ECE253 – Laboratory Exercise 5

Latches, Flip-flops, Registers, and Counters

The purpose of this exercise is to investigate sequential circuits (i.e. storage circuits).

Preparation and In-Lab Marks

Preparation marks: you are required to show the simulation output for Parts I and III, report the differences you observed for the counter in Part IV, and provide circuit diagrams in block form (hand drawn) for the circuits that you design for Parts V and VI.
In-lab marks: You are required to demonstrate to a TA Parts V and VI.

Part I

Figure 1 shows a circuit with three different storage elements: a gated D latch, a positive-edge triggered D flip-flop, and a negative-edge triggered D flip-flop.

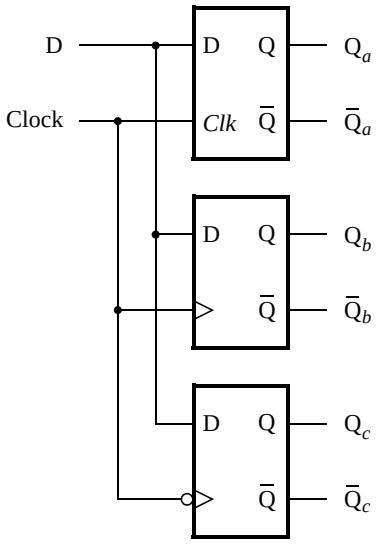
Implement and simulate this circuit using the Quartus software as follows:

1. Create a new Quartus project.
2. Write a Verilog file that instantiates the three storage elements.
3. Use the Quartus Waveform Editor to create a Vector Waveform File (.vwf) which specifies the inputs and outputs of the circuit. Draw the inputs D and *Clock* as indicated in Figure 1. Use functional simulation to obtain the three output signals. Observe the different behavior of the three storage elements.

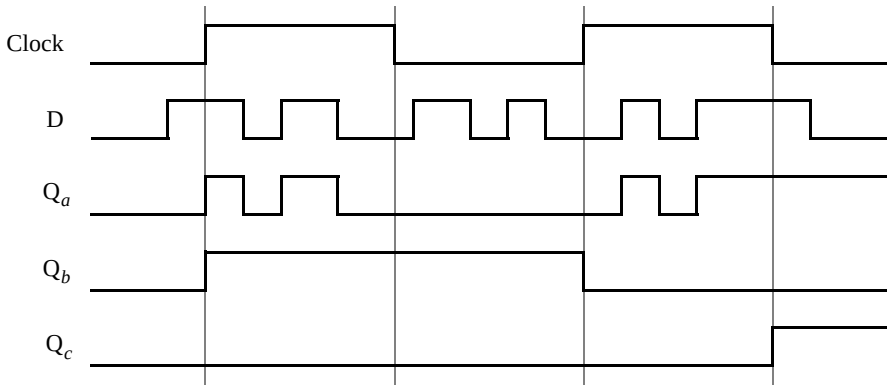
Part II

We wish to display the hexadecimal value of an 8-bit number A on the two 7-segment displays $HEX3-2$. We also wish to display the hex value of an 8-bit number B on the two 7-segment displays $HEX1-0$. The values of A and B are inputs to the circuit which are provided by means of switches SW_{7-0}. To input the values of A and B, first set the switches to the desired value of A, store these switch values in a register, and then change the switches to the desired value of B. Finally, use an adder to generate the arithmetic sum $S = A + B$ (the + symbol means *addition* in this expression), and display this sum on the 7-segment displays $HEX5-4$. Show the carry-out produced by the adder on LEDR[0].

1. Create a new Quartus project which will be used to implement the desired circuit on the DE1-SoC board.
2. Write a Verilog file that provides the necessary functionality. Use KEY_0 as an active-low asynchronous reset, and use KEY_1 as a clock input.
3. Simulate your Verilog using a set of inputs that demonstrates your code is correct.
4. Include the necessary pin assignments for the pushbutton switches and 7-segment displays, and then compile the circuit.
5. Download the circuit onto the DE1-SoC board and test its functionality by operating the switches and observing the output displays.



(a) Circuit



(b) Timing diagram

Figure 1: Circuit and waveforms for Part I.

Part III

Consider the circuit in Figure 2. 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 (logic-1). The counter is reset to 0 by setting the *asynchronous Clear* signal low; i.e., the *Clear* signal is an asynchronous reset. You are to implement a 16-bit counter of this type.

1. Write a Verilog file that defines a 16-bit counter by using the structure depicted in Figure 2. Your code should include a T flip-flop module that is instantiated sixteen times to create the counter. Compile the circuit. Use the Technology Map and/or RTL netlist viewers in Quartus to examine the structure of the synthesized circuit.

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* (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 DE1-SoC board, and compile the code. Download and test your circuit.

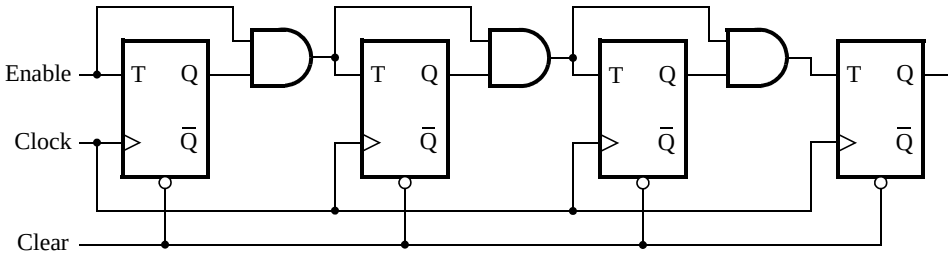


Figure 2: A 4-bit counter.

4. Determine the number of ALMs needed for the FPGA implementation of your counter. An ALM is an "adaptive logic module", which is Intel/Altera lingo for a 6-input look-up table, with extra circuitry that can implement the functionality of two full adders.

Part IV

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, which would be within an `always` block:

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

Compile a 16-bit version of this counter and, as in Part III, determine the number of ALMs needed. Use the RTL Viewer to see the structure of this implementation and comment on differences with the design from Part III. Implement the counter on the DE1-SoC board, using the displays *HEX3-0* to show the counter value.

Part V

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 DE1-SoC board (named $CLOCK\_50$). 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. A partial design of the required circuit is shown in Figure 3. The figure shows how a large bit-width counter can be used to produce an enable signal for a smaller counter. The rate at which the smaller counter increments can be controlled by choosing an appropriate number of bits in the larger counter.

1. Draw and label a block diagram of the circuit, as part of your pre-lab.
2. Write Verilog for the circuit. The only input to your circuit should be $CLOCK\_50$; the only outputs should be for $HEX0$.
3. Simulate your circuit to verify its correctness. Think carefully about a limited set of simulation input vectors that will allow you to verify correctness.
4. Make the necessary pin assignments needed to implement the circuit on the DE1-SoC board, and compile the code. Download and test your circuit.
5. Demonstrate the working circuit to the TA for the in-lab-marked portion of the lab.

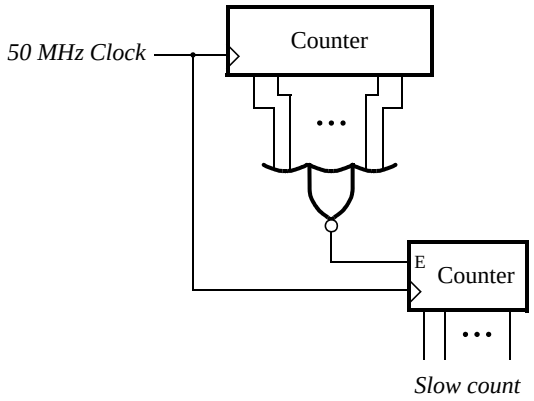


Figure 3: Making a slow counter.

Part VI

Design and implement a circuit that displays the word dE1, in a rotating fashion, on six 7-segment displays. Make the letters move from right to left in intervals of about one second. The patterns that should be displayed in successive clock intervals, as shown in Table 1.

1. Draw and label a block diagram of the circuit, as part of your pre-lab.
2. Write Verilog for the circuit. The only input to your circuit should be $CLOCK\_50$; the only outputs should be for the HEX displays.
3. Simulate your circuit to verify its correctness. Think carefully a way to simulate the circuit, with a limited set of simulation input vectors that will allow you to verify correctness.
4. Make the necessary pin assignments needed to implement the circuit on the DE1-SoC board, and compile the code. Download and test your circuit.
5. Demonstrate the working circuit to the TA for the in-lab-marked portion of the lab.

Clock cycle	Display					
0				d	E	1
1			d	E	1	
2		d	E	1		
3	d	E	1			
4	E	1				d
5	1				d	E

Table 2. Rotating the word dE1 on six displays.