Laboratory Exercise 5 – ECE241 Fall 2013

Counters, Arithmetic

Preparation

You are required to write the Verilog code for Parts I to V. For marking by the teaching assistants, you need to bring with you (pasted into your lab book) your Verilog code for Parts III, IV and V. Also print out and paste into your lab book, for Part I, the portion of the Quartus II report showing the F_{max} of your counter circuit (i.e. the maximum clock frequency at which your counter may be safely operated) and the number of logic elements (LEs) required to implement the counter. Finally, for Part III, print out a simulation that exercises the circuit for several clock cycles.

In-lab Work

You are required to implement and test all of Parts I to V of the lab. But you only need to demonstrate to the teaching assistants Parts III, IV and V. Your mark will be based on these three parts of the lab.

Part I

Consider the circuit in Figure 1. It is a 4-bit synchronous counter which uses four T-type flip-flops. The counter increments its value on each positive edge of the clock if the *Enable* signal is asserted. The counter is reset to 0 by setting the *Clear* signal low – it is an active-low asynchronous clear. You are to implement a 8-bit counter of this type.

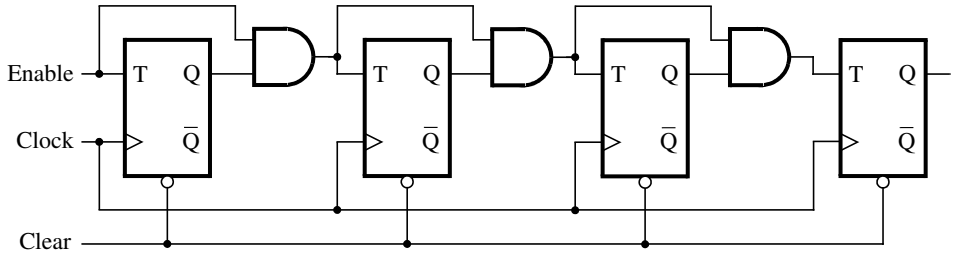

Figure 1: A 4-bit counter.

1. Write a Verilog file that defines a 8-bit counter by using the structure depicted in Figure 1. Your code should include a T flip-flop module that is instantiated 8 times to create the counter (i.e. structural Verilog). Compile the circuit. How many logic elements (LEs) are used to implement your circuit? What is the maximum frequency, F_{max}, at which your circuit can be operated? (Use TimeQuest in Quartus to determine the maximum frequency F_{max}.)
2. Simulate your circuit to verify its correctness.
3. Augment your Verilog file to use the pushbutton KEY_0 as the *Clock* input, switches SW_1 and SW_0 as *Enable* and *Clear* inputs, and 7-segment displays *HEX1-0* to display the hexadecimal count as your circuit operates. Make the necessary pin assignments needed to implement the circuit on the DE2 board, and

compile the circuit. For this part, you should re-use the hexadecimal-to-7-segment display decoder that you created for Lab #4.

4. Download your circuit into the FPGA chip and test its functionality by operating the switches.
5. Use the Quartus II RTL Viewer to see how Quartus II software synthesized your circuit. What are the differences in comparison with Figure 1?

Part II

Another way to specify a counter is by using a register and adding 1 to its value. This can be accomplished using the following Verilog statement:

$$\mathrm{Q} <= \mathrm{Q} + 1;$$

Compile an 8-bit version of this counter and determine the number of LEs needed and the F_{max} that is attainable. Use the same KEY, SW and 7-segment displays as in Part I above. Use the RTL Viewer to see the structure of this implementation versus the design from Part I. Repeat the steps of Part I above for this counter.

Part III

Design and implement a circuit that flashes the red LEDs on the DE2 board in the following pattern (which mimics the "perceived" motion of the lights on an old-style movie house). First, $LEDR_0$ turns on. After about 1 second, $LEDR_0$ turns off and $LEDR_1$ turns on. After 1 second, $LEDR_1$ turns off and $LEDR_2$ turns on, and so on, until $LEDR_{17}$ turns on for 1 second. After $LEDR_{17}$ has been on for a second, it turns off and $LEDR_{16}$ turns on, with the pattern continuing back down to $LEDR_0$. After reaching $LEDR_0$, the pattern begins again.

You may want to use your rotating register (with parallel load) from Lab #4 for this part of the lab. Note that all flip-flops used in your design *must* be clocked directly by the 50 MHz clock on the DE2 board. This means that your flip-flops/counters may need an "enable" signal that is asserted only in specific clock cycles.

1. Write a Verilog file that realizes the behaviour described above. You should use the 50 MHz clock on the DE2 board as the clock input to your circuit. Your circuit should have no other inputs beside the clock.
2. Simulate your circuit with QSim to verify its correctness.
3. Include the pin constraints for the DE2 board, and synthesize the circuit with Quartus II.
4. Download your circuit into the FPGA chip and test its functionality.

Part IV

Design and implement a circuit that displays the word PASS, in ticker-tape fashion, on the four 7-segment displays $HEX3-0$. Make the letters move from right to left in intervals of about one second. The patterns that should be displayed in successive clock intervals are given in Table 1.

Notice that the word PASS can be displayed using a 7-segment decoder you designed in a previous lab.

Clock cycle	Displayed pattern
0	P A S S
1	S P A S
2	S S P A
3	A S S P
4	P A S S
...	and so on

Table 1. Scrolling the word PASS in ticker-tape fashion.

1. Write a Verilog file that realizes the behaviour described above. You should use the 50 MHz clock on the DE2 board as the clock input to your circuit. Your circuit should have no other inputs beside the clock. As in Part III, the only clock in your design should be the 50 MHz clock.
2. Simulate your circuit with QSim to verify its correctness.
3. Include the pin constraints for the DE2 board, and synthesize the circuit with Quartus II.
4. Download your circuit into the FPGA chip and test its functionality.

Part V

Figure 2 below shows an 8-bit multiplier circuit with registers on its inputs. Each register has an active-high load enable input; that is, at a positive clock edge, the register loads the 8-bit value on its input only if its corresponding enable input is logic-`1`. Multiplication can be implemented in Verilog with the `*` operator. Observe that the 8-bit multiplier has a 16-bit output; do you know why?

Design and compile your circuit with Quartus II software, download it onto a DE2 board, and test its operation as follows:

1. Create a new Quartus II project.
2. Write Verilog code that describes the circuit in Figure 2. You are encouraged to use procedural Verilog.
3. Connect input $DATA$ to switches SW_{7-0}, and use KEY_0 as a manual clock input. Use SW_8 as $LoadEnable1$ and use SW_9 as $LoadEnable2$. The product output should be displayed on red $LEDR_{15-0}$ lights and also using the four 7-segment displays $HEX3-0$. You should use four instances of the 7-segment display decoder from your prior lab.
4. Simulate the circuit with QSim. Your simulation should load the two registers, allowing you to verify the correct product was computed.
5. Assign the pins on the FPGA to connect to the switches, keys and 7-segment displays.
6. Compile your design and load the circuit onto the DE2 board. Test the circuit by loading the registers with different values and observing the output on the HEX displays and red LEDs.

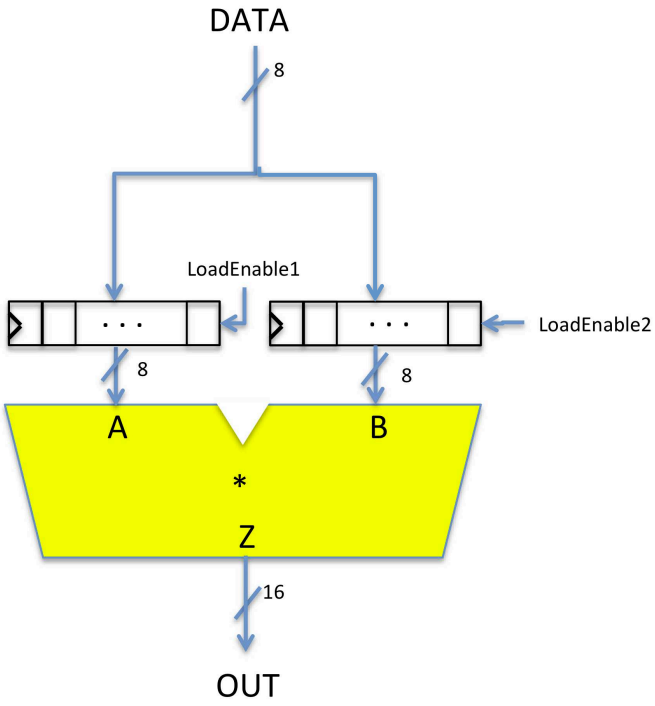

Figure 2: An eight-bit multiplier circuit.