Lab no 04: Finite State Machine - One-hot Counter

The purpose of this Lab is to learn how to:

1) Implement a finite state machine (FSM) for a 10-bit one-hot counter on FPGA.
2) Implement a clock divider to generate a 1-Hz clock From the FPGA 50MHz clock.
3) Connect sub-modules on a top-level module. You will connect the clock divider and the one-hot counter.
4) Use the LEDs on the FPGA board to display the output of the one-hot counter.

Parts: -

1. Code the behavioral description of the clock divider.
2. Code the behavioral description of the 10-bit up/down one-hot counter.
3. Connect the clock divider and the one-hot counter on the top-level module and run it on FPGA.

Part 1. <u>Code the behavioral description of the clock divider</u>

The clock frequency of the MAX 10 FPGA is 50MHz.
To generate a 1-Hz clock, you need to implement a clock divider that slows down the FPGA clock. A clock divider is a counter. Our implementation divides the FPGA clock by 50 million counts, 25 million counts for One pulse, and 25 million counts for Zero pulse.

Figure 1. illustrates an example of a clock divider using a 3-bit counter, with three flip-flops. It counts from 0 to 7. As shown, the initial input frequency is "**<u>divided-by-two</u>**" by the first flip-flop ($f \div 2$) and then "divided-by-two" again by the second flip-flop ($(f \div 2) \div 2$), giving an output frequency which has effectively been divided four times, then its output frequency becomes one-quarter value (25%) of the original clock frequency, ($f \div 4$), and so on.

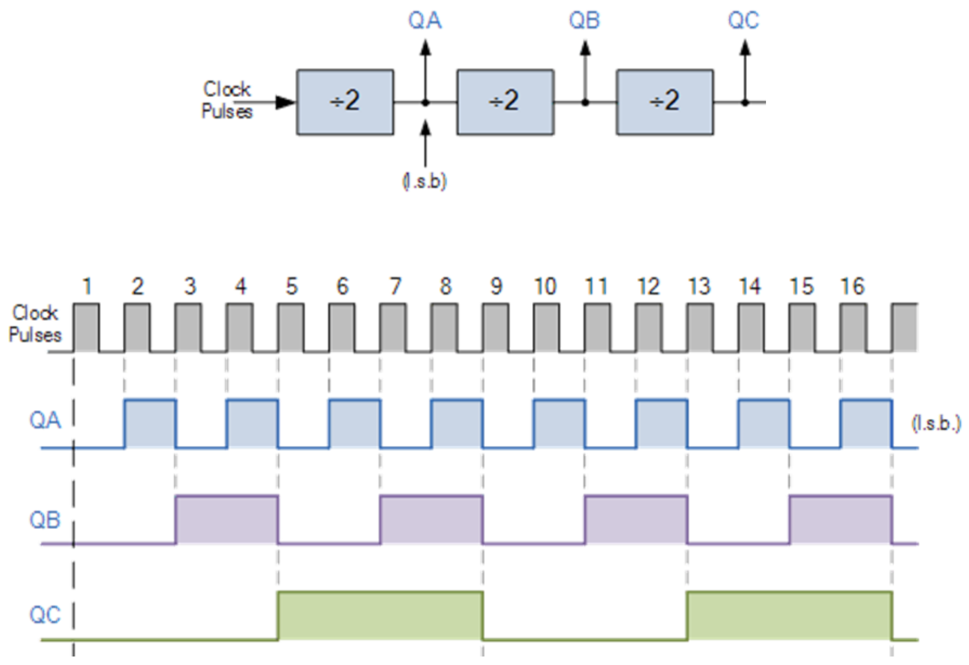

Figure 1. 3-bit counter.

Behavioral Description Verilog Code for Clock Divider

```
module clock_divider (clk, reset, CLK1Hz);
input clk, reset;
output CLK1Hz;
// ------------------------------------------------- //
reg CLK1Hz;
reg [24:0] count; // log2(25 million)
// ------------------------------------------------- //
always @(posedge clk or posedge reset)
begin
     if(reset) // initial (zero)
          begin
               count <= 0;
               CLK1Hz <= 0;
          end
     else
          begin
               if(count < 25_000_000)
                 count <= count + 1; // count 25 million
               else
                 begin
                    CLK1Hz = ~CLK1Hz; // toggle the clk high\low
                    count <= 0;
                 end
          end
end
// ------------------------------------------------- //
endmodule
```

Part 2. Code the behavioral description of 10-bit up/down One-hot counter

Based on the FSM in Figure 2, build the up/down one-hot counter.

Describe the FSM by using a Verilog ***case*** statement in an ***always*** block.

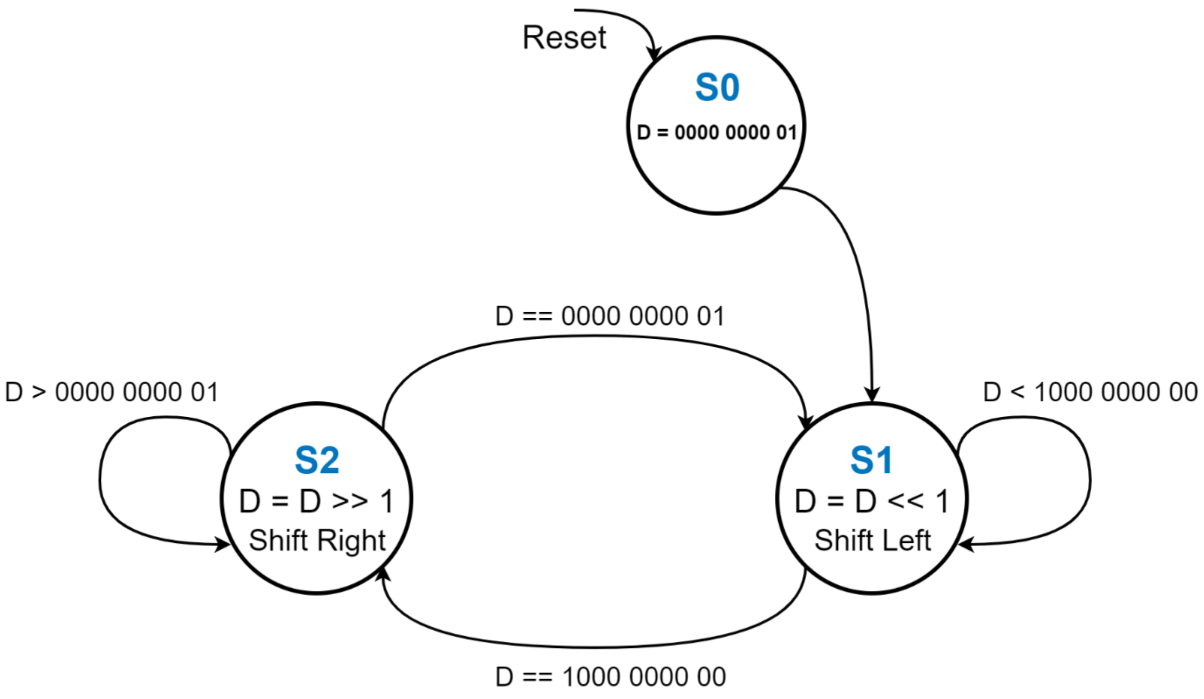

Figure 2. 10-bit up-down one-hot counter FSM.

Behavioral Description Verilog Code for One-hot Counter

```
module one_hot_counter_fsm (clk, reset, oneHot);
input clk, reset;
output [9:0] oneHot;
// ---------------------------------------------------- //
parameter S0 = 2'b00, // initial state
          S1 = 2'b01, // shift left state
          S2 = 2'b10; // shift right state
reg [9:0] oneHot;
reg [1:0] State;
// ---------------------------------------------------- //
always @(posedge clk or posedge reset)
begin
    if(reset) // reset the state to initial state and output to zeros
      begin
        State = S0;
        oneHot <= 10'b0000_0000_01;
      end
    else
      begin
      case(State) // state transitions
        S0: State <= S1; // move to state S1 (shift left state)
        S1: if(oneHot < 10'b1000_0000_00)
                oneHot <= oneHot << 1; // shift one left
            else
                State <= S2; // move to S2 (shift right state)
        S2: if(oneHot > 10'b0000_0000_01)
                oneHot <= oneHot >> 1; // shift one right
            else
                State <= S1; // move to S1 (shift left state)
      endcase
      end
end
// ---------------------------------------------------- //
endmodule
```

Part 3. Connect the Clock divider and the One-hot counter on the top-level module

```
module one_hot_counter (clk, reset, oneHot);
input clk, reset;
output [9:0] oneHot;
// Instantiation of the  clcok divider
clock_divider clock_divider_1Hz (clk, reset, CLK1Hz);
// Instantiation of the one hot couter fsm
one_hot_counter_fsm one_hot_cnt (CLK1Hz,reset, oneHot);
endmodule
```

Once you are confident that the circuit works properly, as a result of simulation, Download the circuit to the FPGA.

Refer to Lab 2 to program the FPGA by Quartus

Use DE10-Lite kit, Altera MAX 10 based FPGA board

Check DE10-lite user manual (Here) for pin assignment.

Notes:

Assign **clk** to the **MAX10_CLK1_50**.

Assign **reset** to the **switch button (SW0)** on FPGA.

Assign the output of the counter to the **10 LEDs (LEDR [0-9])** on FPGA.