Laboratory Exercise 5

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 Altera DE0 board.

Background

In 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 <= '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. A particular value of this parameter is defined 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 $k-1$ the value that follows should be 0.

Your circuit should use pushbutton KEY_0 as an asynchronous reset, KEY_1 as a manual clock input. The contents of the counter should be displayed on green LEDs. Compile your design with Quartus II software, download your design onto a DE0 board, and test its operation. Perform the following steps:

1. Create a new Quartus II project which will be used to implement the desired circuit on the DE0 board.

2. Write a Verilog file that specifies the desired circuit.
3. Include the Verilog file in your project and compile the circuit.
4. Simulate the designed circuit to verify its functionality.
5. Assign the pins on the FPGA to connect to the lights and pushbutton switches, by importing the pin-assignment file *DE0_pin_assignments.qsf*.
6. Recompile the circuit and download it into the FPGA chip.
7. Verify that your circuit works correctly by observing the display.

Part II

Implement a 3-digit BCD counter. Display the contents of the counter on the 7-segment displays, *HEX2−0*. Derive a control signal, from the 50-MHz clock signal provided on the DE0 board, to increment the contents of the counter at one-second intervals. Use the pushbutton switch KEY_0 to reset the counter to 0.

Part III

Design and implement a circuit on the DE0 board that acts as a real-time clock. It should display the minutes (from 0 to 60) on *HEX3−2* and the seconds (from 0 to 60) on *HEX1−0*. Use the switches SW_{7-0} to preset the minute part of the time displayed by the clock.

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 green 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 $LEDG_0$ light on for either 0.5 or 1.5 seconds.

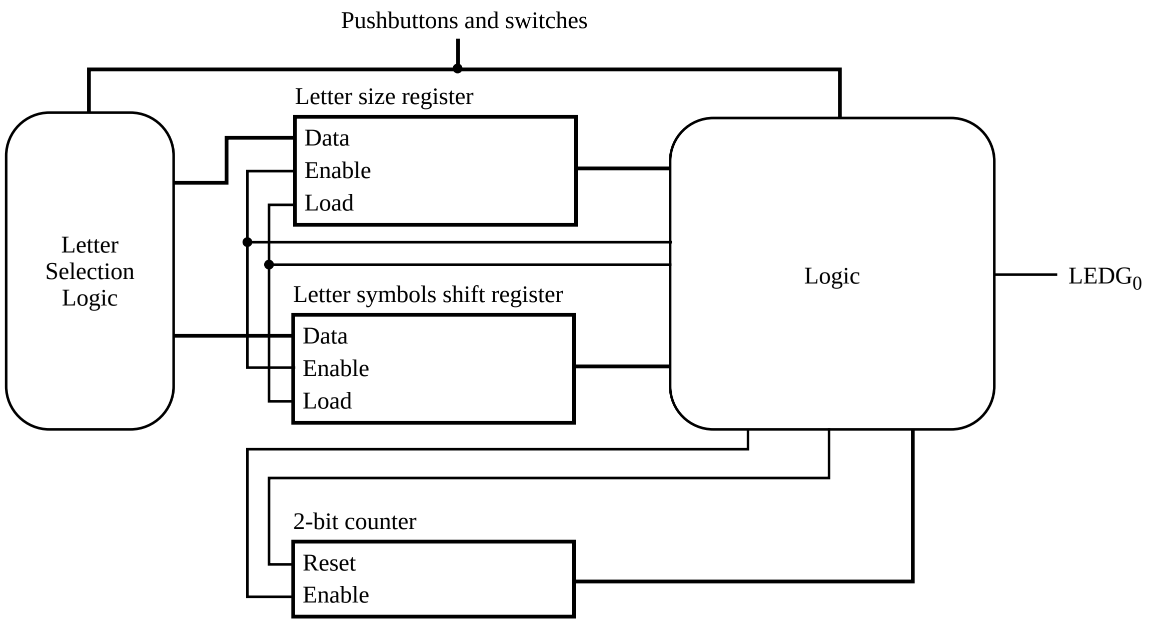

Figure 2: High-level schematic diagram of the circuit for part IV.

Preparation

The recommended preparation for this laboratory exercise includes:

1. Verilog code for **Part I**
2. Simulation of the Verilog code for **Part I**
3. Verilog code for **Part II**
4. Verilog code for **Part III**