Laboratory Exercise A

Which is done in teams and
requires a formal report

Counters

This is Altera's Laboratory Exercise 4.

Part I I'll do this part in class (except for #2 below), but include this part in your lab report anyway (cite me as the source).

Consider the circuit in Figure 1. It is a 4-bit synchronous counter which uses four T-type flip-flops. The counter increments its count on each positive edge of the clock if the Enable signal is asserted. The counter is reset to 0 by using the Reset signal. You are to implement a 16-bit counter of this type.

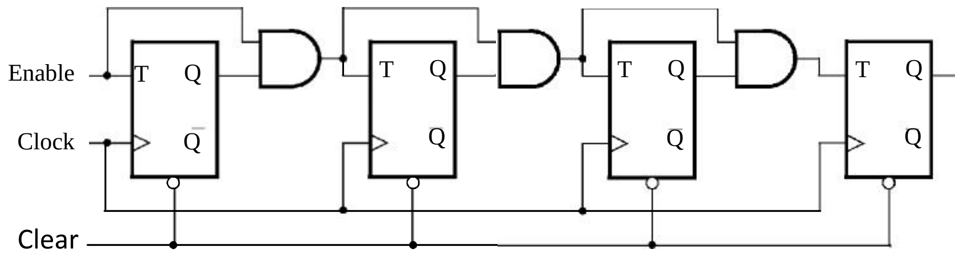

Figure 1. A 4-bit counter.

1. Write a Verilog file that defines a 16-bit counter by using the structure depicted in Figure 1. Your code should include a T flip-flop module that is instantiated 16 times to create the counter. Compile the circuit. How many logic elements (LEs) are used to implement your circuit? What is the maximum frequency, *Fmax*, at which your circuit can be operated?
2. Simulate your circuit to verify its correctness using ModelSim. In your report, figure out some convincing way to demonstrate correct circuit operation without including all 65,536 lines of output!
3. Augment your project with a top-level Verilog file that uses the pushbutton KEY_0 as the *Clock* input, switch SW_1 as an active high *Reset* and switch SW_0 as an active high *Enable*, and 7-segment displays *HEX3-0* 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.
4. Download your circuit into the FPGA chip and test its functionality by operating the implemented 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

Simplify your Verilog code so that the counter specification is based on the Verilog statement

Q <= Q + 1;

1. Write a Verilog file that defines a 16-bit counter by using this approach. Compile the circuit. How many logic elements (LEs) are used to implement your circuit? What is the maximum frequency, *Fmax*, at which your circuit can be operated? Comment on the differences between this and Part I.
2. Simulate your circuit to verify its correctness using ModelSim.
3. Augment your project with a top-level Verilog file that uses the pushbutton KEY_0 as the *Clock* input, switch SW_1 as an active high *Reset* and switch SW_0 as an active high *Enable*, and 7-segment displays *HEX3-0* 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.
4. Download your circuit into the FPGA chip and test its functionality by operating the implemented switches.
5. Use the RTL Viewer to see the structure of this implementation and comment on the differences with the design from Part I.

Part III

Use an LPM from the Library of Parameterized modules to implement a 16-bit counter. Choose the LPM options to be consistent with the above design, i.e. with enable and synchronous clear.

1. Write a Verilog file that implements a 16-bit counter by using this approach. Compile the circuit. How many logic elements (LEs) are used to implement your circuit? What is the maximum frequency, *Fmax*, at which your circuit can be operated? Comment on the differences between this and Part I and Part II.
2. (no ModelSim for this part).
3. Augment your Verilog file to use the pushbutton KEY_0 as the *Clock* input, switches SW_1 and SW_0 as *Enable* and *Reset* inputs, and 7-segment displays *HEX3-0* 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.

4. Download your circuit into the FPGA chip and test its functionality by operating the implemented switches.

5. Use the RTL Viewer to see the structure of this implementation and comment on the differences with the design from Parts I and II.

Part IV

Design and implement a circuit that successively flashes digits 0 through 9 on the 7-segment display $HEX0$. Each digit should be displayed for about one second. Use a counter to determine the one-second intervals. The counter should be incremented by the 50-MHz clock signal provided on the DE2 board. **Do not derive any other clock signals in your design–make sure that all flip-flops in your circuit are clocked directly by the 50 MHz clock signal. This means you should have <u>no</u> clock warnings in your Quartus compile.**

Part V

Design and implement a circuit that displays the word HELLO, in ticker tape fashion, on the eight 7-segment displays $HEX7-0$. 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.

Clock cycle	Displayed pattern							
0				H	E	L	L	O
1			H	E	L	L	O	
2		H	E	L	L	O		
3	H	E	L	L	O			
4	E	L	L	O				H
5	L	L	O				H	E
6	L	O				H	E	L
7	O				H	E	L	L
8				H	E	L	L	O
...	and so on							

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

You must tell
me how to set the switches
SW[14:0] for this part!!

Note: All of these letters must ultimately be driven by switch settings. That is, the only inputs to the muxes that drive the letters must be swtiches or variables (wires) related to the switches. You can easily test this: if all switches are set to 0 (off), then all of the 7-segment displays should show 'H'. And if all of the switches ([14:0]) are 1 (on) all of the 7-segment displays should show a blank.