NATIONAL UNIVERSITY HO CHI MINH CITY

UNIVERSITY OF TECHNOLOGY – FACULTY OFCOMPUTER SCIENCE AND ENGINEERING

REPORT LAB 3
LOGIC DESIGN WITH HDL

ADVISOR: NGUYỄN THẾ BÌNH

TEAM 8

Name:	**Students's ID**
Phạm Anh Tài	2350023
Phạm Ngọc Huy	2352404
Trần Đình Duy Khương	2352638
Lục Tấn Phúc	2352935

2.1 Exercise 1
Clock Frequency Divider
Police Siren: Design a circuit that generate a 1 Hz output signal using Behavioral Model. This signal is connected to 2 RGB LEDs (1 displays the blue color, 1 display the red color) on Arty-Z7 FPGA Board to make it blink interleave with each other (turn on for 0.5s - turn off for 0.5s). Know that the input clock frequency is 125 MHz.

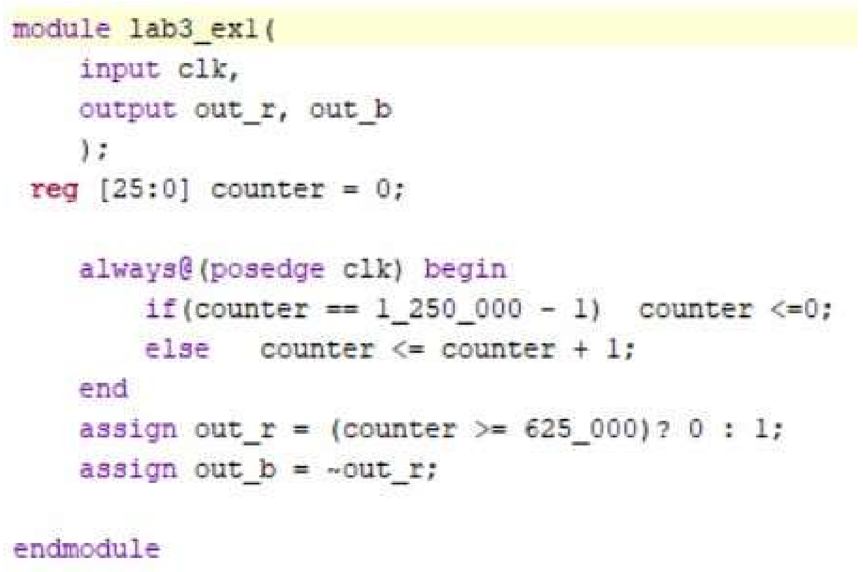
Write test benches to simulate the circuits.

Test the circuits on FPGA board using LEDs and RGB LED

- *Theoretical basis:*

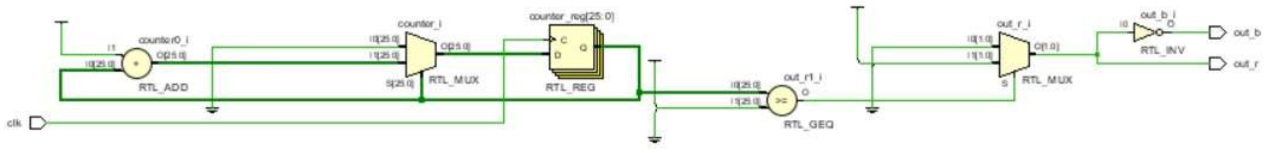
+ In order to generate frequency's input from 125 MHz to frequency's output 1Hz, it means that when the input completes 125.000.000 periods, the output just finishes 1 period. Therefore, the output should be at the state 1 in 125.000.000/2 periods, and then at 0 in 125.000.000/2 periods left.
+ In addition, to create a police siren when the led red is on, the led blue should be off, similar to the opposite.

-Module code:

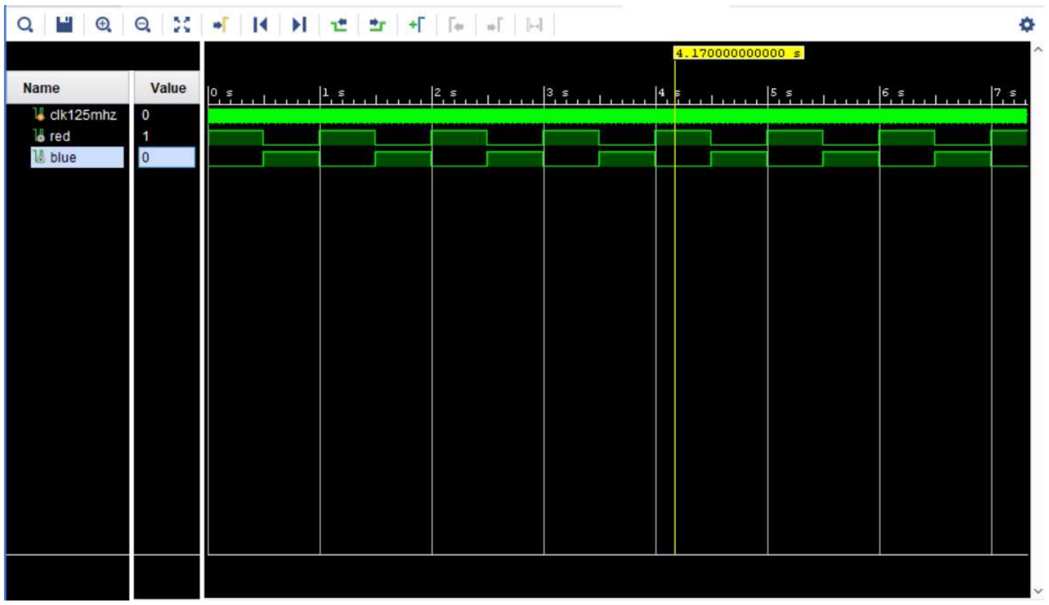


```
module lab3_ex1(
    input clk,
    output out_r, out_b
    );
  reg [25:0] counter = 0;

    always@(posedge clk) begin
        if(counter == 1_250_000 - 1)  counter <=0;
        else    counter <= counter + 1;
    end
    assign out_r = (counter >= 625_000)? 0 : 1;
    assign out_b = ~out_r;

endmodule
```

-RTL analysis:



-Simulation:



Name	Value
clk125mhz	0
red	1
blue	0

Explanation:
+ Because 125MHz is too large to simulate, we consider the clock frequency is 6Hz.
+ As result, while the led 4 red is off, the led 5 blue is on. Similarly, the opposite is true.
+ Another noticeable observation is that the period of output is six times the period clock, which means the frequency of output led is 1Hz. Therefore, when the clock frequency is 125MHz, the output frequency is 1 Hz, too.

2.2 Exercise 2

Edge Detection circuit.

a. Design a Rising Edge Detection circuit. This circuit will use at least 2 flip-flops. The behavior of the circuit is similar to the waveform in Figure 1. Assume that the in signal's HIGH levels last equal to or longer than a clock cycle.

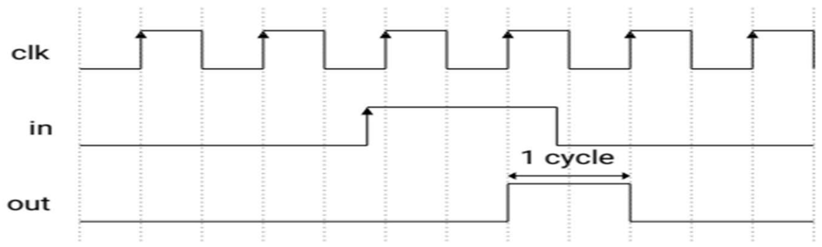
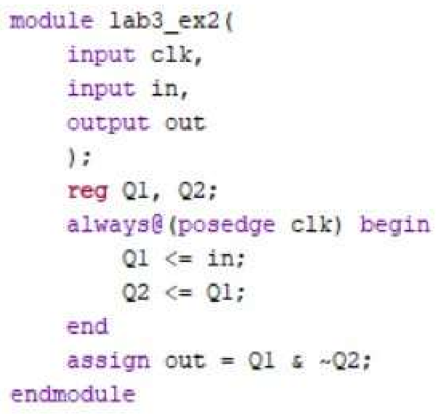


Figure 1: Rising Edge Detector behavior

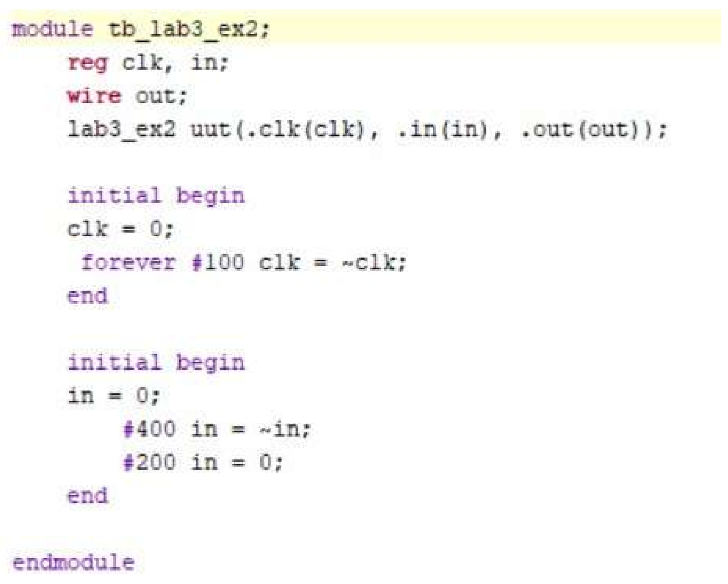
The output is active HIGH in 1 cycle of clock when a rising edge occurs in input signal. Delay is within 0-2 clock cycles. Write RTL code and test benches to simulate the circuit.

-Module code:



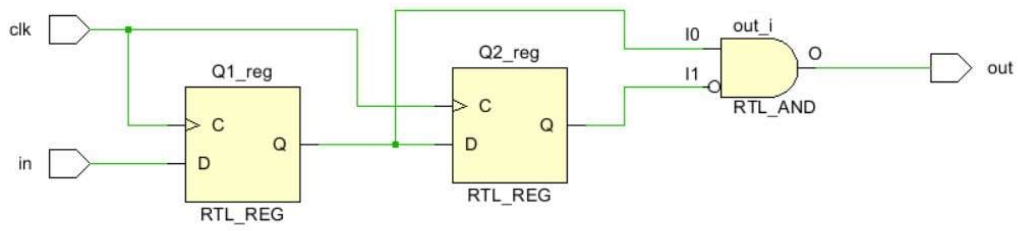
```
module lab3_ex2(
    input clk,
    input in,
    output out
    );
    reg Q1, Q2;
    always@(posedge clk) begin
        Q1 <= in;
        Q2 <= Q1;
    end
    assign out = Q1 & ~Q2;
endmodule
```

-Testbench:

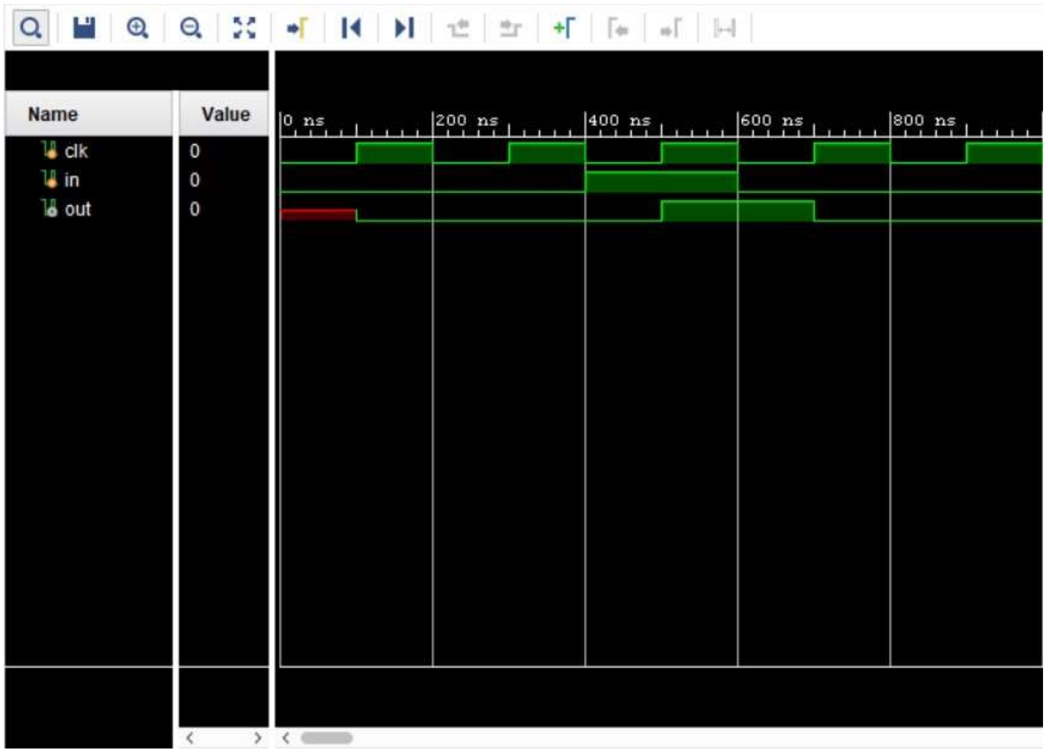


```
module tb_lab3_ex2;
    reg clk, in;
    wire out;
    lab3_ex2 uut(.clk(clk), .in(in), .out(out));

    initial begin
    clk = 0;
     forever #100 clk = ~clk;
    end

    initial begin
    in = 0;
        #400 in = ~in;
        #200 in = 0;
    end

endmodule
```

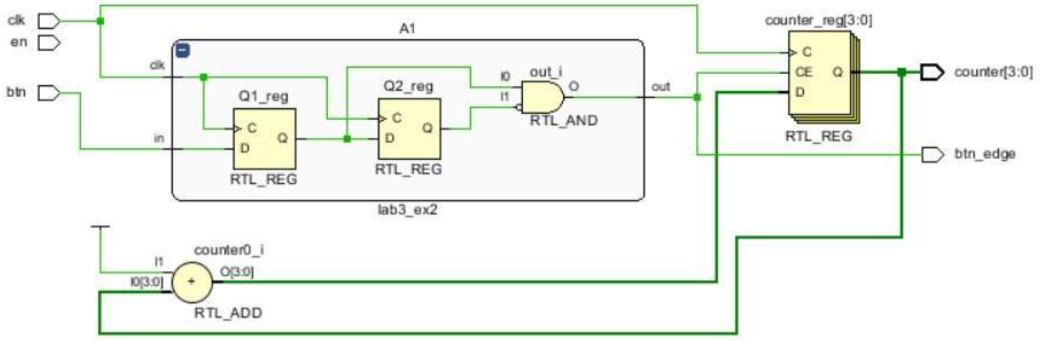
-RTL analysis:



-Simulation:

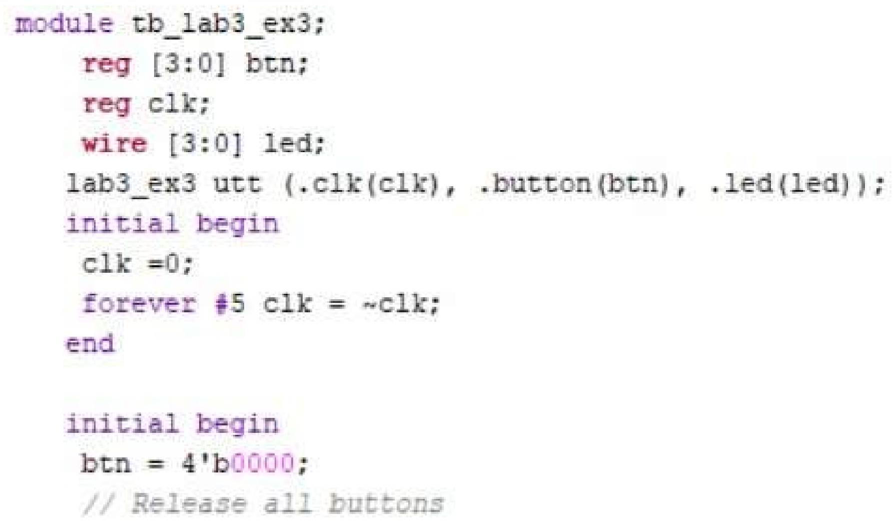


b. Write a 4-bit binary counter that counts up 1 unit when a button is pushed. Use the edge detection circuit to generate an enable signal for the counter when pushing the button. Test the design on FPGA board.

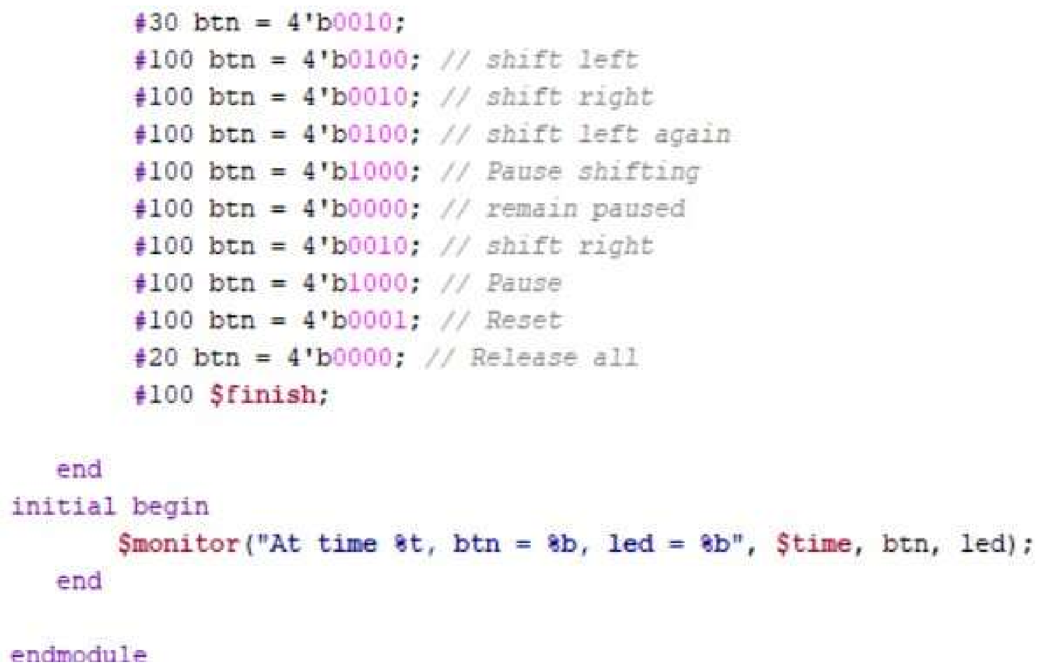


2.3 Exercise 3
Change mode String bit LED circuit. Use Verilog HDL to model a state machine for a circuit that changes display mode of a bit string. In initial, LEDs show the default 4-bit random string which is performed by a reset signal, example string: 0011. And buttons in board will set the display mode as follow:
• Button 0: Mode Reset: Show the default 4-bit string on LEDs.
• Button 1: Mode Circular Shift Left Ring : Shift 4-bit string to left in a ring every 1s.
• Button 2: Mode Circular Shift Right Ring: Shift 4-bit string to right in a ring every 1s.
• Button 3: Pause: Pause the current shifting string.
Draw a state diagram to illustrate the designed FSM. Student can use Moore or Mealy model. Write a test bench to simulate the circuit and test the circuit on the Arty-Z7 board.

-Testbench code:

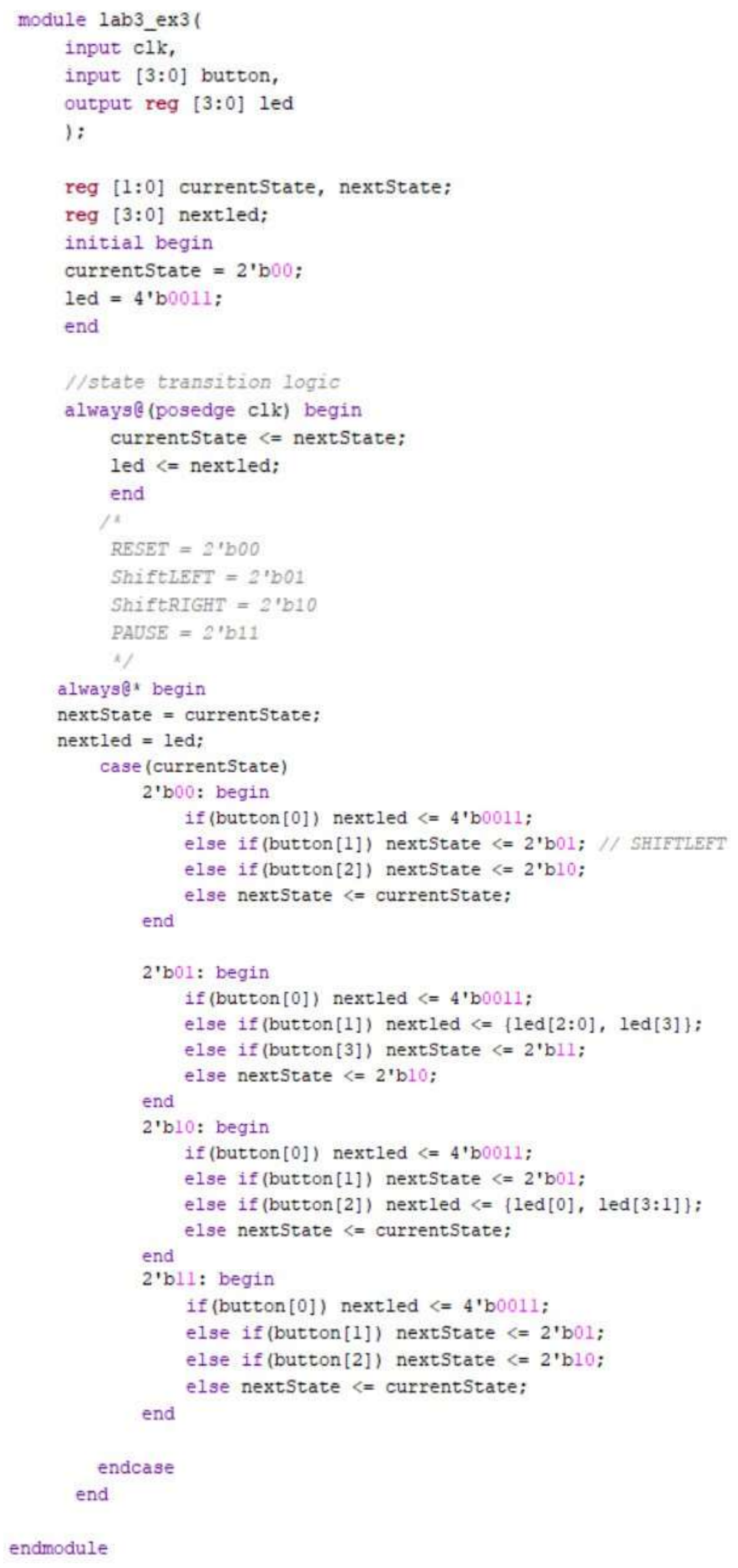


```
module tb_lab3_ex3;
    reg [3:0] btn;
    reg clk;
    wire [3:0] led;
   lab3_ex3 utt (.clk(clk), .button(btn), .led(led));
   initial begin
    clk =0;
    forever #5 clk = ~clk;
   end

   initial begin
    btn = 4'b0000;
    // Release all buttons
```



```
        #30 btn = 4'b0010;
        #100 btn = 4'b0100; // shift left
        #100 btn = 4'b0010; // shift right
        #100 btn = 4'b0100; // shift left again
        #100 btn = 4'b1000; // Pause shifting
        #100 btn = 4'b0000; // remain paused
        #100 btn = 4'b0010; // shift right
        #100 btn = 4'b1000; // Pause
        #100 btn = 4'b0001; // Reset
        #20 btn = 4'b0000; // Release all
        #100 $finish;

   end
initial begin
       $monitor("At time %t, btn = %b, led = %b", $time, btn, led);
   end

endmodule
```

-Module code:

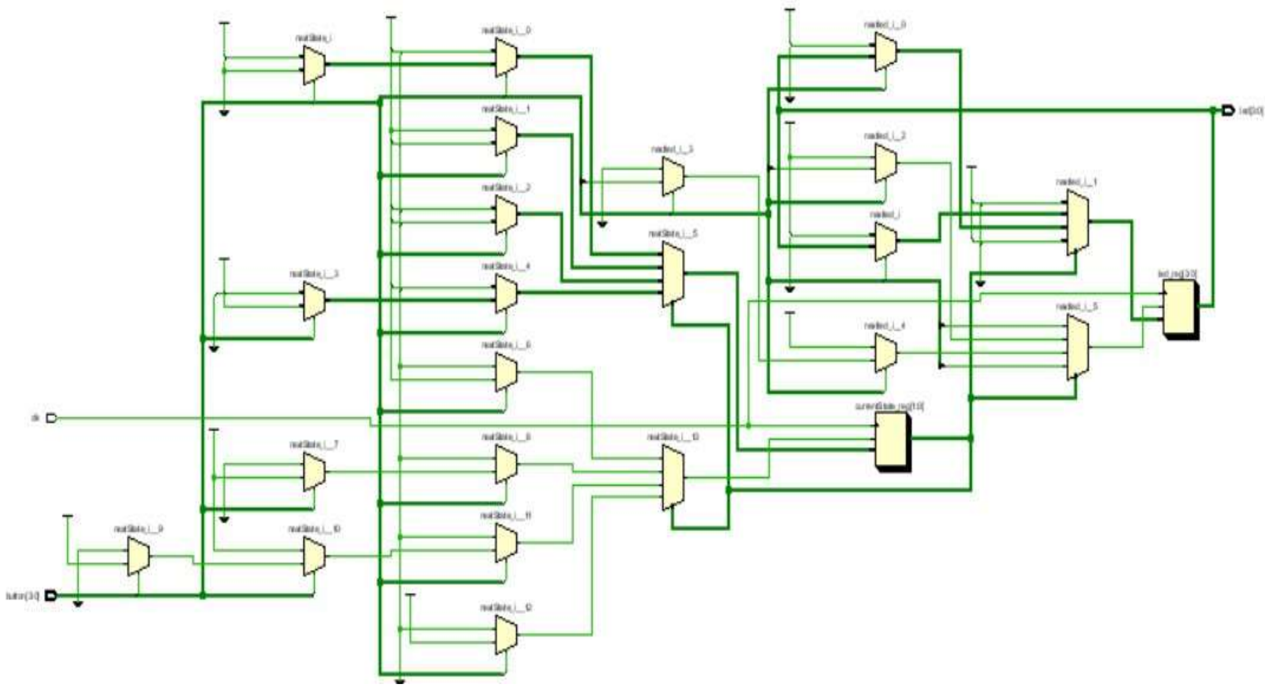


```
module lab3_ex3(
    input clk,
    input [3:0] button,
    output reg [3:0] led
    );

    reg [1:0] currentState, nextState;
    reg [3:0] nextled;
    initial begin
    currentState = 2'b00;
    led = 4'b0011;
    end

    //state transition logic
    always@(posedge clk) begin
        currentState <= nextState;
        led <= nextled;
        end
       /*
        RESET = 2'b00
        ShiftLEFT = 2'b01
        ShiftRIGHT = 2'b10
        PAUSE = 2'b11
        */
    always@* begin
    nextState = currentState;
    nextled = led;
        case(currentState)
            2'b00: begin
                if(button[0]) nextled <= 4'b0011;
                else if(button[1]) nextState <= 2'b01; // SHIFTLEFT
                else if(button[2]) nextState <= 2'b10;
                else nextState <= currentState;
            end

            2'b01: begin
                if(button[0]) nextled <= 4'b0011;
                else if(button[1]) nextled <= {led[2:0], led[3]};
                else if(button[3]) nextState <= 2'b11;
                else nextState <= 2'b10;
            end
            2'b10: begin
                if(button[0]) nextled <= 4'b0011;
                else if(button[1]) nextState <= 2'b01;
                else if(button[2]) nextled <= {led[0], led[3:1]};
                else nextState <= currentState;
            end
            2'b11: begin
                if(button[0]) nextled <= 4'b0011;
                else if(button[1]) nextState <= 2'b01;
                else if(button[2]) nextState <= 2'b10;
                else nextState <= currentState;
            end

        endcase
     end

endmodule
```

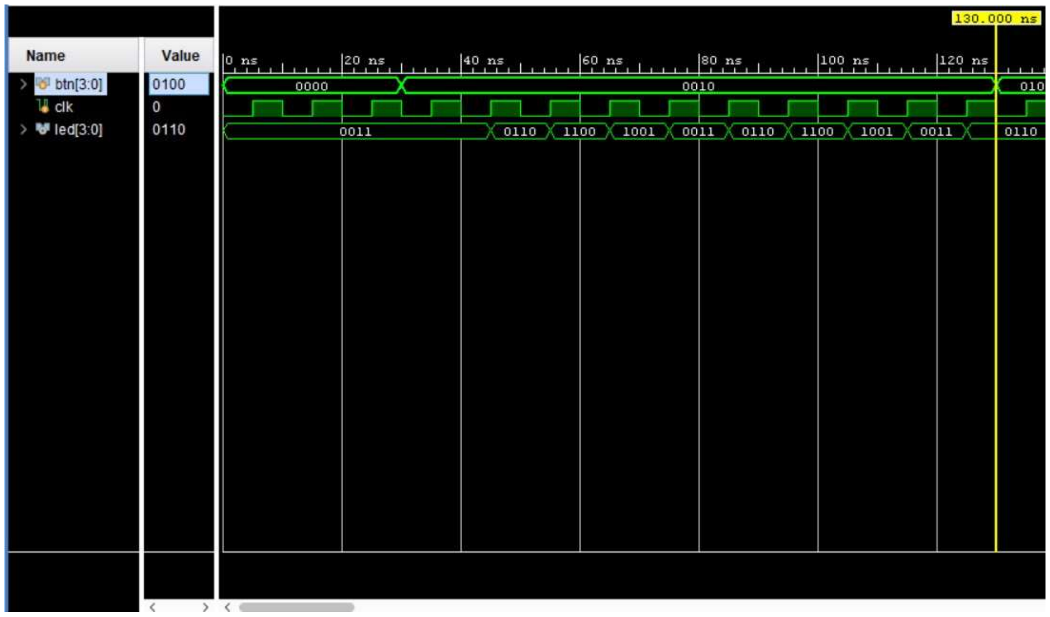
Explanation:

We use finite state machine (FSM) of Moore model in this exercise. Putting "button" in always block, create "temp" to register immediately signals of user. In this model, there are 4 inputs which are 0001, 0010, 0100, 1000 and 4 states which are 0011 (default), 0110, 1100, 1001

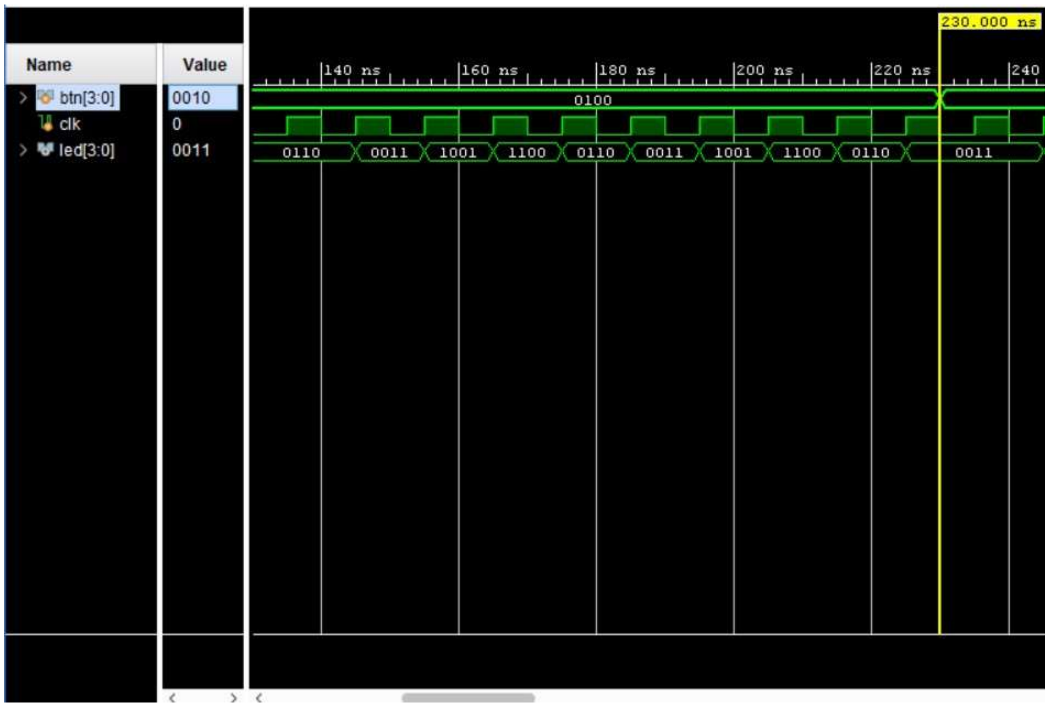
-RTL analysis:



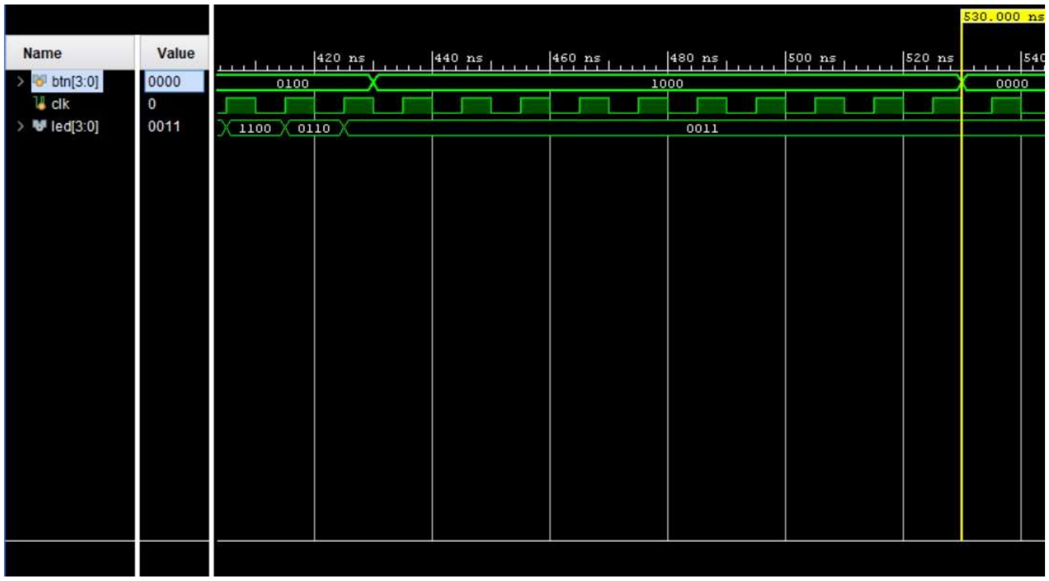
-Simulation:



Button 1: shift left



Button 2 : shift right



Button 3 : pause

+ When button 0 is selected (0001), the output is 0011 by default. Next, when button 2 is selected (0010), the output shifts left to 1001, 1100, 0110, 0011, 1001 respectively to clock signals.
+ Then, the button 3 is selected (1000), the output is paused at 1001. The button 1 is selected afterwards, and the output is 0011 by default.
+ When button 2 is selected (0100) the output shifts right to 0110, 1100, 1001, 0011, 0110 respectively to clock signals.
+ Finally, when button 3 is selected, the output is paused at 0110.