ENGR 304
Hardware Lab 3: Iterative Multiplication on the Altera DE2 Board

Objectives

After completing this lab you should:

- recall binary arithmetic operations,
- recall how to use Quartus design software,
- be able to write VHDL code for registers and for combinational logic,
- be able to use a VHDL testbench to test your design, and
- be able to program the Altera part on the Altera DE2 board and test it out.

Altera DE2 Board

In this lab, you will be doing a simple logic design for implementation on the Altera DE2 board. During this lab, you will focus on inputting data from switches and displaying information on LEDs and the 7-segment displays.

Specifications of the Design

In the design for this lab, you shall input two 8-bit unsigned numbers that will then be multiplied to produce a 16-bit result. The multiply operation will be based on adding together a number of partial products as described on page 292 of the textbook and in the powerpoint presented in lab. The partial products are gradually accumulated (added together) in a register and after the last partial product is added, the register will hold the final product. Along with the 16 slide switches for inputting the numbers you will also use one slide switch (SW16) to control the mode of operation when the clock edges arrive and two pushbuttons: one that operates as the system reset and one that operates as the system clock. Seven segment displays shall show the bit patterns of the input numbers and the result of the operation performed, all in hexadecimal format. In addition, LEDs will be used to show the step-by-step execution progress and the state of the registers.

Specifically, these inputs and outputs are assigned as follows:

PIN Name	Description
SW(7 downto 0)	8-bit Multiplier (Q) with SW(7) as the most significant bit (unsigned format)
SW(15 downto 8)	8-bit Multiplicand (M) with SW(15) as the most significant bit (unsigned format)
SW(16)	When SW16 is high, the registers in your system shall be either pre-loaded from the values found on SW(15..0) or initialized to a known state every time there is a rising edge on the system clock. When SW16 is low, the system performs an accumulate-multiply step every time there is a rising edge on the system clock.
KEY(3)	System Clock. All registers should "operate" on the rising edge of this signal (release of the button).
KEY(0)	System Reset. All registers should be reset to zero when this signal is asserted (Asserted = logic 0 or when the button is pressed).

HEXi(6 downto 0)	7 segment display signals for each of the 8 displays (HEX7 … HEX0). • HEX7&6 display the multiplicand (M) value in hex, • HEX5&4 display the multiplier (Q) value in hex, • HEX3&2 display the upper portion of the product, and • HEX1&0 display the lower portion of the product (initially the multiplier). HEX7-4 shall not change during the accumulate-multiply operation (SW16 = ‘0’), even if the input switches get changed.
LEDG(7 downto 0)	LED pattern to indicate progress through the multiplication operation (8 steps to complete). Choose an appropriate pattern.
LEDR(15 downto 0)	LED pattern to match the bits currently stored in the A (LEDR(15..8)) and Q (LEDR(7..0)) shift registers combined (upper and lower portions of the product).

Steps:

1. Draw a detailed block diagram of the system. A template is provided. Be sure to label and size any internal signals that are needed. Remember to add the switches and keys as external components, similarly to HW02.
2. Determine how many flip-flops (or “registers” as Quartus calls them) should be needed in your final design.
3. Create a new folder for this lab on your H: drive, and copy any important files needed for this design.
4. Create a new project in Quartus in your new design directory. The testbench (extra credit) assumes that you have named this project/entity/component as “HWLab3”. Again, specify the target device as the Cyclone II: EP2C35F672C6. Modify your HW02 design (VHDL file) to implement the new requirements. You should build your VHDL code based on the blocks and signals that are in your block diagram. In other words, you should not add any VHDL code that does not have a corresponding signal or block in your block diagram.
5. Compile your design. Get a screen capture of the compilation report giving the number of flip/flops (registers) and a screen capture of all warnings. You must fix any “sensitivity list” or “inferred latch” warnings. Timing and pin capacitance warnings are ok.
6. Program the DE2 board with your design. Choose a small set of tests to run on your design, including some of the “edge” cases. Identify why you chose each test and what the results were. Include this information in your writeup.
7. Either demonstrate for your instructor or record a video of your design multiplying 0xC7 and 0xF7 together. If you record a video, email a link (not the video) to your instructor. Please avoid doing high resolution videos for this as the video can be too large for some download services.
8. EXTRA CREDIT: A template of a testbench has been provided on the shared drive. Complete the testbench code and simulate your design in Modelsim. In particular, look for “TBD” place holders in the code and put in proper code in place of them. “doit.do” should setup your simulation (without running it). Running through a complete test of your design should take at least 1.3 ms of simulation time, but you will have to enter the run command in modelsim.

HAND IN:

1. Detailed block diagram identifying the components of the system. Labels of signals and functional modules should match the labels found in your VHDL code.

2. Well-commented VHDL code (you do not have to print out the 7-segment code). You will lose points if the code is printed without adhering to the standards for code printouts.
3. Printout of a screen capture showing the number of registers in your design based on the compilation summary in Quartus.
4. A summary of the lab in a one-page writeup that describes how you used the green LEDs, compares the number of actual and predicted registers, and includes a description of your DE2 testing plan and results.
5. EXTRA CREDIT ONLY: Printout of your completed testbench code, also well-formatted for printing. Highlight the changes you made to the testbench code. Also, include a screen capture of the runtime messages from the modelsim transcript window.