55:131 Introduction to VLSI Design
Project #3 -- Fall 2010

64-bit PCI Target with EDAC
Due Date: Friday November 19, 2010

Introduction

In this project we will modify the PCI Target from project 2 to change from a 32-bit datapath to a 64-bit datapath. Several blocks in the design will need to be changed, as well as the test bench. Three implementation options are permitted, each with different grading:

Option	Description	Points	Comments
1	64-bit PCI and 64-bit EDC. Only 64-bit PCI transactions are allowed.	100%	
2	64-bit PCI, 2 X 32-bit EDC. Both 32-bit and 64-bit PCI transactions are allowed.	125%	1KB of total data memory is still needed. Consider using 2 512B memories, one each for the top and bottom half of the 64-bit dword.
3	64-bit PCI, 64-bit EDC. Both 32-bit and 64-bit PCI transactions are allowed.	150%	Requires a Read-Modify-Write memory cycle for 32-bit writes. This option is only recommended for students confident with both Verilog and simulation debugging.

Connection of the device in a system is the same as it was for project 2 (shown in figure 1). A representative block diagram of the device is shown in figure 2. As in project 2, this diagram is shown for guidance only – you may use any internal architecture you desire. The design uses a combination of Verilog structural and behavioral styles.

Code for the EDAC is provided by Xilinx application note XAPP 645. Use the 32-bit or 64-bit design as appropriate (top_32b_edc.v or top_64b_edc.v respectively).

http://www.xilinx.com/support/documentation/application_notes/xapp645.pdf

Code for project 2 will be provided, along with a testbench. Your test bench should ensure that the 64-bit implementation works correctly. This includes correction of single-bit errors and detection (and reporting) of double-bit errors. A method to inject errors is provided, see the EDAC requirements. Your grade for the project will depend on how well your design meets those requirements.

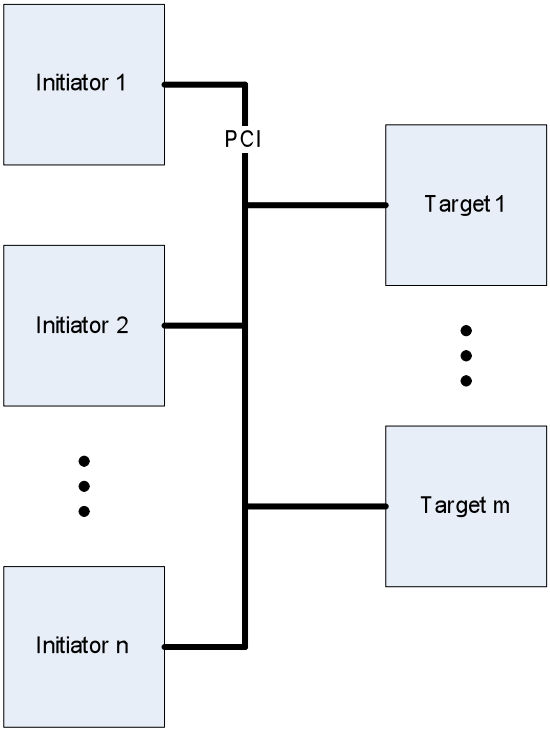

Figure 1 - PCI Bus System

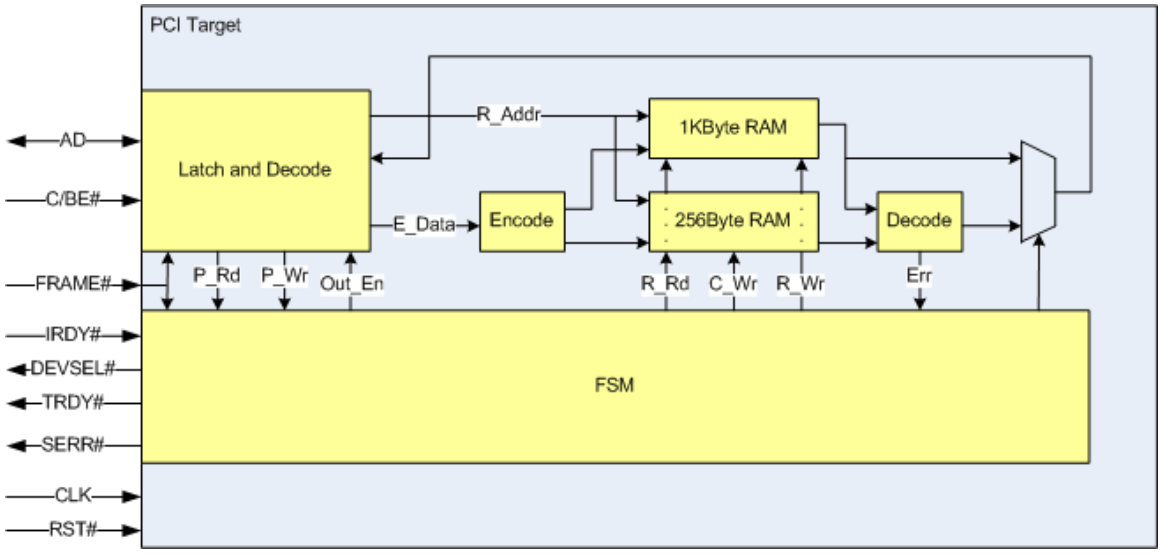

Figure 2 - Simple PCI Target with EDAC

Goals

- To understand and interpret the 64-bit requirements for the simple PCI target.
- To develop a suitable architecture that will meet those requirements.
- To integrate the code from 64-bit EDAC code from Xilinx.
- To write a test bench to test the Target's functionality.

Requirements

1. Interface signals

Name	Number of signals	Direction	Description
CLK	1	In	66MHz clock
RESET#	1	In	Asynchronous reset (active low)
FRAME#	1	In	Sent by initiator to indicate the start of a transaction
AD[63:0]	64	Bidir	Address/Data lines
C/BE[7:0]#	8	In	Command/Byte Enable
IRDY#	1	In	Initiator-ready
TRDY#	1	Out	Target-ready
DEVSEL#	1	Out	Response by target to indicate that it has recognized its address and is ready for the data transfer transaction
SERR#	1	Out	System error indicates a double-bit error has occurred while reading the target's memory.

2. Resets and Clocks

2.1. All signals are clocked on the rising edge of the CLK signal except RESET#.

2.2. Maximum clock frequency is 66MHz.

2.3. Assertion of RESET# shall initialize all flip-flops in the PCI target.

3. Protocol

3.1. FRAME# shall asserted for 1 clock at the start of each transaction.

3.2. On the clock in which FRAME# is active:

3.2.1. The address shall be decoded from the AD bus.

3.2.2. Valid addresses for this target are 0x00001000 through 0x000013FF.

3.2.3. These addresses will be word-aligned and will correspond to a word in the 1KByte memory inside the PCI target. All 256 32-bit words shall be readable/writable from the PCI bus.

3.2.4. The command shall be decoded from the C/BE# bus.

3.2.5. Valid memory commands for this device are 0x6 (read) and 0x7 (write).

3.3. If the transaction is meant for this target (i.e. valid address and command for this device):

3.3.1. The target shall assert the DEVSEL# output within 3 clocks of the assertion of FRAME#.

3.3.2. DEVSEL# shall remain asserted until, and while, TRDY# is asserted.

3.3.3. The target shall assert TRDY# when it accepts data (on a write) or has provided data (on a read).

3.3.4. TRDY# shall be asserted for 1 clock period.

3.3.5. Data shall be accepted from the AD bus when TRDY# is asserted for write transactions.

3.3.6. Data shall be driven onto the AD bus when TRDY# is asserted for read transactions.

3.3.7. The AD bus shall be tri-stated by the target when TRDY# is not driven.

3.3.8. TRDY# shall be asserted no later than 16 clock cycles after the assertion of FRAME#

3.4. TRDY# shall be driven high for one clock after assertion, then tri-stated (hi-Z).

3.5. DEVSEL# shall be driven high for one clock after assertion, then tri-stated (hi-Z).

3.6. AD and C/BE# shall be tri-stated by the initiator(s) when no transactions are in progress.

3.7. The initiator shall assert IRDY# on the clock after assertion of FRAME#.

3.8. IRDY# and FRAME# shall be driven high one clock after they have been asserted, then they will be tri-stated.

3.9. The initiator shall remove IRDY# if the target doesn't assert DEVSEL# within 3 clock periods of the assertion of FRAME#. This could occur for cases when the address or command is not valid for the target.

3.10. IRDY# shall remain asserted until TRDY# is asserted or when an initiator timeout occurs.

3.11. All data transactions shall be 4 bytes wide (hereafter called a word) for grading option 1, or 4 bytes or 8 bytes (hereafter called a dword) for grading options 2 and 3. This effectively makes AD[1:0] "don't care" during the address phase of the transaction.

3.12. No burst transactions shall be performed.

4. Error detection and correction (EDAC)

4.1. EDAC encoding and decoding shall implement single bit correction, double-bit detection for the 1KB target memory.

4.2. For grading option 2, EDAC shall use 7 check bits in a byte wide memory. The remaining bit input should be tied high or low, not left "floating".

4.3. For grading options 1 and 3, EDAC shall use 8 check bits in a byte wide memory.

4.4. To enable error injection, IO commands 0x2 (IO read) and 0x3 (IO write), shall be used.

4.4.1. The IO read command shall read the memory without EDAC decoding at the same addresses as used for the memory commands (0x1000 through 0x13FF).
4.4.2. The IO write command shall write to memory without EDAC encoding at the same addresses as used for the memory commands (0x1000 through 0x13FF). In other words, only the data is written, not the check bits.
4.4.3. EDAC (decode) shall be used for the memory read command (0x6).
4.4.4. EDAC (encode) shall be used for the memory write command.

4.5. SERR# shall be asserted when a double bit error is detected.
4.6. Transactions with errors shall terminate normally, with SERR# asserted at the same time as TRDY#.
4.7. SERR# shall be driven low when asserted and tri-stated at other times.
4.8. The data returned on a transaction with an error is a ”don’t care” – it can be whatever you decide.

Discussion:

As in project 2, your design should be completely synchronous. In other words, all flip-flops should be clocked by the same CLK signal and reset or preset by the RST# signal. The memory you create for the Target device need not be initialized by the reset signal.

What to turn in:

Submit your design code, test bench code and any “do files” you create to ICON. The design (and test bench) should compile and run under Modelsim/Questsim. All files should be ascii text files.