Reference DESIGN

Synthesizable PCI Bridge Design Examples

November 1, 1999 Data Sheet

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

Introduction

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

General Description

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

www.xilinx.com/pci

These designs are general purpose data transfer engines to be used with the LogiCORE PCI Interfaces. Figure 1 presents a block diagram of the bridge design. Typically, the user will customize the local interface to conform to a particular peripheral bus (ISA, VME, i960) or attach to a memory device. The design is modular so that unused portions may be removed. Some versions are subsets of the complete design, and do not contain parts of the target functionality as indicated in the facts table. SB08, SB05, and SB04 support Block Mode DMA transfer.

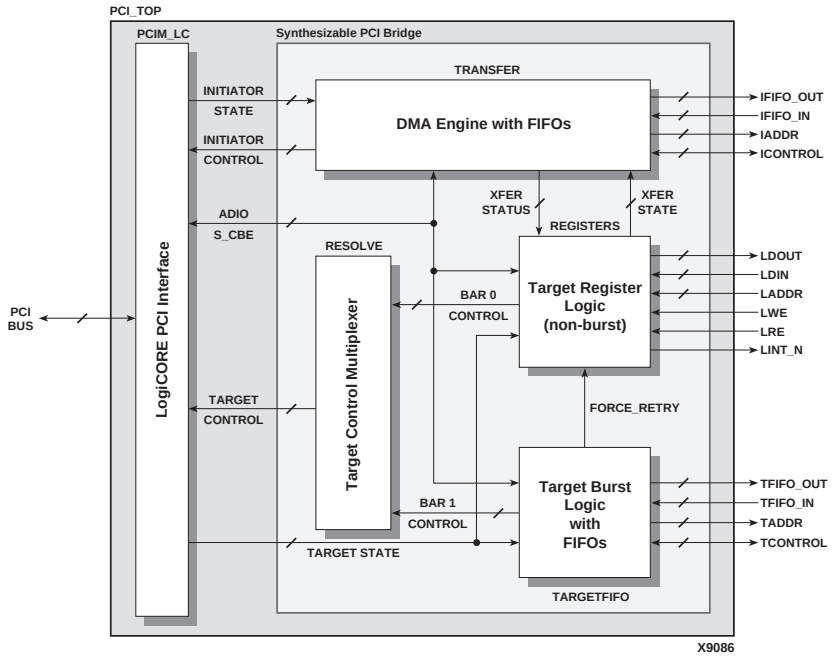

Figure 1: Synthesizable PCI Bridge Block diagram

Design Example Facts

Features	SB04	SB05	SB08
Initiator Functions			
Separate read and write FIFOs (unidirectional)	✔		✔
Block data transfer engine (DMA)	✔		✔
Programmable Burst sizes fixed by transfer counter	✔		✔
Auto data delivery (handles terminations)	✔		✔
Discard counter to prevent deadlock	✔		✔
Initiator address counter	✔		✔
Target Functions			
BAR 0 - Supports single data phase transfers	✔	✔	✔
BAR 0 - Region 1 demonstrates doorbells	✔		✔
BAR 0 - Region 2 demonstrates mailboxes	✔		✔
BAR 0 - Region 3 demonstrates long latency accesses	✔		✔
BAR 0 - Region 4 contains control registers for initiator	✔	✔	✔
BAR 1 - Separate read and write FiFOs (unidirectional)	✔		✔
BAR 1 - Delayed completion discard after time-out	✔		✔
BAR 1 - Generates target abort on address wrap	✔		✔
Target address counter	✔		✔
Target functions independent of initiator	✔		✔
Bus Width			
32 Bit	✔	✔	✔
64 Bit			
Bus Frequency			
33 MHz	✔	✔	✔
66 MHz			
HDL Support			
Verilog	✔	✔	✔
VHDL			
Supported Families/LogiCORE Product			
SpartanXL / PCI32 V3.0		✔	
XLA / PCI32 V3.0	✔		
Virtex / PCI32 V3.0			✔
Virtex / PCI64 V3.0			
Device Features Used			
Distributed RAM FIFOs	✔	✔	
Block RAM FIFOs			✔
Resource Utilization			
CLBs Used[1]	1054	543	943

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

Functional Description

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

BAR0 Configuration

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

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

Region One: Doorbells

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

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

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

Region Two: Mailboxes

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

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

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

Region Three: Bounded Latency Accesses

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

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

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

Region Four: Control Registers

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

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

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

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

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

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

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

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

BAR1 Configuration

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

Posted Writes

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

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

Prefetched Reads

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

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

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

Register File Interface

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

Table 1: Local Bus Register Interface

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

Target FIFO Interface

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

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

Table 2: Local Bus Target Fifo Interface

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

Initiator FIFO Interface

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

The FIFOs are identical, but data flows in opposite directions. Table 3 lists the signals used in the interface.

Table 3: Local Bus Initiator Fifo Interface

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

Block Mode DMA Transfer

To initiate block mode DMA transfer the following steps are involved in a sequence:

- Write the target PCI address to the bridge's PCI transfer address register, XFER_PADR, in the Target Register Module
- Write the local address of the source or destination to the bridge's XFER_LADR register in the Target Register Module (only the lower 16 bits of the address are latched into the XFER_LADR register
- Write the length of the transaction to the bridge's 16-bit XFER_LEN register (only the upper 16 bits of the address are written to the XFER_LEN register and the count is in bytes rather than DWORDS, for example, to transfer 2 DWORDS, write 0X00080000 to XFER_LEN)
- Write to the bridge's command register, CONTROL, to initiate a read and write transaction and set interrupts on done or error. The relevant bits of the CONTROL register are : FAIL (bit 8), DONE (bit 9), NOTIFY_P (bits 12 &13), NOTIFY_L (bits 14 & 15), DIR (bit 16), GO (bit 17), ENA (bit 30) and RESET (bit 31). Refer to Table 2-3 in the Synthesizable PCI Bridge Design User Guide for more details on the definition and control of these bits.

Pinout

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

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

Core Modifications

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

Verification Methods

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

Recommended Design Experience

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

Reference Design License

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