Data Sheet
April 2002

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

Introduction

Lattice 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 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

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 **intan** 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 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.

FPSC Highlights (continued)

- — 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).
- — 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 5 MHz to 120 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.

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

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.

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

Lattice'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.

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, 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.

Boundary Scan

Boundary scan is implemented in the OR3LP26B device as with any of the OR3LXXB family of parts. The PCI core side of the device contains the same boundary-scan registers. After performing a boundary-scan test, it is highly recommended that the device be reset through the PCI **rstn** pin. This reset will clear out any PCI core internal registers that may have been set during the boundary-scan tests.

More Series 3 Information

For more information on Series 3 FPGAs, please refer to the Series 3 FPGA data sheet, available on the Lattice website .

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 inter-quad 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.

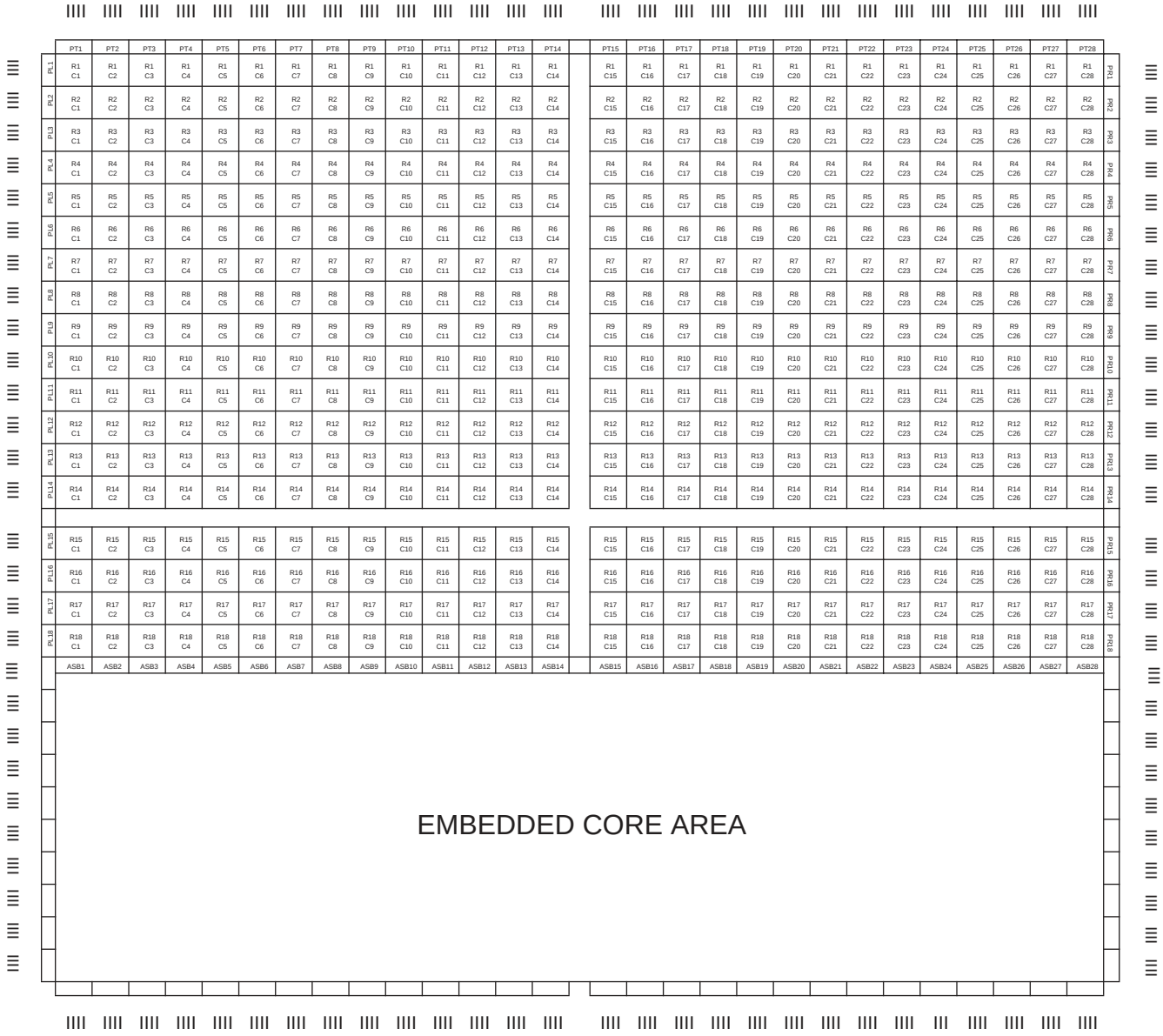
Table 2. PCI Local Bus Data Rates

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.

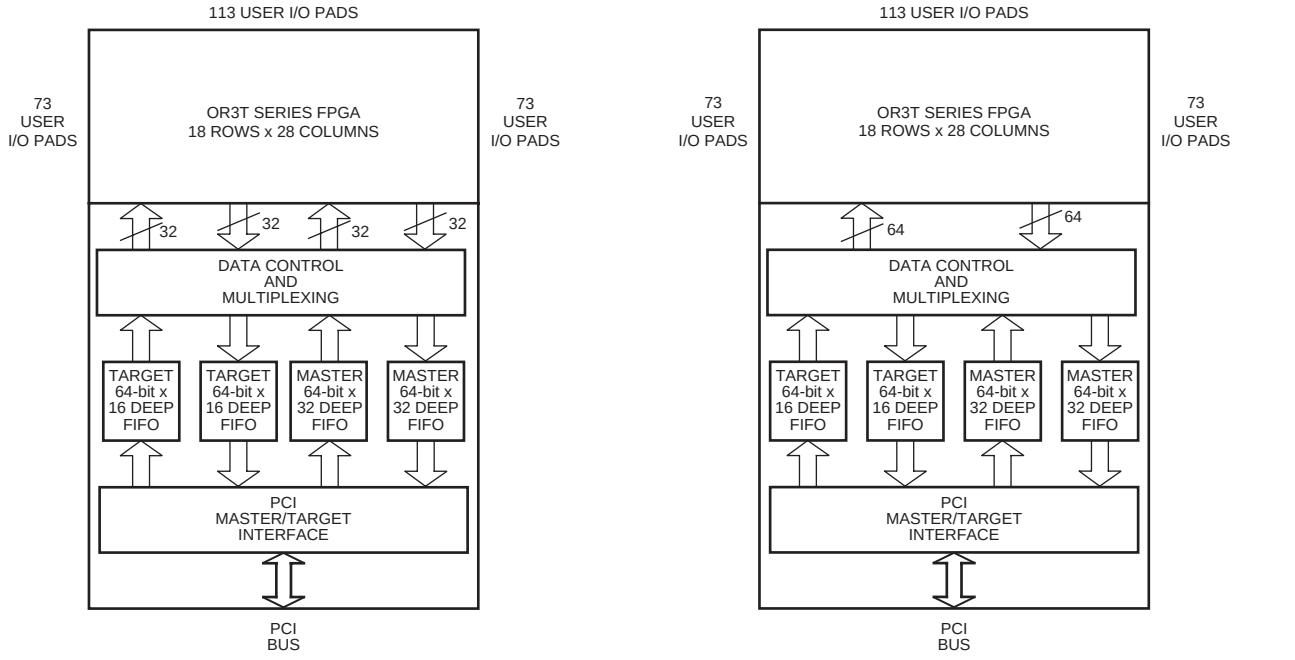
OR3LP26B Overview (continued)

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.



Note: 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; 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; 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. 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	Configuration 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	Configuration 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 (devseln)

The target is responsible for responding to a master's request by asserting the PCI bus signal **devseln**. **devseln** 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.

Reset Operation

The PCI bus contains a signal, **rstn**, that performs a PCI reset function. When the reset occurs, all state machines in the ASIC are placed in their idle state, the configuration space BARs are reset to their mask values, and the command registers are reset. The reset does not reset the FPGA logic. The PCI reset signal is fed from the ASIC to the FPGA logic to be used by the designer.

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.

PCI Bus Core Detailed Description

(continued)

Arbitration Parking

The PCI Specification requires that all master agents properly handle bus parking, which means that when that agent receives an asserted **gntn** without the agent having asserted **reqn**, the agent still must drive signal **par** and buses AD and c_ben. 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 **perrn** 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 also reduce 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 always configured to operate in 64-bit mode 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). During 32-bit transactions, the FPSC will pack the data to fully utilize the memories. 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 70.

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 **rstn** and **intan**, are sampled on the rising edge of **clk**, and all other PCI bus timing parameters are defined with respect to this edge. The signal **clk** operates up to 66 MHz, and the minimum frequency is dc.
rstn	I	**Reset.** An active-low signal used to reset the entire PCI bus. **rstn** is asynchronous to **clk**. During **rstn**, 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_ben[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_ben[3:0]** define the bus command. During the data phase, **c_ben[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_ben[3:0]**. **par** is stable and valid one clock after the address phase. For data phases, **par** is stable and valid one clock after **irdyn** is asserted on a write transaction or **trdyn** 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		
framen	I/O	**Cycle Frame.** An active-low signal driven by the current Master to indicate the beginning and duration of an access. The signal framen is asserted to indicate a bus transaction is beginning. While **framen** is asserted, data transfers continue. When **framen** is deasserted, the transaction is in the final phase or has completed.
irdyn	I/O	**Initiator Ready.** An active-low signal indicating the bus Master's ability to complete the current data phase of the transaction. The signal **irdyn** is used in conjunction with **trdyn**. A data phase is completed on any clock cycle during which both **irdyn** and **trdyn** are asserted. During a write, **irdyn** 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 **irdyn** and **trdyn** are asserted together.
trdyn	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. The signal **trdyn** is used in conjunction with **irdyn**. A data phase is completed on any clock where both **trdyn** and **irdyn** are sampled active. During reads, **trdyn** indicates that valid data is present on **ad[31:0]** lines. During write cycles, **trdyn** indicates that the Target is prepared to accept data.
stopn	I/O	**STOPn.** 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]**.
devseln	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)		
reqn	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.
gntn	I	**Grant.** An active-low signal that indicates to the OR3LP26B that access to the PCI bus has been granted.
Error Reporting Pins		
perrn	I/O	**Parity Error.** An active-low signal for the reporting of data parity errors during all PCI transactions except a special cycle. The **perrn** 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 **perrn** 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 **perrn** signal will be asserted for more than a single clock. **perrn** is driven high for one clock before being 3-stated. The signal **perrn** is not asserted until it has claimed the access by asserting **devseln** and completed a data phase.

PCI Bus Core Detailed Description (continued)

Table 7. PCI Bus Pin Descriptions (continued)

Symbol	I/O	Description
serrn	O	**System Error.** An active-low open drain signal pulsed by agents to report errors other than parity. **serrn** is sampled every **clk** edge, so any agent asserting **serrn** must ensure it is valid for at least one clock period. The OR3LP26B asserts **serrn** if a Master abort sequence is asserted when the Master controller is accessing the PCI bus.
Interrupt Pins		
intan	O	**PCI Interrupt.** The OR3LP26B asserts this active-low open drain 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 **req64n** is asserted), these address bits are transferred. During a data phase, the data is valid when **req64n** and **ack64n** are both asserted. Otherwise, these bits are 3-stated.
c_ben[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 **req64n** 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.
req64n	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. The signal **req64n** has the same meaning as **framen** for 32-bit transfers.
ack64n	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. The signal **ack64n** has the same timing as **devseln** in 32-bit transfers.
par64	I/O	**Upper Double-Word Parity.** The even parity bit that covers **ad[63:32]** and **c_ben[7:4]**. PAR64 is valid one clock after the initial address phase when **req64n** is asserted and the DAC command is indicated on **c_ben[7:4]**. It is also valid the clock cycle after the second address phase of a DAC command when **req64n** is asserted.
Hot Swap Function Pins		
enumn	O	Active-low open drain 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.
ledn	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 Dual Port

Pages 19—67 will refer to the dual-port mode of the OR3LP26B device. For quad-port mode, please refer to pages 68—120.

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
Data FIFO Signals		
datafmfpga[63:0] datafmfpgax[7:0]	O	Main data bus into the master write FIFO and target read FIFO. Refer to Table 10 on page 28 for bus usage and bit descriptions. These signals must be synchronous to **fclk**.
datatofpga[63:0] datatofpgax[7:0]	I	Main data bus out of the master read FIFO and target write FIFO. Refer to Table 10 on page 28 for bus usage and bit descriptions. These signals are synchronous to **fclk**.
Master General Signals		
fpga_mbusyn	O	**FPGA Master Is Busy.** This signal is used in modes currently not implemented in the core. Tie off this signal to a 1.
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. This signal is synchronous to **fclk**.
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. Both signals are synchronous to **fclk**.
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. On each rising edge of the clock that this signal is sampled low, command, burst length, and address will be registered. This signal must be synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
mstatecntr[2:0]	I	**Internal State Counter.** Used for Master reads and writes. Details of the Master state machine operation can be found in tables at the end of each operation section. This signal is synchronous to **fclk**.
mfifoclrn	O	**Master FIFO Clear.** This active-low signal is asserted by the FPGA Master to clear all Master FIFOs. This signal must be synchronous to **fclk**.

PCI Bus Core Detailed Description Dual Port (continued)

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

Symbol	I/O	Description
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. This signal is synchronous to **fclk**.
mcmd[3:0]	O	**Master Command Code.** Command code for the current Master read/write operation. Refer to Table 10 on page 28. This signal must be synchronous to **fclk**.
Master Write Data FIFO Signals		
mwdataenn	O	**Master Write FIFO Data Enable.** This active-low signal enables the registering of bus **datafmfpga** during Master write operations into the PCI core Master write data FIFOs on the rising edge of the Master FIFO clock signal. The signal **mwdataenn** should not be asserted when the Master write data FIFOs are full, or data may be lost. This signal must be synchronous to **fclk**.
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. When asserted, **mwpcihold** must be held low for a minimum of two **pciclk** periods. This signal must be synchronous to **pcilk**.
mw_fulln	I	**Master Write Data FIFO Full Flag.** This active-low signal indicates that the Master write data FIFOs are full. This signal is synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
mw_emptyn	I	**Master Write Data FIFO Empty Flag.** This active-low signal indicates that the Master write data FIFO is empty. Refer to Master write description on signal usage. This signal is synchronous to **pciclk**.
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. This signal must be synchronous to **fclk**.
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 **datatofpga** during Master read operations on the rising edge of the Master FIFO clock signal. Valid data will be read from the FIFO whenever it is not empty. This signal must be synchronous to **fclk**.
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. This signal is synchronous to **fclk**.

PCI Bus Core Detailed Description Dual Port (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. This signal is synchronous to **fclk**.
mr_fulln	I	**Master Read Data FIFO Full Flag.** This active-low signal indicates that the Master read data FIFO is full. Refer to Master read description on signal usage. This signal is synchronous to **pciclk**.
fpga_mstopburstn	O	**Stop Burst Reads.** This active-low signal is used by the FPGA Master to terminate burst reads before completion. When asserted, it must stay asserted for a minimum of two **pciclk** periods. When asserted, **fpga_mstopburstn** must stay asserted until **ma_fulln** goes inactive (high). This signal must be synchronous to **pciclk**.
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 (1 **fclk** period). This signal is synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
fpga_tabort	O	**Target Abort.** This active-high signal is asserted by the FPGA Target application to abort all future PCI cycles. Once asserted, this signal needs to remain asserted for a minimum of two **pciclk** cycles. This signal must be synchronous to **pciclk**.
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 **pciclk** cycles. This signal must be synchronous to **pciclk**.
deltrn	O	**Target Delayed Transaction.** Used for Target I/O write (page 48) and Target read operations (page 57). Target memory writes are always posted. Once asserted, this signal needs to remain asserted for a minimum of two **pciclk** cycles. This signal must be synchronous to **pciclk**.
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. Both signals are synchronous to **fclk**.

PCI Bus Core Detailed Description Dual Port (continued)

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

Symbol	I/O	Description
Target FIFO Address and Command Register Control Signals		
tfifoclrn	O	**Target FIFO Clear.** This active-low signal is asserted by the FPGA Target to clear all Target FIFOs. This signal must be synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
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). This signal must be synchronous to **fclk**.
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. Because it is synchronous to **pciclk**, it must be qualified with **treqn**.
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. Because it is synchronous to **pciclk**, it must be qualified with **treqn**.
tstatecntr[2:0]	I	**Internal State Counter.** Used for target reads and writes. Details of the target state machine operation can be found in tables at the end of each operation section. This signal is synchronous to **fclk**.
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 **datatofpga** 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. This signal must be synchronous to **fclk**.
tw_emptyn	I	**Target Write FIFO Empty.** This signal active indicates that the Target write FIFO is empty. This signal is synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
tw_fulln	I	**Target Write Data FIFO Full Flag**. This active-low signal indicates that the target write data FIFO is full. Refer to target write description on signal usage. This signal is synchronous to **pciclk**.

PCI Bus Core Detailed Description Dual Port (continued)

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

Symbol	I/O	Description
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 Target 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 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. This signal is synchronous to **fclk**.
Target Read Data FIFO Signals		
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). Once asserted, this signal needs to remain asserted for a minimum of two **pciclk** periods. This signal must be synchronous to **pciclk**.
trdataenn	O	**Target Read FIFO Data Enable.** This active-low signal enables the registering of bus **datafmfpga** during Target read operations into the PCI core Target read data FIFOs on the rising edge of the Target FIFO clock signal. The signal **trdataenn** should not be asserted when the Target read data FIFOs are full, or data may be lost. This signal must be synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
tr_emptyn	I	**Target Read Data FIFO Empty Flag.** This active-low signal indicates that the target read data FIFO is empty. Refer to target read description on signal usage. This signal is synchronous to **pciclk**.
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. Once asserted, this signal needs to remain asserted for a minimum of two **pciclk** periods. This signal must be synchronous to **pciclk**.
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. This signal is synchronous to **fclk**.

PCI Bus Core Detailed Description Dual Port (continued)

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

Symbol	I/O	Description
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). Once asserted, this signal needs to remain asserted for a minimum of two **pciclk** periods. This signal must be synchronous to **pciclk**.
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 **intan** onto the PCI bus. Once asserted, this signal needs to remain asserted for a minimum of two **pciclk** cycles. This signal must be synchronous to **pciclk**.
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**. External or user-defined clocks may also be used. The signals **fclk1** and **fclk2** must be the same clock in dual-port mode.
pciclk	I	**PCI Clock**. The signal **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 (**rstn**).
fpga_syserror	O	**System Error**. This active-high signal is used by the FPGA to generate a system error on the PCI bus. This is passed to the PCI bus as **serrn**. This signal must be synchronous to **pciclk**.
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 **req64n** as active (low) on the inactive-going (rising) edge of PCI signal **rstn**. Note that this does not imply that any particular transaction is 64-bit, since each transaction is individually negotiated using PCI signals **req64n** and **ack64n**. This signal is synchronous to **pciclk**.
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). This signal must be synchronous to **fclk**.

PCI Bus Core Detailed Description Dual Port (continued)

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
ASB1A	pci_rstn	pci_intan
ASB1B	pci_64bit	(unused)
ASB1C	(unused)	fpga_syserror
ASB1D	(unused)	fpga_mbusyn
ASB2A	datatofpga31	datafmfpga31
ASB2B	datatofpga30	datafmfpga30
ASB2C	datatofpga29	datafmfpga29
ASB2D	datatofpga28	datafmfpga28
ASB3A	datatofpga27	datafmfpga27
ASB3B	datatofpga26	datafmfpga26
ASB3C	datatofpga25	datafmfpga25
ASB3D	datatofpga24	datafmfpga24
ASB4A	datatofpga23	datafmfpga23
ASB4B	datatofpga22	datafmfpga22
ASB4C	datatofpga21	datafmfpga21
ASB4D	datatofpga20	datafmfpga20
ASB5A	datatofpga19	datafmfpga19
ASB5B	datatofpga18	datafmfpga18
ASB5C	datatofpga17	datafmfpga17
ASB5D	datatofpga16	datafmfpga16
ASB6A	datatofpgax3	datafmfpgax3
ASB6B	datatofpgax2	datafmfpgax2
ASB6C	datatofpgax1	datafmfpgax1
ASB6D	datatofpgax0	datafmfpgax0
ASB7A	datatofpga15	datafmfpga15
ASB7B	datatofpga14	datafmfpga14
ASB7C	datatofpga13	datafmfpga13
ASB7D	datatofpga12	datafmfpga12
ASB8A	datatofpga11	datafmfpga11
ASB8B	datatofpga10	datafmfpga10
ASB8C	datatofpga9	datafmfpga9
ASB8D	datatofpga8	datafmfpga8
ASB9A	datatofpga7	datafmfpga7
ASB9B	datatofpga6	datafmfpga6
ASB9C	datatofpga5	datafmfpga5
ASB9D	datatofpga4	datafmfpga4
CKTOASB9	(unused)	fclk1

PCI Bus Core Detailed Description Dual Port (continued)

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

PCI Core/FPGA Interface Site	FPGA Input Signal Name	FPGA Output Signal Name
ASB10A	datatofpga3	datafmfpga3
ASB10B	datatofpga2	datafmfpga2
ASB10C	datatofpga1	datafmfpga1
ASB10D	datatofpga0	datafmfpga0
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
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	datafmfpga32
ASB19B	datatofpga33	datafmfpga33
ASB19C	datatofpga34	datafmfpga34
ASB19D	datatofpga35	datafmfpga35
CKTOASB19	(unused)	fclk2
ASB20A	datatofpga36	datafmfpga36

PCI Bus Core Detailed Description Dual Port (continued)

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

PCI Core/FPGA Interface Site	FPGA Input Signal Name	FPGA Output Signal Name
ASB20B	datatofpga37	datafmfpga37
ASB20C	datatofpga38	datafmfpga38
ASB20D	datatofpga39	datafmfpga39
ASB21A	datatofpga40	datafmfpga40
ASB21B	datatofpga41	datafmfpga41
ASB21C	datatofpga42	datafmfpga42
ASB21D	datatofpga43	datafmfpga43
ASB22A	datatofpga44	datafmfpga44
ASB22B	datatofpga45	datafmfpga45
ASB22C	datatofpga46	datafmfpga46
ASB22D	datatofpga47	datafmfpga47
ASB23A	datatofpgax4	datafmfpgax4
ASB23B	datatofpgax5	datafmfpgax5
ASB23C	datatofpgax6	datafmfpgax6
ASB23D	datatofpgax7	datafmfpgax7
ASB24A	datatofpga48	datafmfpga48
ASB24B	datatofpga49	datafmfpga49
ASB24C	datatofpga50	datafmfpga50
ASB24D	datatofpga51	datafmfpga51
ASB25A	datatofpga52	datafmfpga52
ASB25B	datatofpga53	datafmfpga53
ASB25C	datatofpga54	datafmfpga54
ASB25D	datatofpga55	datafmfpga55
ASB26A	datatofpga56	datafmfpga56
ASB26B	datatofpga57	datafmfpga57
ASB26C	datatofpga58	datafmfpga58
ASB26D	datatofpga59	datafmfpga59
ASB27A	datatofpga60	datafmfpga60
ASB27B	datatofpga61	datafmfpga61
ASB27C	datatofpga62	datafmfpga62
ASB27D	datatofpga63	datafmfpga63
ASB28A	mstatecntr0	mfifoclrn
ASB28B	mstatecntr1	(unused)
ASB28C	mstatecntr2	(unused)
ASB28D	pci_mcfg_stat	mcfgshiftenn

PCI Bus Core Detailed Description Dual Port (continued)

Table 10. Bit Definitions on FPGA/PCI Core Interface

Bits	Name	Description
A. Dual-Port Master Write, Command and Address		**mstatecntr = 0**
datafmfpgax[7:3]	—	Unused
datafmfpgax[2]	DA	Dual address indicator (active-high)
datafmfpgax[1:0]	—	Unused
datafmfpga[63:32]	A3 & A2	Address words 3 and 2 (if DA = 1; else must set all bits to 0s)
datafmfpga[31:0]	A1 & A0	Address words 1 and 0
mcmd[3:0]	mcmd	Master command opcode*
B. Dual-Port Master Write, Data		**mstatecntr = 4**
datafmfpgax[7:0]	BE7—BE0	Byte enables (active-low)
datafmfpga[63:0]	D7—D0	Data bytes 7 to 0
C. Dual-Port Master Read (Burst Length Cycle)		**mstatecntr = 0**
datafmfpgax[7:3]	—	Unused
datafmfpgax[2]	DA	Dual address indicator (active-high)
datafmfpgax[1:0]	—	Unused
datafmfpga[63:56]	mrd_benn	Byte enables (active-low)
datafmfpga[55:50]	—	Unused
datafmfpga[49:32]	BL	Burst length (in Quadwords)
datafmfpga[31:0]	A1 & A0	Address words 1 and 0 (set to all 0s if 64-bit address required—A1 & A0 supplied in next cycle)
mcmd[3:0]	mcmd	Master command opcode*
D. 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
E. Dual-Port Master Read, Data		**mstatecntr = 4**
datatofpgax[7:0]	—	Unused
datatofpga[63:0]	D7—D0	Data bytes 7 to 0

* 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 Dual Port (continued)

Table 10. Bit Definitions on FPGA/PCI Core Interface (continued)

Bits	Name	Description
F. Dual-Port Target Write & Read, Command and Address		**tstatecntr = 0**
datatofpgax[7:4]	—	Unused
datatofpgax[3]	Burst_I	Burst indication (active-high)
datatofpgax[2]	DA	Dual address indicator (active-high)
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*
G. Dual-Port Target Write, Data		**tstatecntr = 4**
datatofpgax[7:0]	BE7—BE0	Byte enables (active-low)
datatofpga[63:0]	D7—D0	Data bytes 7 to 0
H. Dual-Port Target Read, Data		**tstatecntr = 4**
datafmfpgax[7:0]	—	Unused
datafmfpga[63:0]	D7—D0	Data bytes 7 to 0

* 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

Table 11. Address Cycle Sequences for Various Operations

Operation	Address Mode	Supplied Address	New Burst Length	Address Cycle Sequence (Once Only)	Data Cycle Sequence (Repeats)
Master Write	SA	31:0	NA	A	B
	DA	63:0	NA	A	B
Master Read	SA	31:0	C	NA	E
	DA	63:0	C	D	E
Target Write	SA	31:0	NA	F	G
	DA	63:0	NA	F	G
Target Read	SA	31:0	NA	F	H
	DA	63:0	NA	F	H

PCI Bus Core Detailed Description Dual Port (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 Dual Port (continued)

Understanding FIFO Packing/Unpacking

In dual-port mode, the interface from the core to the FPGA is always 64 bits wide. However, data packing through the FIFOs will differ depending on whether the transfers on the PCI bus are 32 bits or 64 bits. The following discussions pertain to target write or master read operations where data will be read from the FIFOs.

- **64-bit Transfers**: Since the FIFOs are always in 64-bit mode, the data will flow through without any repacking. Keep in mind that 64-bit transfers must start on a Quadword aligned address (AD2 = 0).
- **32-bit Transfers**: The FIFOs are always in 64-bit mode, so depending upon what address the transfer begins, the data coming out of the FIFOs will be packed differently. The following two cases provide examples with different starting addresses and word counts. Case 1 is also true for Master read operations.

Case 1: Target write burst, 32-bit. Even 32-bit starting address, and even number of 32-bit words transferred on the PCI bus.

Table 13. Dual-Port FIFO Packing/Unpacking, Case 1, PCI Side

PCI Address	PCI Data	PCI Byte Enables (Active-Low)
00001000	32-bit Word1	0000
(00001004)	32-bit Word2	0000
(00001008)	32-bit Word3	0000
(0000100C)	32-bit Word4	0000
(00001010)	32-bit Word5	0000
(00001014)	32-bit Word6	0000

Table 14. Dual-Port FIFO Packing/Unpacking, Case 1, FPGA Side

Master Write FIFO Slot	FIFO Data Bits 63:32	FIFO Data Bits 31:0	FIFO Byte Enables (Active-Low)
	datatofpga[63:0]		datatofpgax[7:0]
1	32-bit Word2	32-bit Word1	00000000
2	32-bit Word4	32-bit Word3	00000000
3	32-bit Word6	32-bit Word5	00000000

Note: PCI addresses in parentheses are not actually sent across the PCI bus during a burst. They are used for illustrative purposes only. Dummy words are unknown data words in the FIFOs with their byte enables disabled.

Case 2: Target write burst, 32-bit. Even 32-bit starting address, odd number of 32-bit words transferred on the PCI bus.

Table 15. Dual-Port FIFO Packing/Unpacking, Case 2, PCI Side

PCI Address	PCI Data	PCI Byte Enables (Active-Low)
00001000	32-bit Word1	0000
(00001004)	32-bit Word2	0000
(00001008)	32-bit Word3	0000
(0000100C)	32-bit Word4	0000
(00001010)	32-bit Word5	0000

PCI Bus Core Detailed Description Dual Port (continued)

Table 16. Dual-Port FIFO Packing/Unpacking, Case 2, FPGA Side

Master Write FIFO Slot	FIFO Data Bits 63:32	FIFO Data Bits 31:0	FIFO Byte Enables (Active-Low)
	datatofpga[63:0]		datatofpgax[7:0]
1	32-bit Word2	32-bit Word1	00000000
2	32-bit Word4	32-bit Word3	00000000
3	Dummy Word	32-bit Word5	FFFF0000

Note: PCI addresses in parentheses are not actually sent across the PCI bus during a burst. They are used for illustrative purposes only. Dummy words are unknown data words in the FIFOs with their byte enables disabled.

Embedded Core/FPGA Interface Operation

Target Address Holding Register and BAR Number Indicator

The PCI core provides two features that reduce overhead on setup of Target transfers.

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. The **bar** bus is synchronous to **pciclk**, so it must be qualified with **treqn**.

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.

Interrupt Request and System Error Generation

Two additional signals are available on the user side interface to request an interrupt on **intan** (**pci_intan**) and force a system error on the PCI **serrn** pin (**fpga_syserror**). The **pci_intan** signal may be asserted low at any time. It is not directly tied to any bus cycle. The **fpga_syserror**, as well, may be asserted high at any time. The **serrn** signal will be subsequently asserted low during the next PCI transaction to this device. In generating **pci_intan** and **fpga_syserror**, keep in mind that both signals need to be synchronous to **pciclk**.

Working in 32-bit and 64-bit Modes

The OR3LP26B works equally well in 32-bit and 64-bit PCI systems. In a 64-bit system, it is required that, during reset, the host assert **req64n** low indicating that the bus width is 64 bits. The core will evaluate this signal at reset, and automatically configure itself in either 32-bit or 64-bit mode. When configured in 32-bit mode, the core will 3-state all upper PCI bus pins and apply a weak pull-up.

32-Bit Transfers in a 64-bit System

Although designed as a 64-bit interface, the OR3LP26B also works efficiently in 32-bit mode. For single 32-bit transfers, the core will perform a 32-bit PCI transfer. For burst transactions, the core will attempt 64-bit transfers, and then back down to 32-bit mode if **ack64n** was not received. In general, the core will perform the PCI bus transaction that is most efficient on the bus.

PCI Bus Core Detailed Description Dual Port (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 the states of each operation for each mode.

Table 17 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 17. Index to State Sequence Tables

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
Master	Write	Config, Memory, I/O	Nonburst	Figure 3	Table 18	Figure 2
			Burst	Figure 5		Figure 4
	Read	Config, Memory, I/O	Nonburst	Figure 7	Table 19* Table 20†	Figure 6
			Burst	Figure 9		Figure 8
Target	Write	Config	Nonburst	Figure 10	Table 21	‡
		I/O	Delayed	Figure 11		Figure 13
		Memory, I/O	Nonburst, Not Delayed	Figure 12		Figure 15
		Memory	Burst	Figure 14		
	Read	Config	Nonburst	Figure 16	Table 22	‡
		I/O	Delayed	Figure 17		Figure 20
			Not Delayed	Figure 18		
		Memory	Nonburst	Figure 21		
			Nonburst Delayed	Figure 19		
			Burst	Figure 24		Figure 23
			Burst Delayed	Figure 22		

* 64-bit address supplied.
† 32-bit address supplied.
‡ The FPGA interface does not participate in Target configuration operations.

PCI Bus Core Detailed Description Dual Port (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 18. 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[2:0]**, 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 **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 **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.

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 **datafmfpga**.

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.

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. During a master write burst transaction, the master write FIFO may go empty, especially if the user side application is slow at filling the FIFO. When this condition occurs, the master will insert wait-states continuously until another word (or the last word) is written into the FIFO and will not terminate the transaction. On the target side, if the target is ready to accept more data, it will have **trdyn** asserted which will disable it from terminating the transaction as well. This can create a deadlock condition on the PCI bus. If the user application cannot supply any more data, and wishes to terminate the burst, additional FPGA logic must be incorporated to detect and accomplish the termination. The way to terminate the transaction is to provide one last piece of data (either real data or a dummy data word with all byte enables disabled) along with **mwlastcycn** asserted.

PCI Bus Core Detailed Description Dual Port (continued)

Designing a Deadlock Timer

This design example is a method by which the user application can detect the deadlock condition and terminate the burst transaction. Since the **mw_emptyn** signal is on the **pciclk** clock domain, it must be resynchronized to the fclk domain. To accomplish this, double register **mw_emptyn** with fclk driven registers. The **mw_emptyn** signal is fed as a clock enable and a synchronous clear to a counter, driven by **fclk**. The counter's length may be designed to guarantee a certain time-out latency on the PCI bus. When the FIFO is not empty (**mw_emptyn** = 1), the counter will stay cleared. When the FIFO has been empty for an extended period of time, the counter will count and eventually overflow. This overflow indication can be used to write one dummy word into the FIFO with the byte enables disabled along with the **mwlastcycn** bit asserted. The transaction will complete, and the core will go back into an idle state.

Bursting

Instead of using a burst length, the Master write operation relies on **mwlastcycn** 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 **datafmfpga**. The signal **mwlastcycn** must remain asserted throughout a nonburst Master write, since the last data phase is the only data phase. The maximum burst length is limited only by the latency timer. To initiate a burst, the starting address must be aligned to a 64-byte boundary. If **ad[2]** is a 1, a single transfer will be executed.

Termination

Once initiated, Master write operations will repeat on the PCI bus until either one of the following occurs:

1. All data is sent.
2. An abort occurs (either Master or Target).
3. The PCI bus's reset signal (**rstn**) 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 **mfifoclrn** to place the core's master logic in a known state. Normally, the clear signal will not be used unless a severe problem has occurred in the data flow. The **mfifoclrn** signal is synchronous with **fclk** and must be asserted for a minimum of three clock periods. During reset, the **m_ready** signal will go low. After the reset signal is deasserted high, **m_ready** will continue to be low for 8—10 clock periods. The FPGA application should not continue normal operation until **m_ready** is asserted high.

Understanding and Using the pci_mcfg_stat Status Signals

On the Master interface, there are two signals that control and provide status to the FPGA application. The signal **pci_mcfg_stat** provides the status, and **mcfgshiftenn** controls what information the status line provides. The **pci_mcfg_stat** signal is always active and duplicates the status contained in configuration status register at location offset 0x04, bits 24, 28, and 29. To use this status output, the FPGA application must keep **mcfgshiftenn** = 1. When high, **pci_mcfg_stat** provides the wired-OR of the three status lines. If **pci_mcfg_stat** gets set to a 1, indicating an error, then the FPGA application may set **mcfgshiftenn** = 0 to determine individual status. Once low, the **pci_mcfg_stat** signal will output data parity error detected on the first clock, target abort received on the second clock, and master abort received on the third clock.

PCI Bus Core Detailed Description Dual Port (continued)

Master Write, Nonburst Transaction

Figure 2 (FPGA bus) and Figure 3 (PCI bus) show the timing of a Master write, nonburst 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.

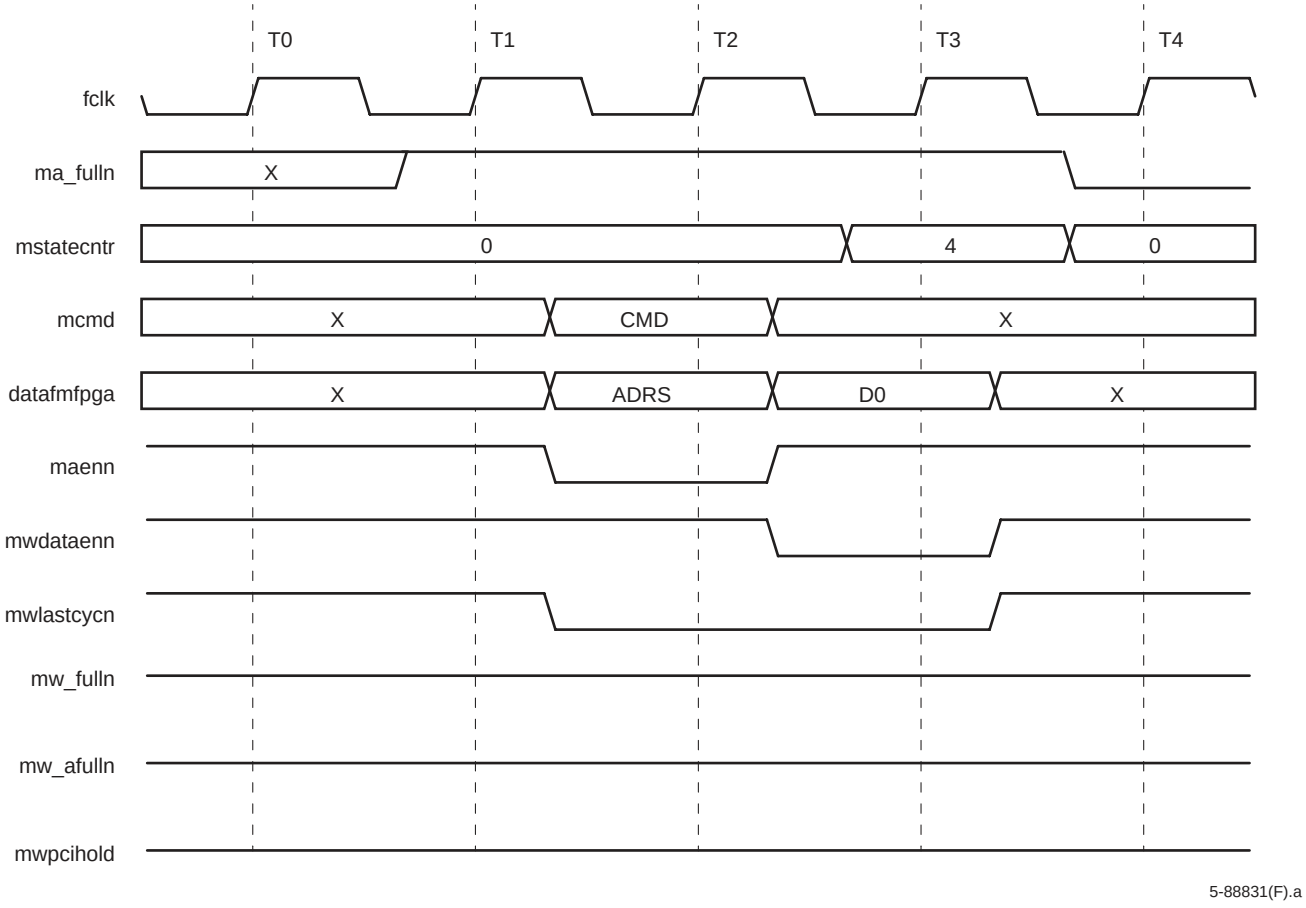


Figure 2. Master Write Single (FPGA Bus, Dual-Port)

PCI Bus Core Detailed Description Dual Port (continued)

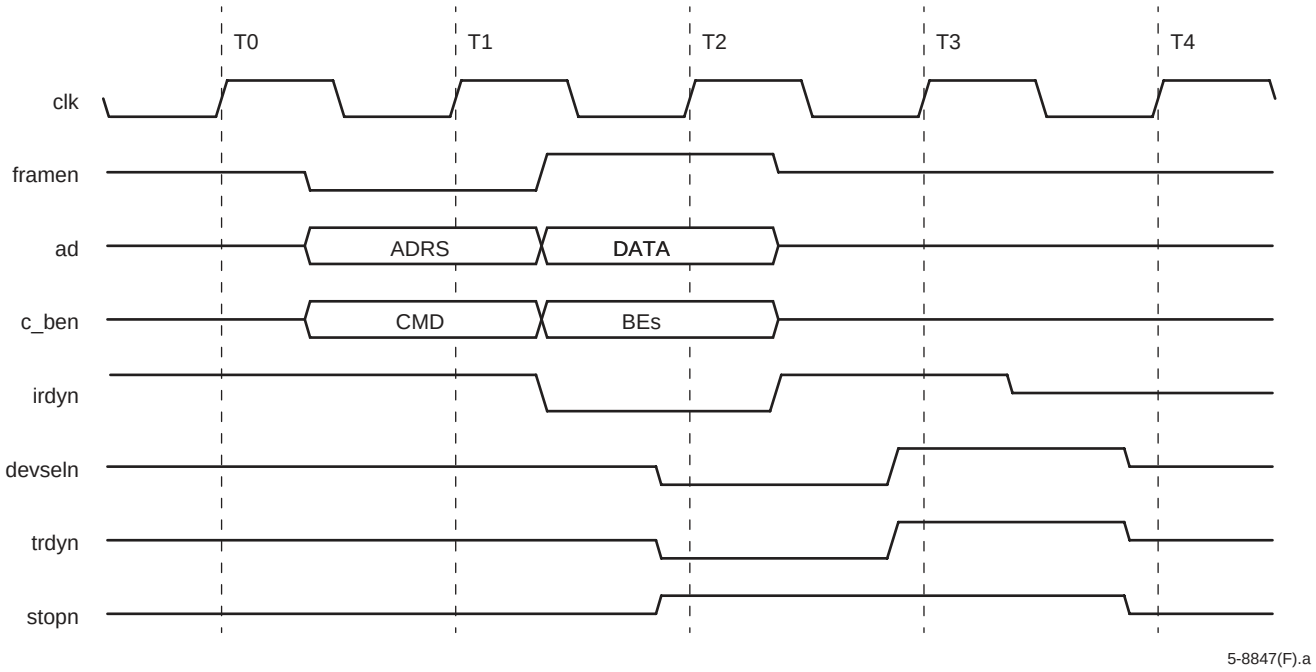


Figure 3. Master Write Single (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

Master Write, Burst Transaction

Figure 4 (FPGA bus) and Figure 5 (PCI bus) show the timing of a four Quadword Master write burst transaction. Operation is similar to that in the previous Master write, nonburst 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.

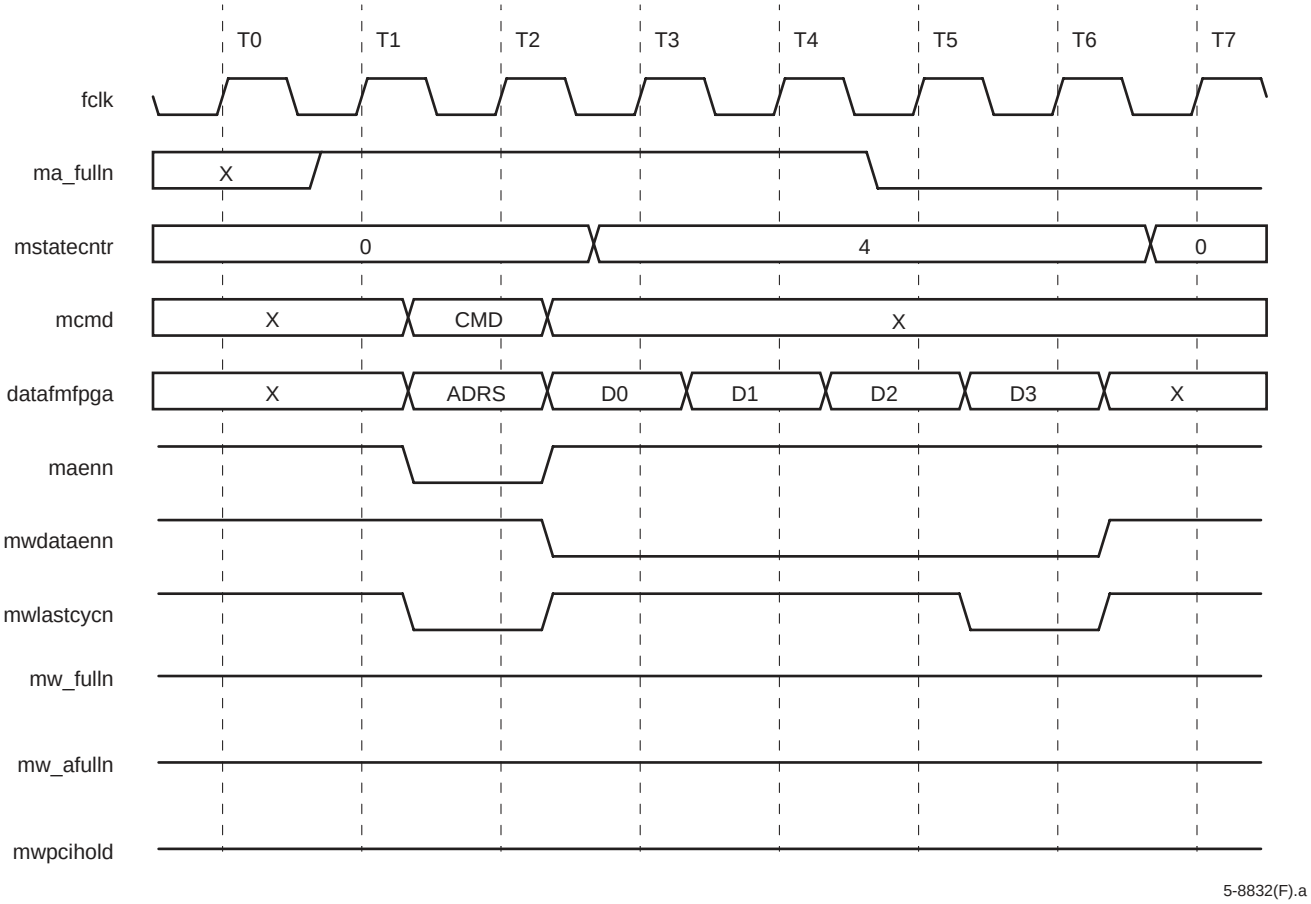


Figure 4. Master Write 32-Byte Burst (FPGA Bus, Dual-Port)

PCI Bus Core Detailed Description Dual Port (continued)

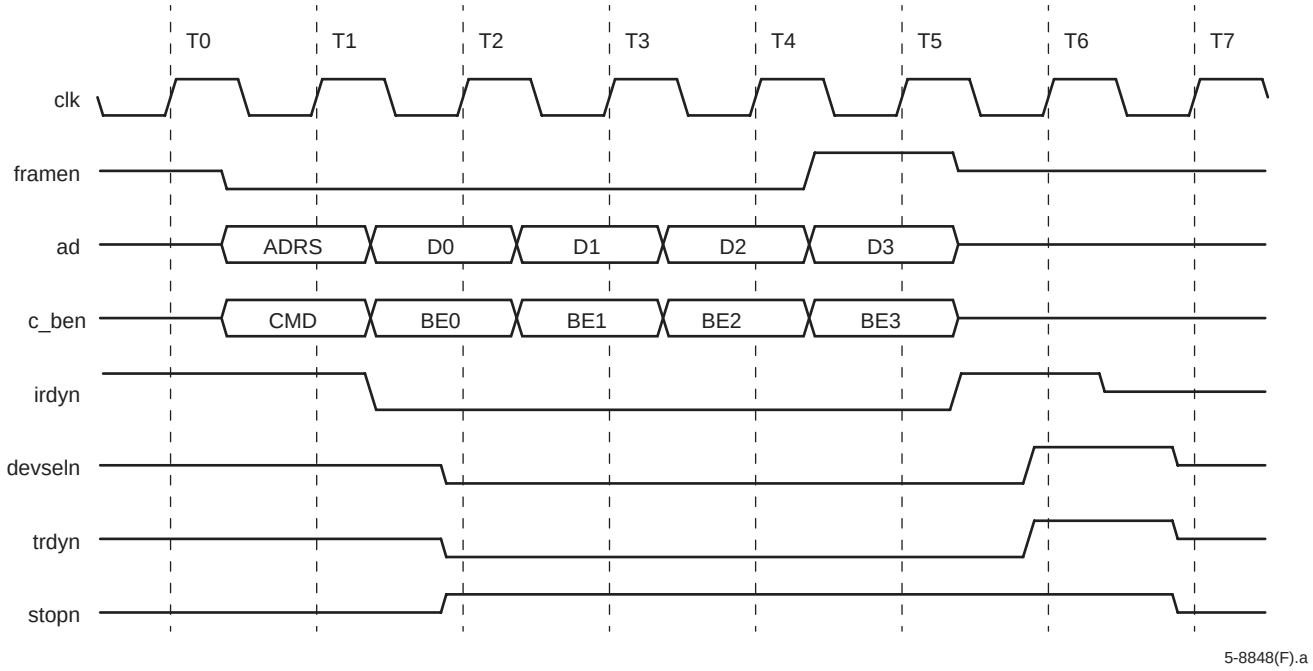


Figure 5. Master Write 32-Byte Burst (PCI Bus, 64-Bit)

Table 18. Dual-Port Master Write

mstatecntr	Next State of mstatecntr	Description	Bus	mwlastcycn	maenn	mwdataenn
0	0	Idle	—	1	1	1
0	4	Address[63:0]	datafmfpgax[7:0] datafmfpga[63:0]	0	0	1
4	4 or 0	Data[63:0], be[7:0]	datafmfpgax[7:0] datafmfpga[63:0]	0*	1	0

* **mwlastcycn** is only 0 during the last data Quadword sent.

PCI Bus Core Detailed Description Dual Port (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 19 through Table 20. The command word, burst length, 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 **datatofpga**.

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. During a master read burst transaction, the master read FIFO may go full, especially if the user side application is slow at unloading the FIFO. When this condition occurs, the master will insert wait-states continuously until another word is read from the FIFO, or the word count is exhausted. On the target side, if the target is ready to send more data, it will have **trdyn** asserted which will disable it from terminating the transaction as well. This can create a deadlock condition on the PCI bus. If the user application cannot unload any more data, and wishes to terminate the burst, additional FPGA logic must be incorporated to detect and accomplish the termination. Two operations must occur to terminate the current transaction. First, the **fpga_mstopburstn** signal must be asserted indicating to the core the master request to terminate. Second, one additional word of data must be read from the FIFO (only if the FIFO is full). The signal **fpga_mstopburstn** needs to stay asserted low until the **ma_fulln** flag is asserted low indicating that the transaction has been terminated and cleared.

Designing a Deadlock Timer

This design example is a method by which the user application can detect this condition and terminate the burst transaction. Since the **mr_fulln** and **fpga_mstopburstn** signals are on the **pciclk** clock domain, the deadlock counter will run on the **pciclk** clock. The **mr_fulln** signal is fed as a clock enable and a synchronous clear to a counter, driven by **pciclk**. The counter's length may be designed to guarantee a certain time-out latency on the PCI bus. When the FIFO is not full (**mr_fulln** = 1), the counter will stay cleared. When the FIFO has been full for an extended period of time, the counter will count and eventually overflow. This overflow indication can be used to set the **fpga_mstopburstn** signal indicating a request to stop the burst. The overflow signal is then detected and synchronized onto the fclk domain to be used to read one additional word from the FIFO. The transaction will complete, and the core will go back into an idle state.

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**). The burst length of 18 bits allows bursts of up to 2^{18}–1 quad words to be specified. To initiate a burst, the starting address must be aligned to a 64-byte boundary, and all of the byte enables must be enabled. If **ad[2]** is a 1, a single transfer will executed.

PCI Bus Core Detailed Description Dual Port (continued)

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 **trdyn** 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 one of the following occurs:

1. All data is received.
2. An abort occurs (either Master or Target).
3. The **fpga_mstopburstn** signal is asserted.
4. The PCI bus' reset signal (**rstn**) 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 **mfifoclrn** to place the core's master logic in a known state. Normally, the clear signal will not be used unless a severe problem has occurred in the data flow. The **mfifoclrn** signal is synchronous with **fclk** and must be asserted for a minimum of three clock periods. During reset, the **m_ready** signal will go low. After the reset signal is deasserted high, **m_ready** will continue to be low for 8—10 clock periods. The FPGA application should not continue normal operation until **m_ready** is asserted high.

Understanding and Using the pci_mcfg_stat Status Signals

On the Master interface, there are two signals that control and provide status to the FPGA application. The signal **pci_mcfg_stat** provides the status, and **mcfgshiftenn** controls what information the status line provides. The **pci_mcfg_stat** signal is always active and duplicates the status contained in configuration status register at location offset 0x04, bits 24, 28, and 29. To use this status output, the FPGA application must keep **mcfgshiftenn** = 1. When high, **pci_mcfg_stat** provides the wired-OR of the three status lines. If **pci_mcfg_stat** gets set to a 1, indicating an error, then the FPGA application may set **mcfgshiftenn** = 0 to determine individual status. Once low, the **pci_mcfg_stat** signal will output data parity error detected on the first clock, target abort received on the second clock, and master abort received on the third clock.

PCI Bus Core Detailed Description Dual Port (continued)

Master Read, Nonburst Transaction

Figure 6 (FPGA bus) and Figure 7 (PCI bus) show the timing of a single Quadword Master read. In Figure 6, 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 7. Once data is transferred on the PCI bus and **mr_emptyn** is deasserted high, the FPGA application asserts **mrdataenn** and one Quadword of data is transferred on bus **datatofpga**.

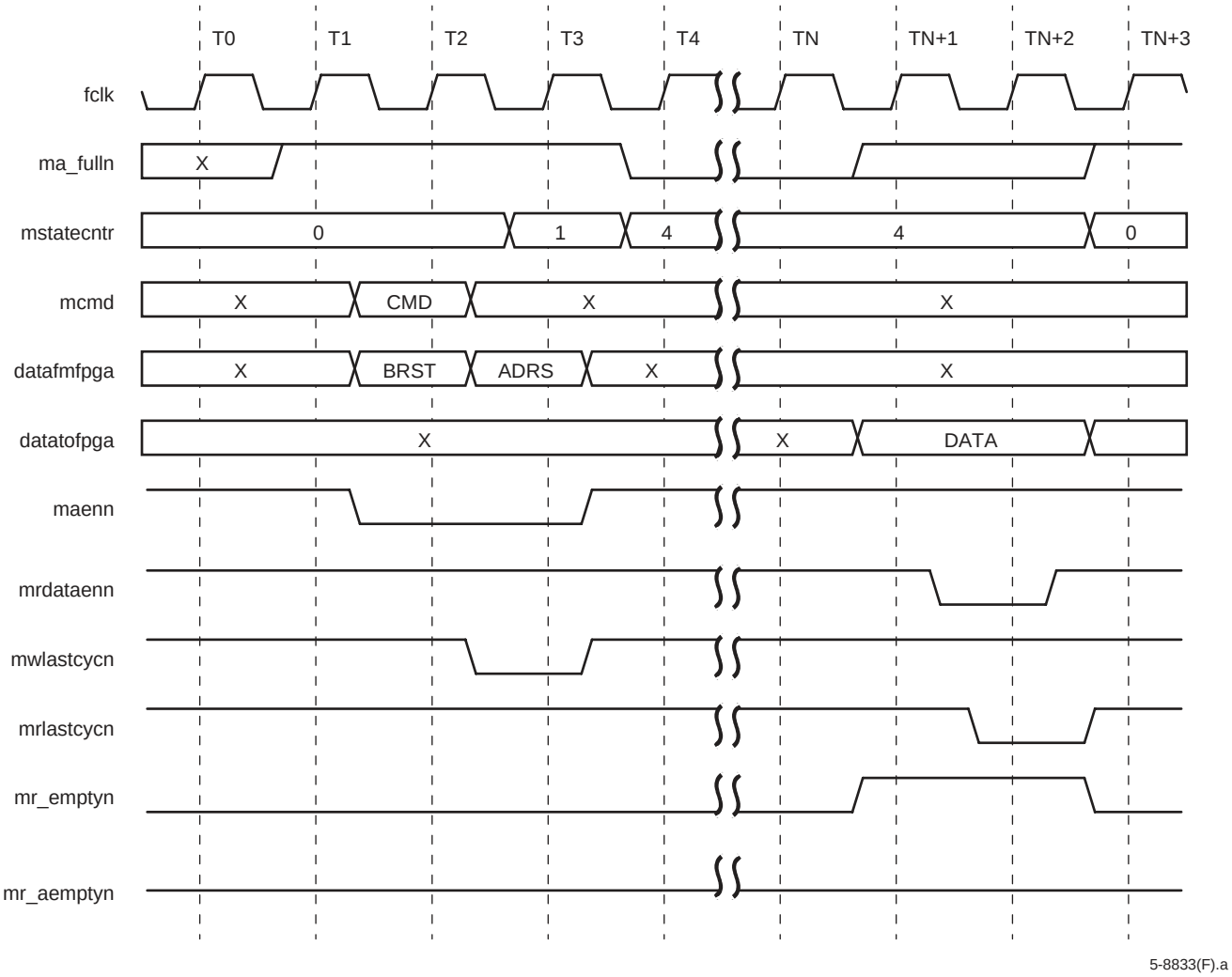


Figure 6. Master Read Single (FPGA Bus, Dual-Port, Specified Burst Length, 64-Bit Address)

PCI Bus Core Detailed Description Dual Port (continued)

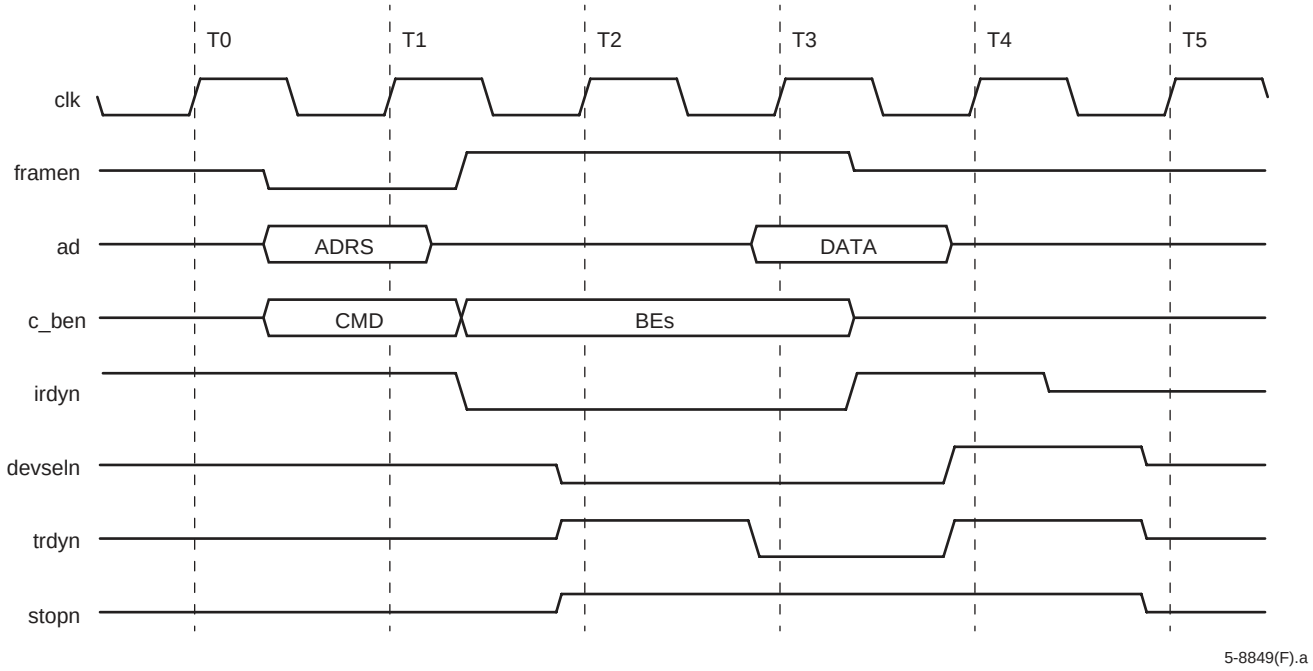


Figure 7. Master Read Single (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

Master Read, Burst Transaction

Figure 8 (FPGA bus dual port) and Figure 9 (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 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 transferred on the PCI bus and **mr_emptyn** is deasserted high, the FPGA application asserts **mrdataenn** and four Quadwords of data are transferred on bus **datatofpga**.

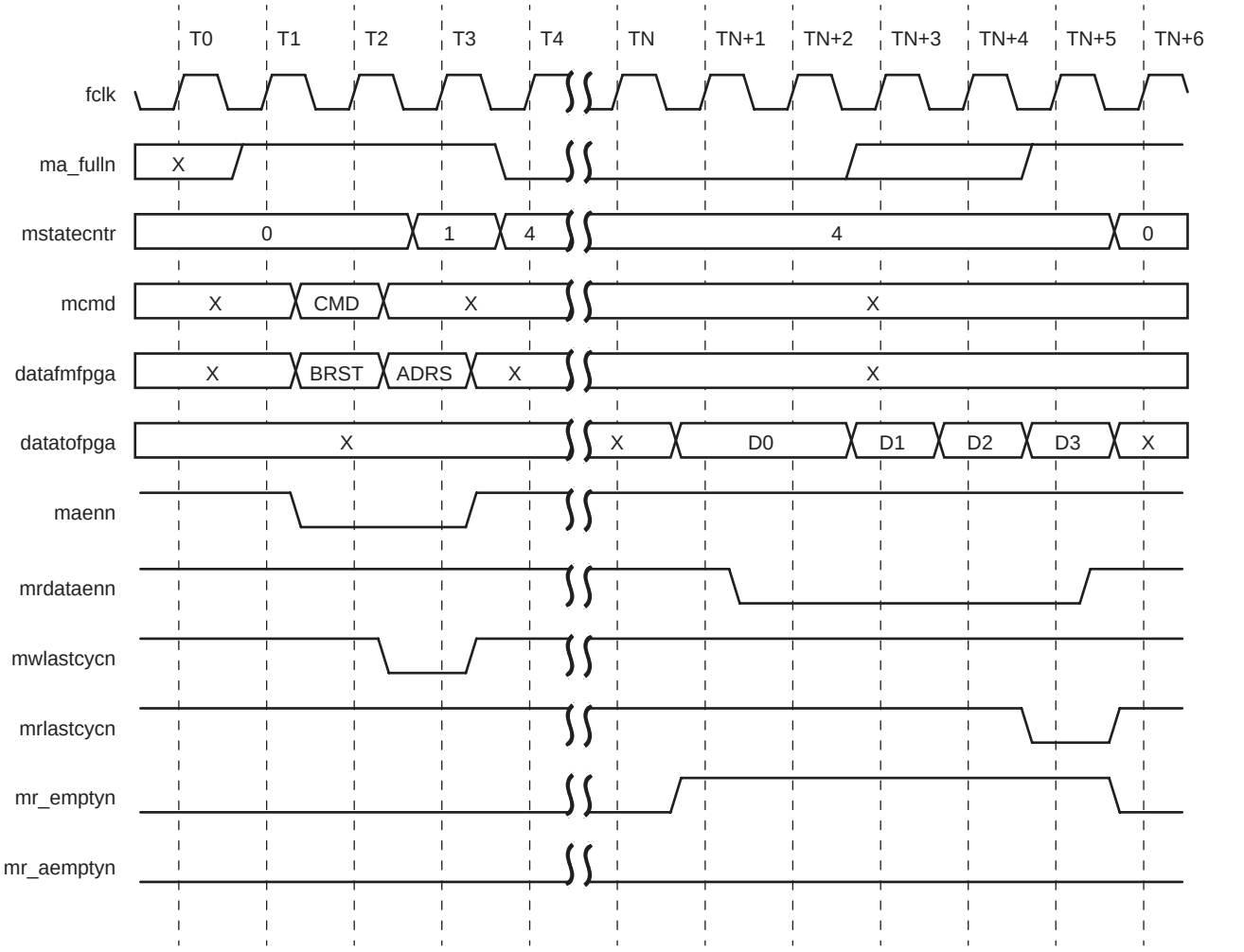


Figure 8. Master Read 32-Byte Burst (FPGA Bus, Dual-Port, Burst Length, and 64-Bit Address)

PCI Bus Core Detailed Description Dual Port (continued)

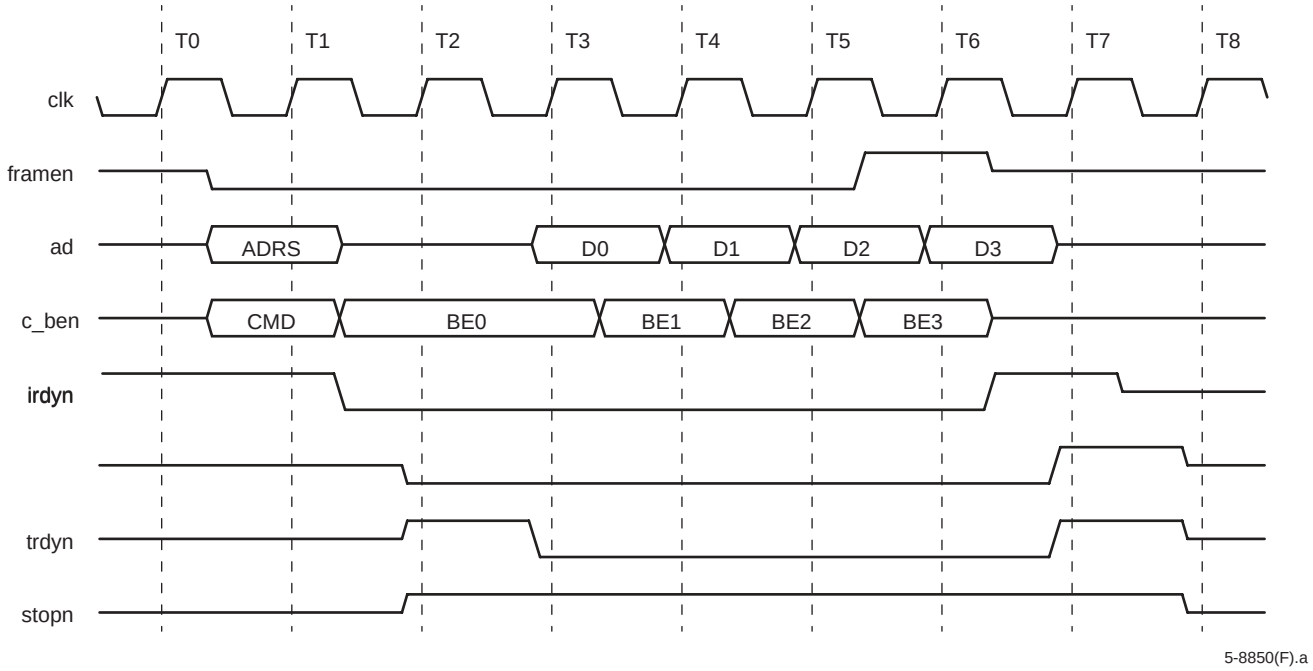


Figure 9. Master Read 32-Byte Burst (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

Table 19. Dual-Port Master Read, 64-Bit Address Supplied

mstatecntr	Next State of mstatecntr	Description	Bus	maenn	mwlastcycn	mrlastcycn	mrdataenn
0	0	Idle	—	1	1	1	1
0	1	BE[7:0], Burst Length	datafmfpgax[7:0] datafmfpga[63:0]	0	1	1	1
1	4	Address[63:0]	datafmfpga[63:0]	0	0	1	1
4	4 or 0	Data[63:0]	datatofpga[63:0]	1	1	0*	0

* **mrlastcycn** is 0 during the last Quadword transferred.

Table 20. Dual-Port Master Read, 32-Bit Address Supplied

mstatecntr	Next State of mstatecntr	Description	Bus	maenn	mwlastcycn	mrlastcycn	mrdataenn
0	0	Idle	—	1	1	1	1
0	4	BE[7:0], Burst Length, Address[31:0]	datafmfpgax[7:0] datafmfpga[63:0]	0	0	1	1
4	4 or 0	Data[63:0]	datatofpga[63:0]	1	1	0*	0

* **mrlastcycn** is 0 during the last Quadword transferred.

PCI Bus Core Detailed Description Dual Port (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 describes 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[2:0]**, 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 **datatofpga** 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 these state counters in this case.

The data required from the FPGA application by the PCI core on bus **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.

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 **datatofpga**.

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. If the FIFO does go full, the core will do one of two things. If **twburstpendn** is deasserted high, the target will disconnect. If **twburstpendn** is asserted low, the target will assert up to eight wait-states and then disconnect if still full. The FIFO full flag is not generally used in user designs. If it is, however, keep in mind that it is synchronous to **pciclk**.

Bursting

Signal **twlastcycn** tells the FPGA application whether the current write is a burst. The FPGA application continues to unload data from the FIFO as long as **twlastcycn** is inactive. The bursting will continue until either **twlastcycn** is received, the FIFO becomes full, or the BAR boundary is crossed. There is no fixed maximum transfer word count.

PCI Bus Core Detailed Description Dual Port (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 **devseln** and **trdyn** 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**. The signal **deltrn** was designed to be a static signal. This signal should be tied off high or low depending upon whether the FPGA application wishes to run delayed transactions. When asserting **deltrn** low, the PCI core will execute delayed transactions for I/O writes as well as all target reads. In delayed transaction mode, 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, 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. Internal to the ASIC, there is also a 15-bit time-out timer (known as the discard timer). During a delayed I/O write transaction, this counter will begin counting. If the same master does not come back within $2^{15} - 1$ **pciclk**'s to complete the write, this timer will expire, resetting the target state machines and setting a user side signal (**disctimerexp** = 1). From this point forward, any master performing a write (including the original master coming back to complete the transfer) will be treated as a new transaction. If monitoring this signal, keep in mind that **disctimerexp** is synchronous to **pciclk** and asserts high for one clock period.

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**.

Reset

The FPGA application can apply the PCI core's reset signal **tfifoclrn** to place the core's target logic in a known state. Normally, the clear signal will not be used unless a severe problem has occurred in the data flow. The **tfifoclrn** signal is synchronous with **fclk** and must be asserted for a minimum of three clock periods. During reset, the **t_ready** signal will go low. After the reset signal is deasserted high, **t_ready** will continue to be low for 8—10 clock periods. The FPGA application should not continue normal operation until **t_ready** is asserted high.

PCI Bus Core Detailed Description Dual Port (continued)

Understanding and Using the pci_tcfg_stat Status Signals

On the Target interface, there are two signals that control and provide status to the FPGA application. The signal **pci_tcfg_stat** provides the status and **tcfgshiftenn** controls what information the status line provides. The **pci_tcfg_stat** signal is always active and duplicates the status contained in configuration status register at location offset 0x04, bits 24, 28, and 29. To use this status output, the FPGA application must keep **tcfgshiftenn** = 1. When high, **pci_tcfg_stat** provides the wired-OR of the three status lines. If **pci_tcfg_stat** gets set to a 1, indicating an error, then the FPGA application may set **tcfgshiftenn** = 0 to determine individual status. Once low, the **pci_tcfg_stat** signal will output target abort signaled on the first clock, system error signaled on the second clock, and parity error detected on the third clock.

Initiating Target Aborts

There may be a need in an application to initiate a target abort condition on the PCI bus. In general, this is asserted for only the most severe cases. The interface signal, **fpga_tabort**, is used for this purpose. From the PCI core's point of view, it needs to know whether to perform a target abort at the very beginning of a transaction, so it is not possible to have a transaction started, and then assert the **fpga_tabort** signal. The signal **fpga_tabort** needs to be asserted before the transaction begins, and it was not designed to be toggled on and off from transaction to transaction. Once an FPGA application determines that it wants to apply a target abort to any master that accesses it, it would assert the **fpga_tabort** signal high. All future target accesses will be terminated in an abort. In generating this signal, keep in mind that this signal needs to be synchronous to **pciclk**.

Initiating PCI Target Retries

In contrast to target abort, many applications may require to assert PCI target retries. In general, this may be asserted for times when the FPGA application is temporarily busy and unavailable to service PCI requests. The interface signal, **fpga_tretryn**, is used for this purpose. From the PCI core's point of view, it needs to know whether to perform a target retry at the very beginning of a transaction, so it is not possible to have a transaction started and then assert the **fpga_tretryn** signal. The signal **fpga_tretryn** needs to be asserted before the transaction begins, and it was not designed to be toggled on and off from transaction to transaction. Once an FPGA application determines that it wants to apply a target retry to any master that accesses it, it would assert the **fpga_tretryn** signal low. All future target accesses will be terminated in a retry (disconnect without data). On the FPGA application side, no activity will occur. In generating this signal, keep in mind that this signal needs to be synchronous to **pciclk**.

PCI Bus Core Detailed Description Dual Port (continued)

Target Write to Configuration Space Transaction

Figure 10 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 **trdyn** one cycle after **devseln** is asserted.

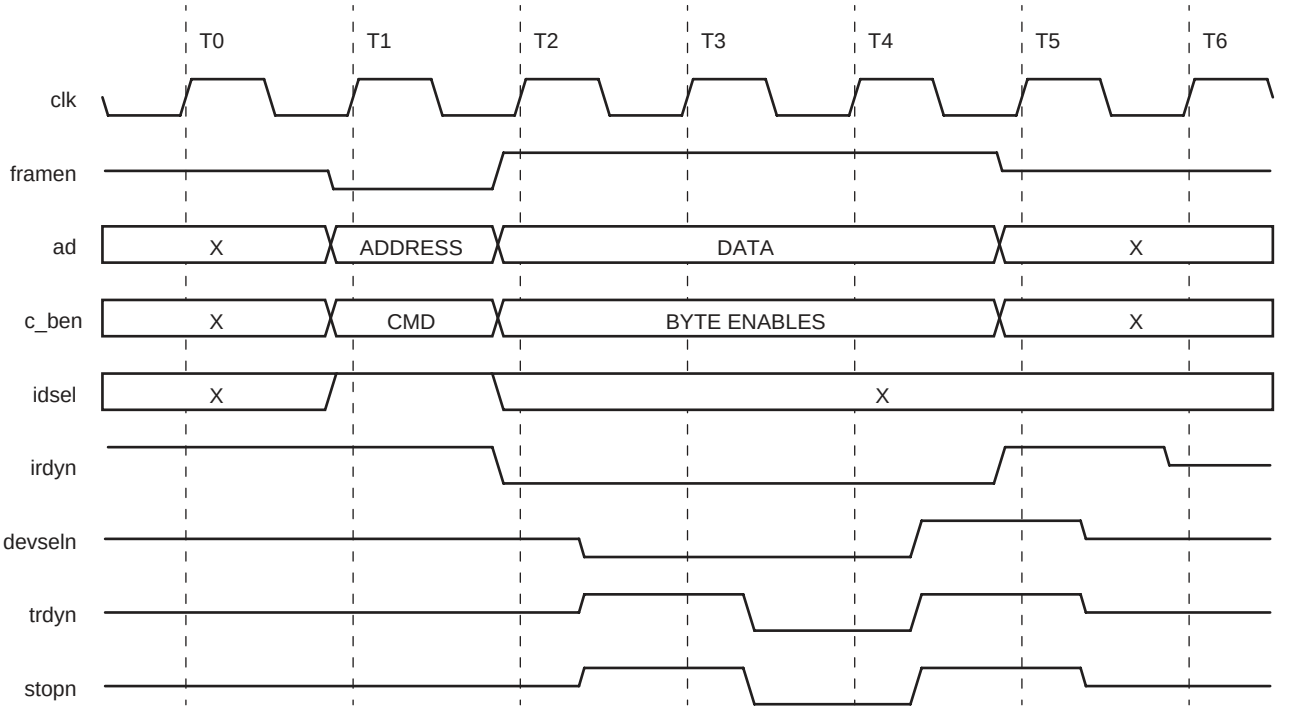


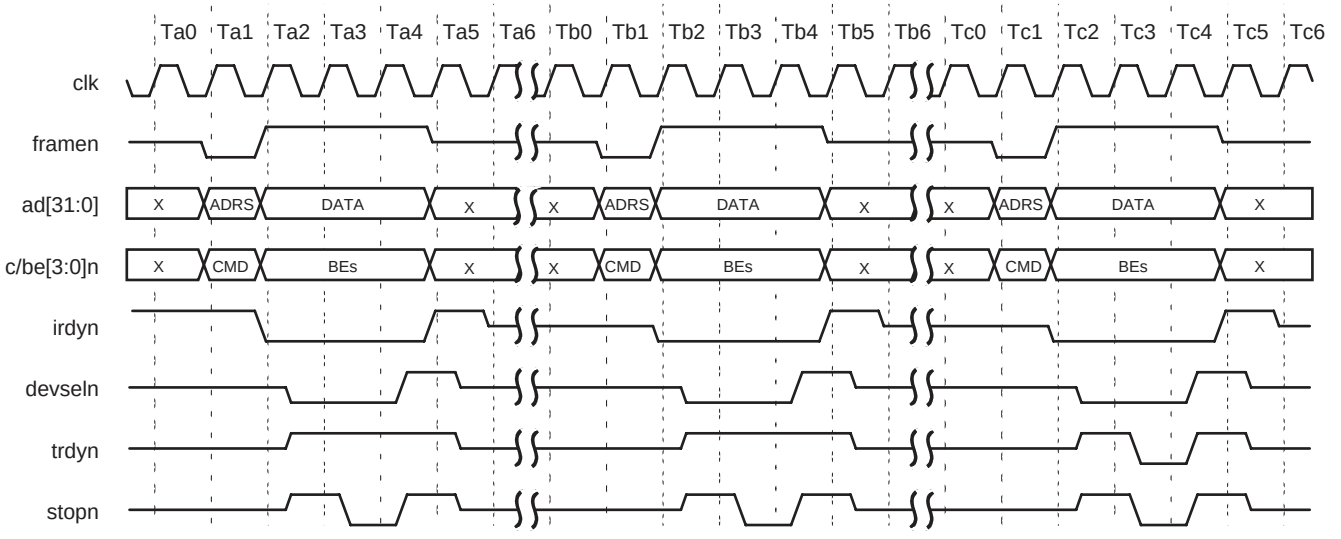
Figure 10. Target Configuration Write (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

Target Write I/O, Delayed Transaction

Figure 11 (PCI bus) and Figure 13 (FPGA bus) show the timing 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 11, 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 13. 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 11. 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 **trdyn** accompanies **stopn** to indicate the disconnect with data.

The timing on the FPGA interface (Figure 13) 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 (**twdataenn**), and the receiving of the data on bus **datatofpga**. 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.



5-7372(F).a

Figure 11. Target I/O Write, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

Target Write Nonburst Transaction

Figure 12 (PCI bus) and Figure 13 (FPGA bus) show the timing on the PCI and FPGA interfaces, respectively, for a Target memory nonburst write transaction. The timing on the PCI interface (Figure 12) is similar to that of an I/O write except that, since bursts to memory space are allowed, the signal **stopn** is not asserted. The FPGA interface timing is as shown in Figure 13, and is the same as the timing for memory and I/O write transactions.

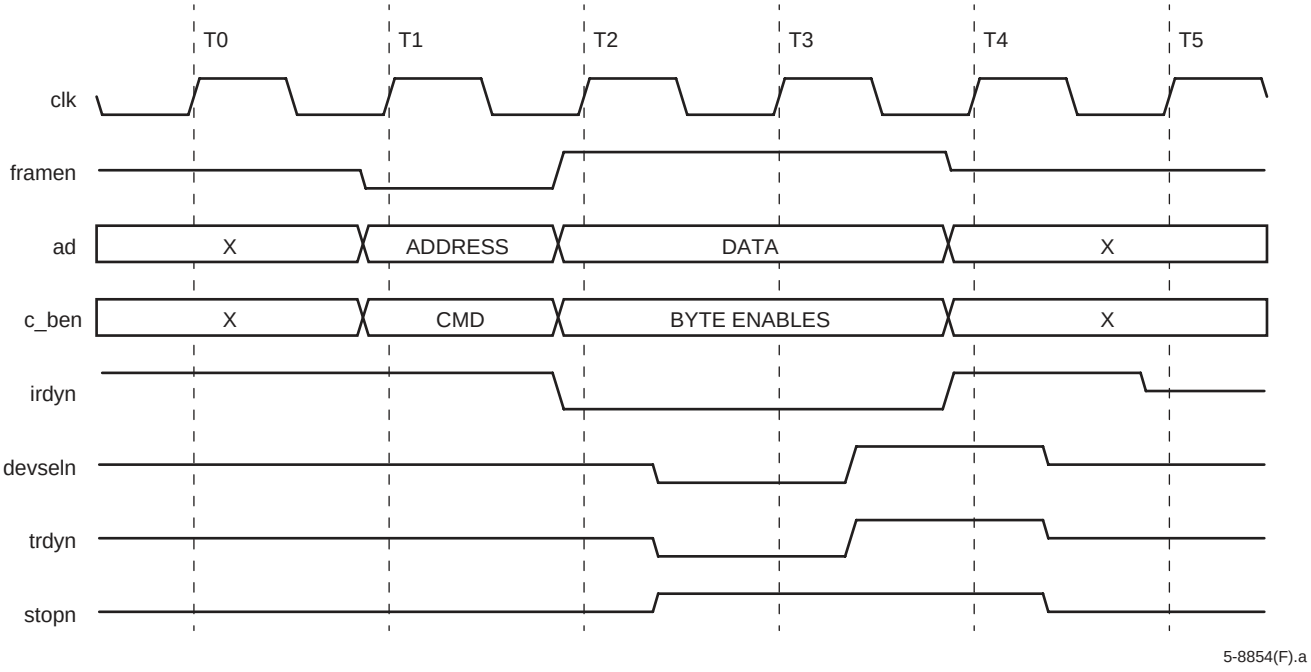


Figure 12. Target Write Memory Single (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

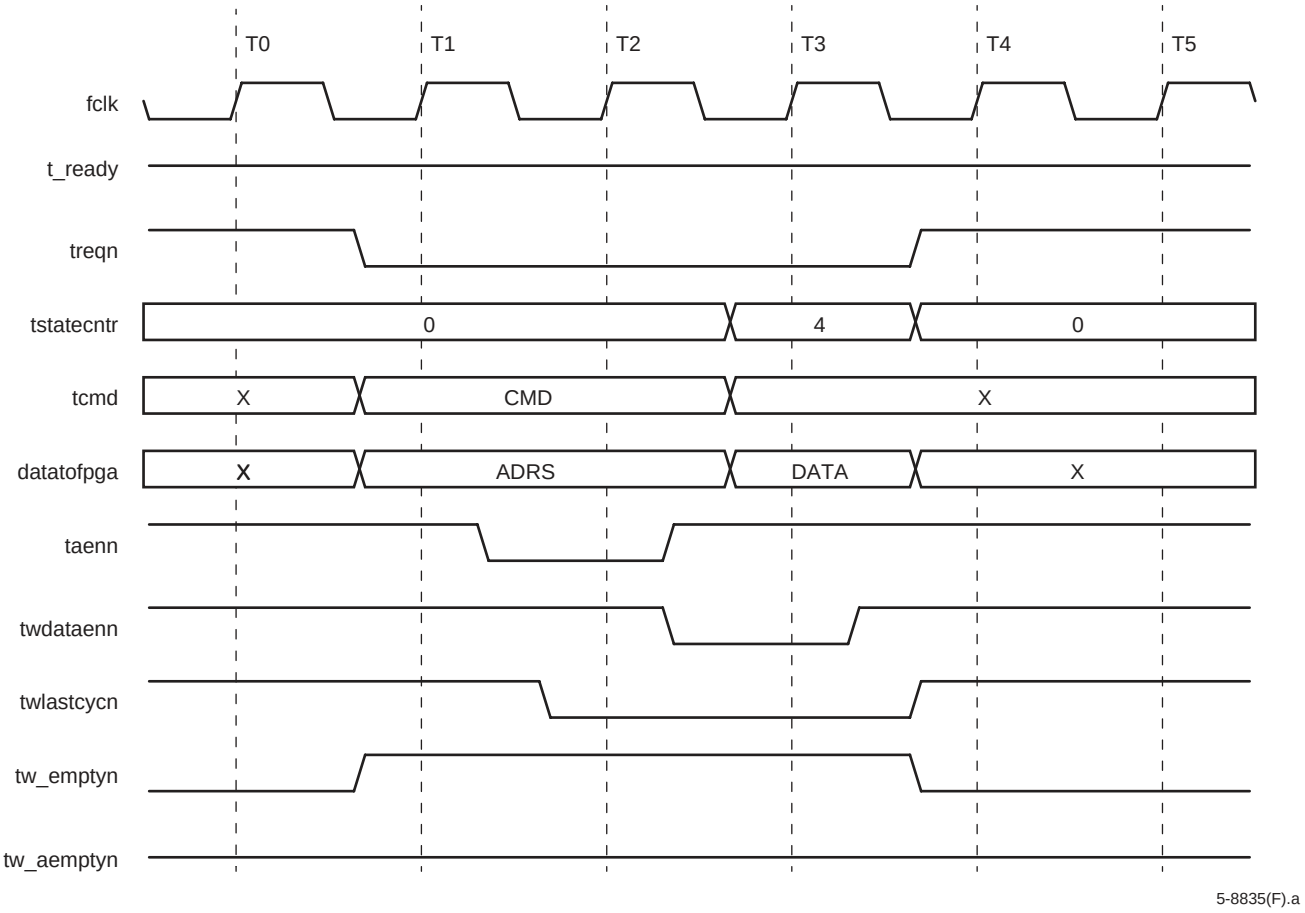


Figure 13. Target Write Single (FPGA Bus, Dual-Port)

PCI Bus Core Detailed Description Dual Port (continued)

Target Write Memory Burst Transaction

Figure 14 (PCI bus) and Figure 15 (FPGA bus) show the timing for a Target memory write burst of four Quadwords. The timing on the PCI interface (Figure 14) is typical for a medium-speed decode Target. Note that **trdyn** is asserted at the earliest possible time, which is concurrent with assertion of **devseln**. 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 15) 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.

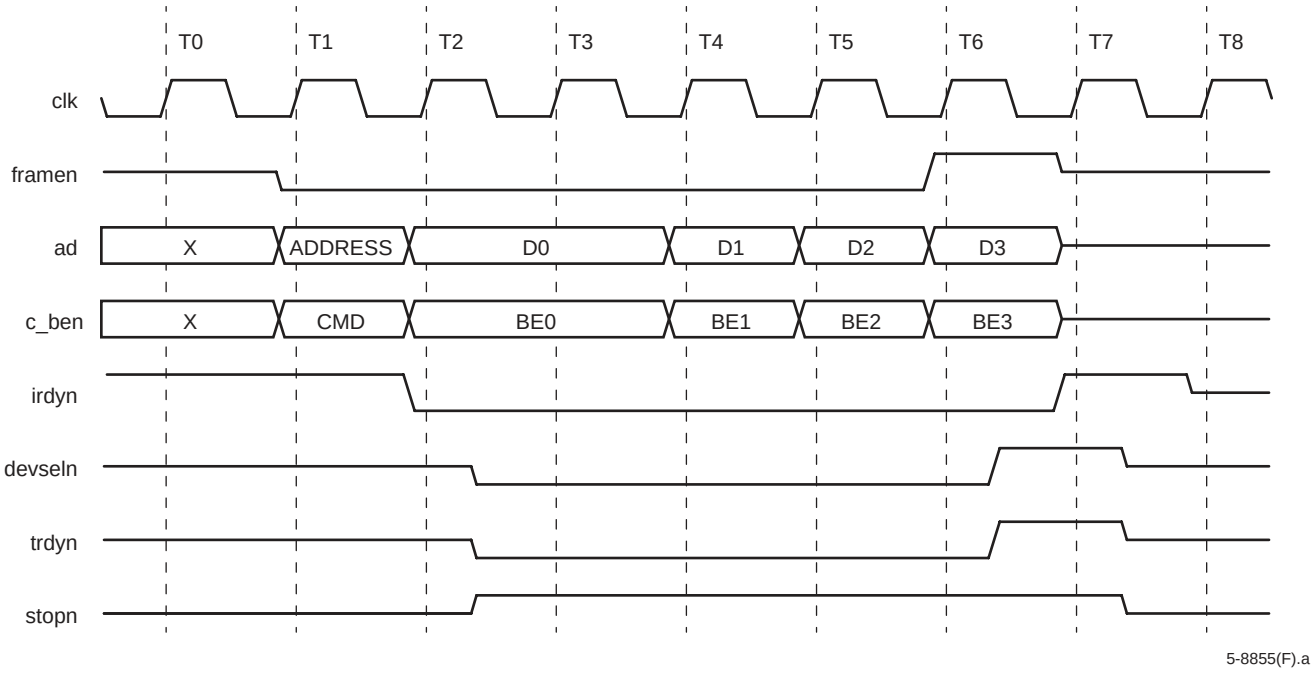


Figure 14. Target Memory Write 32-Byte Burst (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

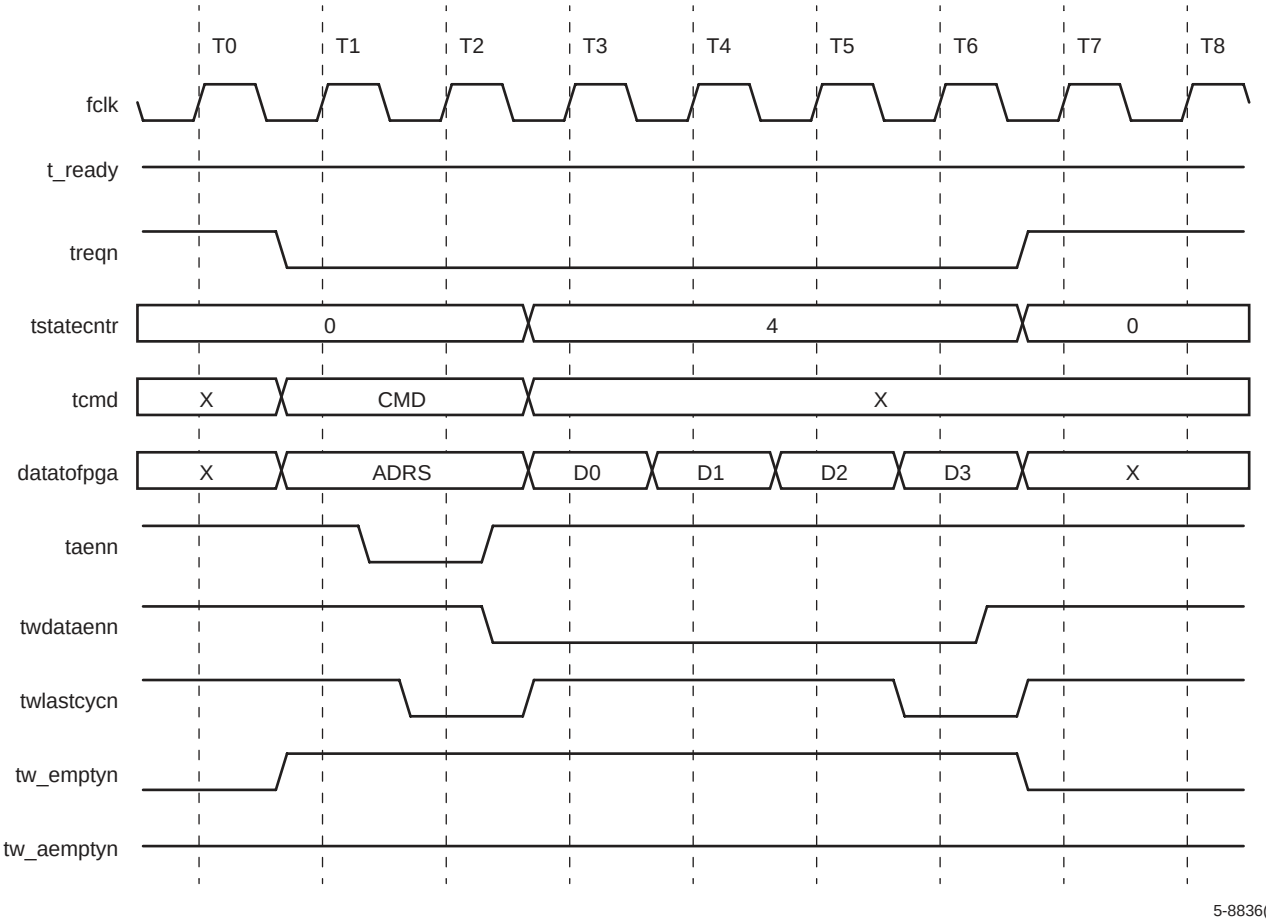


Figure 15. Target Write Memory 32-Byte Burst (FPGA Bus, Dual-Port)

Table 21. Dual-Port Target Write

tstatecntr	Next State of tstatecntr	Description	Bus	treqn	twlastcycn	taenn
0	0	Idle	—	1	1	1
0	4	Address[63:0]	datatofpgax[7:0] datatofpga[63:0]	0	0	0
4	4 or 0	Data[63:0], BE[7:0]	datatofpgax[7:0] datatofpga[63:0]	1*	0†	1

* **treqn** is deasserted high on the last data Quadword.
† **twlastcycn** is asserted low on the last data Quadword.

PCI Bus Core Detailed Description Dual Port (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 22 describes the specific order of operation for a Target read transaction.

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 **datafmfpga**. 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. This is explained in the Delayed Transactions section.

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. If the FIFO does go empty, the core will do one of two things. If **twburstpendn** is deasserted high, the target will disconnect. If **twburstpendn** is asserted low, the target will assert up to eight wait-states and then disconnect if still empty. The FIFO empty flag is not generally used in user designs. If it is, however, keep in mind that it is synchronous to **pciclk**.

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. Bursts will continue until either **trlastcycn** is received, the BAR boundary is crossed, or a 2^{18} physical page address is crossed.

PCI Bus Core Detailed Description Dual Port (continued)

Delayed Transactions

Delayed transactions can be executed by assesting **deltrn** low. 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.

Delayed transactions are very similar to a target retry except that the address is actually stored in the core. Delayed transactions are usually implemented in systems where the user side interface cannot supply the first piece of data in 16 clock cycles. An example of this may be that the user interface is connected to another bus system. On a PCI target read, the user interface must arbitrate for the user bus and get the necessary data. Delayed transaction mode is used when the **deltrn** bit is asserted low. This bit is not a dynamic bit. It must be set ahead of a transaction occurring. It is not recommended to switch between delayed and non-delayed transactions dynamically.

When **deltrn** is low, a master read request is terminated in a target retry. On the user interface side, the address is stored in the target address FIFO, and **treqn** is asserted low. All future master requests are terminated in a retry until the address is read out of the FIFO, data is loaded into the FIFO, and the same request comes back to complete the transaction. In generating this signal, keep in mind that this signal needs to be synchronous to **pciclk**.

Another option the designer has using delayed transactions is to use the signal **trpcihold**. The signal **trpcihold** should be used when the user side interface is slow loading requested data, and the designer wishes to utilize the PCI in the most efficient manner. Without this signal, an external master will request data and hold onto the PCI bus until either it has received it or it gets terminated by latency timers, etc. A more efficient method to utilize the PCI bus is to assert **trpcihold**, load the FIFOs, and then deassert it. While the **trpcihold** signal is asserted, the core thinks that the FIFOs stay empty even though they are slowly filling with data. Requests from an external master are terminated in retries. When the **trpcihold** signal is deasserted (or the FIFO becomes full), the core will allow an external master to come in, the data will be burst across the PCI bus as fast as the master will allow, and the transaction will end. In generating **trpcihold**, keep in mind that this signal needs to be synchronous to **pciclk**.

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**. When **treqn** deasserts, the FPGA application must immediately deassert **trdataenn**.

PCI Bus Core Detailed Description Dual Port (continued)

Reset

The FPGA application can apply the PCI core's reset signal **tfifoclrn** to place the core's target logic in a known state. Normally, the clear signal will not be used unless a severe problem has occurred in the data flow. The **tfifoclrn** signal is synchronous with **fclk** and must be asserted for a minimum of three clock periods. During reset, the **t_ready** signal will go low. After the reset signal is deasserted high, **t_ready** will continue to be low for 8—10 clock periods. The FPGA application should not continue normal operation until **t_ready** is asserted high.

Understanding and Using the pci_tcfg_stat Status Signals

On the Target interface, there are two signals that control and provide status to the FPGA application. The signal **pci_tcfg_stat** provides the status, and **tcfgshiftenn** controls what information the status line provides. The **pci_tcfg_stat** signal is always active and duplicates the status contained in configuration status register at location offset 0x04, bits 24, 28, and 29. To use this status output, the FPGA application must keep **tcfgshiftenn** = 1. When high, **pci_tcfg_stat** provides the wired-OR of the three status lines. If **pci_tcfg_stat** gets set to a 1, indicating an error, then the FPGA application may set **tcfgshiftenn** = 0 to determine individual status. Once low, the **pci_tcfg_stat** signal will output target abort signaled on the first clock, system error signaled on the second clock, and parity error detected on the third clock.

Initiating Target Aborts

There may be a need in an application to initiate a target abort condition on the PCI bus. In general, this is asserted for only the most severe cases. The interface signal, **fpga_tabort**, is used for this purpose. From the PCI core's point of view, it needs to know whether to perform a target abort at the very beginning of a transaction, so it is not possible to have a transaction started, and then assert the **fpga_tabort** signal. The signal **fpga_tabort** needs to be asserted before the transaction begins, and it was designed to be toggled on and off from transaction to transaction. Once an FPGA application determines that it wants to apply a target abort to any master that accesses it, it would assert the **fpga_tabort** signal high. All future target accesses will be terminated in an abort. In generating this signal, keep in mind that this signal needs to be synchronous to **pciclk**.

PCI Bus Core Detailed Description Dual Port (continued)

Target Read from Configuration Space

Figure 16 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 **trdyn** one cycle after **devseln** is asserted.

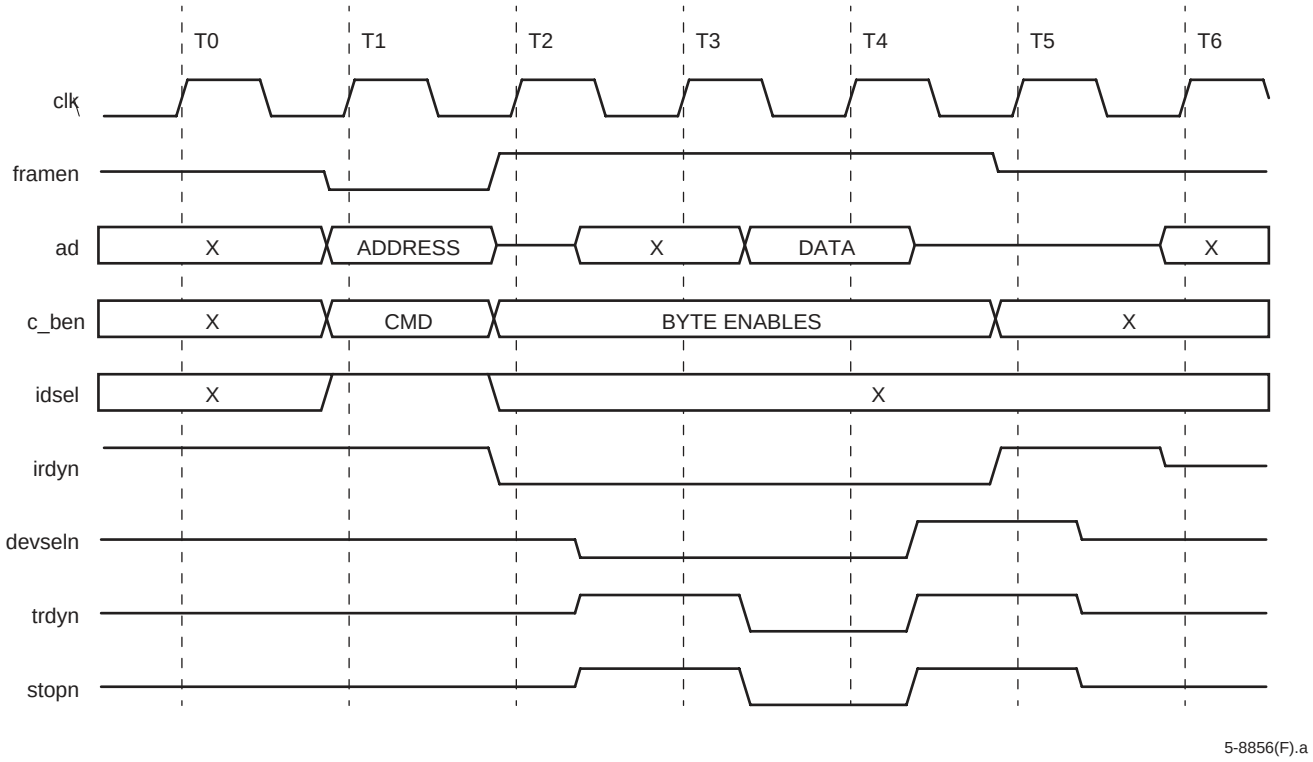


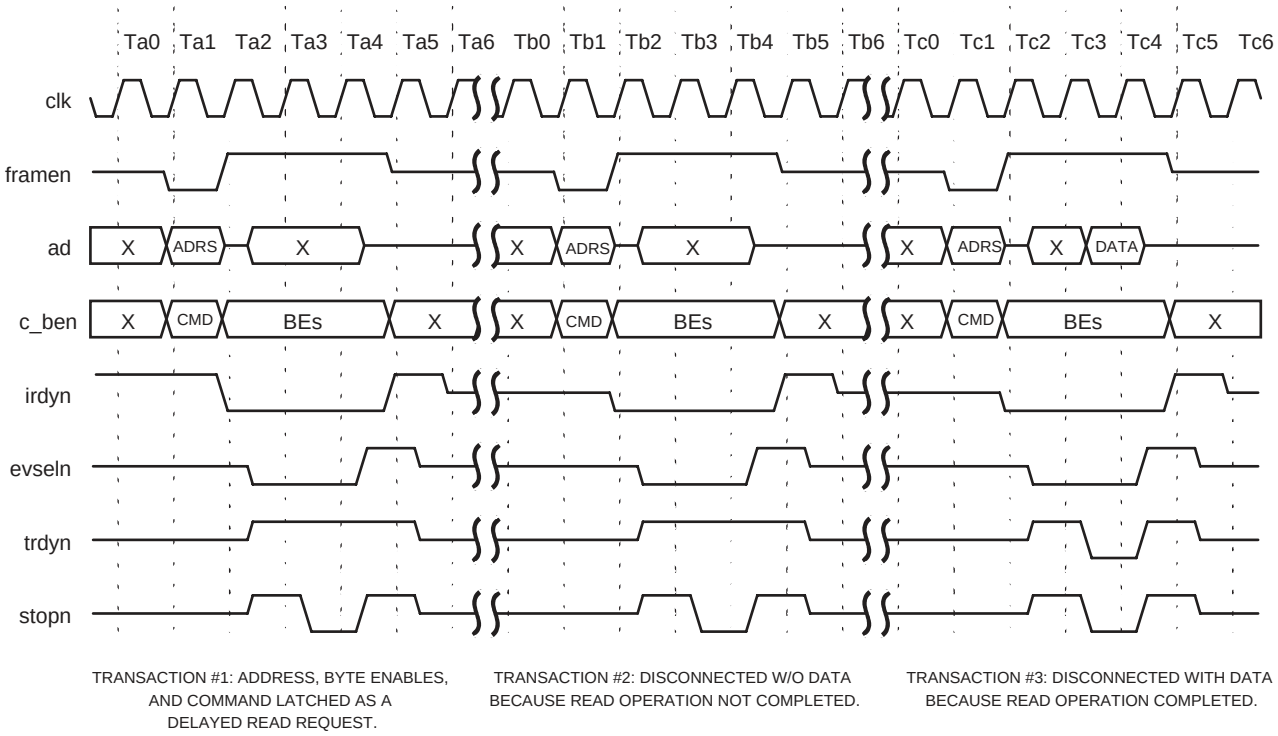
Figure 16. Target Configuration Read (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

Target Read I/O, Delayed Transaction

Figure 17 (PCI bus) and Figure 20 (FPGA bus) 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 17, 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 20. All subsequent read or write requests to memory or I/O space will result in retries, as shown in the second transaction of Figure 17. 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 **trdyn** accompanies **stopn** to indicate the disconnect with data.

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 **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 nonburst transaction; therefore, Target read burst (**trlastcycn**) is kept asserted.



5-8858(F).a

Figure 17. Target I/O Read, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

Target Read I/O, No Delayed Transaction

Figure 18 (PCI bus) and Figure 20 (FPGA bus) 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 18 is identical to the timing of the third (final) transaction of Target I/O read, delayed transaction (Figure 17), which shows a Target I/O read with delayed transaction. Also, the FPGA interface timing is as shown in Figure 20, regardless of whether delayed transactions are enabled.

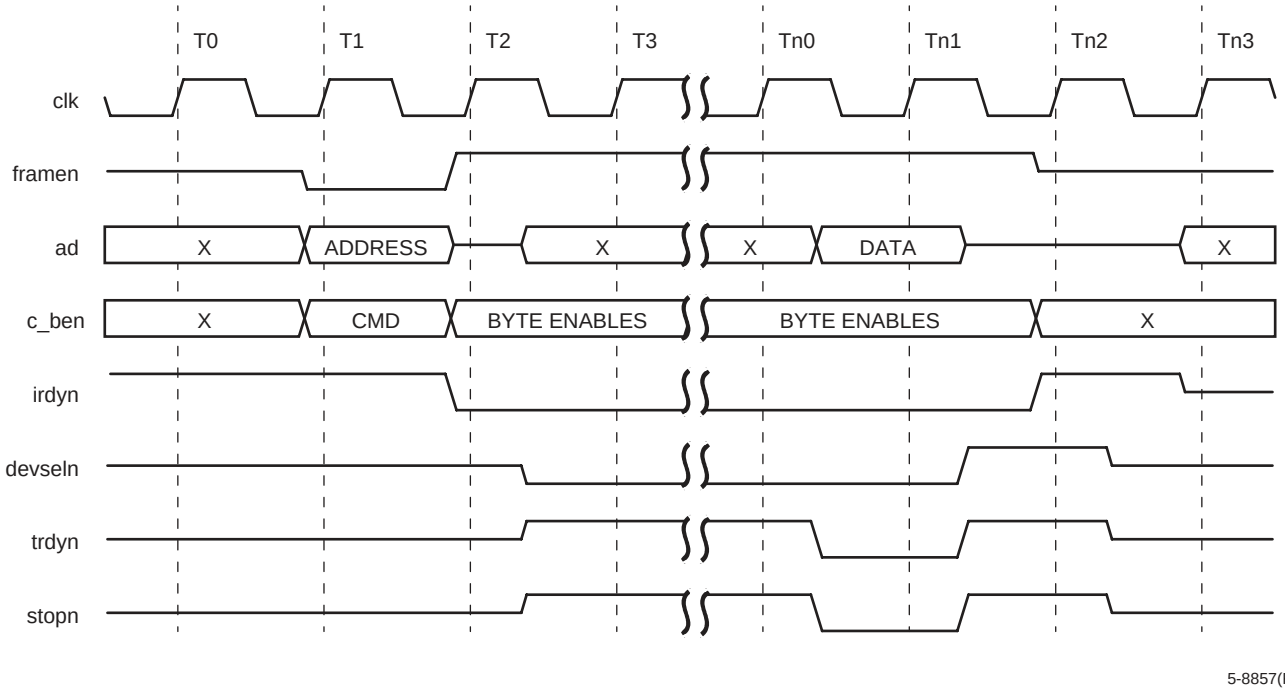


Figure 18. Target I/O Read, Not Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

Target Read Memory, Nonburst, Delayed Transaction

Figure 19 (PCI bus) and Figure 20 (FPGA bus) 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 19) is similar to that of an I/O read (Figure 17) 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 20 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.

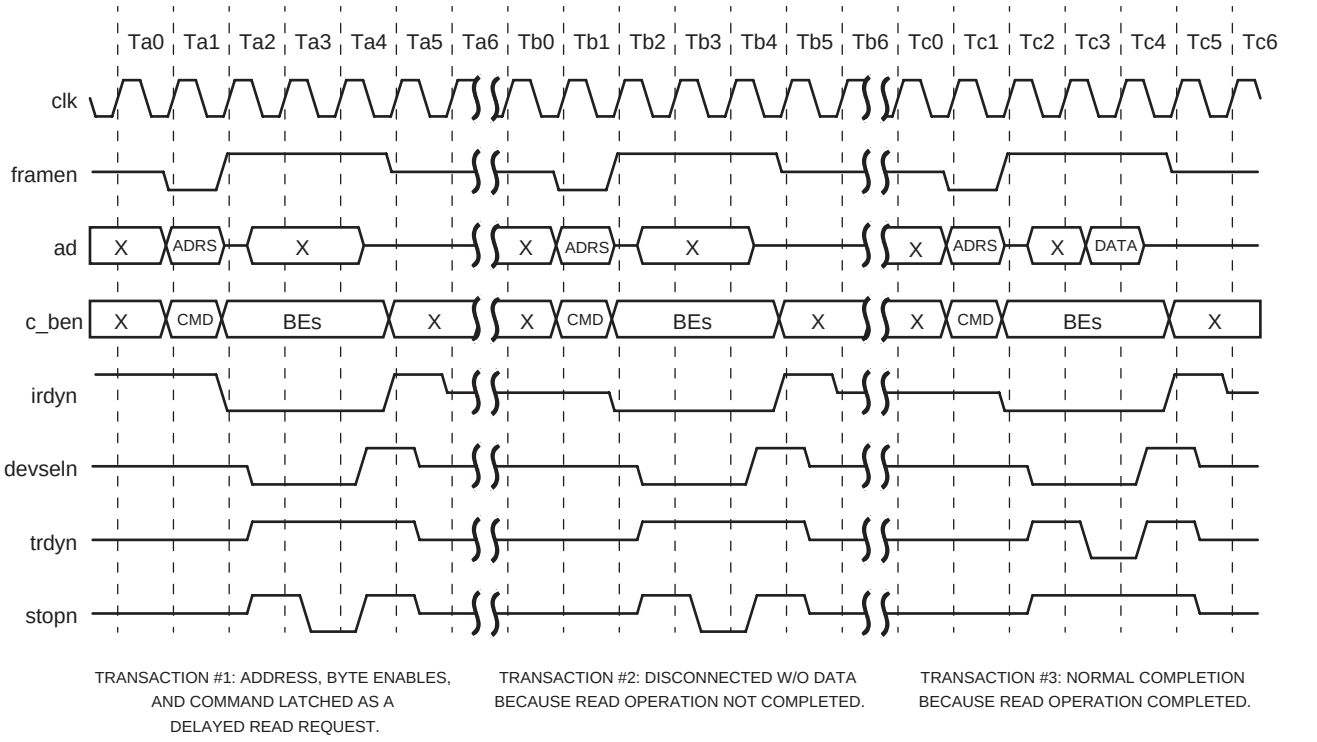


Figure 19. Target Memory Single Read, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

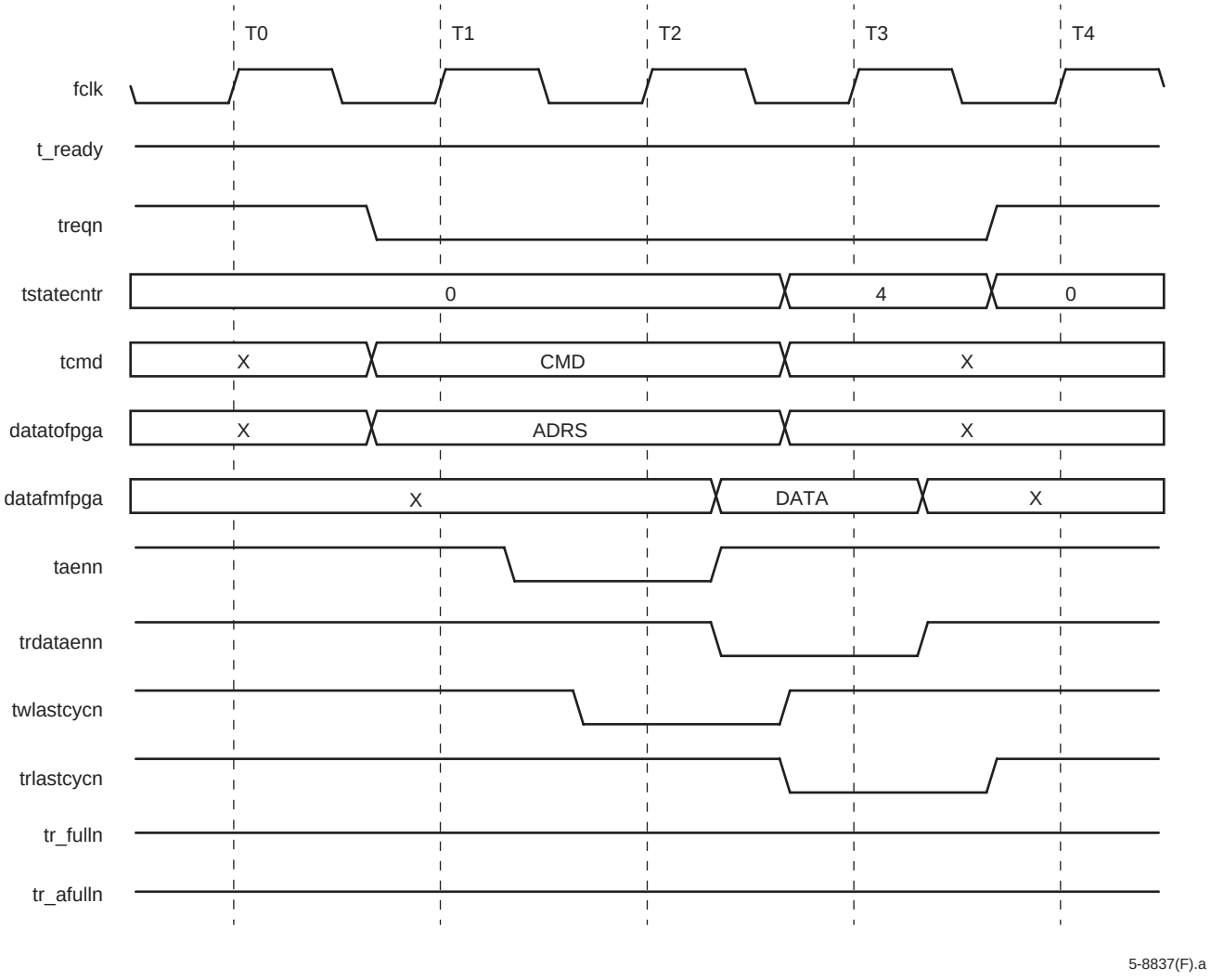


Figure 20. Target Read Single (FPGA Bus, Dual-Port)

PCI Bus Core Detailed Description Dual Port (continued)

Target Read Memory, Nonburst, No Delayed Transaction

Figure 21 (PCI bus) and Figure 20 (FPGA bus) 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 21. Here the PCI core accepts the transaction without issuing a retry but does not immediately assert **trdyn**. Wait-states are inserted until the requested data is placed in the Target read FIFO, at which time **trdyn** 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 20, and is the same as the timing in the accesses of Target I/O read, delayed transaction, Target I/O read, no delayed transaction, and Target read memory nonburst, delayed transaction.

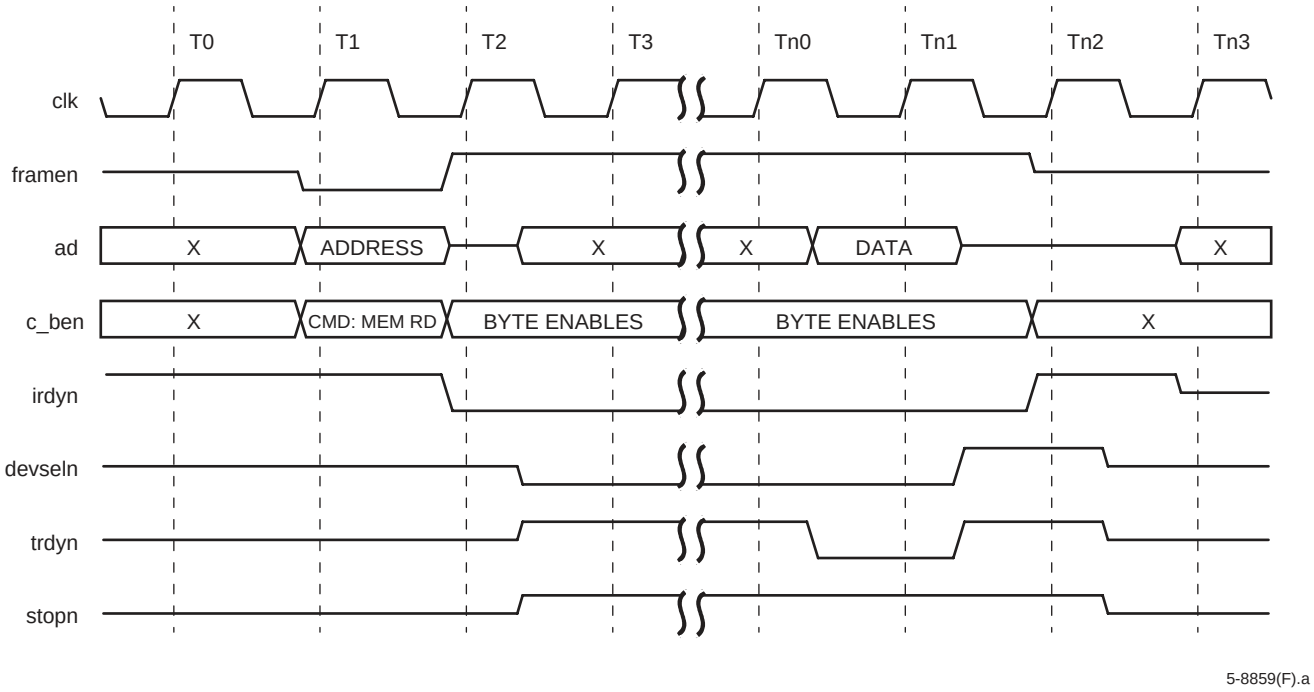
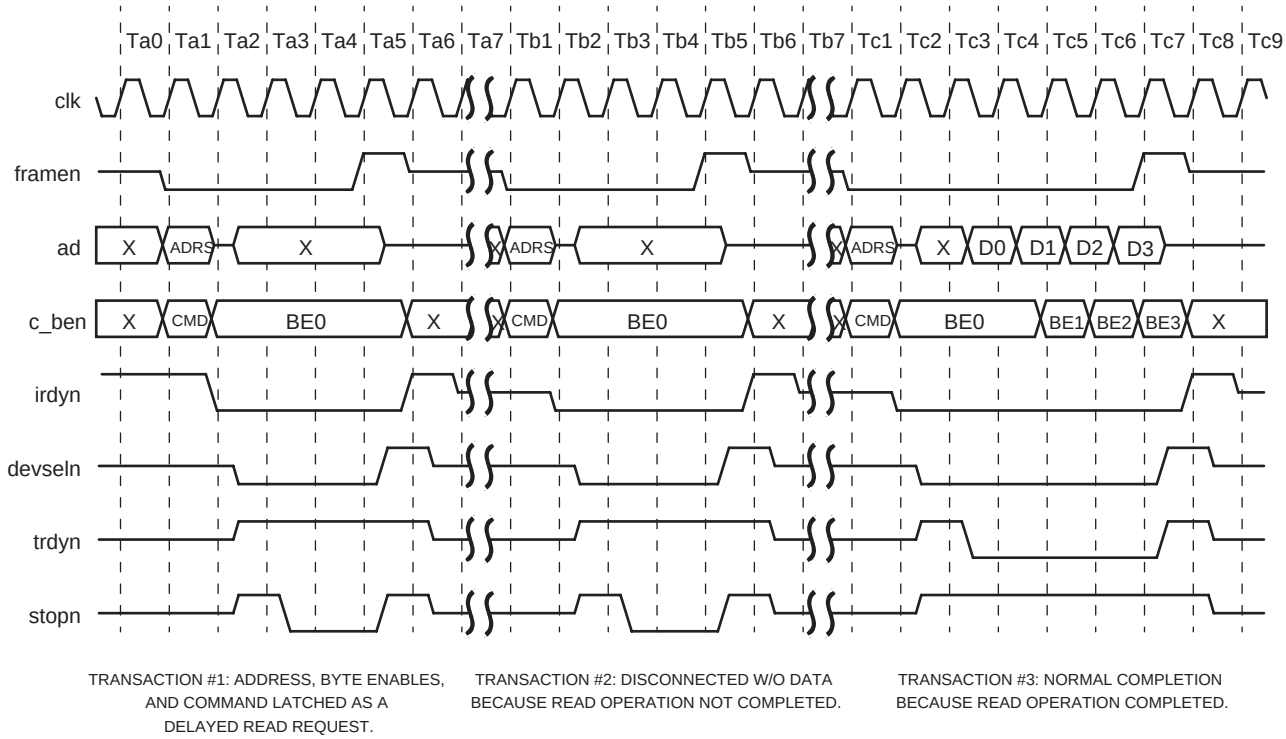


Figure 21. Target Memory Read Single, Not Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

Target Read Memory Burst, Delayed Transaction

Figure 22 (PCI bus) and Figure 23 (FPGA bus) 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 22), 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 **devseln**. 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 22. 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 22. The FPGA interface timing is as shown in Figure 23. This is similar to the timing for a Target non-burst read as shown in Figure 20 except that multiple data cycles are required as long as **trlastcycn** is inactive-high.



5-8862fF).a

Figure 22. Target Memory Read 32-Byte Burst, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Dual Port (continued)

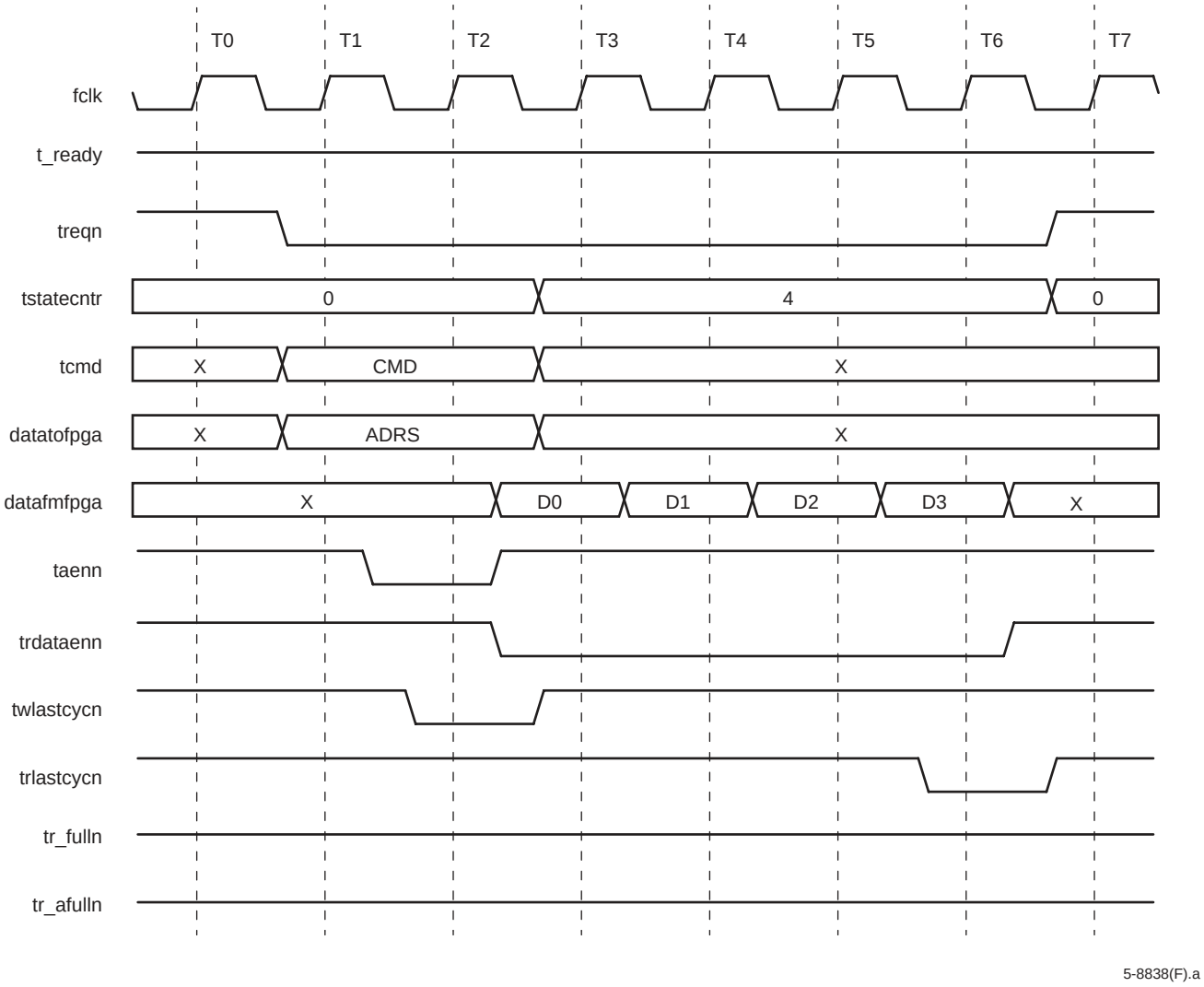


Figure 23. Target Read Memory 32-Byte Burst (FPGA, Dual-Port)

PCI Bus Core Detailed Description Dual Port (continued)

Target Read Memory Burst, No Delayed Transaction

Figure 24 (PCI bus) and Figure 23 (FPGA bus) show the timing for a Target memory burst read of four Quadwords handled as a nondelayed transaction. Figure 24 shows the timing on the PCI interface is similar to that of an I/O read (Figure 18) 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.

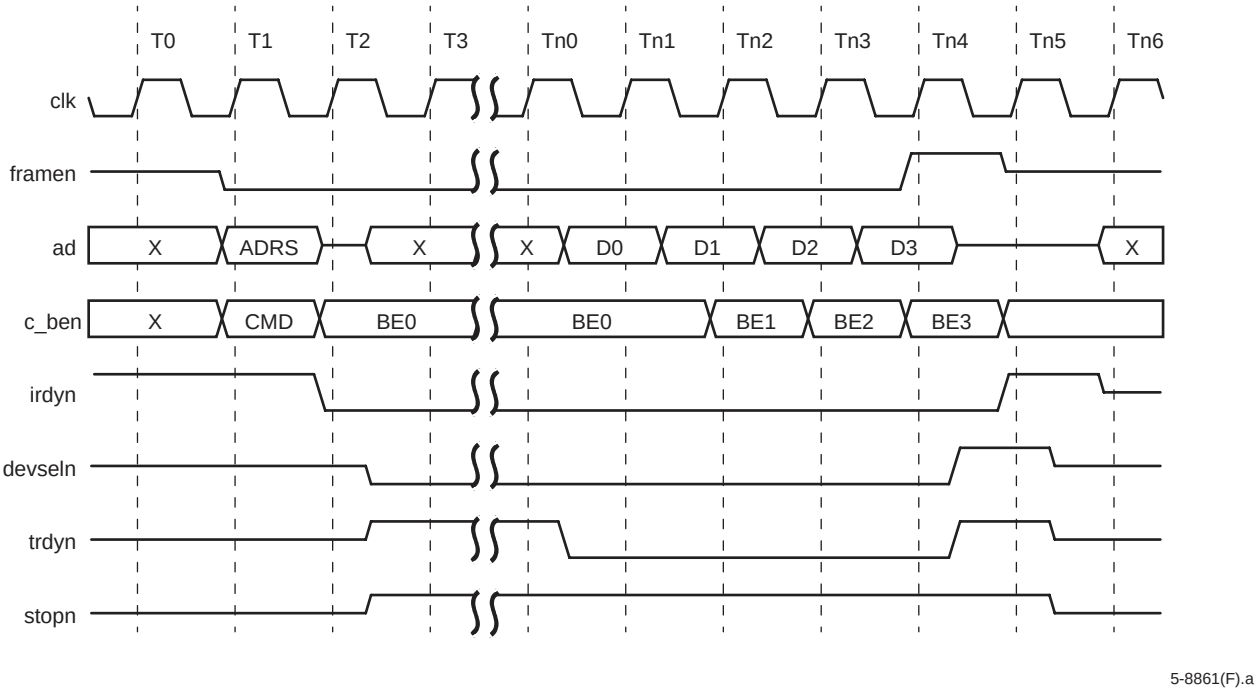


Figure 24. Target Read Memory Burst, No Delayed (PCI Bus, 32-Bit)

Table 22. Dual-Port Target Read

tstatecntr	Next State of tstatecntr	Description	Bus	treqn	trdataenn	twlastcycn	taenn	trlastcycn
0	0	Idle	—	1	1	1	1	1
0	4	Address[63:0]	datatofpgax[7:0] datatofpga[63:0]	0	1	1	0	0
4	4 or 0	Data[63:0]	datafmfpga[63:0]	1*	0	0†	1	1

* **treqn** is deasserted high on the last data Quadword.
† **twlastcycn** is asserted low on the last data Quadword.

PCI Bus Core Detailed Description Quad Port

Pages 68—120 will refer to the quad-port mode of the OR3LP26B device. For dual-port mode, please refer to pages 19—67.

Embedded Core/FPGA Interface Signal Descriptions

In Table 23, 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 23. Embedded Core/FPGA Interface Signals

Symbol	I/O	Description
Master Data FIFO Signals		
mwdata[35:0]	O	Main data bus into the master write FIFO. Refer to Table 25 on page 77 for bus usage and bit descriptions. These signals must be synchronous to **fclk**.
mrdata[35:0]	I	Main data bus out of the master read FIFO. Refer to Table 25 on page 77 for bus usage and bit descriptions. These signals are synchronous to **fclk**.
Master General Signals		
fpga_mbusyn	O	**FPGA Master Is Busy.** This signal is used in modes currently not implemented in the core. Tie off this signal to a 1.
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. This signal is synchronous to **fclk**.
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. Both signals are synchronous to **fclk**.
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. On each rising edge of the clock that this signal is sampled low, command, burst length, and address will be registered. This signal must be synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
mstatecntr[2:0]	I	**Internal State Counter.** Used for Master reads and writes. Details of the Master state machine operation can be found in tables at the end of each operation section. This signal is synchronous to **fclk**.
mfifoclrn	O	**Master FIFO Clear.** This active-low signal is asserted by the FPGA Master to clear all Master FIFOs. This signal must be synchronous to **fclk**.

PCI Bus Core Detailed Description Quad Port (continued)

Table 23. Embedded Core/FPGA Interface Signals (continued)

Symbol	I/O	Description
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. This signal is synchronous to **fclk**.
mcmd[3:0]	O	**Master Command Code.** Command code for the current Master read/write operation. Refer to Table 25 on page 77. This signal must be synchronous to **fclk**.
Master Write Data FIFO Signals		
mwdataenn	O	**Master Write FIFO Data Enable.** This active-low signal enables the registering of bus **datafmfpga** during Master write operations into the PCI core Master write data FIFOs on the rising edge of the Master FIFO clock signal. The signal **mwdataenn** should not be asserted when the Master write data FIFOs are full, or data may be lost. This signal must be synchronous to **fclk**.
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. When asserted, **mwpcihold** must be held low for a minimum of two **pciclk** periods. This signal must be synchronous to **pciclk**.
mw_fulln	I	**Master Write Data FIFO Full Flag.** This active-low signal indicates that the Master write data FIFOs are full. This signal is synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
mw_emptyn	I	**Master Write Data FIFO Empty Flag.** This active-low signal indicates that the Master write data FIFO is empty. Refer to Master write description on signal usage. This signal is synchronous to **pciclk**.
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. This signal must be synchronous to **fclk**.
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 **datatofpga** during Master read operations on the rising edge of the Master FIFO clock signal. Valid data will be read from the FIFO whenever it is not empty. This signal must be synchronous to **fclk**.
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. This signal is synchronous to **fclk**.

PCI Bus Core Detailed Description Quad Port (continued)

Table 23. 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. This signal is synchronous to **fclk**.
mr_fulln	I	**Master Read Data FIFO Full Flag.** This active-low signal indicates that the Master read data FIFO is full. Refer to Master read description on signal usage. This signal is synchronous to **pciclk**.
fpga_mstopburstn	O	**Stop Burst Reads.** This active-low signal is used by the FPGA Master to terminate burst reads before completion. When asserted, it must stay asserted for a minimum of two **pciclk** periods. When asserted, **fpga_mstopburstn** must stay asserted until **ma_fulln** goes inactive (high). This signal must be synchronous to **pciclk**.
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 (1 **fclk** period). This signal is synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
fpga_tabort	O	**Target Abort.** This active-high signal is asserted by the FPGA Target application to abort all future PCI cycles. Once asserted, this signal needs to remain asserted for a minimum of two **pciclk** cycles. This signal must be synchronous to **pciclk**.
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 **pciclk** cycles. This signal must be synchronous to **pciclk**.
deltrn	O	**Target Delayed Transaction.** Used for Target I/O write (page 100) and Target read operations (page 109). Target memory writes are always posted. Once asserted, this signal needs to remain asserted for a minimum of two **pciclk** cycles. This signal must be synchronous to **pciclk**.
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. Both signals are synchronous to **fclk**.

PCI Bus Core Detailed Description Quad Port (continued)

Table 23. Embedded Core/FPGA Interface Signals (continued)

Symbol	I/O	Description
Target Data FIFO Signals		
twdata[35:0]	I	Target side data bus into the FPGA from the target write FIFOs. These signals are synchronous to **fclk**.
trdata[35:0]	O	Target side data bus out of the FPGA into the target read FIFOs. These signals must be synchronous to **fclk**.
Target FIFO Address and Command Register Control Signals		
tfifoclrn	O	**Target FIFO Clear.** This active-low signal is asserted by the FPGA Target to clear all Target FIFOs. This signal must be synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
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). This signal must be synchronous to **fclk**.
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. Because it is synchronous to **pciclk**, it must be qualified with **treqn**.
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. Because it is synchronous to **pciclk**, it must be qualified with **treqn**.
tstatecntr[2:0]	I	**Internal State Counter.** Used for target reads and writes. Details of the target state machine operation can be found in tables at the end of each operation section. This signal is synchronous to **fclk**.
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 **datatofpga** 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. This signal must be synchronous to **fclk**.
tw_emptyn	I	**Target Write FIFO Empty.** This signal active indicates that the Target write FIFO is empty. This signal is synchronous to **fclk**.

PCI Bus Core Detailed Description Quad Port (continued)

Table 23. Embedded Core/FPGA Interface Signals (continued)

Symbol	I/O	Description
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. This signal is synchronous to **fclk**.
tw_fulln	I	**Target Write Data FIFO Full Flag**. This active-low signal indicates that the target write data FIFO is full. Refer to target write description on signal usage. This signal is synchronous to **pciclk**.
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 Target 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 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. This signal is synchronous to **fclk**.
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). Once asserted, this signal needs to remain asserted for a minimum or two **pciclk** periods. This signal must be synchronous to **pciclk**.
Target Read Data FIFO Signals		
trdataenn	O	**Target Read FIFO Data Enable.** This active-low signal enables the registering of bus **datafmfpga** during Target read operations into the PCI core Target read data FIFOs on the rising edge of the Target FIFO clock signal. The signal **trdataenn** should not be asserted when the Target read data FIFOs are full, or data may be lost. This signal must be synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
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. This signal is synchronous to **fclk**.
tr_emptyn	I	**Target Read Data FIFO Empty Flag.** This active-low signal indicates that the target read data FIFO is empty. Refer to target read description on signal usage. This signal is synchronous to **pciclk**.
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. Once asserted, this signal needs to remain asserted for a minimum or two **pciclk** periods. This signal must be synchronous to **pciclk**.

PCI Bus Core Detailed Description Quad Port (continued)

Table 23. Embedded Core/FPGA Interface Signals (continued)

Symbol	I/O	Description
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. This signal is synchronous to **fclk**.
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). Once asserted, this signal needs to remain asserted for a minimum or two **pciclk** periods. This signal must be synchronous to **pciclk**.
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 **intan** onto the PCI bus. Once asserted, this signal needs to remain asserted for a minimum of two **pciclk** cycles. This signal must be synchronous to **pciclk**.
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**. External or user-defined clocks may also be used. The signals **fclk1** and **fclk2** must be the same clock in dual-port mode.
pciclk	I	**PCI Clock**. The signal **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 (**rstn**).
fpga_syserror	O	**System Error**. This active-high signal is used by the FPGA to generate a system error on the PCI bus. This is passed to the PCI bus as **serrn**. This signal must be synchronous to **pciclk**.
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 **req64n** as active (low) on the inactive-going (rising) edge of PCI signal **rstn**. Note that this does not imply that any particular transaction is 64-bit, since each transaction is individually negotiated using PCI signals **req64n** and **ack64n**. This signal is synchronous to **pciclk**.
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). This signal must be synchronous to **fclk**.

PCI Bus Core Detailed Description Quad Port (continued)

Embedded Core/FPGA Interface Signal Locations

Table 24 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 24. OR3LP26B FPGA/PCI Core Interface Signal Locations

PCI Core/FPGA Interface Site	FPGA Input Signal Name	FPGA Output Signal Name
ASB1A	pci_rstn	pci_intan
ASB1B	pci_64bit	(unused)
ASB1C	(unused)	fpga_syserror
ASB1D	(unused)	fpga_mbusyn
ASB2A	twdata31	trdata31
ASB2B	twdata30	trdata30
ASB2C	twdata29	trdata29
ASB2D	twdata28	trdata28
ASB3A	twdata27	trdata27
ASB3B	twdata26	trdata26
ASB3C	twdata25	trdata25
ASB3D	twdata24	trdata24
ASB4A	twdata23	trdata23
ASB4B	twdata22	trdata22
ASB4C	twdata21	trdata21
ASB4D	twdata20	trdata20
ASB5A	twdata19	trdata19
ASB5B	twdata18	trdata18
ASB5C	twdata17	trdata17
ASB5D	twdata16	trdata16
ASB6A	twdata35	trdata35
ASB6B	twdata34	trdata34
ASB6C	twdata33	trdata33
ASB6D	twdata32	trdata32
ASB7A	twdata15	trdata15
ASB7B	twdata14	trdata14
ASB7C	twdata13	trdata13
ASB7D	twdata12	trdata12
ASB8A	twdata11	trdata11
ASB8B	twdata10	trdata10
ASB8C	twdata9	trdata9
ASB8D	twdata8	trdata8
ASB9A	twdata7	trdata7
ASB9B	twdata6	trdata6
ASB9C	twdata5	trdata5
ASB9D	twdata4	trdata4
CKTOASB9	(unused)	fclk1
ASB10A	twdata3	trdata3

PCI Bus Core Detailed Description Quad Port (continued)

Table 24. OR3LP26B FPGA/PCI Core Interface Signal Locations (continued)

PCI Core/FPGA Interface Site	FPGA Input Signal Name	FPGA Output Signal Name
ASB10B	twdata2	trdata2
ASB10C	twdata1	trdata1
ASB10D	twdata0	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
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	mrdata0	mwdata0
ASB19B	mrdata1	mwdata1

PCI Bus Core Detailed Description Quad Port (continued)

Table 24. OR3LP26B FPGA/PCI Core Interface Signal Locations (continued)

PCI Core/FPGA Interface Site	FPGA Input Signal Name	FPGA Output Signal Name
ASB19C	mrdata2	mwdata2
ASB19D	mrdata3	mwdata3
CKTOASB19	(unused)	fclk2
ASB20A	mrdata4	mwdata4
ASB20B	mrdata5	mwdata5
ASB20C	mrdata6	mwdata6
ASB20D	mrdata7	mwdata7
ASB21A	mrdata8	mwdata8
ASB21B	mrdata9	mwdata9
ASB21C	mrdata10	mwdata10
ASB21D	mrdata11	mwdata11
ASB22A	mrdata12	mwdata12
ASB22B	mrdata13	mwdata13
ASB22C	mrdata14	mwdata14
ASB22D	mrdata15	mwdata15
ASB23A	mrdata32	mwdata32
ASB23B	mrdata33	mwdata33
ASB23C	mrdata34	mwdata34
ASB23D	mrdata35	mwdata35
ASB24A	mrdata16	mwdata16
ASB24B	mrdata17	mwdata17
ASB24C	mrdata18	mwdata18
ASB24D	mrdata19	mwdata19
ASB25A	mrdata20	mwdata20
ASB25B	mrdata21	mwdata21
ASB25C	mrdata22	mwdata22
ASB25D	mrdata23	mwdata23
ASB26A	mrdata24	mwdata24
ASB26B	mrdata25	mwdata25
ASB26C	mrdata26	mwdata26
ASB26D	mrdata27	mwdata27
ASB27A	mrdata28	mwdata28
ASB27B	mrdata29	mwdata29
ASB27C	mrdata30	mwdata30
ASB27D	mrdata31	mwdata31
ASB28A	mstatecntr0	mfifoclrn
ASB28B	mstatecntr1	(unused)
ASB28C	mstatecntr2	(unused)
ASB28D	pci_mcfg_stat	mcfgshiftenn

PCI Bus Core Detailed Description Quad Port (continued)

Table 25. Bit Definitions on FPGA/PCI Core Interface

Bits	Name	Description
A. 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 (active-high)
mwdata[33:32]	—	Unused
mwdata[31:0]	A1 & A0	Address words 1 and 0
mcmd[3:0]	mcmd	Master command opcode*
B. 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
C. Quad-Port Master Write, Lower Data DWORD		**mstatecntr = 4**
mwdata[35:32]	BE3—BE0	Byte enables (active-low)
mwdata[31:0]	D3—D0	Data bytes 3 to 0
D. Quad-Port Master Write, Upper Data DWORD		**mstatecntr = 5**
mwdata[35:32]	BE7—BE4	Byte enables (active-low)
mwdata[31:0]	D7—D4	Data bytes 7 to 4
E. 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 (active-high)
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 (active-low)
mwdata[23:16]	—	Unused

* 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 Quad Port (continued)

Table 25. Bit Definitions on FPGA/PCI Core Interface (continued)

Bits	Name	Description
mwdata[15:0]	A0	Address word 0
mcmd[3:0]	mcmd	Master command opcode*
F. 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 (active-high)
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 (active-low)
mwdata[23:18]	—	Unused
mwdata[17:0]	BL	Burst length (In Quadwords)
mcmd[3:0]	mcmd	Master command opcode*
G. 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
H. 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
I. Quad-Port Master Read, Lower Data DWORD		**mstatecntr = 4**
mrdata[35:32]	—	Unused
mrdata[31:0]	D3—D0	Data bytes 3 to 0
J. Quad-Port Master Read, Upper Data DWORD		**mstatecntr = 5**
mrdata[35:32]	—	Unused
mrdata[31:0]	D7—D4	Data bytes 7 to 4

* 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 Quad Port (continued)

Table 25. Bit Definitions on FPGA/PCI Core Interface (continued)

Bits	Name	Description
K. Quad-Port Target Write & Read (Lower Address Cycle)		**tstatecntr = 0**
twdata[35]	Burst_I	Burst indication (active-high)
twdata[34]	DA	Dual address indicator (active-high)
twdata[33:32]	—	Unused
twdata[31:0]	A1 & A0	Address words 1 and 0
tcmd[3:0]	tcmd	Target command opcode*
L. 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
M. Quad-Port Target Write, Lower Data DWORD		**tstatecntr = 4**
twdata[35:32]	BE3—BE0	Byte enables (active-low)
twdata[31:0]	D3—D0	Data bytes 3 to 0
N. Quad-Port Target Write, Upper Data DWORD		**tstatecntr = 5**
twdata[35:32]	BE7—BE4	Byte enables (active-low)
twdata[31:0]	D7—D4	Data bytes 7 to 4
O. Quad-Port Target Read, Lower Data DWORD		**tstatecntr = 4**
trdata[35:32]	—	Unused
trdata[31:0]	D3—D0	Data bytes 3 to 0
P. Quad-Port Target Read, Upper Data DWORD		**tstatecntr = 5**
trdata[35:32]	—	Unused
trdata[31:0]	D7—D4	Data bytes 7 to 4

* 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 Quad Port (continued)

Table 26. Address Cycle Sequences for Various Operations

Operation	Address Mode	Supplied Address	New Burst Length	Address Cycle Sequence (Once Only)	Data Cycle Sequence (Repeats)
Master Write	SA/DA	31:0	NA	A	CD
	DA	63:0	NA	A, B	CD
Master Read	SA/DA	15:0	No	E	I, J
	SA/DA	(none)	Yes	F	I, J
	SA/DA	31:0	Yes	F, G	I, J
	DA	63:0	Yes	F, G, H	I, J
Target Write	SA/DA	31:0	NA	K	M, N
	DA	63:0	NA	K, L	M, N
Target Read	SA/DA	31:0	NA	K	O, P
	DA	63:0	NA	K, L	O, P

PCI Bus Core Detailed Description Quad Port (continued)

Embedded Core Bit Stream Configurable Options

Table 27 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 27. 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 Quad Port (continued)

Understanding FIFO Packing/Unpacking

In quad-port mode, the interface from the core to the FPGA is always 32 bits wide. However, data packing through the FIFOs will differ depending on whether the transfers on the PCI bus are 32 bits or 64 bits. The following discussions pertain to target write or master read operations where data will be read from the FIFOs.

- **64-bit transfers**: Since the FIFOs are always in 64-bit mode, the data will flow through without any repacking. Keep in mind that 64-bit transfers must start on a Quadword aligned address (AD2 = 0). Case 1 provides an example of how the data is read out of the read side of the FIFO.

Case 1: Master read burst, 64-bit. Quadword aligned starting address, even number of 64-bit words transferred on the PCI bus.

Table 28. Quad-Port FIFO Packing/Unpacking, Case 1, PCI Side

PCI Address	PCI Data	PCI Byte Enables (Active-Low)
00001000	64-bit Word1	00000000
(00001008)	64-bit Word2	00000000
(00001010)	64-bit Word3	00000000
(00001018)	64-bit Word4	00000000
(00001020)	64-bit Word5	00000000
(00001028)	64-bit Word6	00000000

Table 29. Dual-Port FIFO Packing/Unpacking, Case 1, FPGA Side

Master Write FIFO Slot	FIFO Data Bits [31:0]	FIFO Byte Enables (Active-Low)
	twdata[31:0]	twdata[35:32]
1	64-bit Word1 [31:0]	0000
1	64-bit Word1 [63:32]	0000
2	64-bit Word2 [31:0]	0000
2	64-bit Word2 [63:32]	0000
3	64-bit Word3 [31:0]	0000
3	64-bit Word3 [63:32]	0000
4	64-bit Word4 [31:0]	0000
4	64-bit Word4 [63:32]	0000
5	64-bit Word5 [31:0]	0000
5	64-bit Word5 [63:32]	0000
6	64-bit Word6 [31:0]	0000
6	64-bit Word6 [63:32]	0000

Note: PCI addresses in parentheses are not actually sent across the PCI bus during a burst. They are used for illustrative purposes only. Dummy words are unknown data words in the FIFOs with their byte enables disabled.

- **32-bit transfers**: The FIFOs are always in 64-bit mode, so depending upon what address the transfer begins, the data coming out of the FIFOs will be packed differently. The following two cases provide examples with different starting addresses and word counts. Case 1 is also true for Master read operations.

Case 1: Target write burst, 32-bit. Quadword aligned starting address, even number of 32-bit words transferred on the PCI bus.

PCI Bus Core Detailed Description Quad Port (continued)

Table 30. Quad-Port FIFO Packing/Unpacking, Case 1, PCI Side

PCI Address	PCI Data	PCI Byte Enables (Active-Low)
00001000	32-bit Word1	0000
(00001004)	32-bit Word2	0000
(00001008)	32-bit Word3	0000
(00001010)	32-bit Word4	0000
(00001014)	32-bit Word5	0000
(00001018)	32-bit Word6	0000

Table 31. Quad-Port FIFO Packing/Unpacking, Case 1, FPGA Side

Master Write FIFO Slot	FIFO Data Bits [31:0]	FIFO Byte Enables (Active-Low)
	twdata[31:0]	twdata[35:32]
1	32-bit Word1	0000
1	32-bit Word2	0000
2	32-bit Word3	0000
2	32-bit Word4	0000
3	32-bit Word5	0000
3	32-bit Word6	0000

Note: PCI addresses in parentheses are not actually sent across the PCI bus during a burst. They are used for illustrative purposes only. Dummy words are unknown data words in the FIFOs with their byte enables disabled.

Case 2: Target write burst, 32-bit. Quadword aligned starting address, odd number of 32-bit words transferred on the PCI bus.

Table 32. Quad-Port FIFO Packing/Unpacking, Case 2, PCI Side

PCI Address	PCI Data	PCI Byte Enables (Active-Low)
00001000	32-bit Word1	0000
(00001004)	32-bit Word2	0000
(00001008)	32-bit Word3	0000
(00001010)	32-bit Word4	0000
(00001014)	32-bit Word5	0000

Table 33. Quad-Port FIFO Packing/Unpacking, Case 1, FPGA Side

Master Write FIFO Slot	FIFO Data Bits [31:0]	FIFO Byte Enables (Active-Low)
	twdata[31:0]	twdata[35:32]
1	32-bit Word1	0000
1	32-bit Word2	0000
2	32-bit Word3	0000
2	32-bit Word4	0000
3	32-bit Word5	0000
3	Dummy Word	FFFF

Note: PCI addresses in parentheses are not actually sent across the PCI bus during a burst. They are used for illustrative purposes only. Dummy words are unknown data words in the FIFOs with their byte enables disabled.

PCI Bus Core Detailed Description Quad Port (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 25). 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 two 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 34 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 34. 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. The **bar** bus is synchronous to the **pciclk**, so it must be qualified with **treq** which is on the **fclk** clock domain.

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 Quad Port (continued)

Interrupt Request and System Error Generation

Two additional signals are available on the user side interface to request an interrupt on **intan** (**pci_intan**) and force a system error on the PCI **serrn** pin (**fpga_syserror**). The **pci_intan** signal may be asserted low at any time. It is not directly tied to any bus cycle. The **fpga_syserror**, as well, may be asserted high at any time. The **serrn** will be subsequently asserted low during the next PCI transaction to this device. In generating **pci_intan** and **fpga_syserror**, keep in mind that both signals need to be synchronous to **pciclk**.

Working in 32- and 64-bit Modes

The OR3LP26B works equally well in 32-bit and 64-bit PCI systems. In a 64-bit system, it is required that, during reset, the host assert **req64n** low indicating that the bus width is 64 bits. The core will evaluate this signal at reset, and automatically configure itself in either 32-bit or 64-bit mode. When configured in 32-bit mode, the core will 3-state all upper PCI bus pins and apply a weak pull-up.

32-bit Transfers in a 64-bit System

Although designed as a 64-bit interface, the OR3LP26B also works efficiently in 32-bit mode. For single 32-bit transfers, the core will perform a 32-bit PCI transfer. For burst transactions, the core will attempt 64-bit transfers, and then back down to 32-bit mode if **ack64n** was not received. In general, the core will perform the PCI bus transaction that is most efficient on the bus.

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 the states of each operation for each mode.

Table 35 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 35. Index to State Sequence Tables

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
Master	Write	Config, Memory, I/O	Nonburst	Figure 25	Table 36	Figure 27
			Burst	Figure 26		Figure 28
	Read	Config, Memory, I/O	Nonburst	Figure 29	Table 37*†	Figure 30
			Burst	Figure 31	Table 38‡	Figure 32
Target	Write	Config	Nonburst	Figure 33	Table 39	§
		I/O	Delayed	Figure 34		Figure 36
		Memory, I/O	Nonburst, Not Delayed	Figure 35		Figure 38
		Memory	Burst	Figure 37		
	Read	Config	Nonburst	Figure 39	Table 40	§
		I/O	Delayed	Figure 40		Figure 43
			Not Delayed	Figure 41		
		Memory	Nonburst	Figure 44		
			Nonburst Delayed	Figure 42		
			Burst	Figure 47		Figure 46
			Burst Delayed	Figure 45		

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

PCI Bus Core Detailed Description
Quad Port (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 36. 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 25. 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[2:0]**, 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 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** 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 25 lists the values of the state counter **mstatecntr** and the appropriate accompanying data.

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**.

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.

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. During a master write burst transaction, the master write FIFO may go empty, especially if the user side application is slow at filling the FIFO. When this condition occurs, the master will insert wait-states continuously until another word (or the last word) is written into the FIFO and will not terminate the transaction. On the target side, if the target is ready to accept more data, it will have **trdyn** asserted which will disable it from terminating the transaction as well. This can create a deadlock condition on the PCI bus. If the user application cannot supply any more data, and wishes to terminate the burst, additional FPGA logic must be incorporated to detect and accomplish the termination. The way to terminate the transaction is to provide one last piece of data (either real data or a dummy data word with all byte enables disabled) along with **mwlastcycn** asserted.

PCI Bus Core Detailed Description
Quad Port (continued)

Designing a Deadlock Timer

This design example is a method by which the user application can detect the deadlock condition and terminate the burst transaction. Since the **mw_emptyn** signal is on the **pciclk** clock domain, it must be resynchronized to the fclk domain. To accomplish this, double register **mw_emptyn** with **fclk** driven registers. The **mw_emptyn** signal is fed as a clock enable and a synchronous clear to a counter, driven by **fclk**. The counter's length may be designed to guarantee a certain time-out latency on the PCI bus. When the FIFO is not empty (**mw_emptyn** = 1), the counter will stay cleared. When the FIFO has been empty for an extended period of time, the counter will count and eventually overflow. This overflow indication can be used to write one dummy word into the FIFO with the byte enables disabled along with the **mwlastcycn** bit asserted. The transaction will complete, and the core will go back into an idle state.

Bursting

Instead of using a burst length, the Master write operation relies on **mwlastcycn** 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**. The signal **mwlastcycn** must remain asserted throughout a nonburst Master write, since the last data phase is the only data phase. The maximum burst length is limited only by the latency timer. To initiate a burst, the starting address must be aligned to a 64-byte boundary. If **ad[2]** is a 1, a single transfer will be executed.

Termination

Once initiated, Master write operations will repeat on the PCI bus until one of the following occurs:

1. All data is sent.
2. An abort occurs (either Master or Target).
3. The PCI bus's reset signal (**rstn**) 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 **mfifoclrn** to place the core's master logic in a known state. Normally, the clear signal will not be used unless a severe problem has occurred in the data flow. The **mfifoclrn** signal is synchronous with **fclk** and must be asserted for a minimum of three clock periods. During reset, the **m_ready** signal will go low. After the reset signal is deasserted high, **m_ready** will continue to be low for 8—10 clock periods. The FPGA application should not continue normal operation until **m_ready** is asserted high.

Understanding and Using the pci_mcfg_stat Status Signals

On the Master interface, there are two signals that control and provide status to the FPGA application. The signal **pci_mcfg_stat** provides the status, and **mcfgshiftenn** controls what information the status line provides. The **pci_mcfg_stat** signal is always active and duplicates the status contained in configuration status register at location offset 0x04, bits 24, 28, and 29. To use this status output, the FPGA application must keep **mcfgshiftenn** = 1. When high, **pci_mcfg_stat** provides the wired-OR of the three status lines. If **pci_mcfg_stat** gets set to a 1, indicating an error, then the FPGA application may set **mcfgshiftenn** = 0 to determine individual status. Once low, the **pci_mcfg_stat** signal will output data parity error detected on the first clock, target abort on received the second clock, and master abort received on the third clock.

PCI Bus Core Detailed Description Quad Port (continued)

Master Write, Nonburst Transaction

Figure 27 (FPGA bus) and Figure 25 (PCI bus) show the timing of a Master write, nonburst transaction. In Figure 27, the transaction is initiated by the FPGA application asserting Master address enable (**maenn**), while providing the command word and the lower DWORD address on bus **mwdata**. On the next clock, for 64-bit address mode, the upper DWORD address is provided on bus **mwdata** while asserting **wmlastcycn**. On the next clock, **maenn** is deasserted and the one DWORD of data is provided on bus **mwdata** along with assertion of the Master write data enable (**mwdataenn**). The forth clock provided the second DWORD of data an assertion of **mwlastcycn**. 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. This completes the setup for this operation. Execution begins on the PCI bus, as shown in Figure 25.

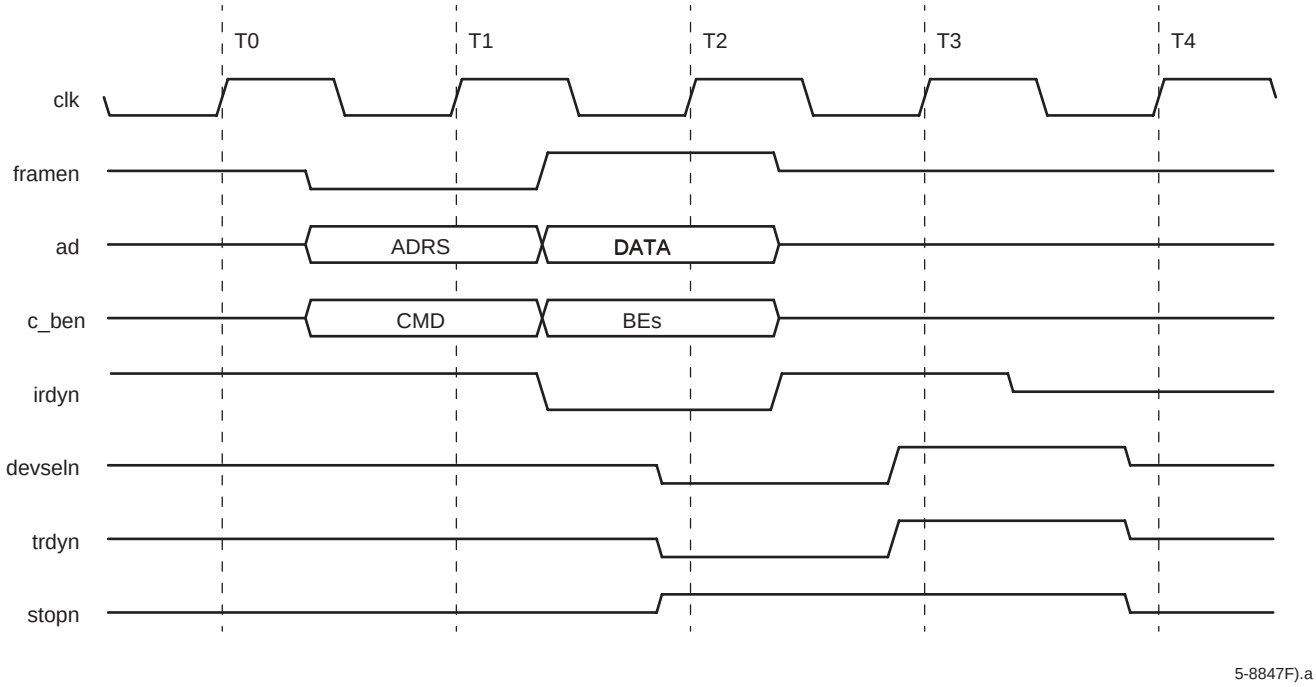


Figure 25. Master Write Single (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

Master Write, Burst Transaction

Figure 28 (FPGA bus) and Figure 26 (PCI bus) show the timing of a 4-Quadword Master write burst transaction. Operation is similar to that in the previous Master write, nonburst transaction, but extra data is supplied by the FPGA application. In Figure 28, the transaction is initiated by the FPGA application asserting Master address enable (**maenn**), while providing the command word and the lower DWORD address on bus **mwdata**. On the second clock, for 64-bit addressing, the upper DWORD address is supplied along with **mwlastcycn**. On the third through tenth clocks, **maenn** is deasserted, the Master write data enable (**mwdataenn**) is asserted, and eight DWORDs of data are provided on bus **mwdata**. On the tenth clock, **mwlastcycn** is asserted along with the last DWORD of data. 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 DWORD 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 26. 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 26.

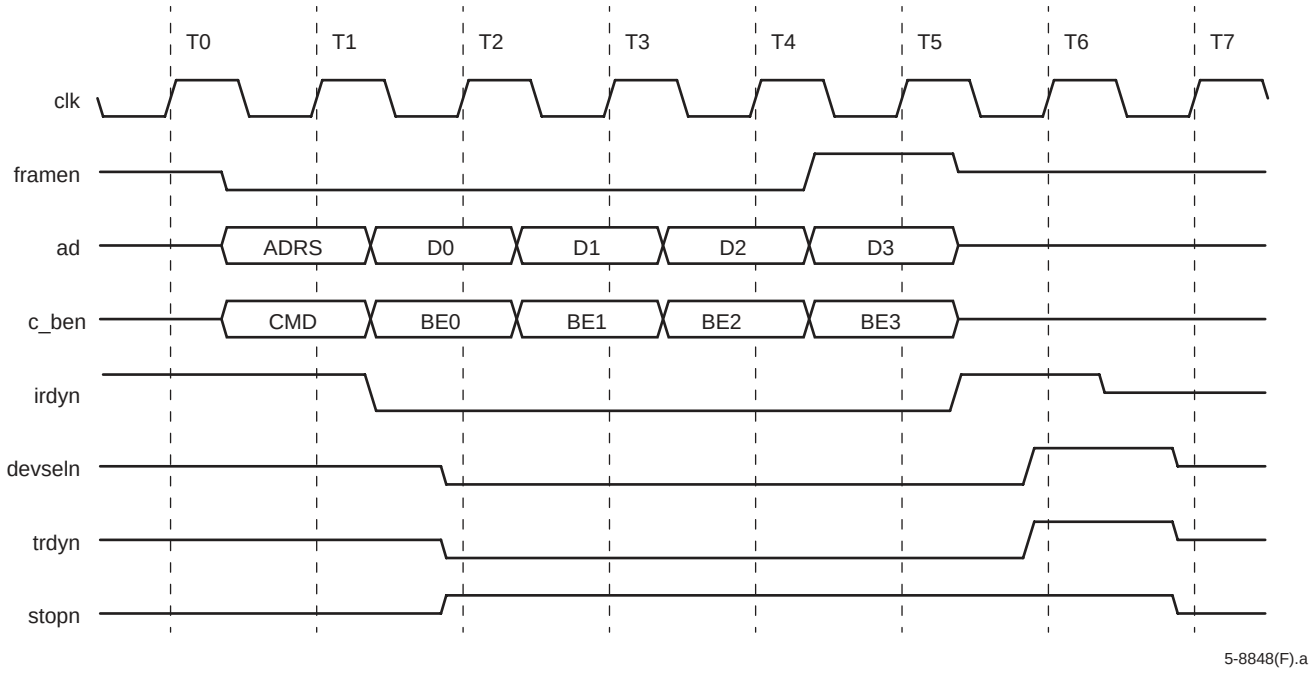


Figure 26. Master Write 32-Byte Burst (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

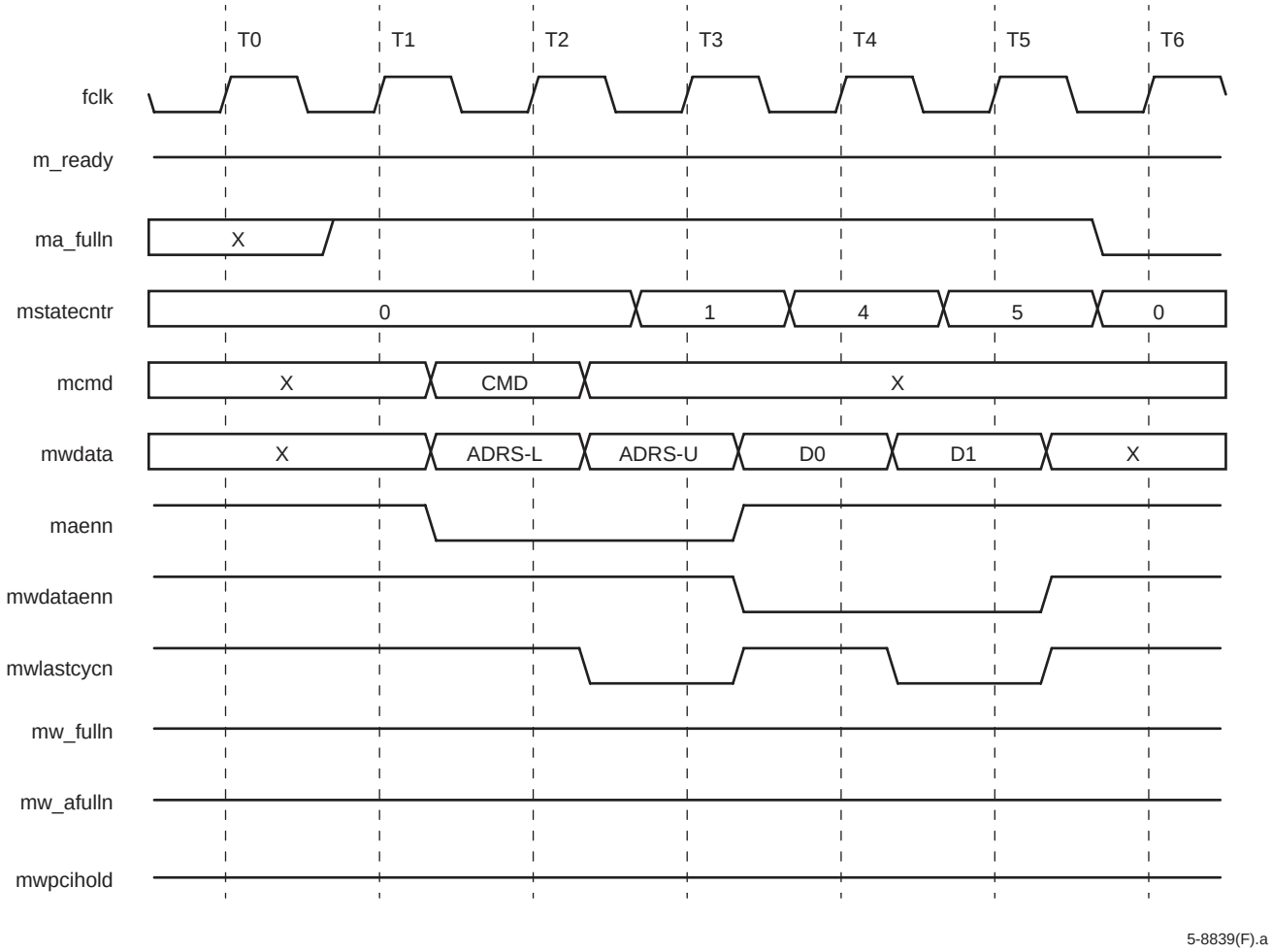


Figure 27. Master Write Single Quadword (FPGA Bus, Quad-Port, 64-Bit Address)

PCI Bus Core Detailed Description Quad Port (continued)

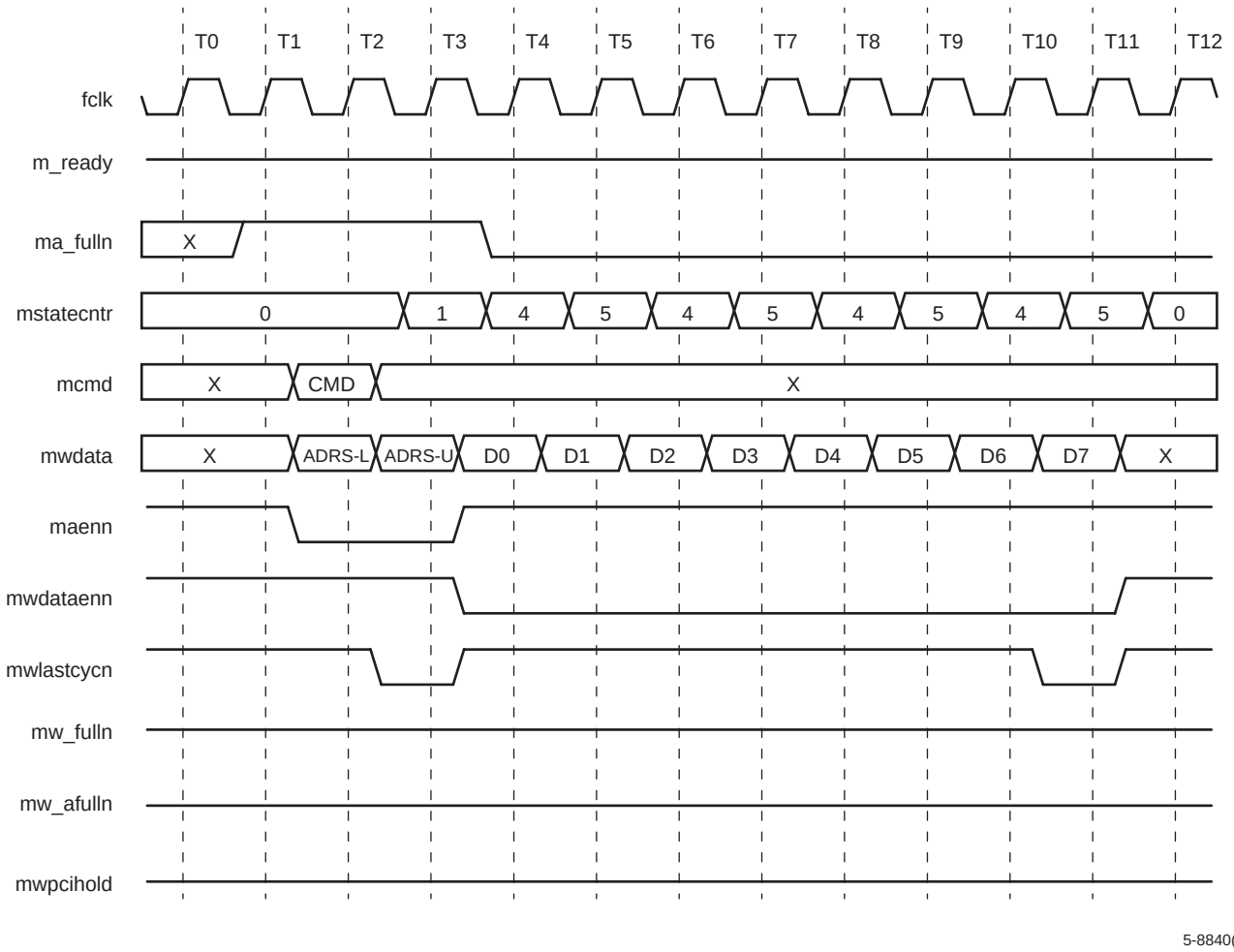


Figure 28. Master Write 32-Byte Burst (FPGA Bus, Quad-Port, 64-Bit Address)

Table 36. Quad-Port Master Write

mstatecntr	Next State of mstatecntr	Description	Bus	maenn	mwdataenn	mwlastcycn
0	0	Idle	—	1	1	1
0	1	Address[31:0]	mwdata[35:0]	0	1	1
1	4	Address[63:32]	mwdata[35:0]	0	1	0
4	5 or 0	Data[31:0], BE[3:0]	mwdata[35:0]	1	0	1
5	4 or 0	Data[63:32], BE[7:4]	mwdata[35:0]	1	0	0*

* **mwlastcycn** is only 0 during the last data DWORD sent.

Notes:
For 32-bit addressing, state 1 is absent.

For 32-bit data, state 5 is absent.

PCI Bus Core Detailed Description
Quad Port (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 38. 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 25. 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**.

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. During a master read burst transaction, the master read FIFO may go full, especially if the user side application is slow at unloading the FIFO. When this condition occurs, the master will insert wait-states continuously until another word is read from the FIFO, or the word count is exhausted. On the target side, if the target is ready to send more data, it will have **trdyn** asserted which will disable it from terminating the transaction as well. This can create a deadlock condition on the PCI bus. If the user application cannot unload any more data, and wishes to terminate the burst, additional FPGA logic must be incorporated to detect and accomplish the termination. Two operations must occur to terminate the current transaction. First, the **fpga_mstopburstn** signal must be asserted indicating to the core the master request to terminate. Second, one additional word of data must be read from the FIFO (only if the FIFO is full). The signal **fpga_mstopburstn** needs to stay asserted low until the **ma_fulln** flag is asserted low indicating that the transaction has been terminated and cleared.

Designing a Deadlock Timer

This design example is a method by which the user application can detect this condition and terminate the burst transaction. Since the **mr_fulln** and **fpga_mstopburstn** signals are on the **pciclk** clock domain, the deadlock counter will run on the **pciclk** clock. The **mr_fulln** signal is fed as a clock enable and a synchronous clear to a counter, driven by **pciclk**. The counter's length may be designed to guarantee a certain time-out latency on the PCI bus. When the FIFO is not full (**mr_fulln** = 1), the counter will stay cleared. When the FIFO has been full for an extended period of time, the counter will count and eventually overflow. This overflow indication can be used to set the **fpga_mstopburstn** signal indicating a request to stop the burst. The overflow signal is then detected and synchronized onto the **fclk** domain to be used to read one additional word from the FIFO. The transaction will complete, and the core will go back into an idle state.

PCI Bus Core Detailed Description
Quad Port (continued)

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**). The burst length of 18 bits allows bursts of up to $2^{18} - 1$ quad words to be specified. To initiate a burst, the starting address must be aligned to a 64-byte boundary. If **ad[2]** is a 1, a single transfer will be executed.

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 **trdyn** 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 the following occurs:

1. All data is received.
2. An abort occurs (either Master or Target).
3. The **fpga_mstopburstn** signal is asserted.
4. The PCI bus' reset signal (**resetn**) 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 **mfifoclrn** to place the core's master logic in a known state. Normally, the clear signal will not be used unless a severe problem has occurred in the data flow. The **mfifoclrn** signal is synchronous with **fclk** and must be asserted for a minimum of three clock periods. During reset, the **m_ready** signal will go low. After the reset signal is deasserted high, **m_ready** will continue to be low for 8—10 clock periods. The FPGA application should not continue normal operation until **m_ready** is asserted high.

Understanding and Using the pci_mcfg_stat Status Signals

On the Master interface, there are two signals that control and provide status to the FPGA application. **pci_mcfg_stat** provides the status, and **mcfgshiftenn** controls what information the status line provides. The **pci_mcfg_stat** signal is always active and duplicates the status contained in configuration status register at location offset 0x04, bits 24, 28, and 29. To use this status output, the FPGA application must keep **mcfgshiftenn** = 1. When high, **pci_mcfg_stat** provides the wired-OR of the three status lines. If **pci_mcfg_stat** gets set to a 1, indicating an error, then the FPGA application may set **mcfgshiftenn** = 0 to determine individual status. Once low, the **pci_mcfg_stat** signal will output data parity error detected on the first clock, target abort received on the second clock, and master abort received on the third clock.

PCI Bus Core Detailed Description Quad Port (continued)

Master Read, Nonburst Transaction

Figure 30 (FPGA bus) and Figure 29 (PCI bus) show the timing of a single Quadword Master read. In Figure 30, the transaction is initiated by the FPGA application asserting Master address enable (**maenn**), while providing the command and burst length on bus **mwdata**. On the next clock, the FPGA application provides the DWORD address and asserts **mwlastcycn**. On the third cycle, both **maenn** and **mwlastcycn** are deasserted. PCI bus activity now begins as shown in Figure 29. Once data is transferred on the PCI bus and **mr_emptyn** is deasserted high, the FPGA application asserts **mrdataenn** and two DWORDs of data are transferred on bus **mrdata**.

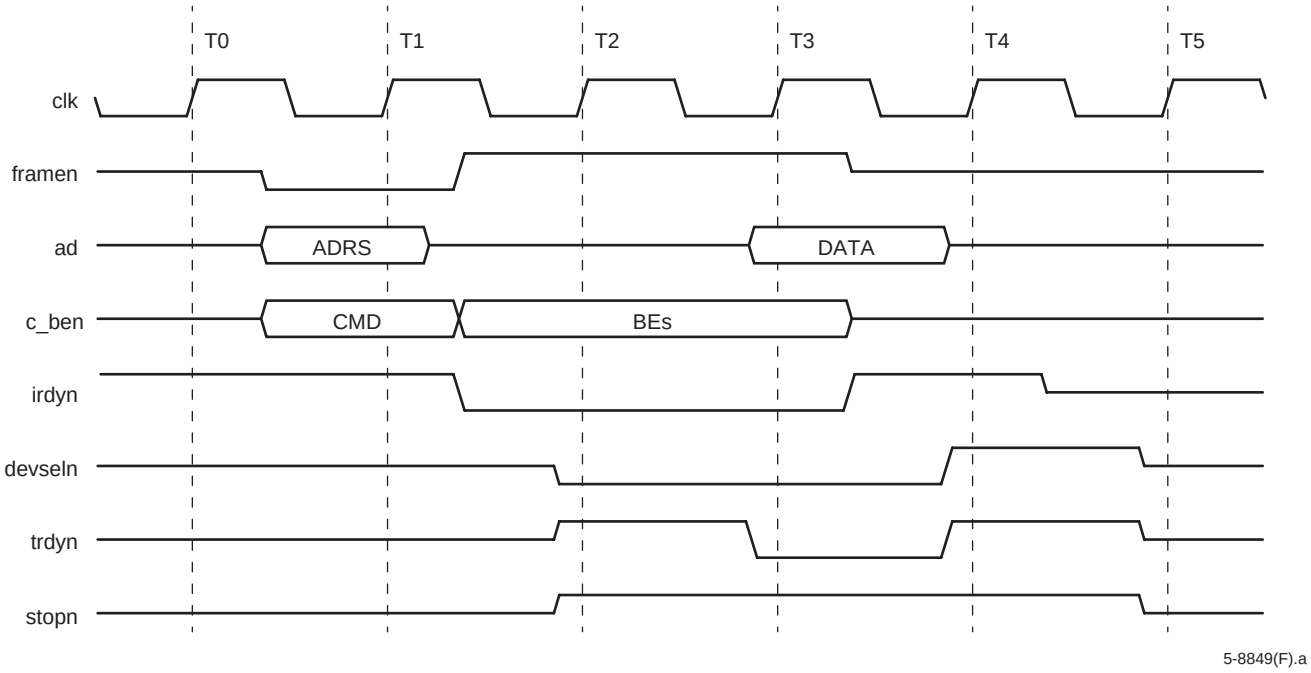


Figure 29. Master Read Single (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

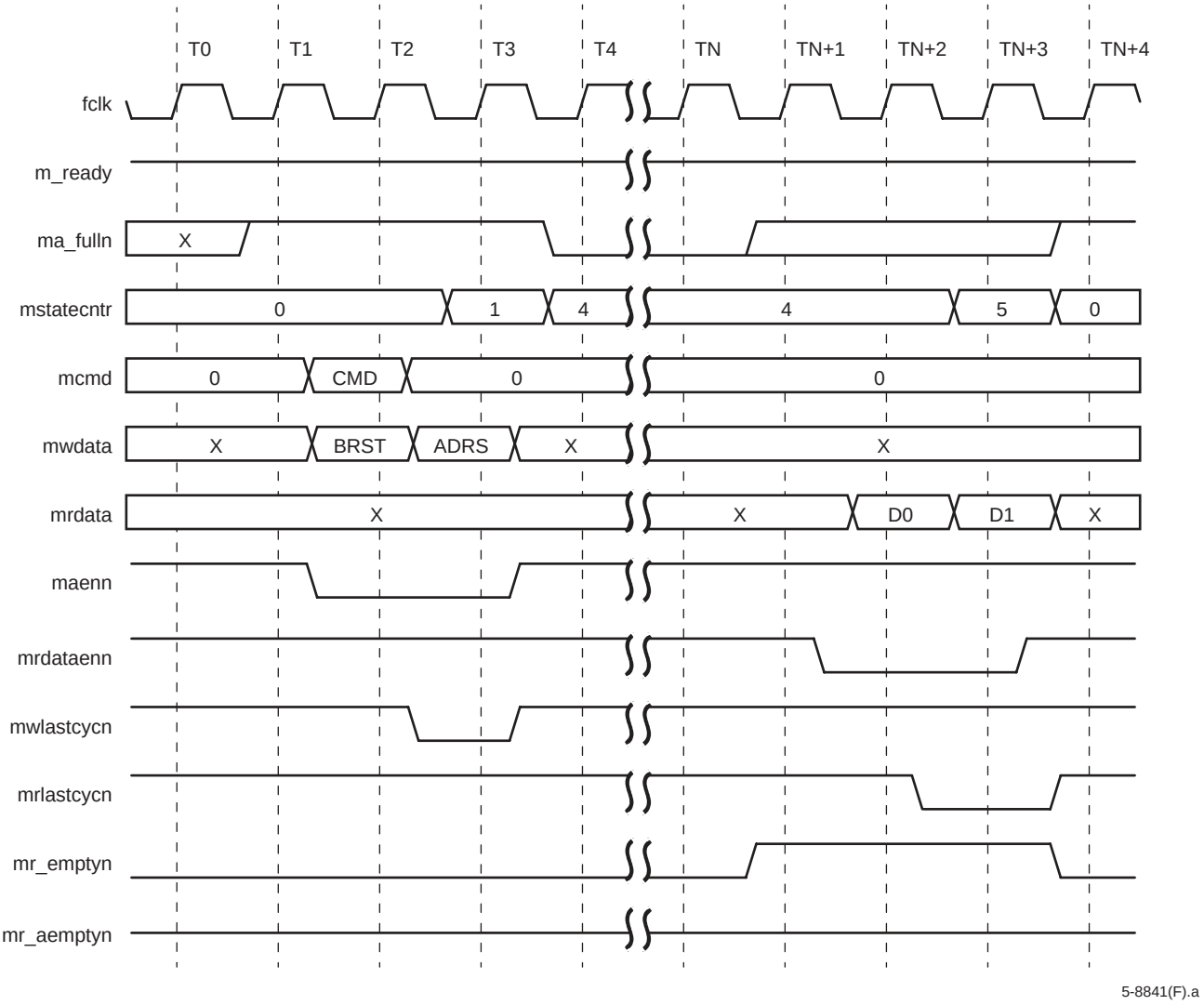


Figure 30. Master Read Single Quadword (FPGA Bus, Quad-Port, Specified Burst Length, 32-Bit Address)

PCI Bus Core Detailed Description Quad Port (continued)

Master Read, Burst Transaction

Figure 32 (FPGA bus) and Figure 31 (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 32, the transaction is initiated by the FPGA application asserting Master address enable (**maenn**), while providing the command and burst length on bus **mwdata**. On the next clock, the FPGA application provides the DWORD address and asserts **mwlastcycn**. On the third cycle, both **maenn** and **mwlastcycn** are deasserted. PCI bus activity now begins as shown in Figure 31. Once data is transferred on the PCI bus and **mr_emptyn** is deasserted high, the FPGA application asserts **mrdataenn** and eight DWORDs of data are transferred on bus **mrdata**.

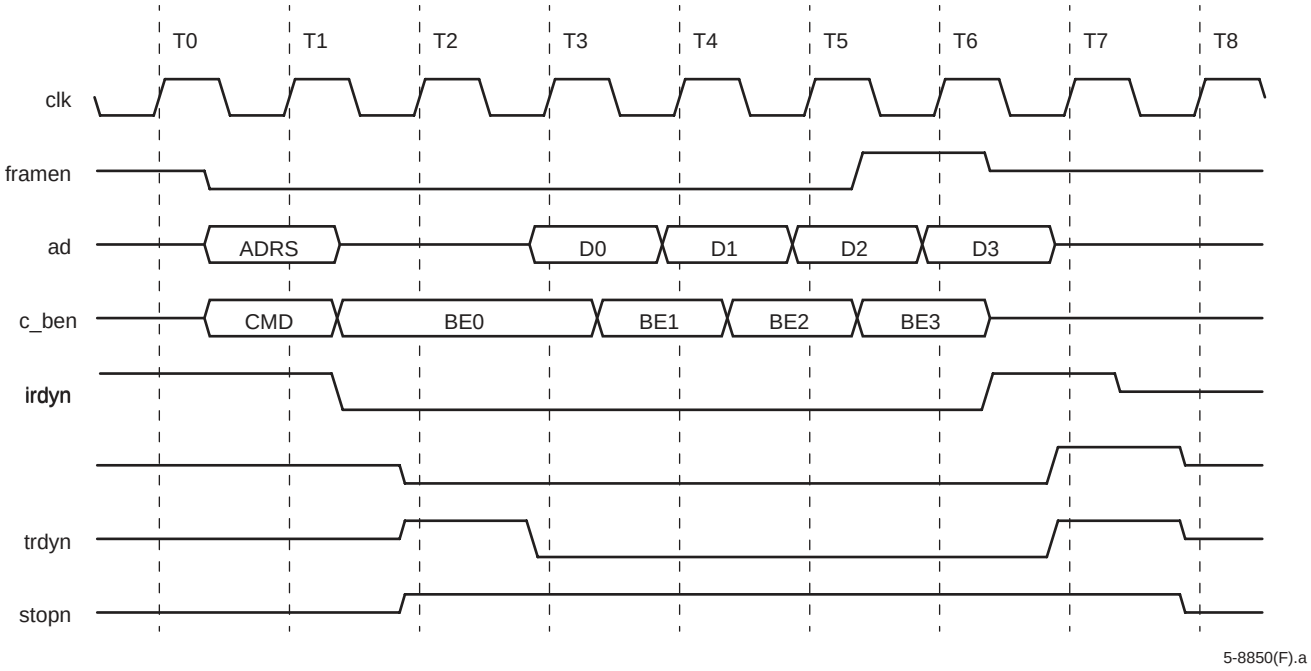


Figure 31. Master Read 32-Byte Burst (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

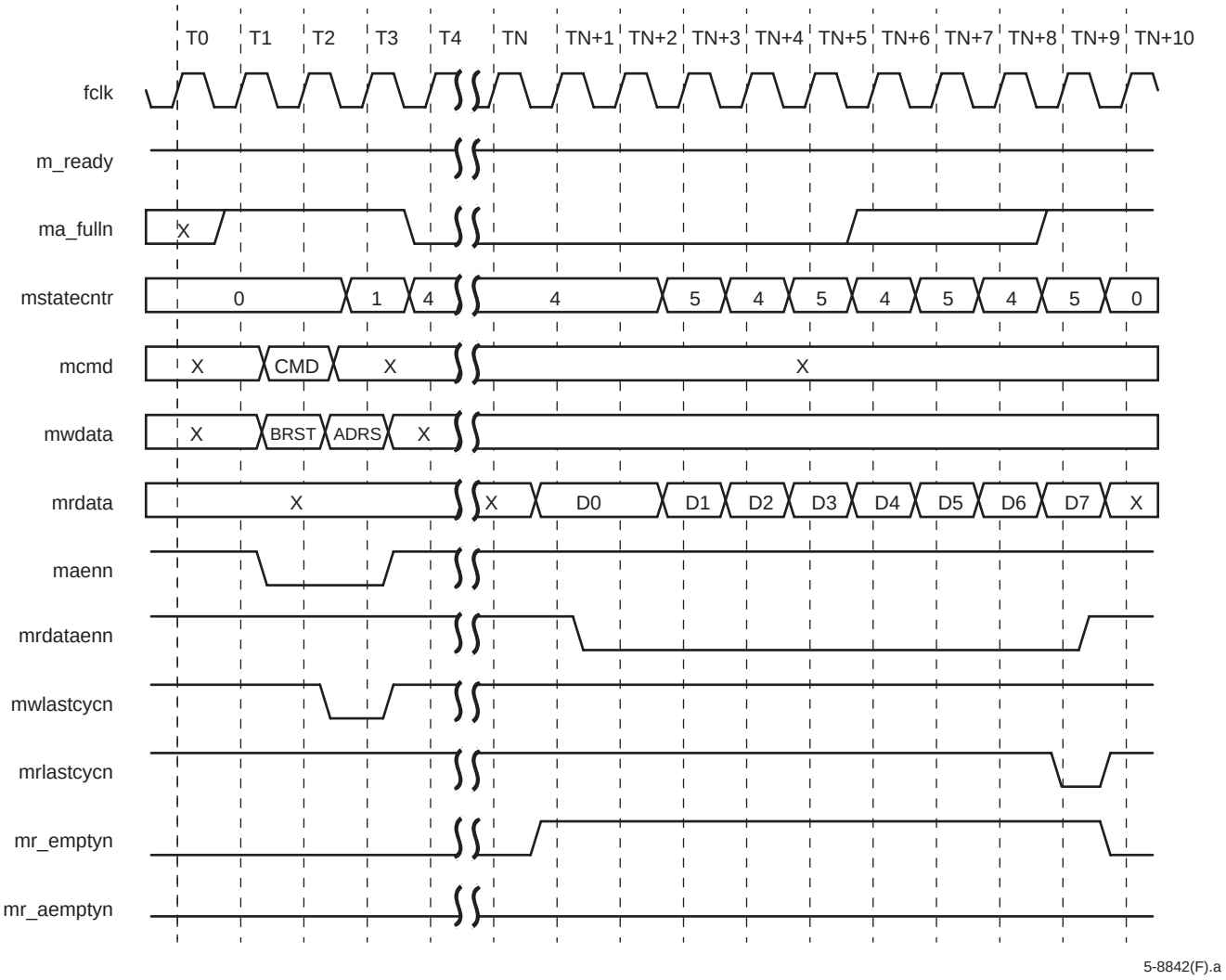


Figure 32. Master Read 32-Byte Burst (FPGA Bus, Quad-Port, Specified Burst Length, 32-Bit Address)

PCI Bus Core Detailed Description Quad Port (continued)

Table 37. Quad-Port Master Read, Duplicate Burst Length and 16-Bit Address

mstatecntr	Next State of mstatecntr	Description	Bus	maenn	mwlastcycn	mrlastcycn	mrdataenn
0	0	Idle	—	1	1	1	1
0	4	BE[7:0], Address[15:0]	mwdata[35:0]	0	0	1	1
4	5 or 0	Data[31:0]	mrdata[31:0]	1	1	1	0
5	4 or 0	Data[63:32]	mrdata[31:0]	1	1	0*	0

* **mrlastcycn** is 0 on the last data DWORD transfer.

Table 38. Quad-Port Master Read, Specified Burst Length and 64-Bit Address

mstatecntr	Next State of mstatecntr	Description	Bus	maenn	mwlastcycn	mrlastcycn	mrdataenn
0	0	Idle	—	1	1	1	1
0	1 or 4	BE[7:0], Burst Length	mwdata[35:0]	0	1	1	1
1	2 or 4	Address[31:0]	mwdata[31:0]	0	1	1	1
2	4	Address[63:32]	mwdata[31:0]	0	0	1	1
4	5 or 0	Data[31:0]	mrdata[31:0]	1	1	1	0
5	4 or 0	Data[63:32]	mrdata[31:0]	1	1	0*	0

* **mrlastcycn** is 0 on the last data DWORD transfer.

PCI Bus Core Detailed Description
Quad Port (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 39 describes 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[2:0]**, 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** 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 these state counters in this case.

The data required from the FPGA application by the PCI core on bus **trdata** 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 25 lists the values of the state counter **tstatecntr** and the appropriate accompanying data.

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**.

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. If the FIFO does go full, the core will do one of two things. If **twburstpendn** is deasserted high, the target will disconnect. If **twburstpendn** is asserted low, the target will assert up to eight wait-states and then disconnect if still full. The FIFO full flag is not generally used in user designs. If it is, however, keep in mind that it is synchronous to **pciclk**.

Bursting

Signal **twlastcycn** tells the FPGA application whether the current write is a burst. The FPGA application continues to unload data from the FIFO as long as **twlastcycn** is inactive. The bursting will continue until either **twlastcycn** is received, the FIFO becomes full, or the BAR boundary is crossed. There is no fixed maximum transfer word count.

PCI Bus Core Detailed Description
Quad Port (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 **devseln** and **trdyn** 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**. The signal **deltrn** was designed to be a static signal. This signal should be tied off high or low depending upon whether the FPGA application wishes to run delayed transactions. When asserting **deltrn** low, the PCI core will execute delayed transactions for I/O writes as well as all target reads. In delayed transaction mode, 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, 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. Internal to the ASIC, there is also a 15-bit time-out timer (known as the discard timer). During a delayed I/O write transaction, this counter will begin counting. If the same master does not come back within $2^{15} - 1$ **pciclk**'s to complete the write, this timer will expire, resetting the target state machines and setting a user side signal (**disctimerexp** = 1). From this point forward, any master performing a write (including the original master coming back to complete the transfer) will be treated as a new transaction. If monitoring this signal, keep in mind that **disctimerexp** is synchronous to **pciclk** and asserts high for one clock period.

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**.

Reset

The FPGA application can apply the PCI core's reset signal **tfifoclrn** to place the core's target logic in a known state. Normally, the clear signal will not be used unless a severe problem has occurred in the data flow. The **tfifoclrn** signal is synchronous with **fclk** and must be asserted for a minimum of three clock periods. During reset, the **t_ready** signal will go low. After the reset signal is deasserted high, **t_ready** will continue to be low for 8—10 clock periods. The FPGA application should not continue normal operation until **t_ready** is asserted high.

Understanding and Using the pci_tcfg_stat Status Signals

On the Target interface, there are two signals that control and provide status to the FPGA application. The signal **pci_tcfg_stat** provides the status and **tcfgshiftenn** controls what information the status line provides. The **pci_tcfg_stat** signal is always active and duplicates the status contained in configuration status register at location offset 0x04, bits 24, 28, and 29. To use this status output, the FPGA application must keep **tcfgshiftenn** = 1. When high, **pci_tcfg_stat** provides the wired-OR of the three status lines. If **pci_tcfg_stat** gets set to a 1, indicating an error, then the FPGA application may set **tcfgshiftenn** = 0 to determine individual status. Once low, the **pci_tcfg_stat** signal will output target abort signaled on the first clock, system error signaled on the second clock, and parity error detected on the third clock.

PCI Bus Core Detailed Description Quad Port (continued)

Initiating Target Aborts

There may be a need in an application to initiate a target abort condition on the PCI bus. In general, this is asserted for only the most severe cases. The interface signal, **fpga_tabort**, is used for this purpose. From the PCI core's point of view, it needs to know whether to perform a target abort at the very beginning of a transaction, so it is not possible to have a transaction started, and then assert the **fpga_tabort** signal. The signal **fpga_tabort** needs to be asserted before the transaction begins, and it was not designed to be toggled on and off from transaction to transaction. Once an FPGA application determines that it wants to apply a target abort to any master that accesses it, it would assert the **fpga_tabort** signal high. All future target accesses will be terminated in an abort. In generating this signal, keep in mind that this signal needs to be synchronous to **pciclk**.

Initiating PCI Target Retries

In contrast to target abort, many applications may require to assert PCI target retries. In general, this may be asserted for times when the FPGA application is temporarily busy and unavailable to service PCI requests. The interface signal, **fpga_tretryn**, is used for this purpose. From the PCI core's point of view, it needs to know whether to perform a target retry at the very beginning of a transaction, so it is not possible to have a transaction started and then assert the **fpga_tretryn** signal. The signal **fpga_tretryn** needs to be asserted before the transaction begins, and it was not designed to be toggled on and off from transaction to transaction. Once an FPGA application determines that it wants to apply a target retry to any master that accesses it, it would assert the **fpga_tretryn** signal low. All future target accesses will be terminated in a retry (disconnect without data). On the FPGA application side, no activity will occur. In generating this signal, keep in mind that this signal needs to be synchronous to **pciclk**.

PCI Bus Core Detailed Description Quad Port (continued)

Target Write to Configuration Space Transaction

Figure 33 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 **trdyn** one cycle after **devseln** is asserted.

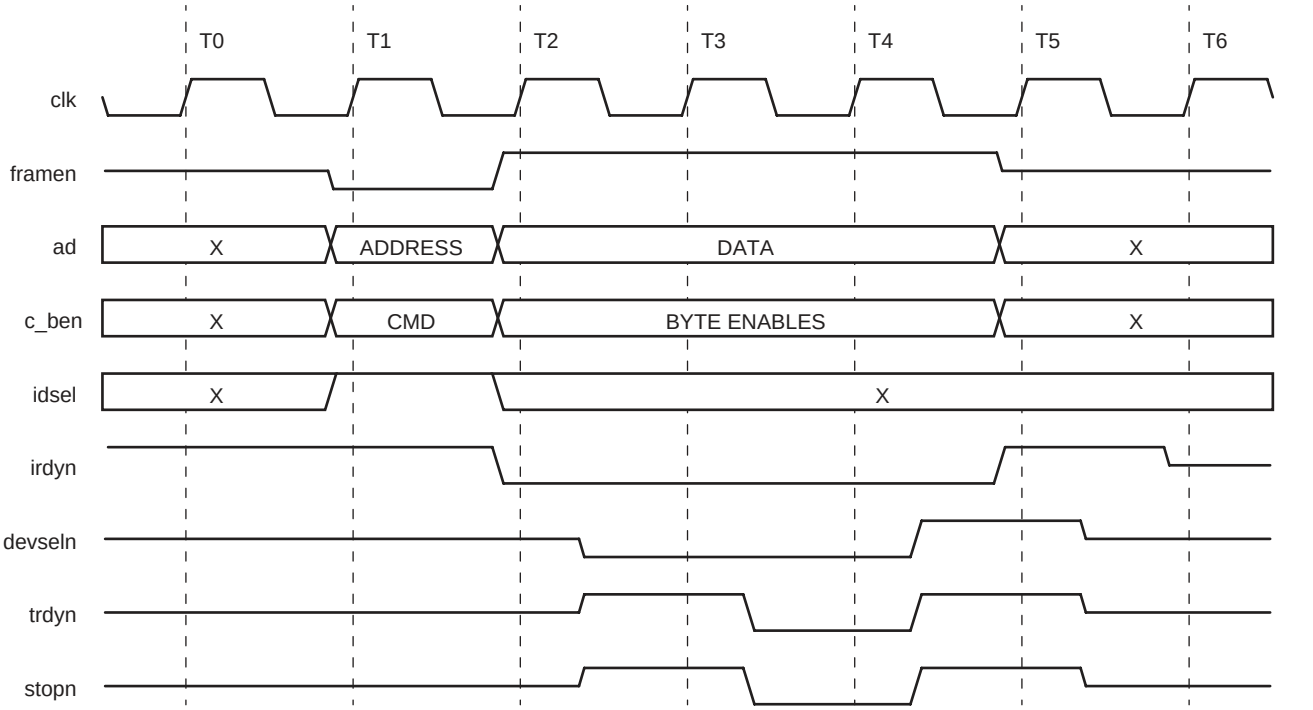


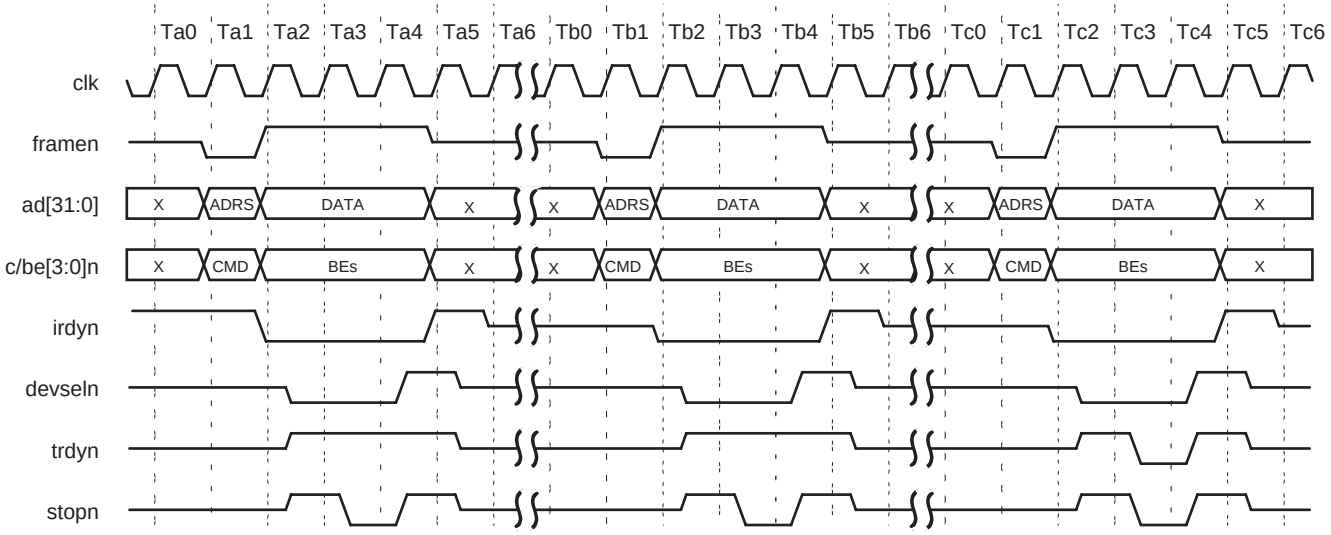
Figure 33. Target Configuration Write (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

Target Write I/O, Delayed Transaction

Figure 34 (PCI bus) and Figure 36 (FPGA bus) show the timing 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 34, 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 36. 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 34. 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 **trdyn** accompanies **stopn** to indicate the disconnect with data.

The timing on the FPGA interface (Figure 36) 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 **twdata**. The FPGA application responds by asserting target address enable (**taenn**) and accepting the command and subsequent lower DWORD address on bus **twdata**. On the next clock, the upper DWORD address is received along with **twlastcycn**. This is followed by deassertion of **taenn**, assertion of Target write data enable (**twdataenn**), and the receiving of the data on bus twdata.



5-7372(F).a

Figure 34. Target I/O Write, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

Target Write Nonburst Transaction

Figure 35 (PCI bus) and Figure 36 (FPGA bus) show the timing on the PCI and FPGA interfaces, respectively, for a Target memory nonburst write transaction. The timing on the PCI interface (Figure 35) is similar to that of an I/O write except that, since bursts to memory space are allowed, the signal **stopn** is not asserted. The FPGA interface timing is as shown in Figure 36, and is the same as the timing for memory and I/O write transactions.

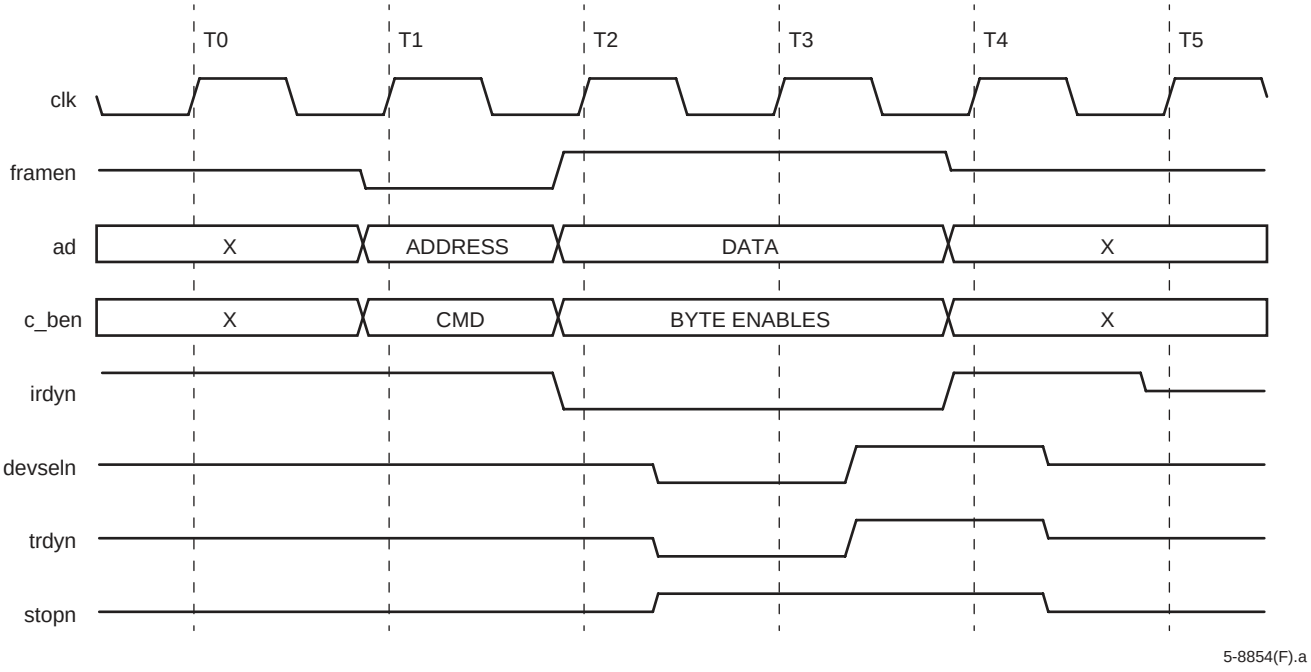


Figure 35. Target Write Memory Single (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

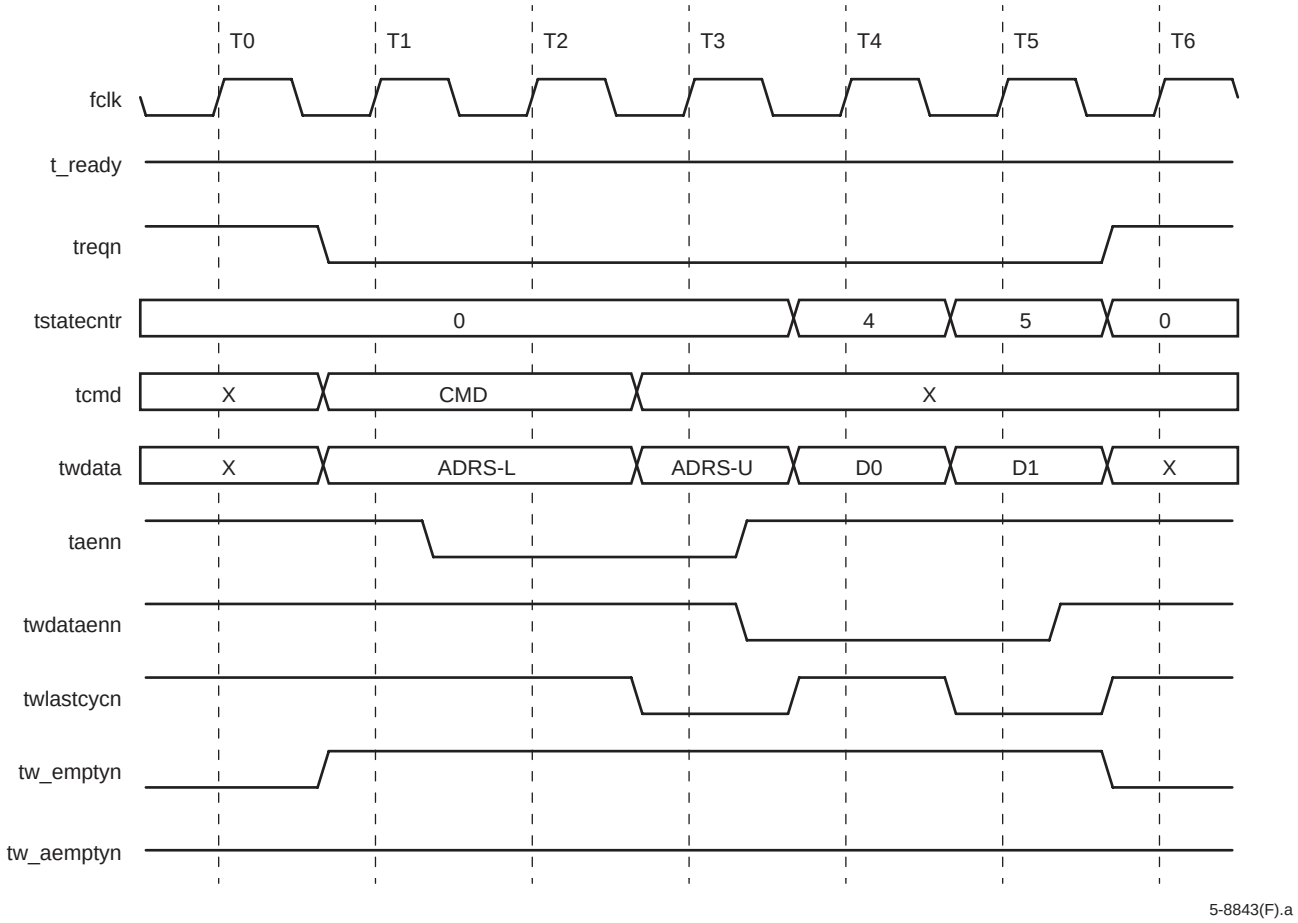


Figure 36. Target Write Single Quadword (FPGA Bus, Quad-Port, 64-Bit Address)

PCI Bus Core Detailed Description Quad Port (continued)

Target Write Memory Burst Transaction

Figure 37 (PCI bus) and Figure 38 (FPGA bus) show the timing for a Target memory write burst of four Quadwords. The timing on the PCI interface (Figure 37) is typical for a medium-speed decode Target. Note that **trdyn** is asserted at the earliest possible time, which is concurrent with assertion of **devseln**. 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 37) 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 twdata. This is followed by deassertion of **taenn**, assertion of Target write data enable (**twdataenn**), and the receiving of the data on bus **twdata**. The FPGA application is informed that the last 32 bits of data is being presented when Target write burst (**twlastcycn**) is asserted.

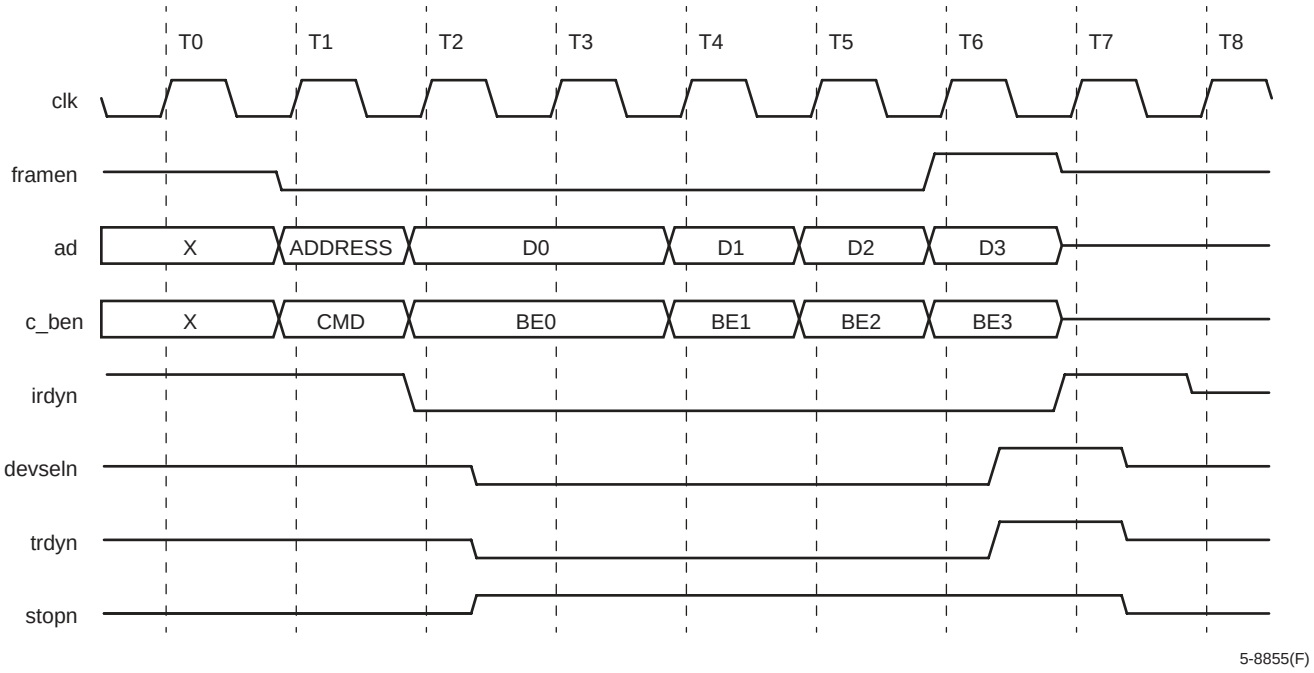


Figure 37. Target Memory Write 32-Byte Burst (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

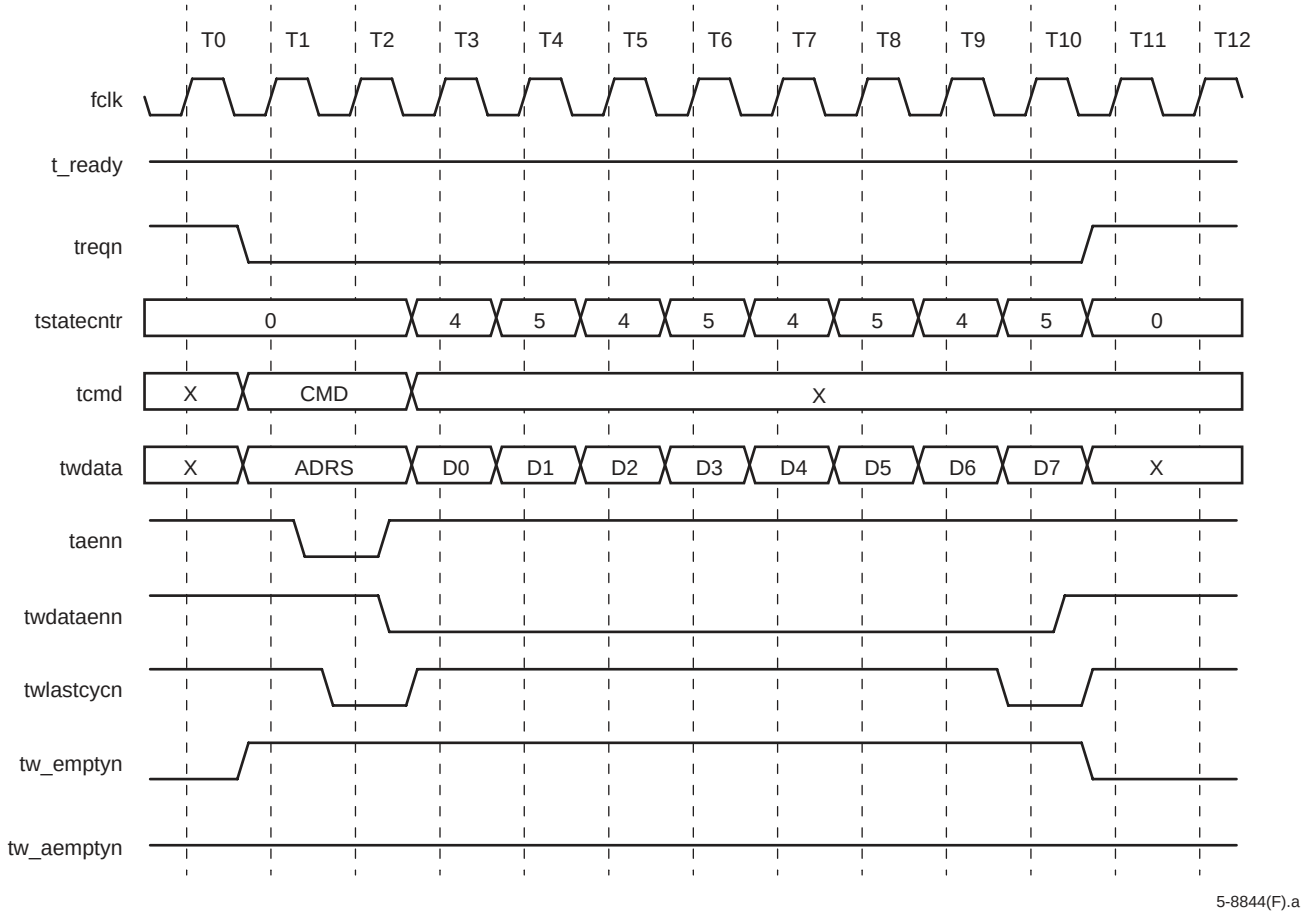


Figure 38. Target Write Memory 32-Byte Burst (FPGA Bus, Quad-Port, 32-Bit Address)

Table 39. Quad-Port Target Write

tstatecntr	Next State of tstatecntr	Description	Bus	treqn	twlastcycn	taenn
0	0	Idle	—	1	1	1
0	1 or 4	Address[31:0]	twdata[35:0]	0	1	0
1	4	Address[63:32]	twdata[32:0]	0	0	0
4	5 or 0	Data[31:0], BE[3:0]	twdata[35:0]	0	1	1
5	4 or 0	Data[63:32], BE[7:4]	twdata[35:0]	1†	0*	1

* **treqn** is deasserted high on the last data DWORD.
† **twlastcycn** is asserted low on the last data DWORD.

PCI Bus Core Detailed Description
Quad Port (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 40 describes 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**. 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. This is explained in the Delayed Transactions section.

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. If the FIFO does go empty, the core will do one of two things. If **twburstpendn** is deasserted high, the target will disconnect. If **twburstpendn** is asserted low, the target will assert up to eight wait-states and then disconnect if still empty. The FIFO empty flag is not generally used in user designs. If it is, however, keep in mind that it is synchronous to **pciclk**.

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. Bursts will continue until either **trlastcycn** is received, the BAR boundary is crossed, or a 2^{18} physical page address is crossed.

PCI Bus Core Detailed Description
Quad Port (continued)

Delayed Transactions

Delayed transactions can be executed by asserting **deltrn** low. 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.

Delayed transactions are very similar to a target retry except that the address is actually stored in the core. Delayed transactions are usually implemented in systems where the user side interface cannot supply the first piece of data in 16 clock cycles. An example of this may be that the user interface is connected to another bus system. On a PCI target read, the user interface must arbitrate for the user bus and get the necessary data. Delayed transaction mode is used when the **deltrn** bit is asserted low. This bit is not a dynamic bit. It must be set ahead of a transaction occurring. It is not recommended to switch between delayed and non-delayed transactions dynamically.

When **deltrn** is low, a master read request is terminated in a target retry. On the user interface side, the address is stored in the target address FIFO, and **treqn** is asserted low. All future master requests are terminated in a retry until the address is read out of the FIFO, data is loaded into the FIFO, and the same request comes back to complete the transaction. In generating this signal, keep in mind that this signal needs to be synchronous to **pciclk**.

Another option the designer has using delayed transactions is to use the signal **trpcihold**. The signal **trpcihold** should be used when the user side interface is slow loading requested data, and the designer wishes to utilize the PCI in the most efficient manner. Without this signal, an external master will request data and hold onto the PCI bus until either it has received it or it gets terminated by latency timers, etc. A more efficient method to utilize the PCI bus is to assert **trpcihold**, load the FIFOs, and then deassert it. While the **trpcihold** signal is asserted, the core thinks that the FIFOs stay empty even though they are slowly filling with data. Requests from an external master are terminated in retries. When the **trpcihold** signal is deasserted (or the FIFO becomes full), the core will allow an external master to come in, the data will be burst across the PCI bus as fast as the master will allow, and the transaction will end. In generating **trpcihold**, keep in mind that this signal needs to be synchronous to **pciclk**.

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**. When **treqn** deasserts, the FPGA application must immediately deassert **trdataenn**.

Reset

The FPGA application can apply the PCI core's reset signal **tfifoclrn** to place the core's target logic in a known state. Normally, the clear signal will not be used unless a severe problem has occurred in the data flow. The **tfifoclrn** signal is synchronous with **fclk** and must be asserted for a minimum of three clock periods. During reset, the **t_ready** signal will go low. After the reset signal is deasserted high, **t_ready** will continue to be low for 8—10 clock periods. The FPGA application should not continue normal operation until **t_ready** is asserted high.

PCI Bus Core Detailed Description
Quad Port (continued)

Understanding and Using the pci_tcfg_stat Status Signals

On the Target interface, there are two signals that control and provide status to the FPGA application. The signal **pci_tcfg_stat** provides the status, and **tcfgshiftenn** controls what information the status line provides. The **pci_tcfg_stat** signal is always active and duplicates the status contained in configuration status register at location offset 0x04, bits 24, 28, and 29. To use this status output, the FPGA application must keep **tcfgshiftenn** = 1. When high, **pci_tcfg_stat** provides the wired-OR of the three status lines. If **pci_tcfg_stat** gets set to a 1, indicating an error, then the FPGA application may set **tcfgshiftenn** = 0 to determine individual status. Once low, the **pci_tcfg_stat** signal will output target abort signaled on the first clock, system error signaled on the second clock, and parity error detected on the third clock.

Initiating Target Aborts

There may be a need in an application to initiate a target abort condition on the PCI bus. In general, this is asserted for only the most severe cases. The interface signal, **fpga_tabort**, is used for this purpose. From the PCI core's point of view, it needs to know whether to perform a target abort at the very beginning of a transaction, so it is not possible to have a transaction started, and then assert the **fpga_tabort** signal. The signal **fpga_tabort** needs to be asserted before the transaction begins, and it was designed to be toggled on and off from transaction to transaction. Once an FPGA application determines that it wants to apply a target abort to any master that accesses it, it would assert the **fpga_tabort** signal high. All future target accesses will be terminated in an abort. In generating this signal, keep in mind that this signal needs to be synchronous to **pciclk**.

PCI Bus Core Detailed Description Quad Port (continued)

Target Read from Configuration Space

Figure 39 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 **trdyn** one cycle after **devseln** is asserted.

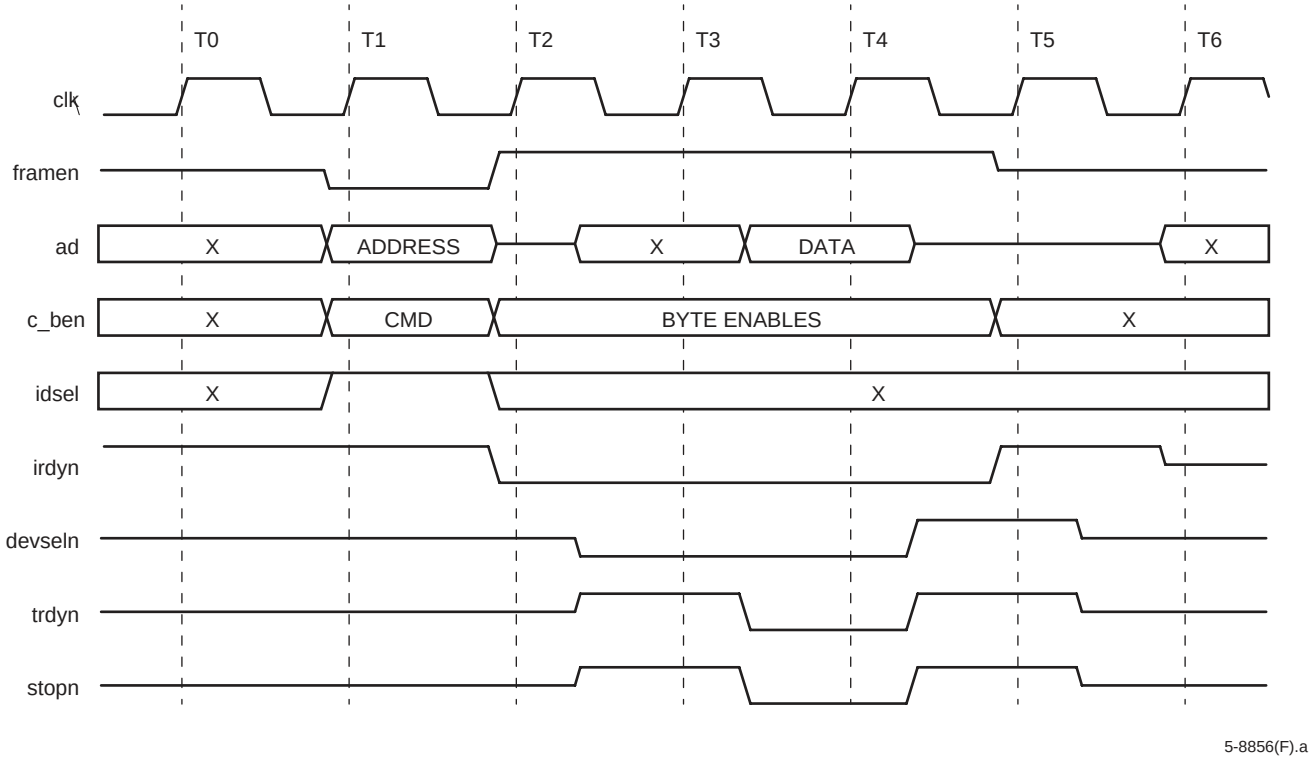


Figure 39. Target Configuration Read (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

Target Read I/O, Delayed Transaction

Figure 40 (PCI bus) and Figure 43 (FPGA bus) 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 40, 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 43. All subsequent read or write requests to memory or I/O space will result in retries, as shown in the second transaction of Figure 40. 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 **trdyn** accompanies **stopn** to indicate the disconnect with data.

The timing on the FPGA interface (Figure 40) 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 **twdata**. The FPGA application responds by asserting Target address enable (**taenn**) and accepting the command and lower DWORD address on bus twdata, after which **taenn** is deasserted. On the next clock, the upper DWORD address is transferred. The FPGA application then accesses the requested data, asserts Target read data enable (**trdataenn**), and transmits the data on bus **trdata**.

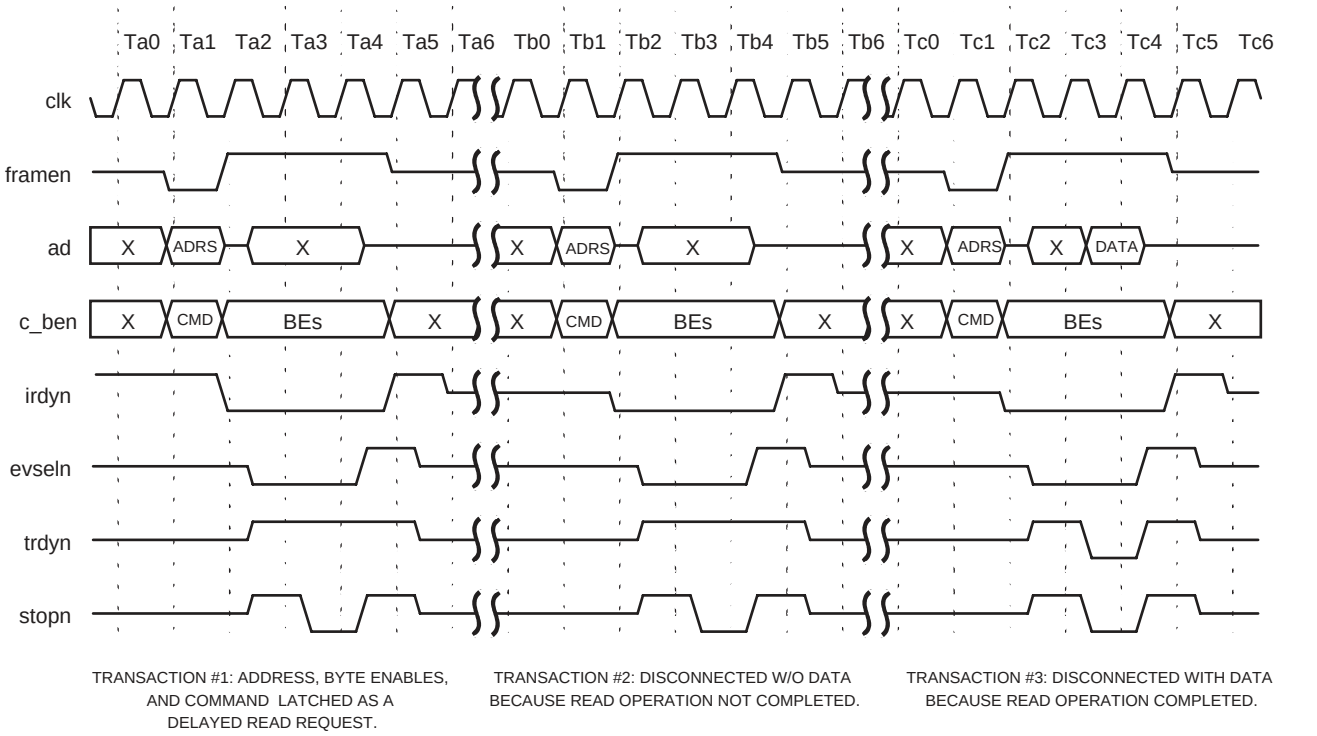


Figure 40. Target I/O Read, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

Target Read I/O, No Delayed Transaction

Figure 41 (PCI bus) and Figure 43 (FPGA bus) 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 41 is identical to the timing of the third (final) transaction of Target I/O read, delayed transaction (Figure 40), which shows a Target I/O read with delayed transaction. Also, the FPGA interface timing is as shown in Figure 43, regardless of whether delayed transactions are enabled.

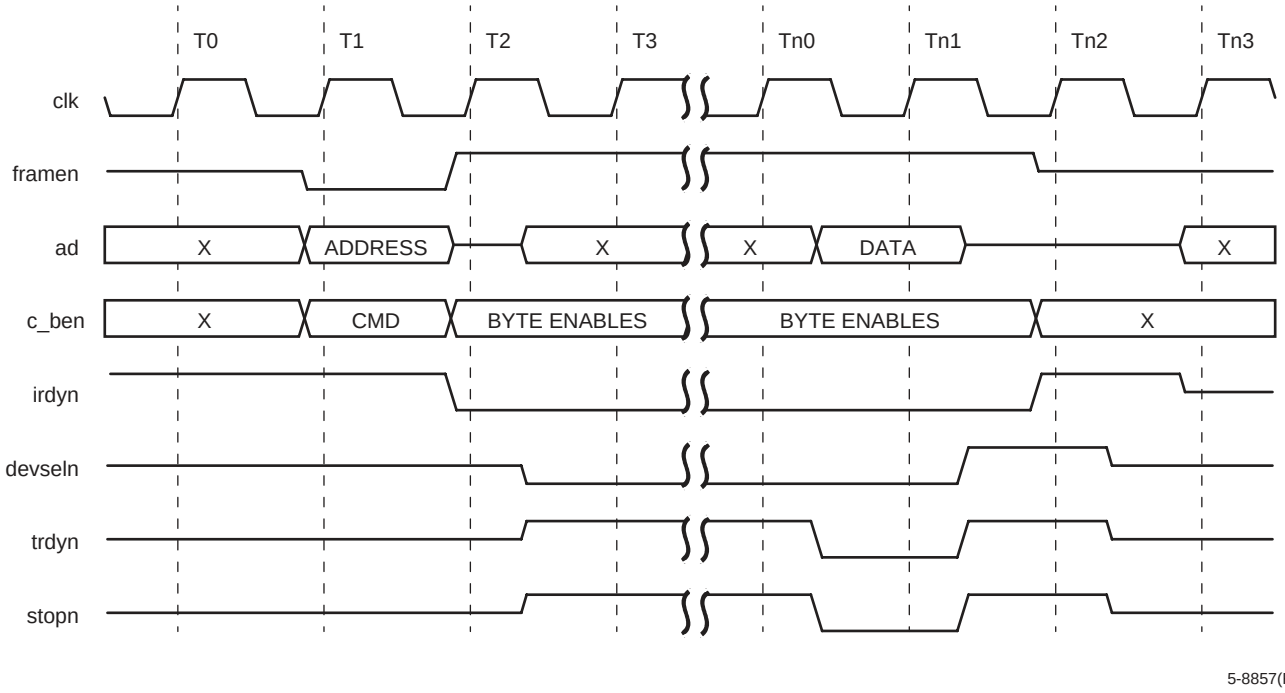


Figure 41. Target I/O Read, Not Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

Target Read Memory, Nonburst, Delayed Transaction

Figure 42 (PCI bus) and Figure 43 (FPGA bus) 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 42) is similar to that of an I/O read (Figure 40) 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 43 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.

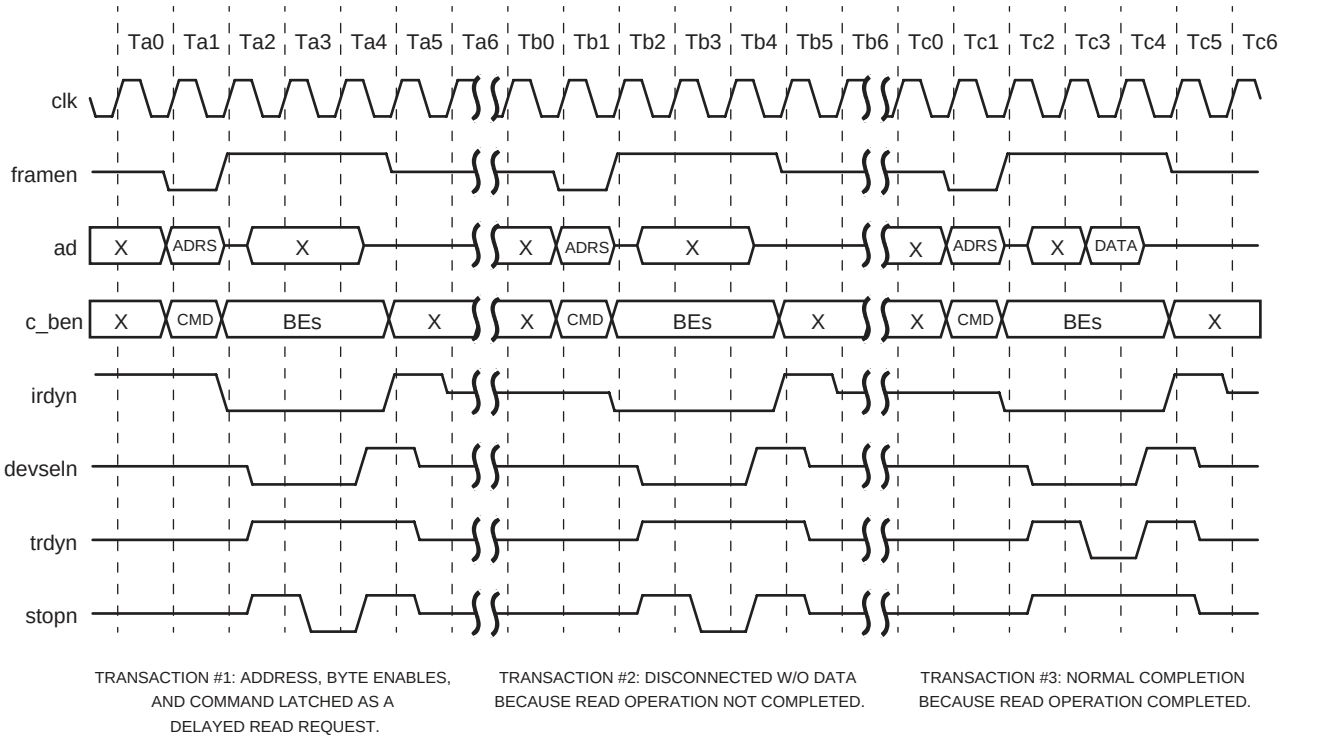


Figure 42. Target Memory Single Read, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

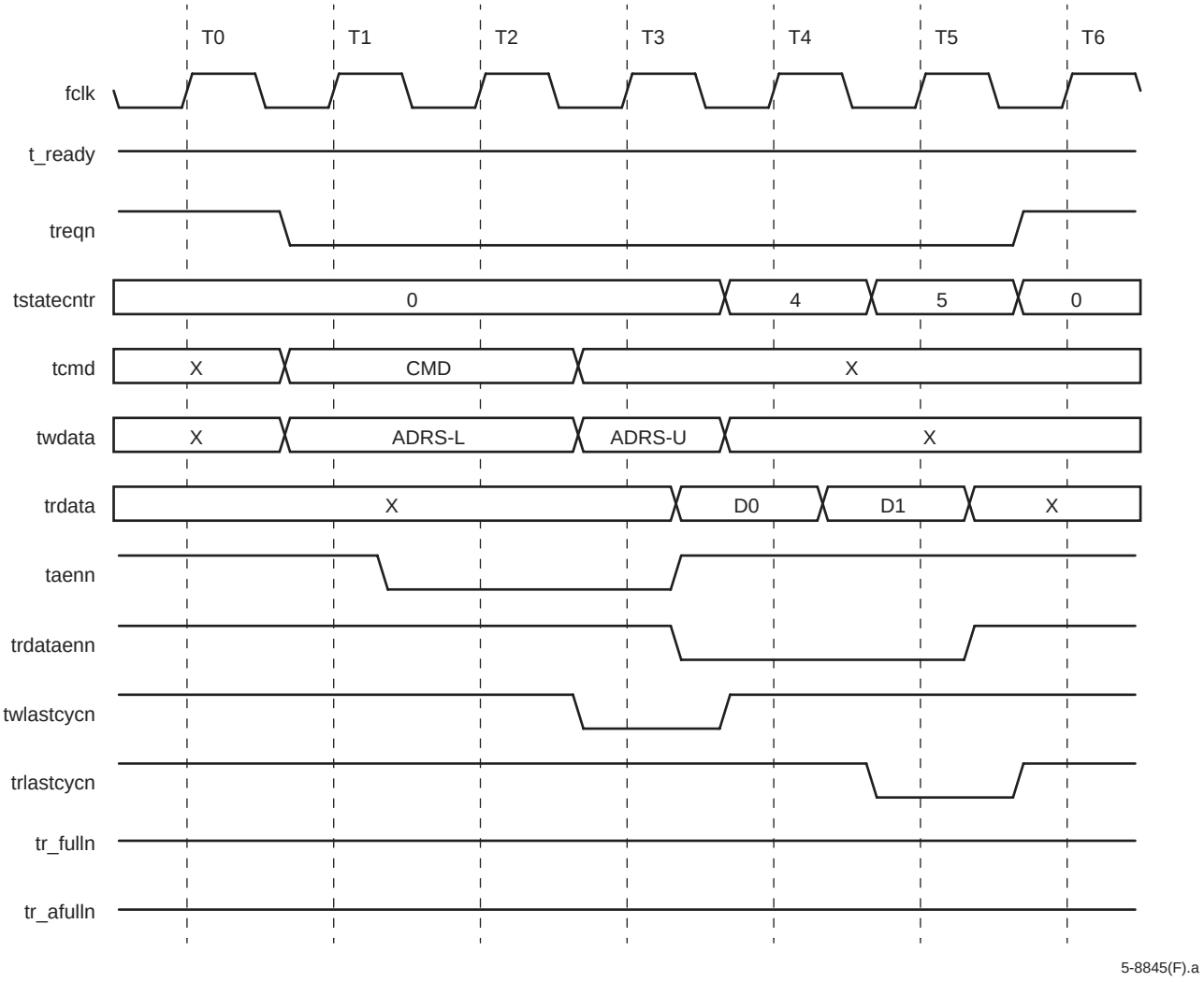


Figure 43. Target Read Single (FPGA Bus, Quad-Port, 64-Bit Address)

PCI Bus Core Detailed Description Quad Port (continued)

Target Read Memory, Nonburst, No Delayed Transaction

Figure 44 (PCI bus) and Figure 43 (FPGA bus) 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 44. Here the PCI core accepts the transaction without issuing a retry but does not immediately assert **trdyn**. Wait-states are inserted until the requested data is placed in the Target read FIFO, at which time **trdyn** 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 43, and is the same as the timing in the accesses of Target I/O read, delayed transaction, Target I/O read, no delayed transaction, and Target read memory nonburst, delayed transaction.

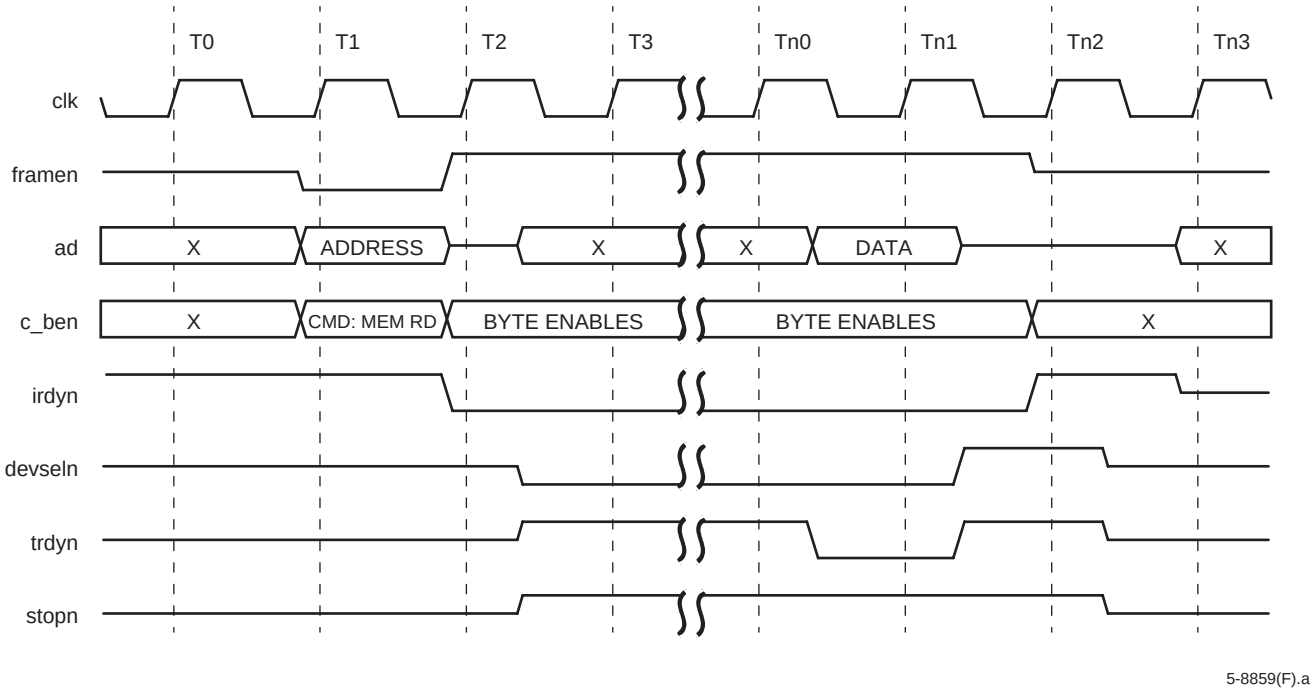
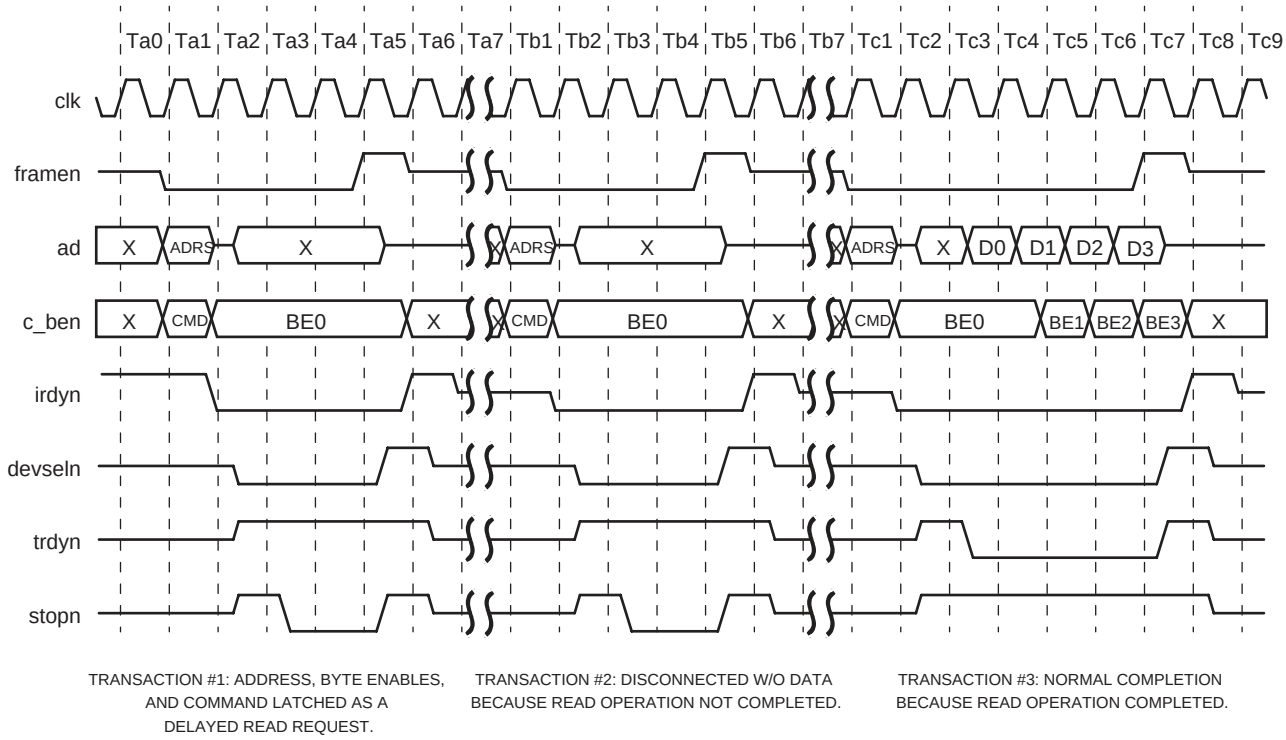


Figure 44. Target Memory Read Single, Not Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

Target Read Memory Burst, Delayed Transaction

Figure 45 (PCI bus) and Figure 46 (FPGA bus) 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 45), 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 **devseln**. 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 45. 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 45. The FPGA interface timing is as shown in Figure 46. This is similar to the timing for a Target non-burst read as shown in Figure 43 except that multiple data cycles are required as long as **trlastcycn** is inactive-high.



5-8862(F).a

Figure 45. Target Memory Read 32-Byte Burst, Delayed (PCI Bus, 64-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

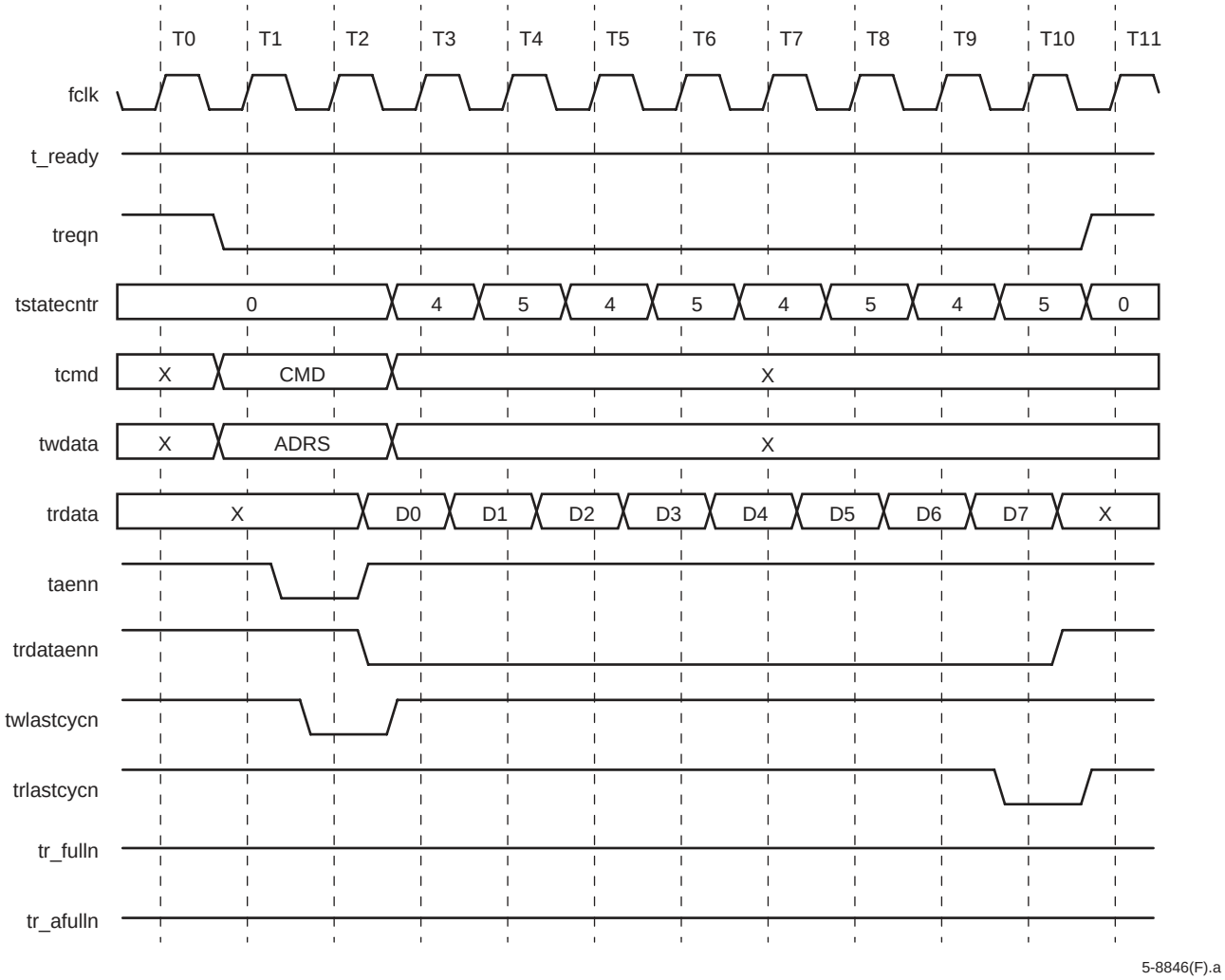


Figure 46. Target Read Memory 32-Byte Burst (FPGA Bus, Quad-Port, 32-Bit Address)

PCI Bus Core Detailed Description Quad Port (continued)

Target Read Memory Burst, No Delayed Transaction

Figure 47 (PCI bus) and Figure 46 (FPGA bus) show the timing for a Target memory burst read of four Quadwords handled as a nondelayed transaction. Figure 47 shows the timing on the PCI interface is similar to that of an I/O read (Figure 40) 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.

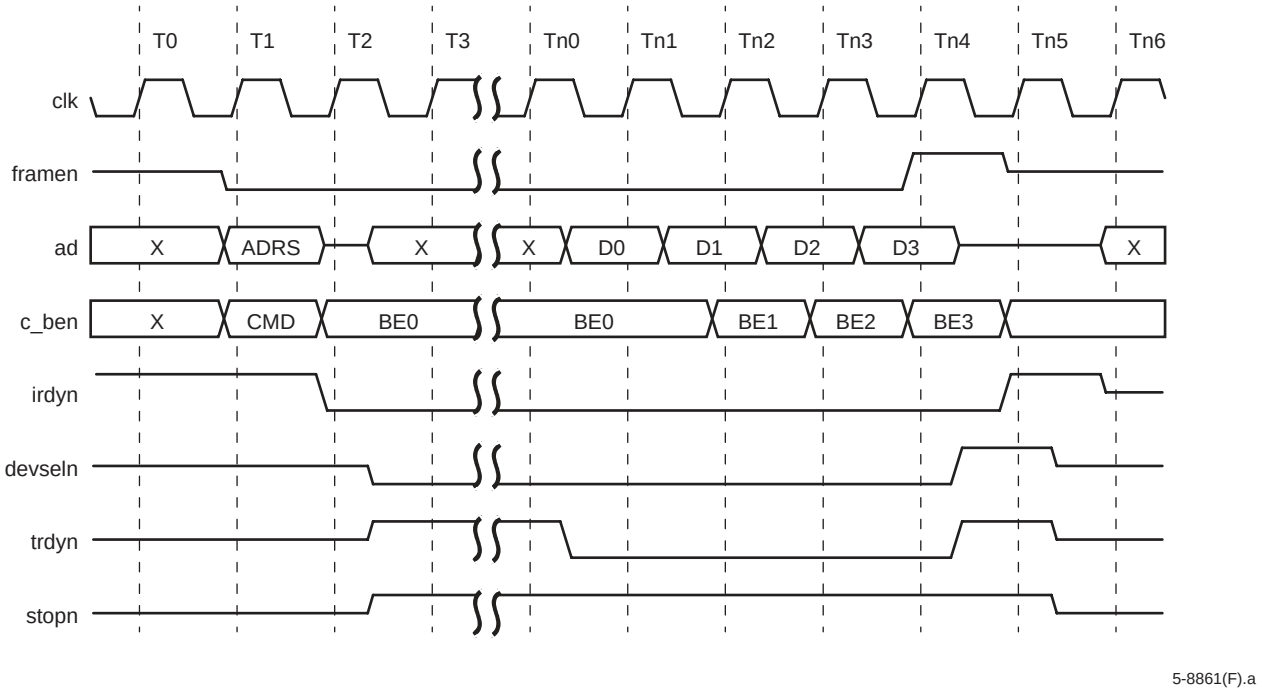


Figure 47. Target Read Memory Burst, No Delayed (PCI Bus, 32-Bit)

PCI Bus Core Detailed Description Quad Port (continued)

Table 40. Quad-Port Target Read

tstatecntr	Next State of tstatecntr	Description	Bus	treqn	taenn	trdataenn	twlastcycn	trlastcycn
0	0	Idle	—	1	1	1	1	1
0	1 or 4	Address[31:0]	datatofpgax[7:0] datatofpga[63:0]	0	0	1	1	1
1	4	Address[63:32]	datatofpga[63:0]	0	0	1	1	0
4	5 or 0	Data[31:0]	datafmfpga[31:0]	0	1	0	1	1
5	4 or 0	Data[63:32]	datafmfpga[31:0]	1*	1	0	0†	1

* **treqn** is deasserted high on the last data DWORD.
† **twlastcycn** is asserted low on the last data DWORD.

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 41 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 42 further details the content and function of each register in the PCI configuration space.

Table 41. Configuration Space Layout

31 — 16	15 — 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

Configuration Space of the PCI Core (continued)

Table 42. Configuration Space Assignment

Bytes	Width	Bit	Description	Read/Write	Initial Value
00—01	16	—	Vendor ID	Read Only	11C1h (Lattice)
02—03	16	—	Device ID	Read Only	5401h (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	*
		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 SERRn	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 PERRn Detected	†	0
		10—9	devseln Timing	Read Only	01b (medium)
		11	Target Abort Signaled	†	0
		12	Target Abort Received	†	0
		13	Master Abort Received	†	0
		14	System Error Signaled	†	0
		15	Parity Error Detected	† ‡	0
08	8	—	Revision ID	Read Only	*
09—0B	24	—	Class Code	Read Only	*
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
0E	8	—	Header Type	Read Only	00h
0F	8	—	BIST	Read Only	zeros
10—27	192	—	BAR	§	*
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	*
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

* These values are intended to be custom assigned, per the intended application, by assigning constants via the FPGA configuration bit stream.

† 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.

‡ This bit is set when the device detects any type of parity error from its own master or target.

§ 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).

Configuration Space of the PCI Core (continued)

Table 42. Configuration Space Assignment (continued)

Bytes	Width	Bit	Description	Read/Write	Initial Value
3C	9	—	Interrupt Line	Read/Write	zeros
3D	8	—	Interrupt Pin	Read Only	01h (INTAn)
3E	8	—	Min_Gnt	Read Only	*
3F	8	—	Max_Lat	Read Only	*
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	**
		1	Done — FPGA's DONE	Read Only	**
		0	Mode — PCI Core Mode	Read Only	0
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	††	Footnote 7
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 — ENUMn Status - Insertion	‡‡	1
		6	EXT — ENUMn Status - Extraction	‡‡	0
		5	Reserved	Read Only	0
		4	Reserved	Read Only	0
		3	LOO	Read/Write	0
		2	Reserved	Read Only	0
		1	EIM — ENUMn Signal Mark	Read/Write	0
		0	Reserved	Read Only	0

* These values are intended to be custom assigned, per the intended application, by assigning constants via the FPGA configuration bit stream.

† 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.

‡ This bit is set when the device detects any type of parity error from its own master or target.

§ 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).

** These signals are tied to the FPGA signal of the same name and are not initialized.

†† 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.

‡‡ These exhibit special behavior per the CompactPCI Hot Swap Specification:
- — Reads behave normally.
- — Writing a 1 clears the bit to zero.
- — Writing a 0 has no effect on the bit.

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.

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 Lattice. If, in addition, the device ID is 5401 hex, the device is a Lattice 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 Lattice 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 [1] until Done goes active-high, signaling that the FPSC preconfiguration operation has completed, typically via a serial configuration PROM.
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 low, then high.
7. Write to the FCCSR with ConfigFPGA high. This will initiate an FPSC configuration session via the PCI interface.
8. Wait for the RAM initialization to complete by monitoring FCCSR [2]. Wait for 1.5 ms, and then send one word of all ones. If **InitN** is high, continue with real data; otherwise, repeat or declare the problem.

FPSC Configuration (continued)

9. 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. The user should always continue to monitor **initn** and Done.
10. Read the FCCSR until **pciregvld** goes inactive-low, and **srempty** goes high indicating that the DWORD it contained has been transferred to the shift register that feeds the serial configuration data to the FPSC. The user should always continue to monitor **initn** and Done.
11. Repeat steps 9 and 10 until all the configuration data has been written. The user should always continue to monitor **initn** and Done.
12. Read the FCCSR and verify that Done went active-high, indicating that the configuration was successful.
13. Write configFPGA low.

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.
6. Write RdCfgN high.
7. Write ConfigFPGA high (no pulse on **prgmn**)
8. Write all 1s to FCDR
9. Loop on FCCSR until **srempty** goes high and **pciregvld** goes low.
10. Write CongfigFPGA low.

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.

Clocking Options at FPGA/Core Boundary

The OR3LP26B supports a wide variety of integrated FPGA/core clocking schemes which, in conjunction with the FIFO interfaces between the PCI bus and the FPGA, 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 48 illustrates the special clock paths provided to service the clocking needs of PCI functions. The various clocking options shown in Figure 48 are discussed below.

Although there are many clocking options, minimum clock skew is obtained by following the following recommendations. This section is divided into internally generated clocks, external system clocks, external express clocks, and external corner clocks that utilize the PLLs. Refer to the Series 3L data sheet and application notes for a full description of all of the clocking options available for the Series 3L parts.

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. 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).

Internally Generated Clocks

- There are no limitations for using 1 or 2 internally generated clocks to connect to the **fclk1** and/or **fclk2** clock input pins.

External System Clocks

External system clocks are clock inputs that do not use the three dedicated **eclk** input clock pins of the device.

- Keep the clocks toward the center of a side instead of in the corners for minimal skew across the FPGA.
- The best skew across the FPGA/ASIC boundary is obtained by selecting pins on the left or right side of the die. Avoid using general I/O as clock inputs on the top of the device.
- Refer to the Series 3 clocking application note for general FPGA clocking rules.

External Express Clocks

External express clocks are externally generated clocks that enter on one of the three eclk pins of the device.

- The best skew across the FPGA/ASIC boundary is obtained by selecting the **eclk** pin on the right side of the device (**eclkr**). Avoid using the top or left side **eclk** inputs.

Externally Generated Clocks Entering Through PCM Input Pins

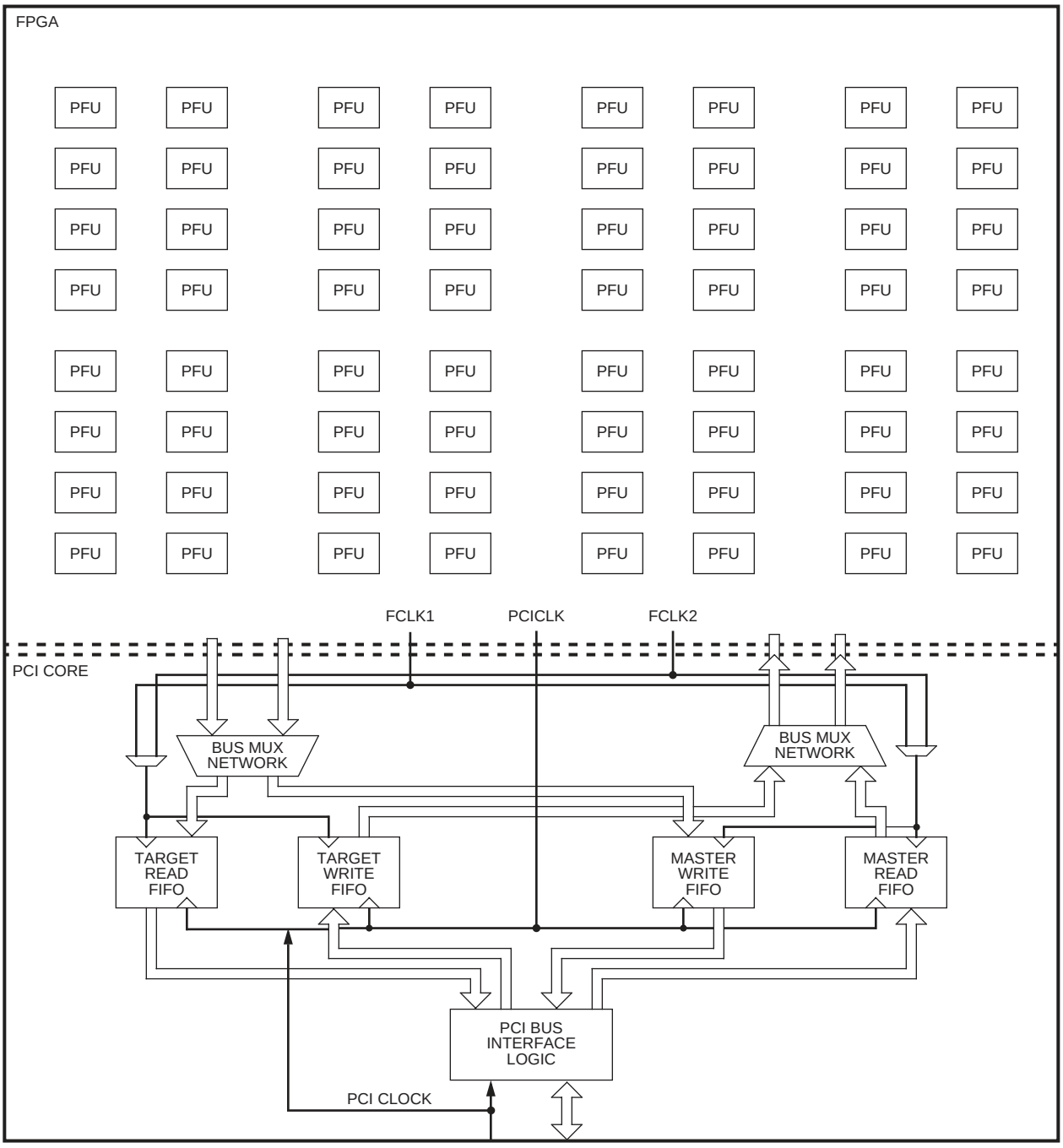
External PCM clocks are clocks entering and going through the programmable clock managers.

- When using a programmable clock manager, either the upper right or lower left clock managers may be used.

Clock Sourced from pciclk

- There are no limitations for using the **pciclk** clock output to connect to the **fclk1** and/or **fclk2** clock input pins.

Clocking Options at FPGA/Core Boundary (continued)



5-7553(F)

Figure 48. 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 49, Figure 50, and Table 43. 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 one-start bit pair and ends with enough one-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)

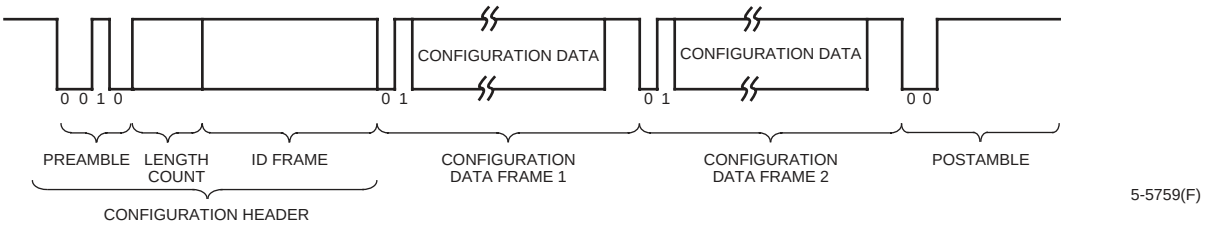


Figure 49. Serial Configuration Data Format—Autoincrement Mode

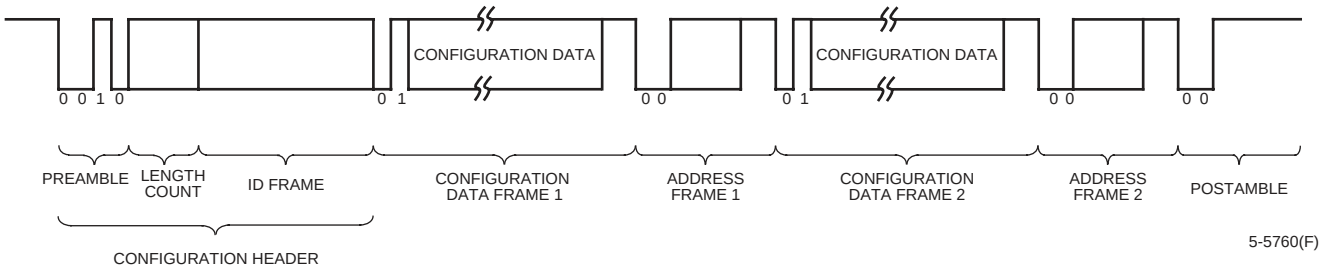


Figure 50. Serial Configuration Data Format—Explicit Mode

Table 43. 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 44.

Table 44. Configuration Frame Size

Devices	OR3LP26B
n 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, eight 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 45. 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 45. 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.

Powerup Sequencing for Series OR3LP26B Device

ORCA Series OR3LP26B device use two power supplies: one to power the device I/Os and the ASIC core (V_{DD}) which is set to 3.3 V for 3.3 V operation and 5 V tolerance, and another supply for the internal FPGA logic ($V_{DD}2$) which is set to 2.5 V. It is understood that many users will derive the 2.5 V core logic supply from a 3.3 V power supply, so the following recommendations are made as to the powerup sequence of the supplies and allowable delays between power supplies reaching stable voltages.

In general, both the 3.3 V and the 2.5 V supplies should ramp-up and become stable as close together in time as possible. There is no delay requirement if the $V_{DD}2$ (2.5 V) supply becomes stable prior to the V_{DD} (3.3 V) supply. There is a delay requirement imposed if the V_{DD} supply becomes stable prior to the $V_{DD}2$ supply.

The requirement is that the $V_{DD}2$ (2.5 V) supply transition from 0 V to 2.3 V within 15.7 ms if the V_{DD} (3.3 V) supply is already stable at a minimum of 3.0 V. If the V_{DD} supply has not yet reached 3.0 V when the $V_{DD}2$ supply has reached 2.3 V, then the requirement is that the $V_{DD}2$ supply reach a minimum of 2.3 V within 15.7 ms of when the V_{DD} supply reaches 3.0 V.

If the chosen power supplies cannot meet this delay requirement, it is always possible to hold-off configuration of the FPGA by asserting $\overline{INIT}$ or $\overline{PRGM}$ until the $V_{DD}2$ supply has reached 2.3 V. This process eliminates any power supply sequencing issues.

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 46. Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Storage Temperature	T_{stg}	–65	150	°C
I/O and ASIC Supply Voltage with Respect to Ground	V_{DD}	—	≤4.2	V
Internal FPGA Supply Voltage with Respect to Ground	$V_{DD}2$	—	≤3.2	V
Input Signal with Respect to Ground CMOS Inputs 5 V Tolerant Inputs	 — —	 –0.5 –0.5	 V_{DD} + 0.3 5.8	 V V
Signal Applied to High-impedance Output	—	–0.5	V_{DD} + 0.3	V

Note: 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 47. 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	2.38 V to 2.63 V

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

Electrical Characteristics

Table 48. Electrical Characteristics

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, $V_{DD}2$ = 2.38 V to 2.63 V, 0 °C < T_A < 70 °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})	 50% V_{DD} GND – 0.5	 V_{DD} + 0.3 30% V_{DD}	 V V
Input Voltage: High Low	 V_{IH} V_{IL}	Input configured as TTL (5 V tolerant)	 50% V_{DD} GND – 0.5	 5.8 V 30% V_{DD}	 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, VIN = 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}, T_A = 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

Description

The most accurate timing characteristics are reported by the timing analyzer in the *ORCA* Foundry Development System. A timing report provided by the development system after layout divides path delays into logic and routing delays. The timing analyzer can also provide logic delays prior to layout. While this allows routing budget estimates, there is wide variance in routing delays associated with different layouts.

The logic timing parameters noted in the Electrical Characteristics section of this data sheet are the same as those in the design tools. In the PFU timing, symbol names are generally a concatenation of the PFU operating mode and the parameter type. The setup, hold, and propagation delay parameters, defined below, are designated in the symbol name by the SET, HLD, and DEL characters, respectively.

The values given for the parameters are the same as those used during production testing and speed binning of the devices. The junction temperature and supply voltage used to characterize the devices are listed in the delay tables. Actual delays at nominal temperature and voltage for best-case processes can be much better than the values given.

It should be noted that the junction temperature used in the tables is generally 85 °C. The junction temperature for the FPGA depends on the power dissipated by the device, the package thermal characteristics (Θ_{JA}), and the ambient temperature, as calculated in the following equation and as discussed further in the Package Thermal Characteristics Summary section:

$$T_{Jmax} = T_{Amax} + (P \bullet \Theta_{JA})\ °C$$

Note: The user must determine this junction temperature to see if the delays from *ORCA* Foundry should be derated based on the following derating tables.

Table 49 and Table 50 provide approximate power supply and junction temperature derating for OR3LP26B commercial devices. The delay values in this data sheet and reported by *ORCA* Foundry are shown as 1.00 in the tables. The method for determining the maximum junction temperature is defined in the Package Thermal Characteristics section. Taken cumulatively, the range of parameter values for best-case vs. worst-case processing, supply voltage, and junction temperature can approach three to one.

Table 49. Derating for Commercial Devices (I/O Supply V_{DD})

T_J (°C)	Power Supply Voltage		
	3.0 V	3.3 V	3.6 V
–40	0.82	0.72	0.66
0	0.91	0.80	0.72
25	0.98	0.85	0.77
85	1.00	0.99	0.90
100	1.23	1.07	0.94
125	1.34	1.15	1.01

Table 50. Derating for Commercial Devices (I/O Supply V_{DD}2)

T_J (°C)	Power Supply Voltage		
	2.38 V	2.5 V	2.63 V
–40	0.86	0.71	0.67
0	0.94	0.79	0.73
25	0.99	0.84	0.77
85	1.00	0.99	0.92
100	1.23	1.05	0.96
125	1.33	1.13	1.03

Note: The derating tables shown above are for a typical critical path that contains 33% logic delay and 66% routing delay. Since the routing delay derates at a higher rate than the logic delay, paths with more than 66% routing delay will derate at a higher rate than shown in the table. The approximate derating values vs. temperature are 0.26% per °C for logic delay and 0.45% per °C for routing delay. The approximate derating values vs. voltage are 0.13% per mV for both logic and routing delays at 25 °C.

Timing Characteristics (continued)

In addition to supply voltage, process variation, and operating temperature, circuit and process improvements of the *ORCA* Series FPGAs over time will result in significant improvement of the actual performance over those listed for a speed grade. Even though lower speed grades may still be available, the distribution of yield to timing parameters may be several speed grades higher than that designated on a product brand. Design practices need to consider best-case timing parameters (e.g., delays = 0), as well as worst-case timing.

The routing delays are a function of fan-out and the capacitance associated with the CIPs and metal interconnect in the path. The number of logic elements that can be driven (fan-out) by PFUs is unlimited, although the delay to reach a valid logic level can exceed timing requirements. It is difficult to make accurate routing delay estimates prior to design compilation based on fan-out. This is because the CAE software may delete redundant logic inserted by the designer to reduce fan-out, and/or it may also automatically reduce fan-out by net splitting.

The waveform test points are given in the Input/Output Buffer Measurement Conditions section of this data sheet. The timing parameters given in the electrical characteristics tables in this data sheet follow industry practices, and the values they reflect are described below.

Propagation Delay—The time between the specified reference points. The delays provided are the worst case of the tphh and tpll delays for noninverting functions, tplh and tphl for inverting functions, and tphz and tplz for 3-state enable.

Setup Time—The interval immediately preceding the transition of a clock or latch enable signal, during which the data must be stable to ensure it is recognized as the intended value.

Hold Time—The interval immediately following the transition of a clock or latch enable signal, during which the data must be held stable to ensure it is recognized as the intended value.

3-State Enable—The time from when a 3-state control signal becomes active and the output pad reaches the high-impedance state.

Clock Timing

Table 51. ExpressCLK (ECLK) and Fast Clock (fclk) Timing Characteristics

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; V_{DD2} = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Device (T_J = 85 °C, V_{DD} = min)	Symbol	Min	Max	Unit
ECLK Delay (middle pad)	eclkm_del	—	1.99	ns
ECLK Delay (corner pad)	eclkc_del	—	4.20	ns
fclk Delay (middle pad)	fclkm_del	—	5.24	ns
fclk Delay (corner pad)	fclkc_del	—	7.46	ns

Notes:
The **ECLK** delays are to all of the PICs on one side of the device for middle pin input, or two sides of the device for corner pin input. The delay includes both the input buffer delay and the clock routing to the PIC clock input.

The **fclk** delays are for a fully routed clock tree that uses the ExpressCLK input into the fast clock network. It includes both the input buffer delay and the clock routing to the PFU CLK input. The delay will be reduced if any of the clock branches are not used.

Timing Characteristics (continued)

Table 52. General-Purpose Clock Timing Characteristics (Internally Generated Clock)

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; V_{DD2} = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Device (T_J = 85 °C, V_{DD} = min)	**Symbol**	**Min**	**Max**	**Unit**
OR3LP26B	clk_del	—	3.95	ns

Notes:
This table represents the delay for an internally generated clock from the clock tree input in one of the four middle PICs (using pSW routing) on any side of the device which is then distributed to the PFU/PIO clock inputs. If the clock tree input used is located at any other PIC, see the results reported by *ORCA* Foundry.

This clock delay is for a fully routed clock tree that uses the general clock network. The delay will be reduced if any of the clock branches are not used. See pin-to-pin timing in Table 55 for clock delays of clocks input on general I/O pins.

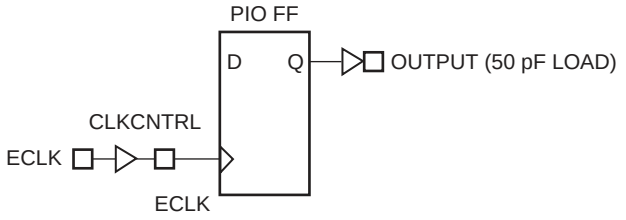
Table 53. OR3LP26B ExpressCLK to Output Delay (Pin-to-Pin)

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; V_{DD2} = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_J = 85 °C, V_{DD} = min)	**Min**	**Max**	**Unit**
ECLK Middle Input Pin→OUTPUT Pin (Fast)	—	5.82	ns
ECLK Middle Input Pin→OUTPUT Pin (Slewlim)	—	6.61	ns
ECLK Middle Input Pin→OUTPUT Pin (Sinklim)	—	11.05	ns
Additional Delay if ECLK Corner Pin Used	—	2.2	ns

Notes:
Timing is without the use of the programmable clock manager (PCM).

This clock delay is for a fully routed clock tree that uses the ExpressCLK network. It includes both the input buffer delay, the clock routing to the PIO CLK input, the clock→Q of the FF, and the delay through the output buffer. The given timing requires that the input clock pin be located at one of the six ExpressCLK inputs of the device, and that a PIO FF be used.



5-4846(F).a

Figure 51. ExpressCLK to Output Delay

Timing Characteristics (continued)

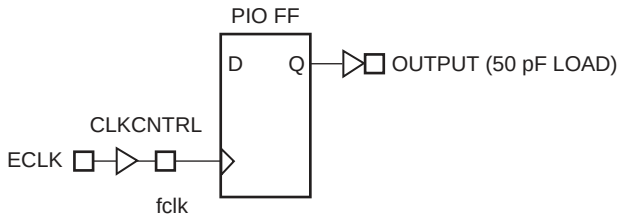
Table 54. OR3LP26B Fast Clock (fclk) to Output Delay (Pin-to-Pin)

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; V_{DD2} = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_J = 85 °C, V_{DD} = min)	Min	Max	Unit
Output Not on Same Side of Device As Input Clock (Fast Clock Delays Using ExpressCLK Inputs)			
ECLK Middle Input Pin →OUTPUT Pin (Fast)	—	9.06	ns
ECLK Middle Input Pin →OUTPUT Pin (Slewlim)	—	9.86	ns
ECLK Middle Input Pin →OUTPUT Pin (Sinklim)	—	14.3	ns
Additional Delay if ECLK Corner Pin Used	—	2.2	ns

Notes:
Timing is without the use of the programmable clock manager (PCM).

This clock delay is for a fully routed clock tree that uses the primary clock network. It includes both the input buffer delay, the clock→Q of the FF, and the delay through the output buffer. The delay will be reduced if any of the clock branches are not used. The given timing requires that the input clock pin be located at one of the six ExpressCLK inputs of the device and that a PIO FF be used.



5-4846(F).b

Figure 52. Fast Clock to Output Delay

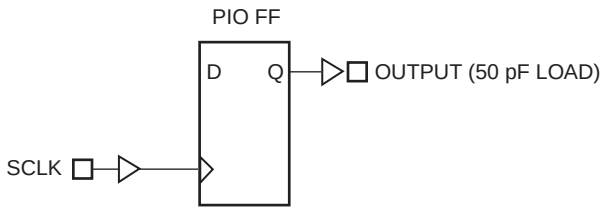
Timing Characteristics (continued)

Table 55. OR3LP26B General System Clock (SCLK) to Output Delay (Pin-to-Pin)

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; V_{DD2} = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_J = 85 °C, V_{DD} = min)	Min	Max	Unit
Output On Same Side of Device As Input Clock (System Clock Delays Using General User I/O Inputs)			
Clock Input Pin (mid-PIC) →OUTPUT Pin (Fast)	—	9.86	ns
Clock Input Pin (mid-PIC) →OUTPUT Pin (Slewlim)	—	10.66	ns
Clock Input Pin (mid-PIC) →OUTPUT Pin (Sinklim)	—	15.10	ns
Additional Delay if Non-mid-PIC Used as Clock Pin	—	0.83	ns
Output Not on Same Side of Device As Input Clock (System Clock Delays Using General User I/O Inputs)			
Additional Delay if Output Not on Same Side as Input Clock Pin	—	0.83	ns

Note: This clock delay is for a fully routed clock tree that uses the primary clock network. It includes both the input buffer delay, the clock routing to the PIO CLK input, the clock→Q of the FF, and the delay through the output buffer. The delay will be reduced if any of the clock branches are not used. The given timing requires that the input clock pin be located at one of the four center PICs on any side of the device and that a PIO FF be used. For clock pins located at any other PIO, see the results reported by *ORCA* Foundry.



5-4846(F)

Figure 53. System Clock to Output Delay

Timing Characteristics (continued)

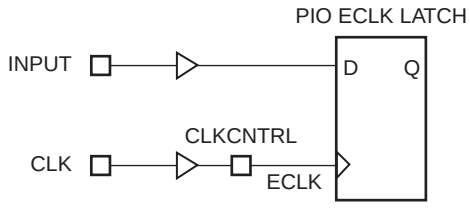
Table 56. OR3LP26B Input to ExpressCLK (ECLK) Fast-Capture Setup/Hold Time (Pin-to-Pin)

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; V_{DD2} = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_J = 85 °C, V_{DD} = min)	Min	Max	Unit
Input to ECLK Setup Time (middle ECLK pin)	0.97	—	ns
Input to ECLK Setup Time (middle ECLK pin, delayed data input)	9.98	—	ns
Input to ECLK Setup Time (corner ECLK pin)	0.0	—	ns
Input to ECLK Setup Time (corner ECLK pin, delayed data input)	8.11	—	ns
Input to ECLK Hold Time (middle ECLK pin)	0.0	—	ns
Input to ECLK Hold Time (middle ECLK pin, delayed data input)	0.0	—	ns
Input to ECLK Hold Time (corner ECLK pin)	0.0	—	ns
Input to ECLK Hold Time (corner ECLK pin, delayed data input)	0.0	—	ns

Notes:
The pin-to-pin timing parameters in this table should be used instead of results reported by *ORCA* Foundry.

The ECLK delays are to all of the PIOs on one side of the device for middle pin input, or two sides of the device for corner pin input. The delay includes both the input buffer delay and the clock routing to the PIO clock input.



5-4847(F).b

Figure 54. Input to ExpressCLK Setup/Hold Time

Timing Characteristics (continued)

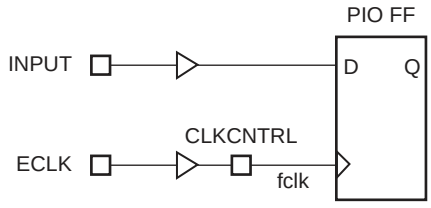
Table 57. OR3LP26B Input to Fast Clock Setup/Hold Time (Pin-to-Pin)

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; V_{DD2} = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_J = 85 °C, V_{DD} = min)	Min	Max	Unit
Output Not on Same Side of Device As Input Clock (Fast Clock Delays Using ExpressCLK Inputs)			
Input to fclk Setup Time (middle ECLK pin)	0.0	—	ns
Input to fclk Setup Time (middle ECLK pin, delayed data input)	5.58	—	ns
Input to fclk Setup Time (corner ECLK pin)	0.0	—	ns
Input to fclk Setup Time (corner ECLK pin, delayed data input)	3.77	—	ns
Input to fclk Hold Time (middle ECLK pin)	4.62	—	ns
Input to fclk Hold Time (middle ECLK pin, delayed data input)	0.0	—	ns
Input to fclk Hold Time (corner ECLK pin)	6.54	—	ns
Input to fclk Hold Time (corner ECLK pin, delayed data input)	0.0	—	ns

Notes:
The pin-to-pin timing parameters in this table should be used instead of results reported by *ORCA* Foundry.

The fclk delays are for a fully routed clock tree that uses the ExpressCLK input into the fast clock network. It includes both the input buffer delay and the clock routing to the PFU CLK input. The delay will be reduced if any of the clock branches are not used.



5-4847(F).a

Figure 55. Input to Fast Clock Setup/Hold Time

Timing Characteristics (continued)

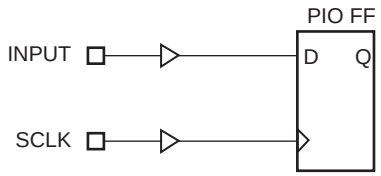
Table 58. OR3LP26B Input to General System Clock (SCLK) Setup/Hold Time (Pin-to-Pin)

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; $V_{DD}2$ = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_J = 85 °C, V_{DD} = min)	**Min**	**Max**	**Unit**
Input to SCLK Setup Time	0.0	—	ns
Input to SCLK Setup Time (delayed data input)	5.02	—	ns
Input to SCLK Hold Time	5.47	—	ns
Input to SCLK Hold Time (delayed data input)	0.0	—	ns
Additional Hold Time if Non-mid-PIC Used as SCLK Pin (no delay on data input)	0.83	—	ns

Notes:
The pin-to-pin timing parameters in this table should be used instead of results reported by *ORCA* Foundry.

This clock delay is for a fully routed clock tree that uses the clock network. It includes both the input buffer delay and the clock routing to the PIO FF CLK input. The delay will be reduced if any of the clock branches are not used. The given setup (delayed and no delay) and hold (delayed) timing allows the input clock pin to be located in any PIO on any side of the device, but a PIO FF must be used. The hold (no delay) timing assumes the clock pin is located at one of the four middle PICs on any side of the device and that a PIO FF is used. If the clock pin is located elsewhere, then the last parameter in the table must be added to the hold (no delay) timing.



5-4847(F)

Figure 56. Input to System Clock Setup/Hold Time

Table 59. OR3LP26B PCI and FPGA Interface Clock Operation Frequencies

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; $V_{DD}2$ = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_I = 85 °C, V_{DD} = min, $V_{DD}2$ = min)	**Speed –8**			**Unit**
Signal	**Min**	**Typ**	**Max**	
Clk (PCI clock)	0	66*	66*	MHz
Fclk1 (user interface clock)	0	66†	100‡	MHz
Fclk2 (user interface clock)	0	66†	100‡	MHz

* The PCI clock frequency is based on the internal register to register frequency and the 66 MHz PCI I/O specifications.
† The maximum user interface clock frequencies are values based on registering all signals at the FPGA/ASIC boundary. This number will be lower depending on the design implementation and number of FPGA logic levels into and out of the ASIC.
‡ This is the typical operating frequency for a real design that does not register signals at the FPGA/ASIC boundary.

Timing Characteristics (continued)

Table 60. OR3LP26B FPGA to PCI, and PCI to FPGA, Combinatorial Path Delays

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; $V_{DD}2$ = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_I = 85 °C, V_{DD} = min, $V_{DD}2$ = min)		**Min**	**Max**	**Unit**
Source	**Destination**			
pci_intan (FPGA side)	intan (PCI side)	—	4.094	ns
clk (PCI side)	pciclk (FPGA side)	—	3.226	ns
rstn (PCI side)	pci_rstn (FPGA side)	—	1.622	ns

Notes:
The FPGA to PCI combinatorial path delays include the ASIC path delay and the output buffer delay under a 10 pF load. They do not include the interbuf delay on the FPGA side.

The PCI to FPGA combinatorial path delays include the ASIC input buffer delay, and ASIC path delay entering the FPGA. They do not include the interbuf delay on the FPGA side.

Table 61. OR3LP26B FPGA Side Interface Combinatorial Path Delay Signals

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; $V_{DD}2$ = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_I = 85 °C, V_{DD} = min, $V_{DD}2$ = min)		**Min**	**Max**	**Unit**
Source	**Destination**			
fifo_sel	datatofpga[63:0]	—	3.253	ns
fifo_sel	datatofpgax[7:0]	—	2.652	ns
twdataenn	twlastcycn	—	5.220	ns
twdataenn	datatofpga[63:0] (dual-port mode)	—	6.114	ns
twdataenn	datatofpgax[7:0] (dual-port mode)	—	5.847	ns
twdataenn	twdata[35:0] (quad-port mode)	—	6.114	ns
trdataenn	trlastcycn	—	5.558	ns
mrdataenn	mrlastcycn	—	5.237	ns
taenn	twlastcycn	—	5.406	ns
taenn	tstatecntr[2:0]	—	4.767	ns
taenn	datatofpga[63:0] (dual-port mode)	—	5.944	ns
taenn	datatofpgax[7:0] (dual-port mode)	—	5.763	ns
taenn	twdata[35:0] (quad-port mode)	—	5.944	ns
taenn	treqn	—	4.958	ns
maenn	mstatecntr[2:0]	—	5.860	ns
mcmd	mstatecntr[2:0]	—	5.662	ns
tcfgshiftenn	pci_tdfg_stat	—	4.227	ns
mcfgshiftenn	pci_mdfg_stat	—	5.300	ns

Note: The combinatorial path parameters are measured from the input to the output (both on the FPGA side), excluding the interbufs, which traverse the ASIC/FPGA boundary. The *ORCA* Foundry Static Analysis Tool, Trace, accounts for clock skew and interbuf delays on the clock and data paths.

Timing Characteristics (continued)

Table 62. OR3LP26B Interbuf Delays

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; $V_{DD}2$ = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_I = 85 °C, V_{DD} = min, $V_{DD}2$ = min)	**Min**	**Max**	**Unit**
Interbuf from FPGA to ASIC	—	0.592	ns
Interbuf from ASIC to FPGA	—	0.429	ns

Note: The interbufs are buffers that interface between the FPGA and the ASIC.

Table 63. OR3LP26B FPGA Side Interface Clock to Output Delays, pciclk Synchronous Signals

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; $V_{DD}2$ = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_I = 85 °C, V_{DD} = min, $V_{DD}2$ = min)	**Min**	**Max**	**Unit**
mw_emptyn	—	4.985	ns
mr_fulln	—	4.458	ns
tr_emptyn	—	4.686	ns
tw_fulln	—	4.703	ns
tcmd[3:0]	—	4.345	ns
bar[2:0]	—	4.139	ns

Note: The clock to out parameters are measured from the **pciclk** clock output pin on the FPGA side, excluding the interbufs, which traverse the ASIC/FPGA boundary. The *ORCA* Foundry Static Analysis Tool, Trace, accounts for clock skew and interbuf delays on the clock and data paths.

Timing Characteristics (continued)

Table 64. OR3LP26B FPGA Side Interface Clock to Output Delays, fclk Synchronous Signals

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; $V_{DD}2$ = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_I = 85 °C, V_{DD} = min, $V_{DD}2$ = min)	**Min**	**Max**	**Unit**
fpga_msyserror	—	3.779	ns
pci_mcfg_stat	—	4.404	ns
ma_fulln	—	4.314	ns
mstatecntr[2:0]	—	5.796	ns
m_ready	—	4.758	ns
mw_fulln	—	4.348	ns
mw_afulln	—	3.734	ns
datatofpga[63:0] (dual-port mode)	—	8.679	ns
datatofpgax[7:0] (dual-port mode)	—	7.974	ns
mrdata[35:0] (quad-port mode)	—	8.479	ns
twdata[35:0] (quad-port mode)	—	6.867	ns
mr_emptyn	—	3.840	ns
mr_aemptyn	—	3.684	ns
mrlastcycn	—	7.536	ns
disctimerexpn	—	3.436	ns
pci_tcfg_stat	—	3.777	ns
treqn	—	4.932	ns
t_ready	—	4.817	ns
tstatecntr[2:0]	—	4.355	ns
tw_emptyn	—	3.893	ns
tw_aemptyn	—	3.759	ns
twlastcycn	—	7.557	ns
tr_fulln	—	4.358	ns
tr_afulln	—	3.915	ns
trlastcycn	—	5.533	ns

Note: The clock to out parameters are measured from the FCLK1 and FCLK2 clock input pins on the FPGA side, excluding the interbufs, which traverse the ASIC/FPGA boundary. The *ORCA* Foundry Static Analysis Tool, Trace, accounts for clock skew and interbuf delays on the clock and data paths.

Timing Characteristics (continued)

Table 65. OR3LP26B FPGA Side Interface Input Setup Delays, pciclk Synchronous Signals

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; $V_{DD}2$ = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_I = 85 °C, V_{DD} = min, $V_{DD}2$ = min)	**Min**	**Max**	**Unit**
fpga_mbusyn	–0.514	—	ns
deltrn	–1.486	—	ns
mwpcihold	–1.190	—	ns
fpga_mstopburstn	–1.208	—	ns
fpga_tabort	1.744	—	ns
fpga_tretryn	0.864	—	ns
twburstpendn	–1.561	—	ns
trpcihold	–1.542	—	ns
trburstpendn	–1.557	—	ns
fpga_syserror	–0.828	—	ns

Note: The input setup parameters are measured from the **pciclk** clock output pin on the FPGA side, excluding the interbufs, which traverse the ASIC/FPGA boundary. The *ORCA* Foundry Static Analysis Tool, Trace, accounts for clock skew and interbuf delays on the clock and data paths.

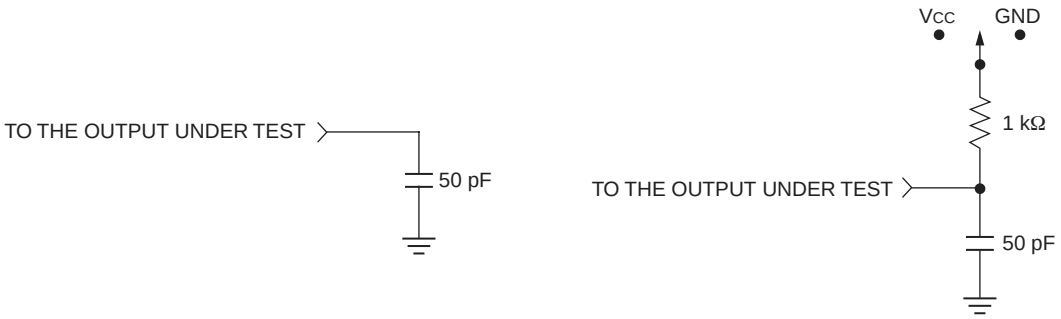
Table 66. OR3LP26B FPGA Side Interface Input Setup Delays, fclk Synchronous Signals

OR3LP26B Commercial: V_{DD} = 3.0 V to 3.6 V, 0 °C < T_A < 70 °C; $V_{DD}2$ = 2.38 V to 2.63 V, 0 °C < T_A < 70 °C.

Description (T_I = 85 °C, V_{DD} = min, $V_{DD}2$ = min)	**Min**	**Max**	**Unit**
mcfgshiftenn	1.752	—	ns
maenn	4.777	—	ns
mfifoclrn	5.934	—	ns
mcmd[3:0]	5.251	—	ns
mwdataenn	4.806	—	ns
datafmfpga[63:0] (dual-port mode)	5.333	—	ns
datafmfpgax[7:0] (dual-port mode)	5.978	—	ns
mwdata[35:0] (quad-port mode)	5.978	—	ns
trdata[35:0] (quad-port mode)	5.226	—	ns
mwlastcycn	4.896	—	ns
mrdataenn	3.246	—	ns
tcfgshiftenn	1.209	—	ns
tfifoclrn	3.395	—	ns
taenn	3.893	—	ns
twdataenn	3.677	—	ns
trdataenn	3.773	—	ns

Note: The input setup parameters are measured from the FCLK1 and FCLK2 clock input pins on the FPGA side, excluding the interbufs, which traverse the ASIC/FPGA boundary. The *ORCA* Foundry Static Analysis Tool, Trace, accounts for clock skew and interbuf delays on the clock and data paths.

Input/Output Buffer Measurement Conditions



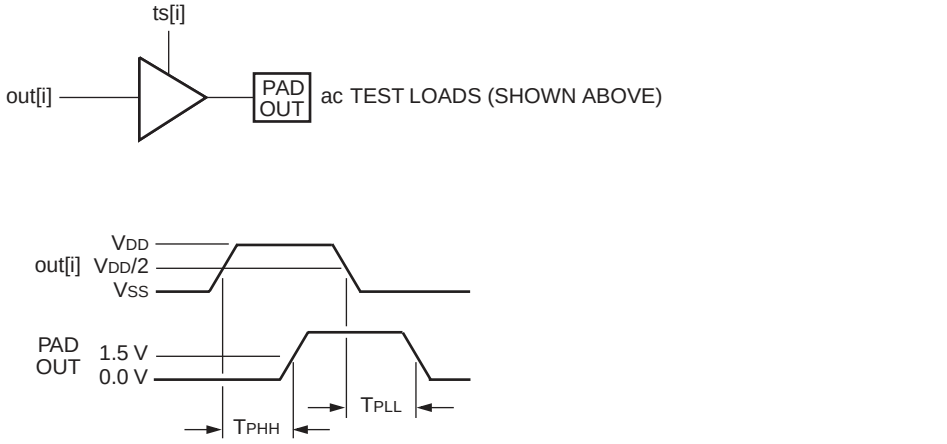
A. Load Used to Measure Propagation Delay

B. Load Used to Measure Rising/Falling Edges

5-3234(F)

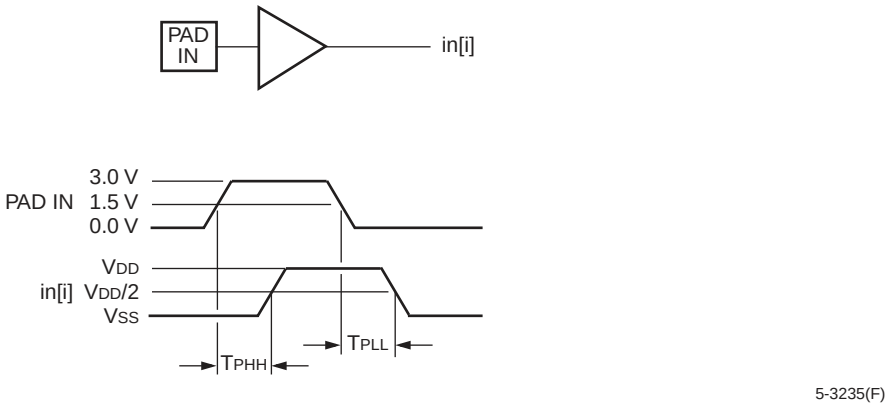
Note: Switch to V_{DD} for T_{PLZ}/T_{PZL}; switch to GND for T_{PHZ}/T_{PZH}.

Figure 57. ac Test Loads



5-3233.a(F)

Figure 58. Output Buffer Delays



5-3235(F)

Figure 59. Input Buffer Delays

Output Buffer Characteristics

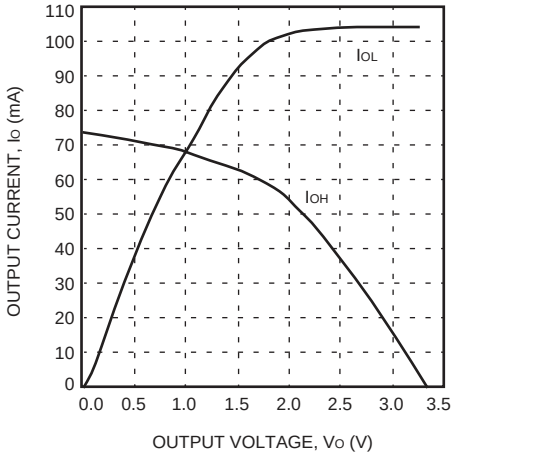


Figure 60. Sinklim (T_J = 25 °C, V_{DD} = 3.3 V)

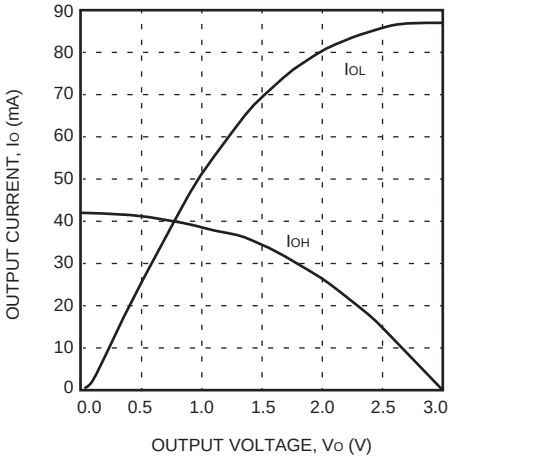


Figure 63. Sinklim (T_J = 125 °C, V_{DD} = 3.0 V)

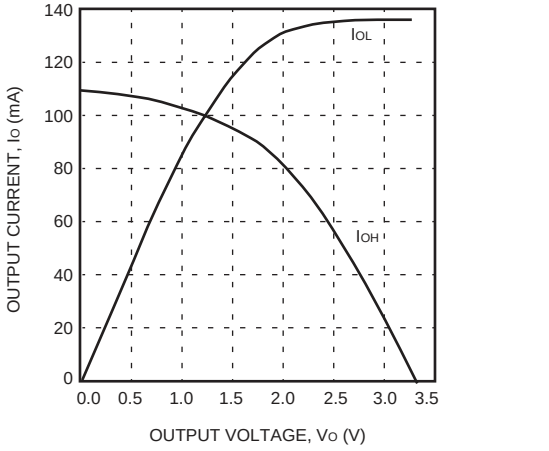


Figure 61. Slewlim (T_J = 25 °C, V_{DD} = 3.3 V)

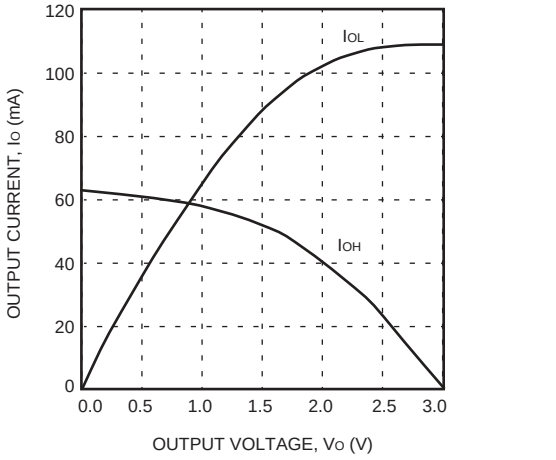


Figure 64. Slewlim (T_J = 125 °C, V_{DD} = 3.0 V)

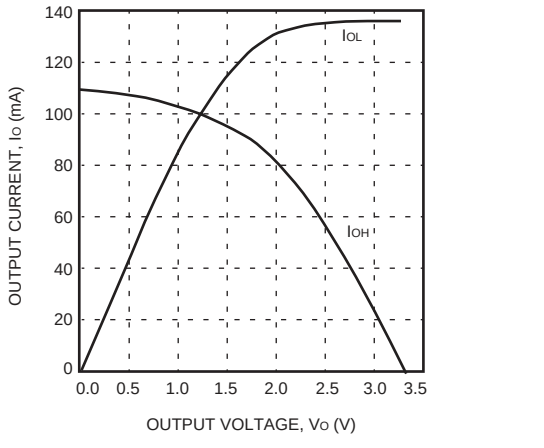


Figure 62. Fast (T_J = 25 °C, V_{DD} = 3.3 V)

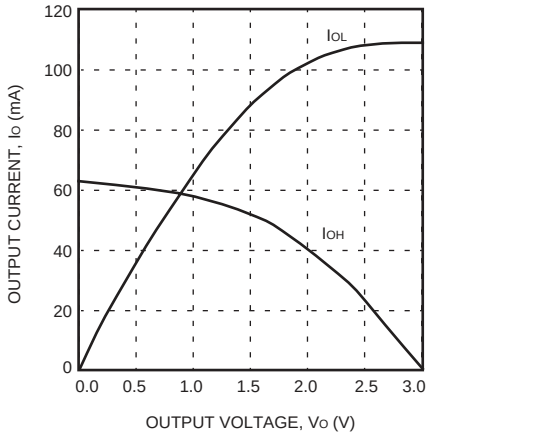


Figure 65. Fast (T_J = 125 °C, V_{DD} = 3.0 V)

Estimating Power Dissipation

The total operating power dissipated is estimated by summing the FPGA standby (I_{DDSB}), internal, and external power dissipated, in addition to the PCI core internal and I/O power.

Table 67. PCI Core Internal Power Dissapation

Operating Frequency (MHz)	Power Dissipated		Unit
	Min	Max	
33	—	292	mW
66	—	584	mW

The following discussion relates to the FPGA portion of the device. The internal and external power is the power consumed in the PLCs and PICs, respectively. In general, the standby power is small and may be neglected. The total operating power is as follows:

$$P_T = \Sigma P_{PLC} + \Sigma P_{PIC}$$

The internal operating power is made up of two parts: clock generation and PFU output power. The PFU output power can be estimated based upon the number of PFU outputs switching when driving an average fan-out of two:

$$P_{PFU} = 0.078 \text{ mW/MHz}$$

For each PFU output that switches, 0.136 mW/MHz needs to be multiplied times the frequency (in MHz) that the output switches. Generally, this can be estimated by using one-half the clock rate, multiplied by some activity factor; for example, 20%.

The power dissipated by the clock generation circuitry is based upon four parts: the fixed clock power, the power/clock branch row or column, the clock power dissipated in each PFU that uses this particular clock, and the power from the subset of those PFUs that are configured as synchronous memory. Therefore, the clock power can be calculated for the four parts using the following equations:

OR3LP26B Clock Power

P = [0.22 mW/MHz
+ (0.39 mW/MHz/Branch) (# Branches)
+ (0.008 mW/MHz/PFU) (# PFUs)
+ (0.002 mW/MHz/PIO (# PIOs)]

For a quick estimate, the worst-case (typical circuit) OR3LP26BB clock power = 4.8 mW/MHz

The following discussions are relavant to FPGA I/Os and the PCI core I/Os. The power dissipated in a PIC is the sum of the power dissipated in the four PIOs in the PIC. This consists of power dissipated by inputs and ac power dissipated by outputs. The power dissipated in each PIO depends on whether it is configured as an input, output, or input/output. If a PIO is operating as an output, then there is a power dissipation component for P_{IN}, as well as P_{OUT}. This is because the output feeds back to the input.

The power dissipated by an input buffer is ($V_{IH} = V_{DD} - 0.3$ V or higher) estimated as:

$$P_{IN} = 0.09 \text{ mW/MHz}$$

The ac power dissipation from an output or bidirectional is estimated by the following:

$$P_{OUT} = (C_L + 8.8 \text{ pF}) \times V_{DD}^2 \times F \text{ Watts}$$

where the unit for C_L is farads, and the unit for F is Hz.

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 68 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 68. FPGA Common-Function Pin Descriptions

Symbol	I/O	Description
Dedicated Pins		
V_{DD}	—	3.3 V power supply.
$V_{DD}2$	—	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, $\overline{\text{RESET}}$ 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 45 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, M3 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 68. 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.*
$\overline{\text{RD}}$/MPI_$\overline{\text{STRB}}$	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 68. 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 CLK-OUT 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 69 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 70 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 69. *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 70. Pinout Information

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
VSS	VSS	VSS*	VSS*
VDD	VDD	VDD*	VDD*
VSS	VSS	VSS*	VSS*
PL1D	I/O	B1	D1
PL1C	I/O	C2	F4
PL1B	I/O	C1	F3
PL1A	I/O	D2	F2
VDD	VDD	VDD*	VDD*
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*	VSS*
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*	VDD*
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*	VSS*
PL8D	I/O	H2	M5

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PL8C	I/O	J4	M4
PL8B	I/O	H1	M2
PL8A	I/O-A2	H3	M1
PL9D	I/O	J2	N5
PL9C	I/O	J1	N4
PL9B	I/O	K2	N3
PL9A	I/O-A3	J3	N2
VDD	VDD	VDD*	VDD*
PL10D	I/O	K1	—
VDD2	VDD2	—	VDD2
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
VSS	VSS	VSS*	VSS*
PL12D	I/O	L1	R4
PL12C	I/O	—	R2
PL12B	I/O	—	R1
PL12A	I/O	M2	—
VDD2	VDD2	—	VDD2
PL13D	I/O	M1	T5
PL13C	I/O	—	T4
PL13B	I/O	—	T2
PL13A	I/O	L3	T1
VSS	VSS	VSS*	VSS*
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
VDD	VDD	VDD*	VDD*
PL15D	I/O	P2	V1
PL15C	——	—	—
VDD2	VDD2	P4	VDD2

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PL15B	I/O	P1	V2
PL15A	I/O-$\overline{\text{MPI\_RW}}$	N3	V3
VSS	VSS	VSS*	VSS*
PL16D	I/O-$\overline{\text{MPI\_ACK}}$	R2	V4
PL16C	I/O	—	V5
PL16B	I/O	—	W1
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*	VSS*
PL18D	I/O	R3	Y4
PL18C	I/O	—	Y5
PL18B	I/O	—	AA1
PL18A	I/O-SECKLL	T1	AA2
PL19D	No Connect†	R4	AA3
PL19C	No Connect†	—	AA4
PL19B	No Connect†	—	AA5
PL19A	I/O-MPI_IRQ	U2	AB1
VDD	VDD	VDD*	VDD*
PL20D	No Connect†	T3	AB2
PL20C	No Connect†	U1	AB3
PL20B	No Connect†	U4	—
VDD2	VDD2	—	VDD2
PL20A	No Connect†	V2	AB4
PL21D	VDD	U3	AB5
PL21C	VSS	V1	AC1
PL21B	VSS	W2	AC2
PL21A	intan	W1	AC4
VSS	VSS	VSS*	VSS*
PL22D	rstn	V3	AC5
PL22C	gntn	Y2	AD1
PL22B	No Connect†	—	AD2
PL22A	No Connect†	—	AD4
PL23D	reqn	W4	AD5
PL23C	No Connect†	—	AE1

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PL23B	No Connect†	—	AE2
PL23A	No Connect†	—	AE3
VDD	VDD	VDD*	VDD*
PL24D	ad31	Y1	AE4
PL24C	No Connect†	—	AE5
PL24B	No Connect†	—	AF1
PL24A	ad30	W3	AF2
PL25D	No Connect†	—	AF3
PL25C	ad29	AA2	AF4
PL25B	ad28	Y4	AF5
PL25A	ad27	AA1	AG1
VSS	VSS	VSS*	VSS*
PL26D	—	—	—
VDD2	VDD2	Y3	VDD2
PL26C	ad26	AB2	AG2
PL26B	No Connect†	—	AG4
PL26A	ad25	AB1	AG5
PL27D	ad24	AA3	AH1
PL27C	c_be3n	AC2	AH2
PL27B	No Connect†	—	AH4
PL27A	idsel	AB4	AH5
VDD	VDD	VDD*	VDD*
PL28D	ad23	AC1	AJ3
PL28C	No Connect†	AB3	AJ4
PL28B	No Connect†	AD2	AK1
PL28A	vio	AC3	AK2
VSS	VSS	VSS*	VSS*
PCCLK	CCLK	AD1	AL1
VDD	VDD	VDD*	VDD*
VSS	VSS	VSS*	VSS*
VDD	VDD	VDD*	VDD*
VSS	VSS	VSS*	VSS*
PB1A	ad22	AF2	AP4
PB1B	No Connect†	AE3	AN5
PB1C	ad21	AF3	AM6
PB1D	ad20	AE4	AN6
VDD	VDD	VDD*	VDD*
PB2A	ad19	AD4	AP6

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PB2B	No Connect†	—	AK7
PB2C	No Connect†	—	AL7
PB2D	—	—	—
VDD2	VDD2	AF4	VDD2
PB3A	ad18	AE5	AN7
PB3B	No Connect†	—	AP7
PB3C	ad17	AC5	AK8
PB3D	ad16	AD5	AL8
VSS	VSS	VSS*	VSS*
PB4A	c_be2n	AF5	AN8
PB4B	perrn	AE6	AP8
PB4C	serrn	AC7	AK9
PB4D	par	AD6	AL9
PB5A	c_be1n	AF6	AM9
PB5B	ad15	AE7	AN9
PB5C	ad14	AF7	AP9
PB5D	ad13	AD7	AK10
VSS	VSS	VSS*	VSS*
PB6A	ad12	AE8	AL10
PB6B	No Connect†	—	AM10
PB6C	No Connect†	—	AN10
PB6D	ad11	AC9	AP10
PB7A	ad10	AF8	AK11
PB7B	No Connect†	—	AL11
PB7C	No Connect†	—	AN11
PB7D	ad9	AD8	AP11
VSS	VSS	VSS*	VSS*
PB8A	ad8	AE9	AK12
PB8B	No Connect†	—	AL12
PB8C	No Connect†	—	AN12
PB8D	c_be0n	AF9	AP12
PB9A	ad7	AE10	AK13
PB9B	No Connect†	—	AL13
PB9C	No Connect†	—	AM13
PB9D	ad6	AD9	AN13
VDD	VDD	VDD*	VDD*
VDD2	VDD2	—	VDD2
PB10A	No Connect†	AF10	—

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PB10B	No Connect†	—	AP13
PB10C	No Connect†	—	AK14
PB10D	ad5	AC10	AL14
PB11A	ad4	AE11	AM14
PB11B	No Connect†	—	AN14
PB11C	No Connect†	—	AP14
PB11D	ad3	AD10	AK15
VDD	VDD	VDD*	VDD*
PB12A	ad2	AF11	AL15
PB12B	No Connect†	—	AN15
PB12C	No Connect†	—	AP15
PB12D	ad1	AE12	AK16
PB13A	ad0	AF12	AL16
PB13B	No Connect†	—	AN16
PB13C	No Connect†	—	AP16
PB13D	framen	AD11	AK17
VSS	VSS	VSS*	VSS*
PB14A	No Connect†	AE13	—
VDD2	VDD2	—	VDD2
PB14B	irdyn	AC12	AM17
PB14C	trdyn	AF13	AP17
PB14D	devseln	AD12	AP18
VSS	VSS	VSS*	VSS*
PECKB	clk	AE14	AN18
PB15A	—	—	—
PB15B	stopn	AC14	AM18
PB15C	ack64n	AF14	AL18
PB15D	req64n	AD13	AK18
VSS	VSS	VSS*	VSS*
PB16A	—	—	—
VDD2	VDD2	AE15	VDD2
PB16B	No Connect†	—	AP19
PB16C	No Connect†	—	AN19
PB16D	c_be7n	AD14	AL19
PB17A	c_be6n	AF15	AK19
PB17B	No Connect†	—	AP20
PB17C	No Connect†	—	AN20
PB17D	c_be5n	AE16	AL20

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
VDD	VDD	VDD*	VDD*
PB18A	HDC	AD15	AK20
PB18B	No Connect†	—	AP21
PB18C	No Connect†	—	AN21
PB18D	c_be4n	AF16	AM21
PB19A	ad63	AC15	AL21
PB19B	No Connect†	—	AK21
PB19C	No Connect†	—	AP22
PB19D	No Connect†	AE17	—
VDD2	VDD2	—	VDD2
VDD	VDD	VDD*	VDD*
PB20A	LDC	AD16	AN22
PB20B	No Connect†	—	AM22
PB20C	No Connect†	—	AL22
PB20D	ad62	AF17	AK22
PB21A	ad61	AC17	AP23
PB21B	No Connect†	—	AN23
PB21C	No Connect†	—	AL23
PB21D	ad60	AE18	AK23
VSS	VSS	VSS*	VSS*
PB22A	ad59	AD17	AP24
PB22B	No Connect†	—	AN24
PB22C	No Connect†	—	AL24
PB22D	No Connect†	—	AK24
PB23A	ad58	AF18	AP25
PB23B	No Connect†	—	AN25
PB23C	ad57	AE19	AM25
PB23D	ad56	AF19	AL25
VSS	VSS	VSS*	VSS*
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

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
VSS	VSS	VSS*	VSS*
PB26A	ad49	AD20	AN27
PB26B	ad48	AE22	AL27
PB26C	No Connect†	—	AK27
PB26D	ad47	AF22	AP28
PB27A	ad46	AD21	AN28
PB27B	No Connect†	AE23	AL28
PB27C	No Connect†	—	AK28
PB27D	ad45	AC22	AP29
VDD	VDD	VDD*	VDD*
PB28A	ad44	AF23	AN29
PB28B	ad43	AD22	AM29
PB28C	No Connect†	AE24	AP30
PB28D	par64	AD23	AN30
VSS	VSS	VSS*	VSS*
PDONE	DONE	AF24	AP31
VDD	VDD	VDD*	VDD*
VSS	VSS	VSS*	VSS*
PRESETN	RESET	AE26	AL34
PPRGMN	PRGM	AD25	AK33
PR28A	M0	AD26	AK34
PR28B	No Connect†	—	AJ31
PR28C	ad42	AC25	AJ32
PR28D	ad41	AC24	AJ33
VDD	VDD	VDD*	VDD*
VDD2	VDD2	—	VDD2
PR27A	No Connect†	AC26	—
PR27B	No Connect†	—	AJ34
PR27C	No Connect†	—	AH30
PR27D	ad40	AB25	AH31
PR26A	ad39	AB23	AH33
PR26B	ad38	AB24	AH34
PR26C	No Connect†	—	AG30
PR26D	ad37	AB26	AG31
VSS	VSS	VSS*	VSS*
PR25A	ad36	AA25	AG33
PR25B	ad35	Y23	AG34
PR25C	ad34	AA24	AF30

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PR25D	No Connect†	—	AF31
PR24A	ad33	AA26	AF32
PR24B	No Connect†	—	AF33
PR24C	No Connect†	—	AF34
PR24D	No Connect†	—	AE30
VDD	VDD	VDD*	VDD*
PR23A	ad32	Y25	AE31
PR23B	enumn	Y26	AE32
PR23C	No Connect†	—	AE33
PR23D	ledn	Y24	AE34
PR22A	No Connect†	—	AD30
PR22B	No Connect†	—	AD31
PR22C	No Connect†	—	AD33
PR22D	M1	W25	AD34
VSS	VSS	VSS*	VSS*
PR21A	ejectsw	V23	AC30
PR21B	No Connect†	W26	AC31
PR21C	No Connect†	W24	AC33
PR21D	No Connect†	—	—
VDD2	VDD2	V25	VDD2
PR20A	No Connect†	V26	AC34
PR20B	No Connect†	U25	AB30
PR20C	No Connect†	V24	AB31
PR20D	No Connect†	U26	AB32
VDD	VDD	VDD*	VDD*
PR19A	I/O-M2	U23	AB33
PR19B	No Connect†	—	AB34
PR19C	No Connect†	—	AA30
PR19D	No Connect†	T25	AA31
PR18A	I/O	U24	AA32
PR18B	I/O	—	AA33
PR18C	I/O	—	AA34
PR18D	I/O	T26	Y30
VSS	VSS	VSS*	VSS*
PR17A	I/O-M3	R25	Y31
PR17B	I/O	—	Y33
PR17C	I/O	—	Y34
PR17D	I/O	R26	W30

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PR16A	I/O	T24	W31
PR16B	I/O	—	W33
PR16C	I/O	—	W34
PR16D	I/O	P25	—
VDD2	VDD2	—	VDD2
VSS	VSS	VSS*	VSS*
PR15A	I/O	R23	V30
PR15B	I/O	P26	V32
PR15C	I/O	R24	V33
PR15D	I/O	N25	V34
VDD	VDD	VDD*	VDD*
PECKR	I-ECKR	N23	U34
PR14A	—	—	—
PR14B	I/O	N26	U33
PR14C	I/O	P24	U32
PR14D	I/O	M25	U31
VSS	VSS	VSS*	VSS*
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*	VSS*
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*	VDD*
PR9A	I/O-CS0	K26	N34

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
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*	VSS*
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*	VDD*
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*	VSS*
PR3A	I/O-WR	E25	H34
PR3B	I/O	E26	H33
PR3C	I/O	—	H31
PR3D	I/O	F24	H30
PR2A	I/O	D25	G34
PR2B	I/O	—	G33
PR2C	I/O	—	G31
PR2D	I/O	E23	G30
VDD	VDD	VDD*	VDD*
PR1A	I/O	D26	F34
PR1B	I/O	E24	F32

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PR1C	I/O	C25	F31
PR1D	I/O	D24	E33
VSS	VSS	VSS*	VSS*
PRD_CFGN	RD_CFGN	C26	D34
VDD	VDD	VDD*	VDD*
VSS	VSS	VSS*	VSS*
VDD	VDD	VDD*	VDD*
VSS	VSS	VSS*	VSS*
PT28D	I/O-SECKUR	A25	A31
PT28C	I/O	B24	A30
PT28B	I/O	A24	C29
PT28A	I/O	B23	B29
VDD	VDD	VDD*	VDD*
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
VSS	VSS	VSS*	VSS*
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
VSS	VSS	VSS*	VSS*
PT23D	—	—	—
VDD2	VDD2	B19	VDD2
PT23C	I/O	D18	C25
PT23B	I/O	A19	B25
PT23A	I/O	—	A25
PT22D	I/O	C19	E24
PT22C	I/O	—	D24

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PT22B	I/O	—	B24
PT22A	I/O	—	A24
VSS	VSS	VSS*	VSS*
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*	VDD*
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*	VDD*
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*	VSS*
PECKT	I-ECKT	B14	D18
PT15D	—	—	—
PT15C	I/O	D15	—
VDD2	VDD2	—	VDD2
PT15B	I/O	A14	C18
PT15A	I/O-D3	C15	A18
VSS	VSS	VSS*	VSS*

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PT14D	I/O	B13	A17
PT14C	I/O	D13	B17
PT14B	—	—	—
VDD2	VDD2	A13	VDD2
PT14A	I/O-D2	C14	C17
VSS	VSS	VSS*	VSS*
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*	VDD*
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*	VDD*
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*	VSS*
PT7D	I/O	C10	E12
PT7C	I/O	—	A11
PT7B	I/O	—	B11
PT7A	I/O	A9	D11

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
PT6D	I/O	B8	E11
PT6C	I/O	—	A10
PT6B	I/O	—	B10
PT6A	I/O-TDI	A8	C10
VSS	VSS	VSS*	VSS*
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*	VSS*
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*	VDD*
PT1D	I/O	A4	A6
PT1C	I/O	C5	C6
PT1B	I/O	B3	D6
PT1A	I/O-TCK	C4	B5
VSS	VSS	VSS*	VSS*
PRD_DATA	RD_DATA/TDO	A3	A4
VDD	VDD	VDD*	VDD*
VDD2	VDD2	—	VDD2
‡	VDD	—	—
‡	VDD	—	A3
‡	VDD	—	A32
‡	VDD	—	—
‡	VDD	—	B3
‡	VDD	—	B4

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
‡	VDD	—	B31
‡	VDD	—	B32
‡	VDD	—	C1
‡	VDD	—	C2
‡	VDD	—	C4
‡	VDD	—	C7
‡	VDD	—	C11
‡	VDD	—	C15
‡	VDD	—	C20
‡	VDD	—	C24
‡	VDD	—	C28
‡	VDD	—	C31
‡	VDD	—	C33
‡	VDD	—	C34
‡	VDD	—	D2
‡	VDD	—	D3
‡	VDD	—	—
‡	VDD	D6	—
‡	VDD	D11	—
‡	VDD	D16	—
‡	VDD	D21	—
‡	VDD	—	—
‡	VDD	—	D32
‡	VDD	—	D33
‡	VDD	—	G3
‡	VDD	—	G32
‡	VDD	F4	L3
‡	VDD	F23	L32
‡	VDD	L4	R3
‡	VDD	L23	R32
‡	VDD	T4	Y3
‡	VDD	T23	Y32
‡	VDD	AA4	AD3
‡	VDD	AA23	AD32
‡	VDD	—	AH3
‡	VDD	—	AH32
‡	VDD	—	AL2
‡	VDD	—	AL3

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
‡	VDD	—	—
‡	VDD	AC6	—
‡	VDD	AC11	—
‡	VDD	AC16	—
‡	VDD	AC21	—
‡	VDD	—	—
‡	VDD	—	AL32
‡	VDD	—	AL33
‡	VDD	—	AM1
‡	VDD	—	AM2
‡	VDD	—	AM4
‡	VDD	—	AM7
‡	VDD	—	AM11
‡	VDD	—	AM15
‡	VDD	—	AM20
‡	VDD	—	AM24
‡	VDD	—	AM28
‡	VDD	—	AM31
‡	VDD	—	AM33
‡	VDD	—	AM34
‡	VDD	—	AN3
‡	VDD	—	AN4
‡	VDD	—	AN31
‡	VDD	—	AN32
‡	VDD	—	—
‡	VDD	—	AP3
‡	VDD	—	AP32
‡	VDD	—	—
‡	VDD2	—	C5
‡	VDD2	—	C30
‡	VDD2	—	D5
‡	VDD2	—	D30
‡	VDD2	—	E3
‡	VDD2	—	E4
‡	VDD2	—	E5
‡	VDD2	—	E6
‡	VDD2	—	E29
‡	VDD2	—	E30

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
‡	VDD2	—	E31
‡	VDD2	—	E32
‡	VDD2	—	F5
‡	VDD2	—	F30
‡	VDD2	—	AJ5
‡	VDD2	—	AJ30
‡	VDD2	—	AK3
‡	VDD2	—	AK4
‡	VDD2	—	AK5
‡	VDD2	—	AK6
‡	VDD2	—	AK29
‡	VDD2	—	AK30
‡	VDD2	—	AK31
‡	VDD2	—	AK32
‡	VDD2	—	AL5
‡	VDD2	—	AL30
‡	VDD2	—	AM5
‡	VDD2	—	AM30
‡	VSS	A1	A1
‡	VSS	A2	A2
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	—	A33
‡	VSS	A26	A34
‡	VSS	—	B1
‡	VSS	B2	B2
‡	VSS	B25	B33
‡	VSS	B26	B34
‡	VSS	—	—
‡	VSS	C3	C3
‡	VSS	—	C8
‡	VSS	—	C12
‡	VSS	—	C16
‡	VSS	—	C19

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
‡	Vss	—	C23
‡	Vss	—	C27
‡	Vss	C24	C32
‡	Vss	—	—
‡	Vss	D4	D4
‡	Vss	D9	—
‡	Vss	D14	—
‡	Vss	D19	—
‡	Vss	D23	D31
‡	Vss	—	H3
‡	Vss	—	H32
‡	Vss	H4	M3
‡	Vss	J23	M32
‡	Vss	—	T3
‡	Vss	—	T32
‡	Vss	N4	—
‡	Vss	P23	—
‡	Vss	—	W3
‡	Vss	—	W32
‡	Vss	V4	AC3
‡	Vss	W23	AC32
‡	Vss	—	AG3
‡	Vss	—	AG32
‡	Vss	AC4	AL4
‡	Vss	AC8	—
‡	Vss	AC13	—
‡	Vss	AC18	—
‡	Vss	AC23	AL31
‡	Vss	—	—
‡	Vss	AD3	AM3
‡	Vss	—	AM8
‡	Vss	—	AM12
‡	Vss	—	AM16
‡	Vss	—	AM19
‡	Vss	—	AM23
‡	Vss	—	AM27
‡	Vss	AD24	AM32
‡	Vss		—

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
‡	VSS	AE1	AN1
‡	VSS	AE2	AN2
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	AE25	AN33
‡	VSS	—	AN34
‡	VSS	AF1	AP1
‡	VSS	—	AP2
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	—	—
‡	VSS	AF25	AP33
‡	VSS	AF26	AP34
‡	VSS	L11	N13
‡	VSS	L12	N14
‡	VSS	L13	N15
‡	VDD2	—	N16
‡	VDD2	—	N17
‡	VDD2	—	N18
‡	VDD2	—	N19
‡	VSS	L14	N20
‡	VSS	L15	N21
‡	VSS	L16	N22
‡	VSS	M11	P13
‡	VSS	M12	P14
‡	VSS	M13	P15
‡	VDD2	—	P16
‡	VDD2	—	P17
‡	VDD2	—	P18
‡	VDD2	—	P19
‡	VSS	M14	P20
‡	VSS	M15	P21
‡	VSS	M16	P22
‡	VSS	N11	R13

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
‡	VSS	N12	R14
‡	VSS	N13	R15
‡	VDD2	—	R16
‡	VDD2	—	R17
‡	VDD2	—	R18
‡	VDD2	—	R19
‡	VSS	N14	R20
‡	VSS	N15	R21
‡	VSS	N16	R22
‡	VDD2	—	T13
‡	VDD2	—	T14
‡	VDD2	—	T15
‡	VSS	—	T16
‡	VSS	—	T17
‡	VSS	—	T18
‡	VSS	—	T19
‡	VDD2	—	T20
‡	VDD2	—	T21
‡	VDD2	—	T22
‡	VDD2	—	U13
‡	VDD2	—	U14
‡	VDD2	—	U15
‡	VSS	—	U16
‡	VSS	—	U17
‡	VSS	—	U18
‡	VSS	—	U19
‡	VDD2	—	U20
‡	VDD2	—	U21
‡	VDD2	—	U22
‡	VDD2	—	V13
‡	VDD2	—	V14
‡	VDD2	—	V15
‡	VSS	—	V16
‡	VSS	—	V17
‡	VSS	—	V18
‡	VSS	—	V19
‡	VDD2	—	V20
‡	VDD2	—	V21

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
‡	VDD2	—	V22
‡	VDD2	—	W13
‡	VDD2	—	W14
‡	VDD2	—	W15
‡	VSS	—	W16
‡	VSS	—	W17
‡	VSS	—	W18
‡	VSS	—	W19
‡	VDD2	—	W20
‡	VDD2	—	W21
‡	VDD2	—	W22
‡	VSS	P11	Y13
‡	VSS	P12	Y14
‡	VSS	P13	Y15
‡	VDD2	—	Y16
‡	VDD2	—	Y17
‡	VDD2	—	Y18
‡	VDD2	—	Y19
‡	VSS	P14	Y20
‡	VSS	P15	Y21
‡	VSS	P16	Y22
‡	VSS	R11	AA13
‡	VSS	R12	AA14
‡	VSS	R13	AA15
‡	VDD2	—	AA16
‡	VDD2	—	AA17
‡	VDD2	—	AA18
‡	VDD2	—	AA19
‡	VSS	R14	AA20
‡	VSS	R15	AA21
‡	VSS	R16	AA22
‡	VSS	T11	AB13
‡	VSS	T12	AB14
‡	VSS	T13	AB15
‡	VDD2	—	AB16
‡	VDD2	—	AB17
‡	VDD2	—	AB18
‡	VDD2	—	AB19

* 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.
† Pins marked No Connect must be left unconnected.
‡ These pins are connected to a power plane in the package rather than to a particular pad.

Pin Information (continued)

Table 70. Pinout Information (continued)

OR3LP26B Pad	Function	PBGA 352	PBGAM 680
‡	Vss	T14	AB20
‡	Vss	T15	AB21
‡	Vss	T16	AB22

* 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.
† Pins marked No Connect must be left unconnected.
‡ 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: ΘJA, ψJC, and ΘJC. 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 (theta-JA, R-theta, etc.).

$$\Theta JA = \frac{TJ - TA}{Q}$$

where TJ is the junction temperature, TA is the ambient air temperature, and Q is the chip power.

Experimentally, ΘJA 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 (TJ) is determined by the forward drop on the diodes, and the ambient temperature (TA) is noted. Note that ΘJA 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:

$$\psi JC = \frac{TJ - TC}{Q}$$

where TC is the case temperature at top dead center, TJ is the junction temperature, and Q is the chip power. During the ΘJA measurements described above, besides the other parameters measured, an additional temperature reading, TC, is made with a thermocouple attached at top-dead-center of the case. ψJC 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:

$$\Theta JC = \frac{TJ - TC}{Q}$$

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 ΘJC from ψJC. ΘJC is a true thermal resistance and is expressed in units of °C/W.

ΘJB

This is the thermal resistance from junction to board (ΘJB). It is defined by:

$$\Theta JB = \frac{TJ - TB}{Q}$$

where TB 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 ΘJB 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, TAmax, and the power dissipated by the device, Q (expressed in °C), the maximum junction temperature is approximated by:

$$TJmax = TAmax + (Q \bullet \Theta JA)$$

Table 71 lists the thermal characteristics for all packages used with the *ORCA* OR3LP26B Series of FPGAs.

Package Thermal Characteristics Summary (continued)

Table 71. *ORCA* OR3LP26B Plastic Package Thermal Guidelines

Package*	Θ_{JA} (°C/W)			T_A = 70 °C Max T_J = 125 °C Max 0 fpm (W)
	0 fpm	200 fpm	500 fpm	
352-Pin PBGA† ‡	19.0	16.0	15.0	2.9
680-Pin PBGAM† ‡	14.5	TBD	TBD	3.8

* Mounted on a four-layer JEDEC standard test board with two power/ground planes.
† With thermal balls connected to board ground plane.
‡ 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 72. Package Coplanarity

Package Type	Coplanarity Limit (mils)
PBGA	8.0
PBGAM	8.0

Package Parasitics

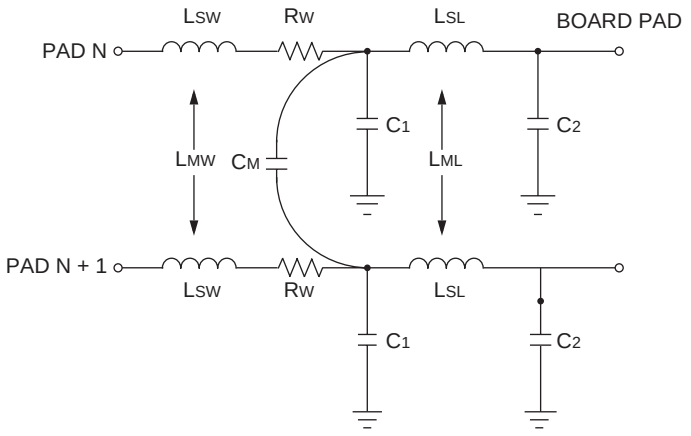
The electrical performance of an IC package, such as signal quality and noise sensitivity, is directly affected by the package parasitics. Table 73 lists eight parasitics associated with the *ORCA* packages. These parasitics represent the contributions of all components of a package, which include the 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 73 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 73. 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 66. Package Parasitics

Package Outline Diagrams

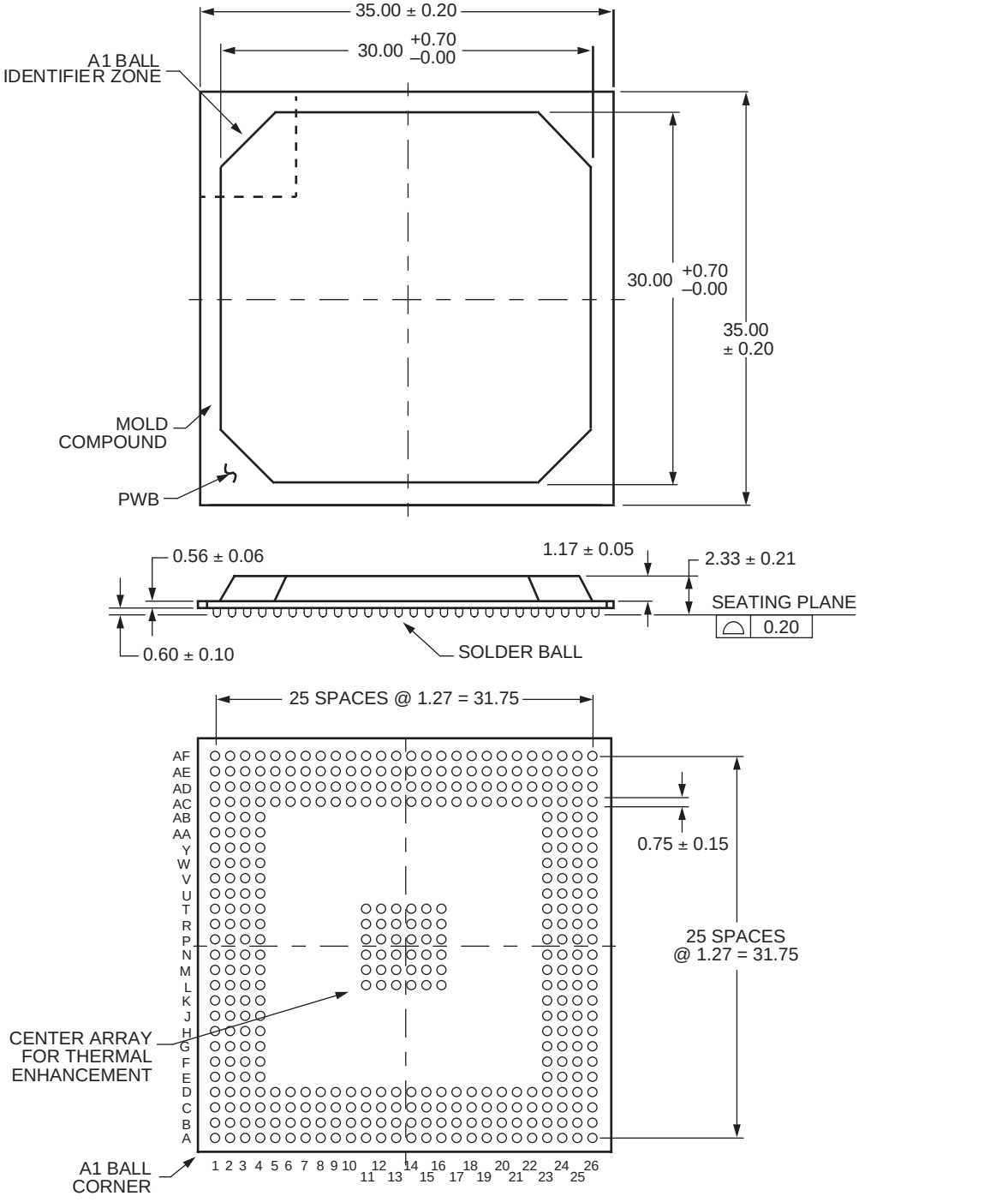
Terms and Definitions

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.

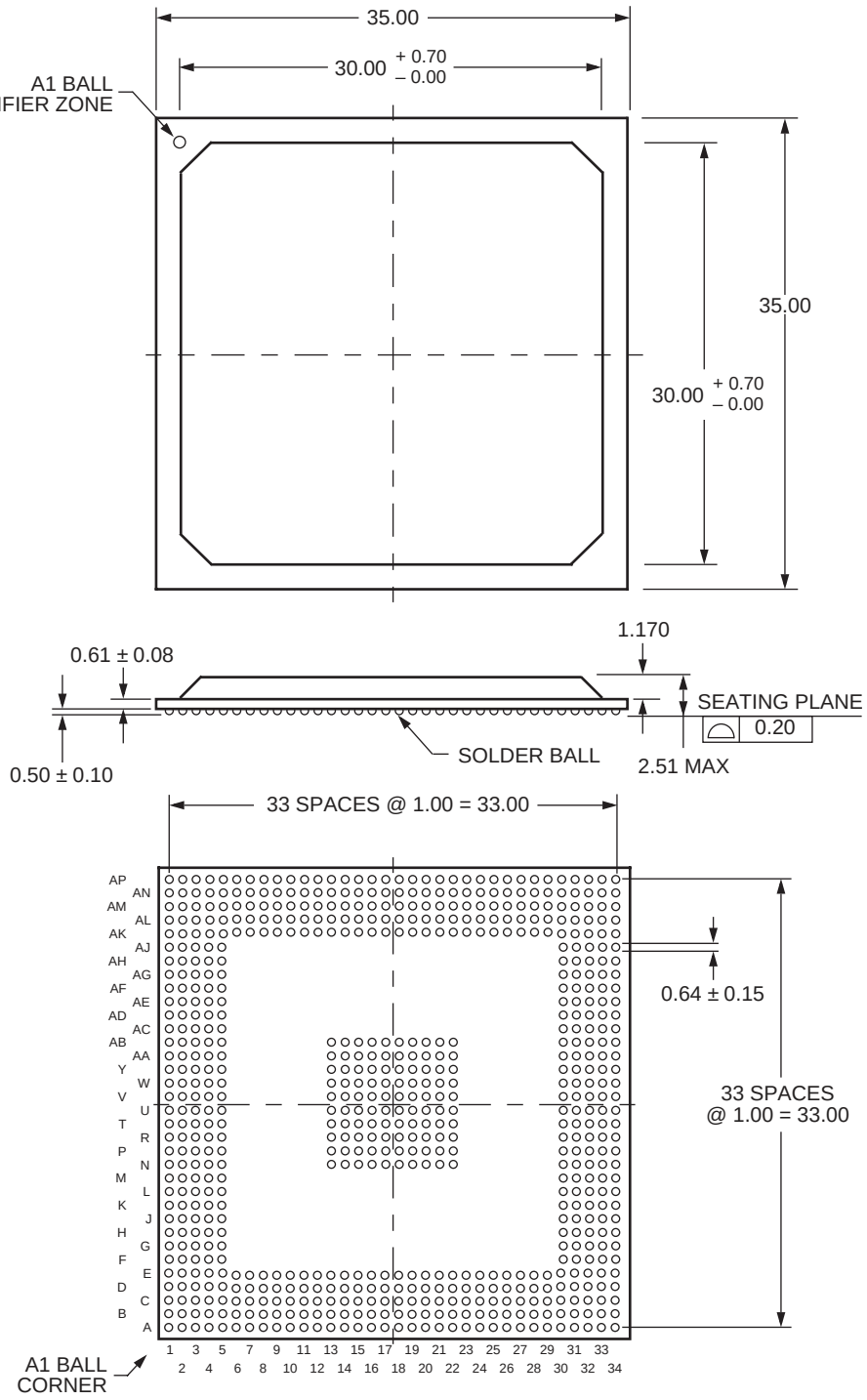


5-4407(F)

Package Outline Diagrams (continued)

680-Pin PBGA

Dimensions are in millimeters.



Ordering Information

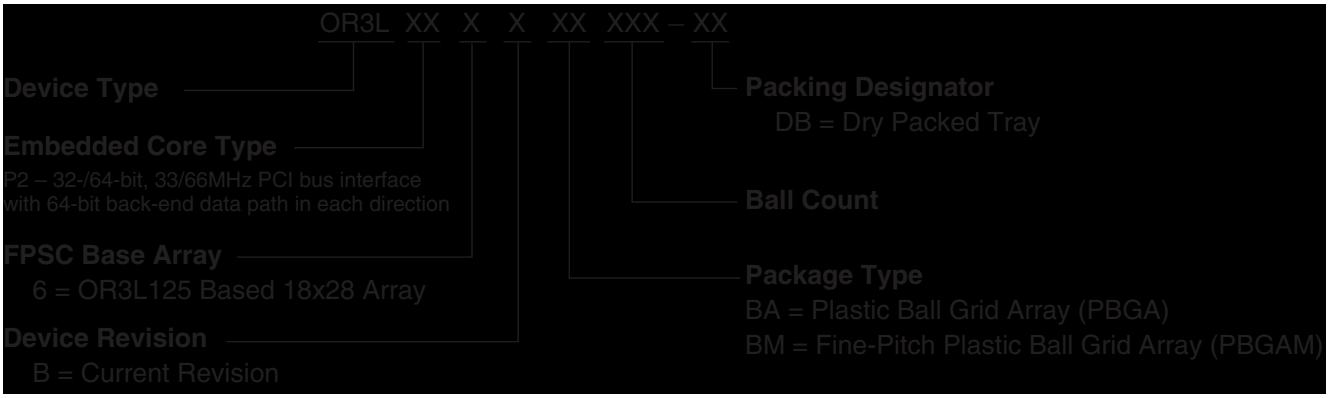


Table 74. Ordering Information

Device Family	Part Number	Package Type	Ball Count	Packing Designator
OR3LP26B	OR3LP26BBA352-DB	PBGA	352	DB
	OR3LP26BBM680-DB	PBGAM	680	DB