Laboratory Exercise 6

Timers and Real-time Clock

The purpose of this exercise is to study the use of clocks in timed circuits. The designed circuits are to be implemented on an Intel FPGA DE10-Lite, DE0-CV, DE1-SoC, or DE2-115 board.

Background

In the Verilog hardware description language we can describe a variable-size counter by using a parameter declaration. An example of an *n*-bit counter is shown in Figure 1.

```
module  counter (Clock, Reset_n, Q);
    parameter n = 4;

    input Clock, Reset_n;
    output [n-1:0] Q;
    reg [n-1:0] Q;

    always @(posedge Clock or negedge Reset_n)
    begin
        if (!Reset_n)
            Q <= 1'd0;
        else
            Q <= Q + 1'b1;
    end
endmodule
```

Figure 1: A Verilog description of an *n*-bit counter.

The parameter *n* specifies the number of bits in the counter. When instantiating this counter in another Verilog module, a particular value of the parameter *n* can be specified by using a **defparam** statement. For example, an 8-bit counter can be specified as:

```
counter eight_bit (Clock, Reset_n, Q);
    defparam eight_bit.n = 8;
```

By using parameters we can instantiate counters of different sizes in a logic circuit, without having to create a new module for each counter.

Part I

Create a modulo-k counter by modifying the design of an 8-bit counter to contain an additional parameter. The counter should count from 0 to $k-1$. When the counter reaches the value $k-1$, then the next counter value should be 0. Include an output from the counter called *rollover* and set this output to 1 in the clock cycle where the count value is equal to $k-1$.
Perform the following steps:

1. Create a new Quartus project which will be used to implement the desired circuit on your DE-series board.

2. Write a Verilog file that specifies the circuit for $k = 20$, and an appropriate value of n. Your circuit should use pushbutton KEY_0 as an asynchronous reset and KEY_1 as a manual clock input. The contents of the counter should be displayed on the red lights *LEDR*. Also display the *rollover* signal on one of the LEDR lights.
3. Include the Verilog file in your project and compile the circuit.
4. Simulate the designed circuit to verify its functionality.
5. Make the necessary pin assignments needed to implement the circuit on your DE-series board, and compile the circuit.
6. Verify that your circuit works correctly by observing the lights.

Part II

Using your modulo-counter from Part I as a subcircuit, implement a 3-digit BCD counter (hint: use multiple counters, not just one). Display the contents of the counter on the 7-segment displays, $HEX2-0$. Connect all of the counters in your circuit to the 50-MHz clock signal on your DE-series board, and make the BCD counter increment at one-second intervals. Use the pushbutton switch KEY_0 to reset the BCD counter to 0.

Part III

Design and implement a circuit on your DE-series board that acts as a real-time clock. It should display the minutes (from 0 to 59) on $HEX5-4$, the seconds (from 0 to 59) on $HEX3-2$, and hundredths of a second (from 0 to 99) on $HEX1-0$. Use the switches SW_{7-0} to preset the minute part of the time displayed by the clock when KEY_1 is pressed. Stop the clock whenever KEY_0 is being pressed and continue the clock when KEY_0 is released.

Part IV

An early method of telegraph communication was based on the Morse code. This code uses patterns of short and long pulses to represent a message. Each letter is represented as a sequence of dots (a short pulse), and dashes (a long pulse). For example, the first eight letters of the alphabet have the following representation:

A	• —
B	— • • •
C	— • — •
D	— • •
E	•
F	• • — •
G	— — •
H	• • • •

Design and implement a circuit that takes as input one of the first eight letters of the alphabet and displays the Morse code for it on a red LED. Your circuit should use switches SW_{2-0} and pushbuttons KEY_{1-0} as inputs. When a user presses KEY_1, the circuit should display the Morse code for a letter specified by SW_{2-0} (000 for A, 001 for B, etc.), using 0.5-second pulses to represent dots, and 1.5-second pulses to represent dashes. Pushbutton KEY_0 should function as an asynchronous reset. A high-level schematic diagram of the circuit is shown in Figure 2.

Hint: Use a counter to generate 0.5-second pulses, and another counter to keep the $LEDR_0$ light on for either 0.5 or 1.5 seconds.

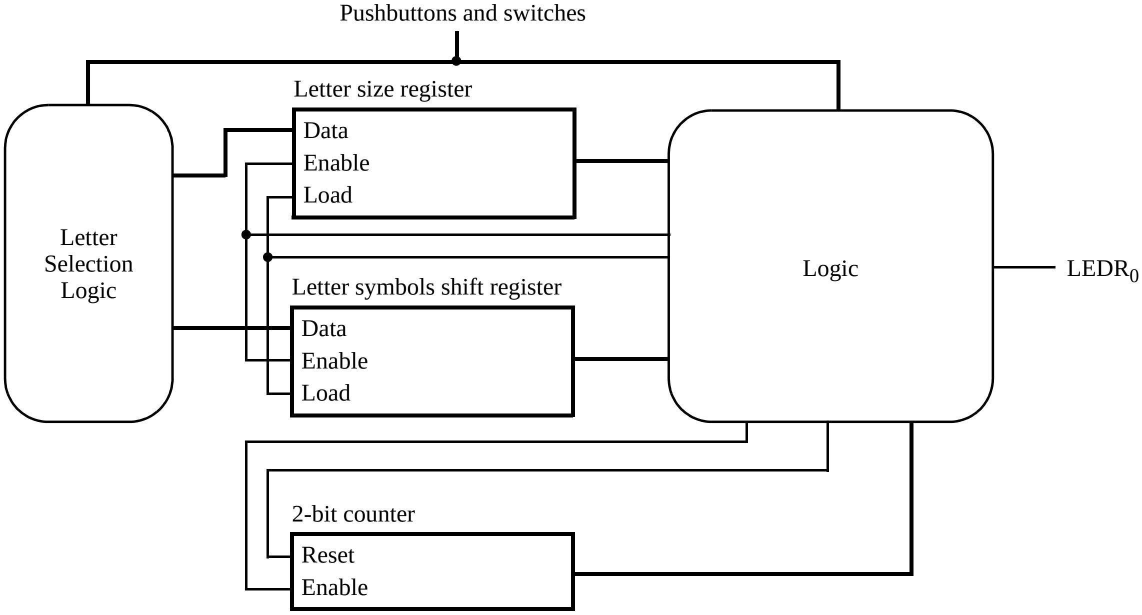

Figure 2: High-level schematic diagram of the circuit for part IV.