ECEN 749 – Lab Report

Lab Number: 1

Lab Title: Using the Vivado

Section Number: 601

Student's Name: Quy Truong Van

Student's UIN: 132005189

Lab Date: 01/24/2023

TA: Kushagra Gupta

Purpose/Introduction

This lab is to get students to review Vivado, Verilog, and procedure to design hardware and program FPGA. Students write Verilog to utilize LEDs, DIP Switches, and Buttons to design simple counter, clock divider, and Jackpot. This lab is important step to review all basic of Vivado, Verilog, and FPGA at the start of the semester.

Procedure

1. Get familiar with Vivado and design a switch.v that will turn on the LEDs corresponding to the DIP Switch in the same column.
    a. Choose the right FPGA xc7z010clg400-1
    b. Create and design switch.v with 4 inputs are DIP Switches and 4 outputs are 4 LEDs
    c. Design XDC for switch.v
    d. Generate bitstream to program FPGA, then demo it to TA

2. Implement 4-bit counter that will go up when pressing the Button one and go down when pressing the button two.
    a. Design counter following the design requirement including clock.
    b. Modifying XDC file to add clock signal at the rate of 125 MHz which used for counter
    c. Add clock divider to slow down the clock cycle which product visible to human eyes.
    d. Demo to TA

3. Design a Jackpot with one-hot encoding sequence.
    a. Design the Jackpot to run one-hot encoding sequence in normal speed that eye can distinguish states.
    b. Design the DIP Switch so that when turning it on as the same time the corresponding LED glow will glow all 4 LEDs.
    c. Demo to TA

Result

All demos are approved by TA.

1. Switch

This code (Figure 1) use blocking assignment, LEDS will reflect the value of SWITCHES immediately. As long as the Switch is on, the corresponding LED will be on.

2. 4-bit Counter

Non-blocking and blocking assignments are both used in this design (Figure 2). Before, going to the counter, the clock is divided into slow clock used Counter theory. The Slow clock cycle will be equal to 2^n the original clock. In this case, n = 26. With that slow clock, the counter going up and down can be visible. In every positive edge of slow clock, if BTN0 on and BTN1 off, the counter going up. BTN0 off and BTN1 on will let counter going down. Other cases will not change the counter. At the end, LEDs reflect the state of the counter using blocking assignment.

3. Jackpot

First version is Figure 3 in Appendix. In this code, Clock is divided to have a slow clock first. When Reset button is pushed, the Counter will turn to 0. In normal execution, if either Switch is on, the LEDs will stop running. If the

Switch is turn back off, the LEDs will keep running in the normal manner. If the Switch is on at the same moment the corresponding LED is on, 4 LEDs will be on. To start over, the switch needs to be off, and the Reset button needs to be pushed.

Even though this code meets the basic requirements of the lab, and the demo was approved by TA, the design needs to be improved. When the switch is on, if switch does not match the LED, the LED can keep running, but it is not the case in this design.

To implement that feature, edge detector is required to catch the edge moment the switch is on so that even if the switch stays on, the LED keep running will not glow 4 LEDs on. The upgraded version in Figures 4 and 5 will do that.

In this version (Figures 4 and 5), edge detector was designed to has catch the edge and let it stay for at least 2 clock cycle.

Testbench (Figure 6) of the upgraded version shows that if edge detector is 1, Switches at 2 match the LEDs at 2, then LEDs will change to f (4'b1111) till the Reset is pushed. After that, even the switch is on, there is no edge detected, the LEDs run in normal manner until the Switches change to 4'b1000 matching the LEDs at 4'b1000, along with the edge detect is 1, the LEDs turns to 4'b1111 after 1 clock cycle.

Even the upgraded version looks good on simulation, the Jackpot programmed on FPGA was hard to win. The Jackpot with super slow clock signal was not been verified due to the unknown error in my Vivado that does not allow me to do testbench with clock divider properly.

Conclusion

This lab provides students a good review of Vivado, Verilog, and FPGA. This lab not only gives students detail instructions to use Vivado, and program FPGA, but also guide students to how write Verilog step by step. This lab also let student practice critical thinking by asking students design Jackpot, a challenging assignment, requiring multiple side modules like clock divider and edge detector.

Questions

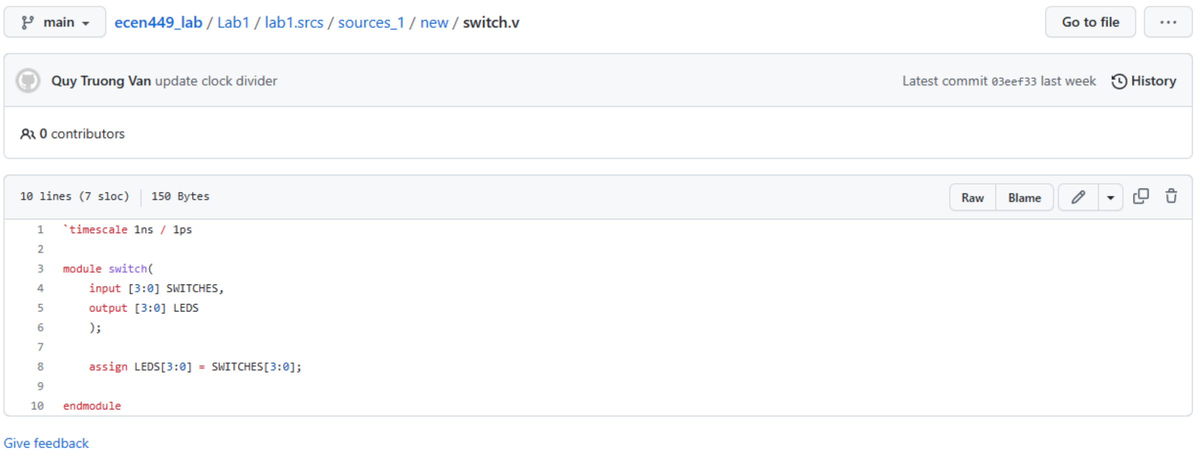
a. How are the user buttons wired on the ZYBO Z7-10 board (i.e. what pins on the FPGA do each of them correspond to and are the signals pulled up and down)? You will have to consult Master XDC file for this information

There are 4 buttons BTN0, BTN1, BTN2, and BTN3 respectively corresponding to 4 pins K18, P16, K19, and Y16. The signals are pulled down according to the Figure 7 got from Zybo Z7 Reference Manual. 4 buttons are switches which is closed when buttons are pushed. A voltage of 3.3 V will go through closed circuit to send signal to corresponding pin.

b. What is the purpose of an edge detection circuit and how should it have been used in this lab?

The purpose of edge detection circuit is to catch the rising moment (like the positive clock edge) of a signal. In this lab, edge detection was used to detect the switch raising signal, so that even after the switch has been staying on, the LED turn on later corresponding to the Switch will not reach the win state (4'b1111).

Appendix



main ecen449_lab / Lab1 / lab1.srcs / sources_1 / new / switch.v

Quy Truong Van update clock divider — Latest commit 03eef33 last week — History

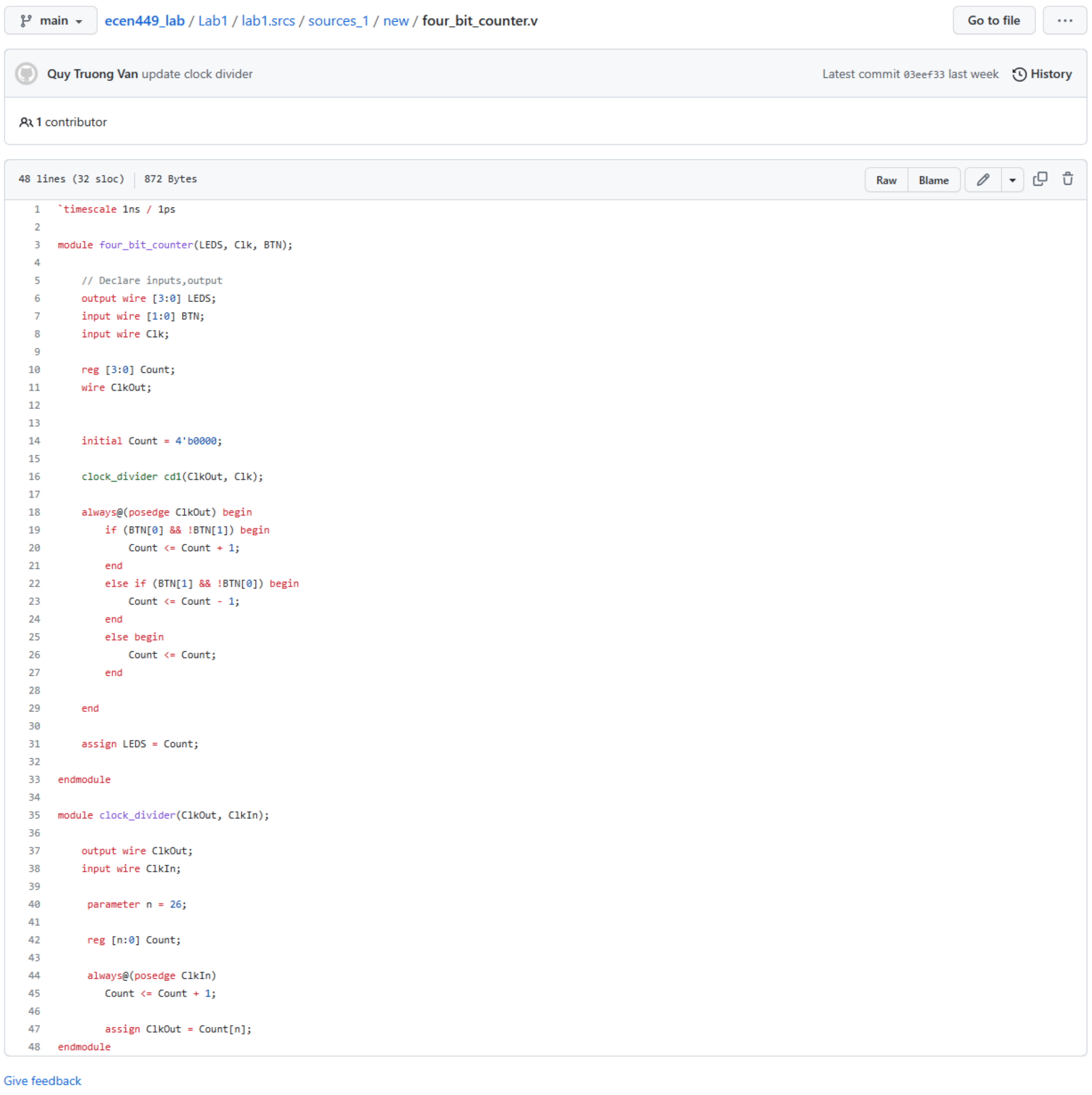
0 contributors

10 lines (7 sloc) | 150 Bytes

```verilog
`timescale 1ns / 1ps

module switch(
    input [3:0] SWITCHES,
    output [3:0] LEDS
    );

    assign LEDS[3:0] = SWITCHES[3:0];

endmodule
```

Give feedback

Figure 1: Switch



main ecen449_lab / Lab1 / lab1.srcs / sources_1 / new / four_bit_counter.v

Quy Truong Van update clock divider — Latest commit 03eef33 last week — History

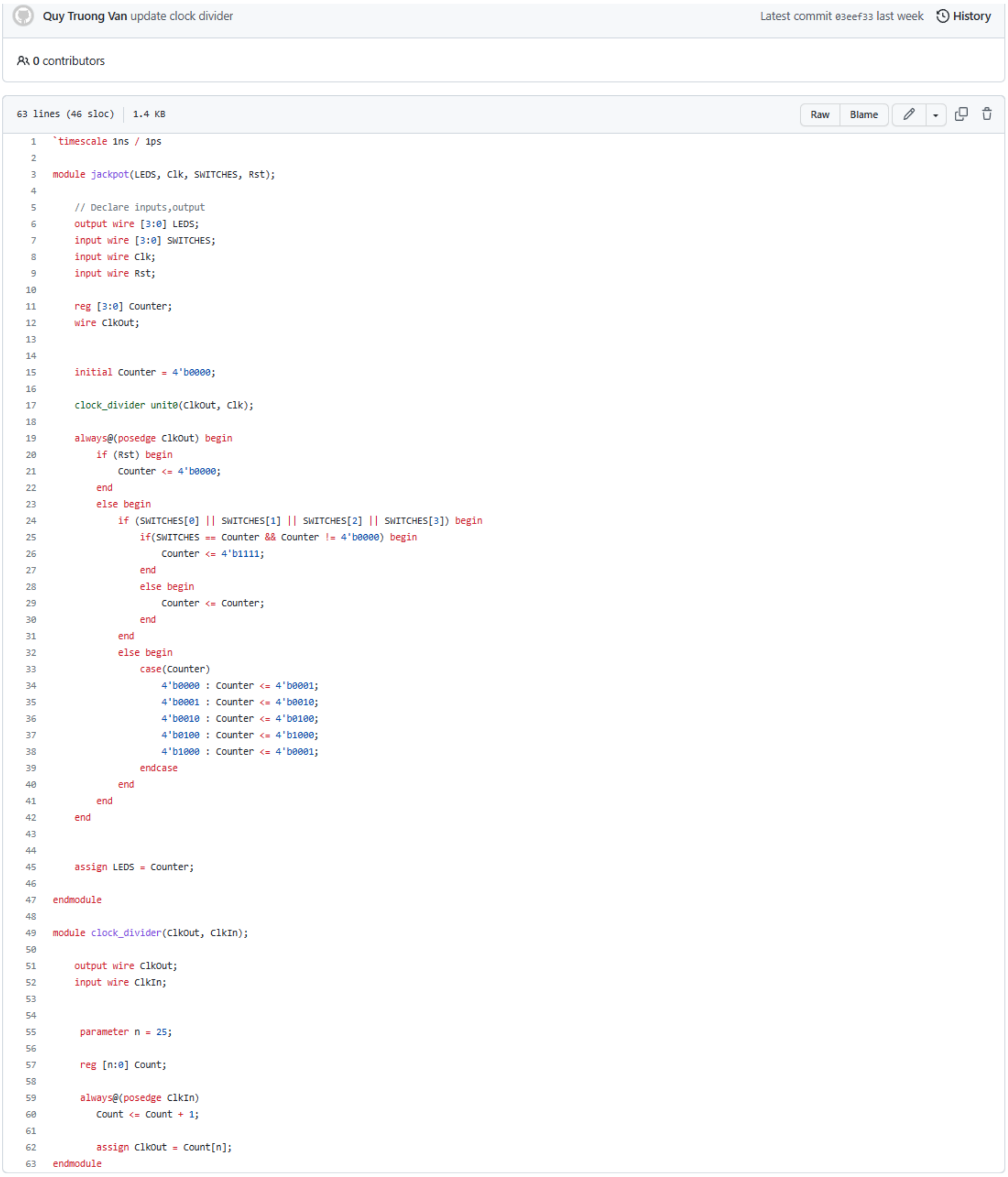
1 contributor

48 lines (32 sloc) | 872 Bytes

```verilog
`timescale 1ns / 1ps

module four_bit_counter(LEDS, Clk, BTN);

    // Declare inputs,output
    output wire [3:0] LEDS;
    input wire [1:0] BTN;
    input wire Clk;

    reg [3:0] Count;
    wire ClkOut;


    initial Count = 4'b0000;

    clock_divider cd1(ClkOut, Clk);

    always@(posedge ClkOut) begin
        if (BTN[0] && !BTN[1]) begin
            Count <= Count + 1;
        end
        else if (BTN[1] && !BTN[0]) begin
            Count <= Count - 1;
        end
        else begin
            Count <= Count;
        end

    end

    assign LEDS = Count;

endmodule

module clock_divider(ClkOut, ClkIn);

    output wire ClkOut;
    input wire ClkIn;

     parameter n = 26;

     reg [n:0] Count;

     always@(posedge ClkIn)
        Count <= Count + 1;

        assign ClkOut = Count[n];
endmodule
```

Give feedback

Figure 2: 4-bit Counter

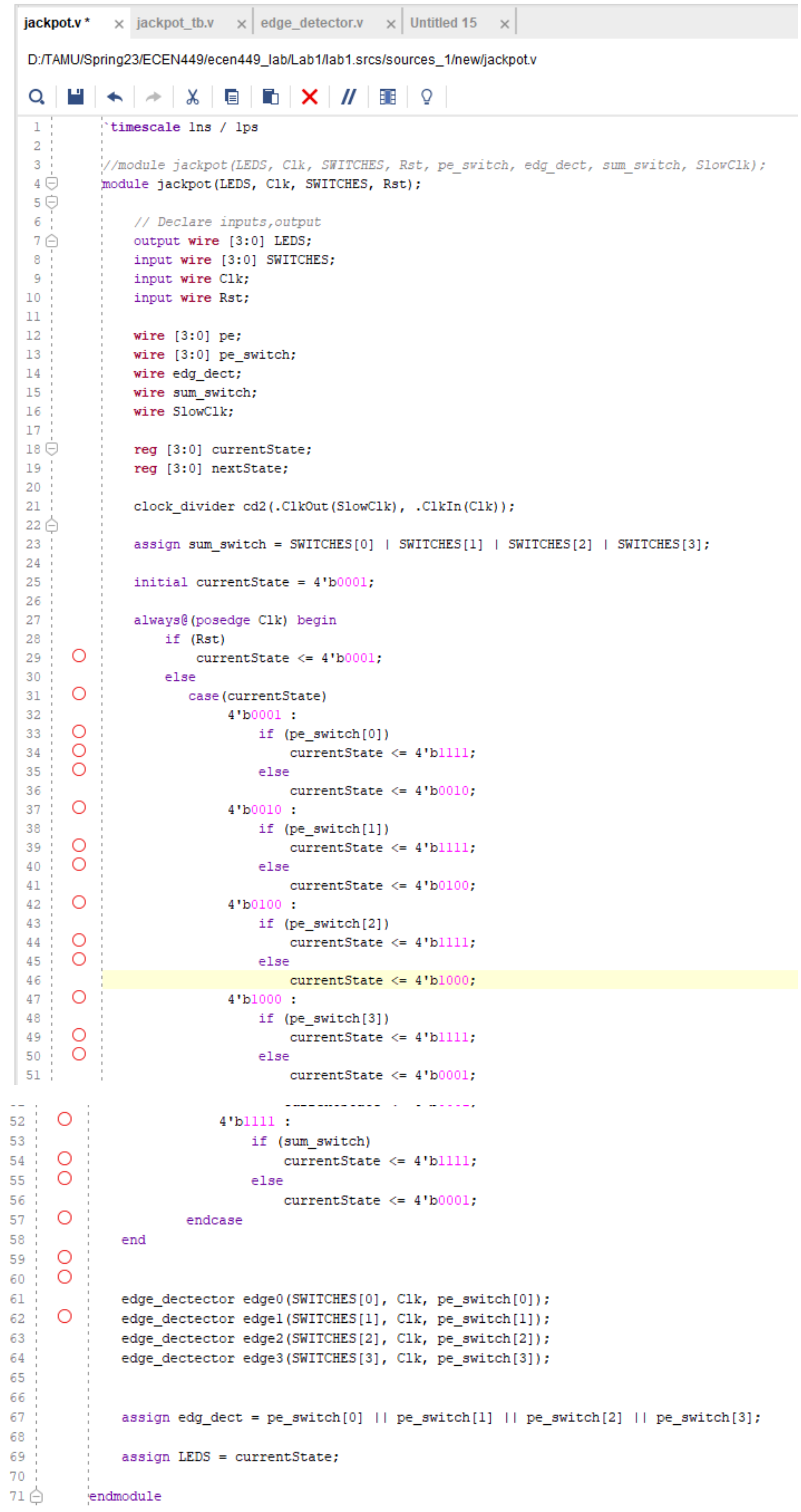

Quy Truong Van update clock divider Latest commit 03eef33 last week History

0 contributors

63 lines (46 sloc) | 1.4 KB

```
`timescale 1ns / 1ps

module jackpot(LEDS, Clk, SWITCHES, Rst);

    // Declare inputs,output
    output wire [3:0] LEDS;
    input wire [3:0] SWITCHES;
    input wire Clk;
    input wire Rst;

    reg [3:0] Counter;
    wire ClkOut;


    initial Counter = 4'b0000;

    clock_divider unit0(ClkOut, Clk);

    always@(posedge ClkOut) begin
        if (Rst) begin
            Counter <= 4'b0000;
        end
        else begin
            if (SWITCHES[0] || SWITCHES[1] || SWITCHES[2] || SWITCHES[3]) begin
                if(SWITCHES == Counter && Counter != 4'b0000) begin
                    Counter <= 4'b1111;
                end
                else begin
                    Counter <= Counter;
                end
            end
            else begin
                case(Counter)
                    4'b0000 : Counter <= 4'b0001;
                    4'b0001 : Counter <= 4'b0010;
                    4'b0010 : Counter <= 4'b0100;
                    4'b0100 : Counter <= 4'b1000;
                    4'b1000 : Counter <= 4'b0001;
                endcase
            end
        end
    end


    assign LEDS = Counter;

endmodule

module clock_divider(ClkOut, ClkIn);

    output wire ClkOut;
    input wire ClkIn;


     parameter n = 25;

     reg [n:0] Count;

     always@(posedge ClkIn)
        Count <= Count + 1;

        assign ClkOut = Count[n];
endmodule
```

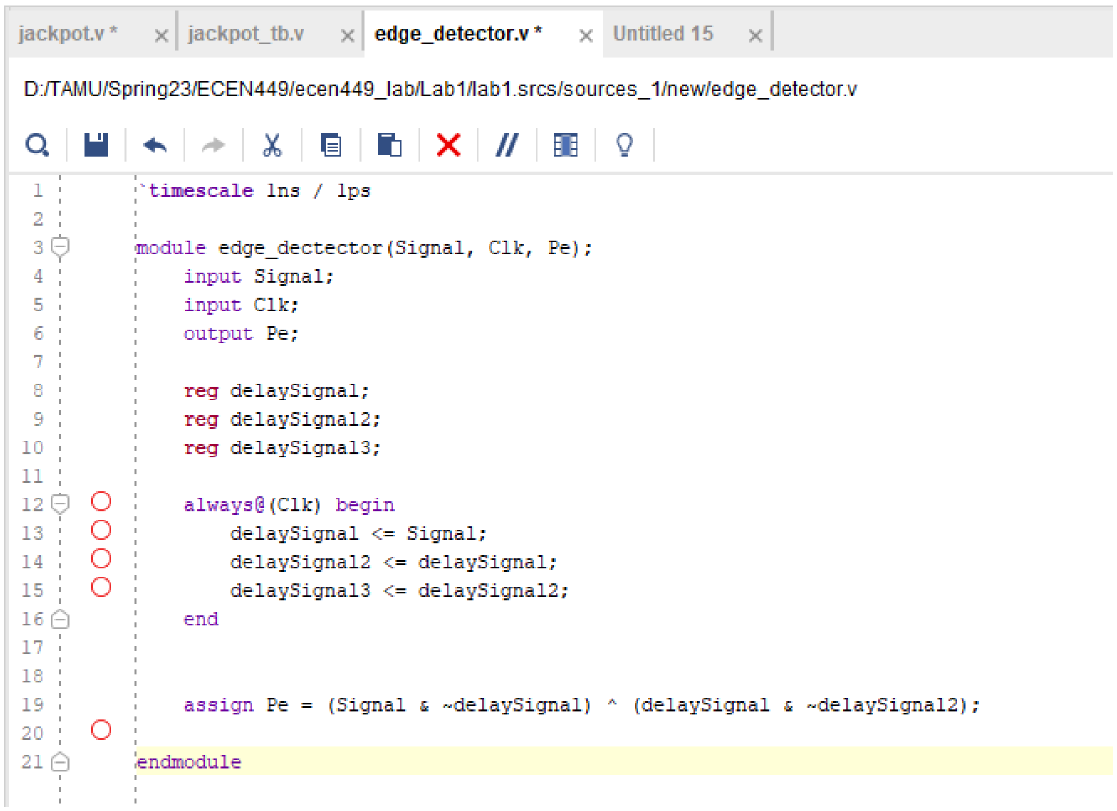
Figure 3: Jackpot first version



D:/TAMU/Spring23/ECEN449/ecen449_lab/Lab1/lab1.srcs/sources_1/new/jackpot.v

```
`timescale 1ns / 1ps

//module jackpot(LEDS, Clk, SWITCHES, Rst, pe_switch, edg_dect, sum_switch, SlowClk);
module jackpot(LEDS, Clk, SWITCHES, Rst);

    // Declare inputs,output
    output wire [3:0] LEDS;
    input wire [3:0] SWITCHES;
    input wire Clk;
    input wire Rst;

    wire [3:0] pe;
    wire [3:0] pe_switch;
    wire edg_dect;
    wire sum_switch;
    wire SlowClk;

    reg [3:0] currentState;
    reg [3:0] nextState;

    clock_divider cd2(.ClkOut(SlowClk), .ClkIn(Clk));

    assign sum_switch = SWITCHES[0] | SWITCHES[1] | SWITCHES[2] | SWITCHES[3];

    initial currentState = 4'b0001;

    always@(posedge Clk) begin
        if (Rst)
            currentState <= 4'b0001;
        else
           case(currentState)
                4'b0001 :
                    if (pe_switch[0])
                        currentState <= 4'b1111;
                    else
                        currentState <= 4'b0010;
                4'b0010 :
                    if (pe_switch[1])
                        currentState <= 4'b1111;
                    else
                        currentState <= 4'b0100;
                4'b0100 :
                    if (pe_switch[2])
                        currentState <= 4'b1111;
                    else
                        currentState <= 4'b1000;
                4'b1000 :
                    if (pe_switch[3])
                        currentState <= 4'b1111;
                    else
                        currentState <= 4'b0001;
                4'b1111 :
                    if (sum_switch)
                        currentState <= 4'b1111;
                    else
                        currentState <= 4'b0001;
            endcase
    end


    edge_dectector edge0(SWITCHES[0], Clk, pe_switch[0]);
    edge_dectector edge1(SWITCHES[1], Clk, pe_switch[1]);
    edge_dectector edge2(SWITCHES[2], Clk, pe_switch[2]);
    edge_dectector edge3(SWITCHES[3], Clk, pe_switch[3]);


    assign edg_dect = pe_switch[0] || pe_switch[1] || pe_switch[2] || pe_switch[3];

    assign LEDS = currentState;

endmodule
```

Figure 4: Upgraded Jackpot

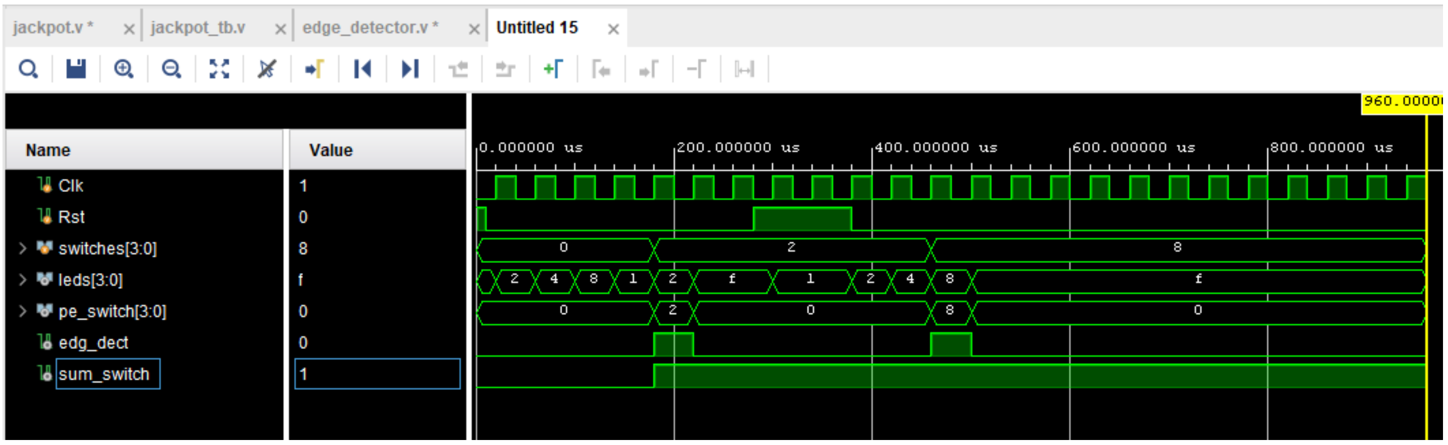


jackpot.v * | jackpot_tb.v | edge_detector.v * | Untitled 15

D:/TAMU/Spring23/ECEN449/ecen449_lab/Lab1/lab1.srcs/sources_1/new/edge_detector.v

```verilog
`timescale 1ns / 1ps

module edge_dectector(Signal, Clk, Pe);
    input Signal;
    input Clk;
    output Pe;

    reg delaySignal;
    reg delaySignal2;
    reg delaySignal3;

    always@(Clk) begin
        delaySignal <= Signal;
        delaySignal2 <= delaySignal;
        delaySignal3 <= delaySignal2;
    end


    assign Pe = (Signal & ~delaySignal) ^ (delaySignal & ~delaySignal2);

endmodule
```

Figure 5: Edge Detector for upgraded Jackpot.



jackpot.v * | jackpot_tb.v | edge_detector.v * | Untitled 15

Name	Value
Clk	1
Rst	0
switches[3:0]	8
leds[3:0]	f
pe_switch[3:0]	0
edg_dect	0
sum_switch	1

Figure 6: Jackpot testbench for upgrade version

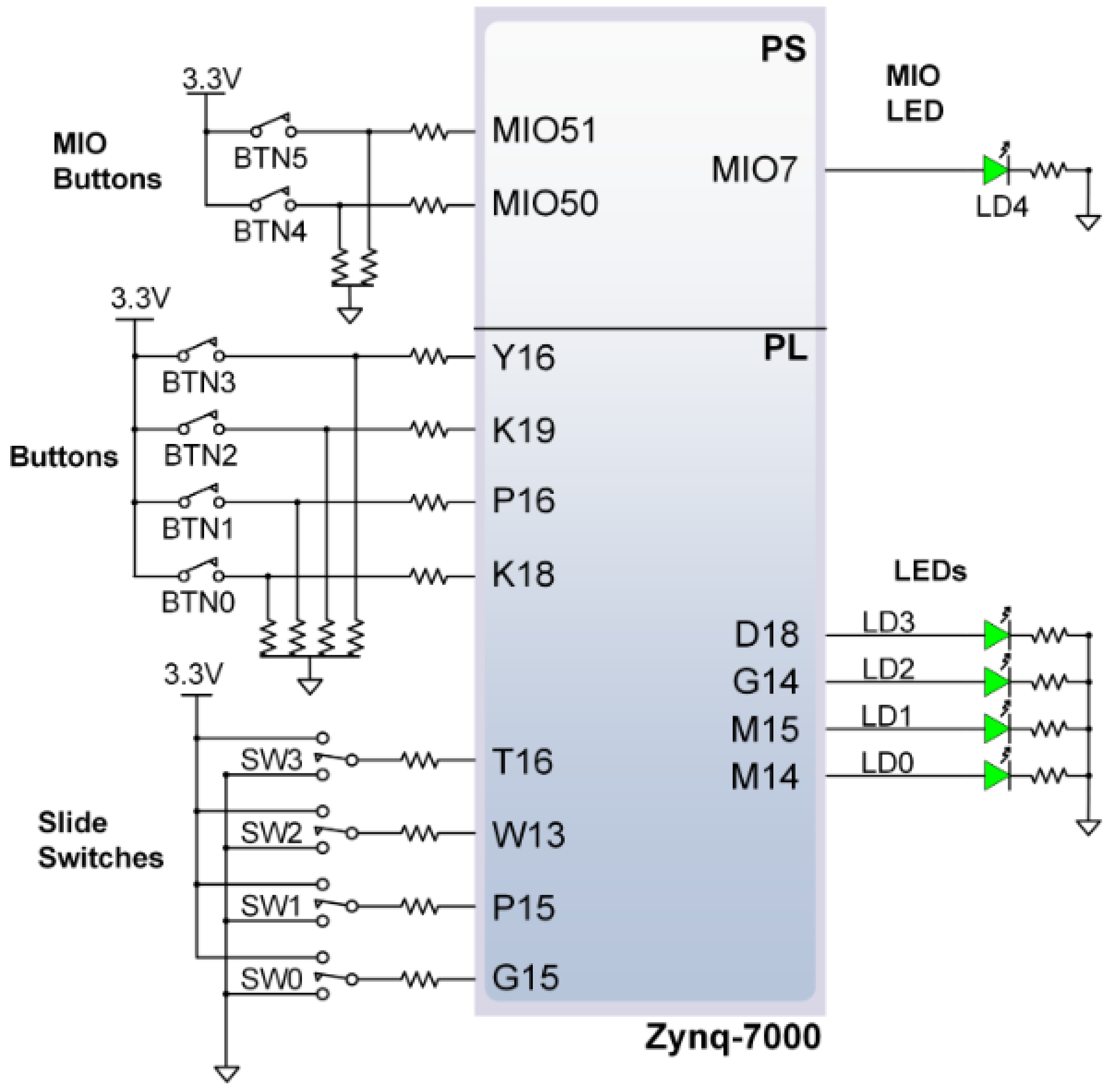


Figure 7: Zybo Z7 Buttons Circuit