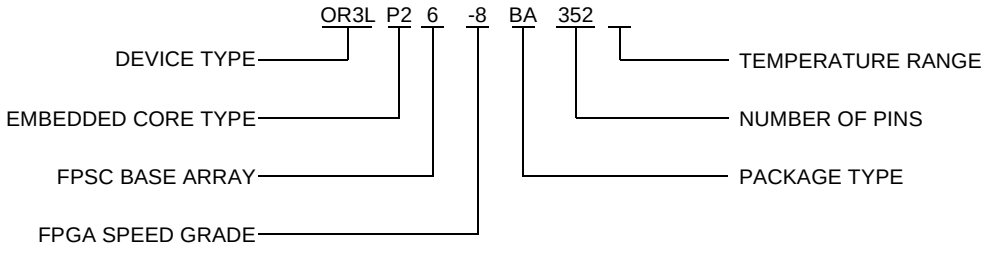


Table 39. Voltage Options

Device	Voltage
OR3L	2.5 V

Table 40. Temperature Options

Symbol	Description	Temperature
(Blank)	Commercial	0 °C to 70 °C
I	Industrial	–40 °C to +85 °C

Table 41. Package Options

Symbol	Description
BA	Plastic Ball Grid Array (PBGA)
BM	Plastic Ball Grid Array Multilayer (PBGAM)

Table 42. *ORCA* Series 3+ Package Matrix

Device	Package	
	352-Pin PBGA	680-Pin PBGAM
	BA352	BM680
OR3LP26B	CI	CI

Key: C = commercial, I = industrial.

Table 43. Embedded Core Type

Symbol	Description
P2	32-/64-bit, 33/66 MHz PCI bus interface with 64-bit back-end data path in each direction.

Table 44. FPSC Base Array

Symbol	Description
6	OR3L125 based 18 x 28 array.

For additional information, contact your Microelectronics Group Account Manager or the following:
INTERNET: **http://www.lucent.com/micro**, or for FPGA information, **http://www.lucent.com/orca**
E-MAIL: **docmaster@micro.lucent.com**
N. AMERICA: Microelectronics Group, Lucent Technologies Inc., 555 Union Boulevard, Room 30L-15P-BA, Allentown, PA 18103
1-800-372-2447, FAX 610-712-4106 (In CANADA: **1-800-553-2448**, FAX 610-712-4106)
ASIA PACIFIC: Microelectronics Group, Lucent Technologies Singapore Pte. Ltd., 77 Science Park Drive, #03-18 Cintech III, Singapore 118256
Tel. (65) 778 8833, FAX (65) 777 7495
CHINA: Microelectronics Group, Lucent Technologies (China) Co., Ltd., A-F2, 23/F, Zao Fong Universe Building, 1800 Zhong Shan Xi Road, Shanghai 200233 P. R. China **Tel. (86) 21 6440 0468**, **ext. 316**, FAX (86) 21 6440 0652
JAPAN: Microelectronics Group, Lucent Technologies Japan Ltd., 7-18, Higashi-Gotanda 2-chome, Shinagawa-ku, Tokyo 141, Japan
Tel. (81) 3 5421 1600, FAX (81) 3 5421 1700
EUROPE: Data Requests: MICROELECTRONICS GROUP DATALINE: **Tel. (44) 7000 582 368**, FAX (44) 1189 328 148
Technical Inquiries: GERMANY: **(49) 89 95086 0** (Munich), UNITED KINGDOM: **(44) 1344 865 900** (Ascot),
FRANCE: **(33) 1 40 83 68 00** (Paris), SWEDEN: **(46) 8 594 607 00** (Stockholm), FINLAND: **(358) 9 4354 2800** (Helsinki),
ITALY: **(39) 02 6608131** (Milan), SPAIN: **(34) 1 807 1441** (Madrid)

November 1999
DS98-318FPGA

microelectronics group

Lucent Technologies
Bell Labs Innovations