Lab 1: Using the Vivado

Alana Ordonez
UIN: 428008825
ECEN 449 506
September 23, 2022

I. Introduction

This lab exercise is for familiarizing ourselves with Xilinx FPGA design flow using Vivado. From creating our first project to writing modules for associations between buttons, LEDs, and switches, we grew accustomed to setting up and running our code. This exercise sets the foundational process for how we start future projects and lab exercises.

II. Procedure

After launching Vivado, creating a subdirectory for this week's lab, and selecting the hardware, a file "switch.v" was created to allow switches to turn their corresponding LED on. A file of design constraints, "switch.xdc, " was created to connect the components (buttons, switches, LEDs, clock, and reset) on the FPGA to the code. After testing that "switch.v" behaved correctly, "4bit-counter.v" was created so that BTN0 increments the count and BTN1 decrements the count, then assigning the LEDS to the counts so that they would turn on per count. This was made using another module called "divider," which is a clock divider to slow down the clock. The count, used as a register, is assigned to the LEDs so the count can correspond to a light.

III. Results

When "switch.v" is run, switches 0 through 3 turn on their corresponding LEDs when toggled. When "4bit-counter.v" is programmed, the LEDs initially count from 0 to 3 then switch direction when BTN1 is pressed, allowing the direction to be shifted depending on which button is pressed. If neither is pressed, the light does not move to another LED. When "jackpot.v" is run, the LEDs count upward from 0 to 3. If a switch is toggled when its corresponding LED is on, then all four LEDs light up, signifying a Jackpot win. If the reset button is pressed, it brings the count back to LED 0 and begins the count again.

When creating these modules, I was having trouble getting Vivado started because it was slowing down the entire computer in Zach (since I cannot run Vivado on my personal laptop). To circumvent this, I had to manually enter new aspects of the module every time I ran the program, meaning that the best chance to finish the lab was to complete it all in one session so I do not lose everything.

IV. Conclusion

This lab really helped in acclimating us to synthesizing and implementing our own modules in Vivado. It aided in understanding exactly how to call the components on the Zybo, as well as mentally work through how to make them behave as desired.

V. Questions

(a) *How are the user push-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 or down)? You will have to consult the Master XDC file for this information.*

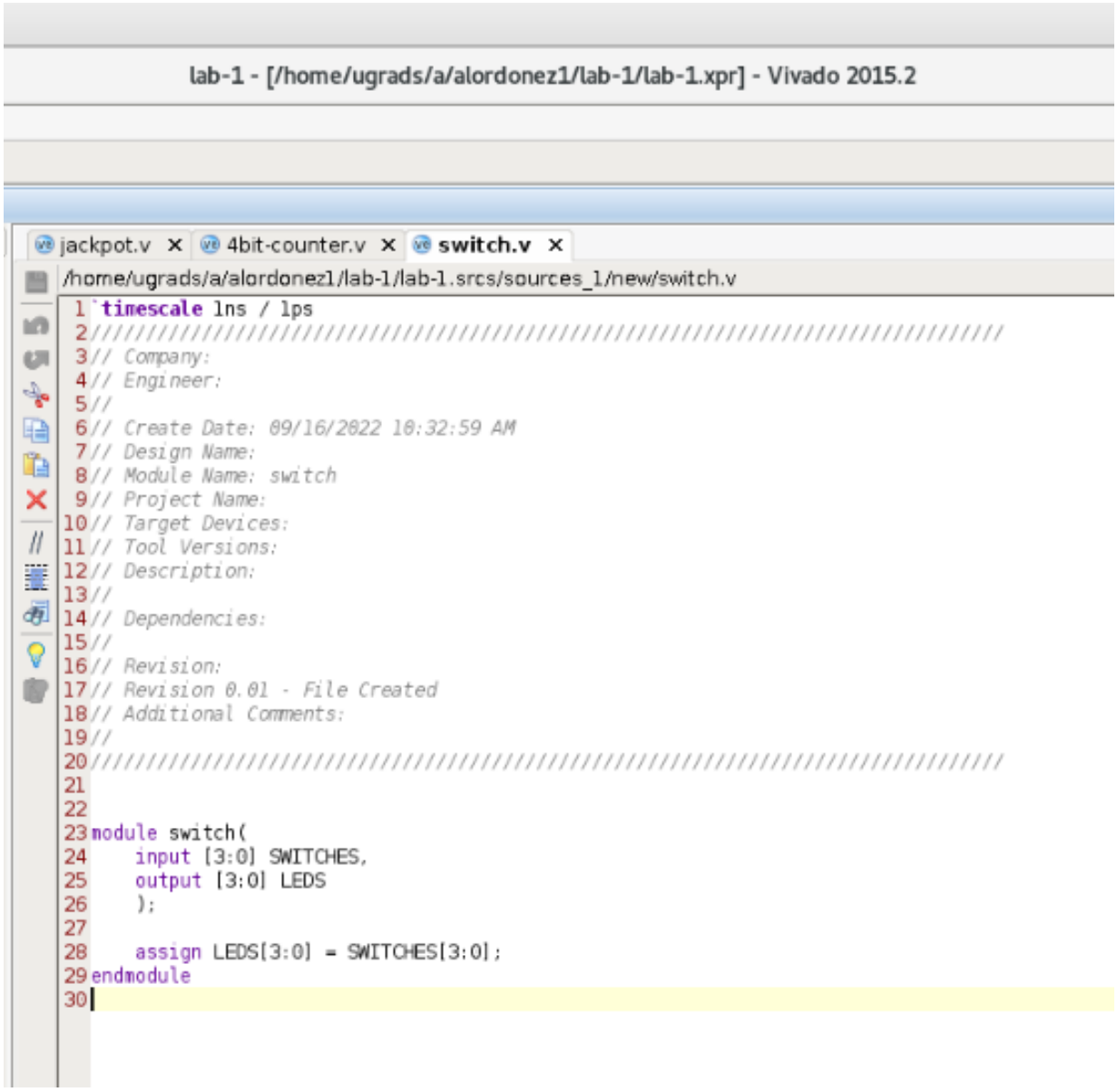
BTN0 is connected to K18, BTN1 is connected to P16, and BTN2 is connected to K19, and they have a pull-down resistor.

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

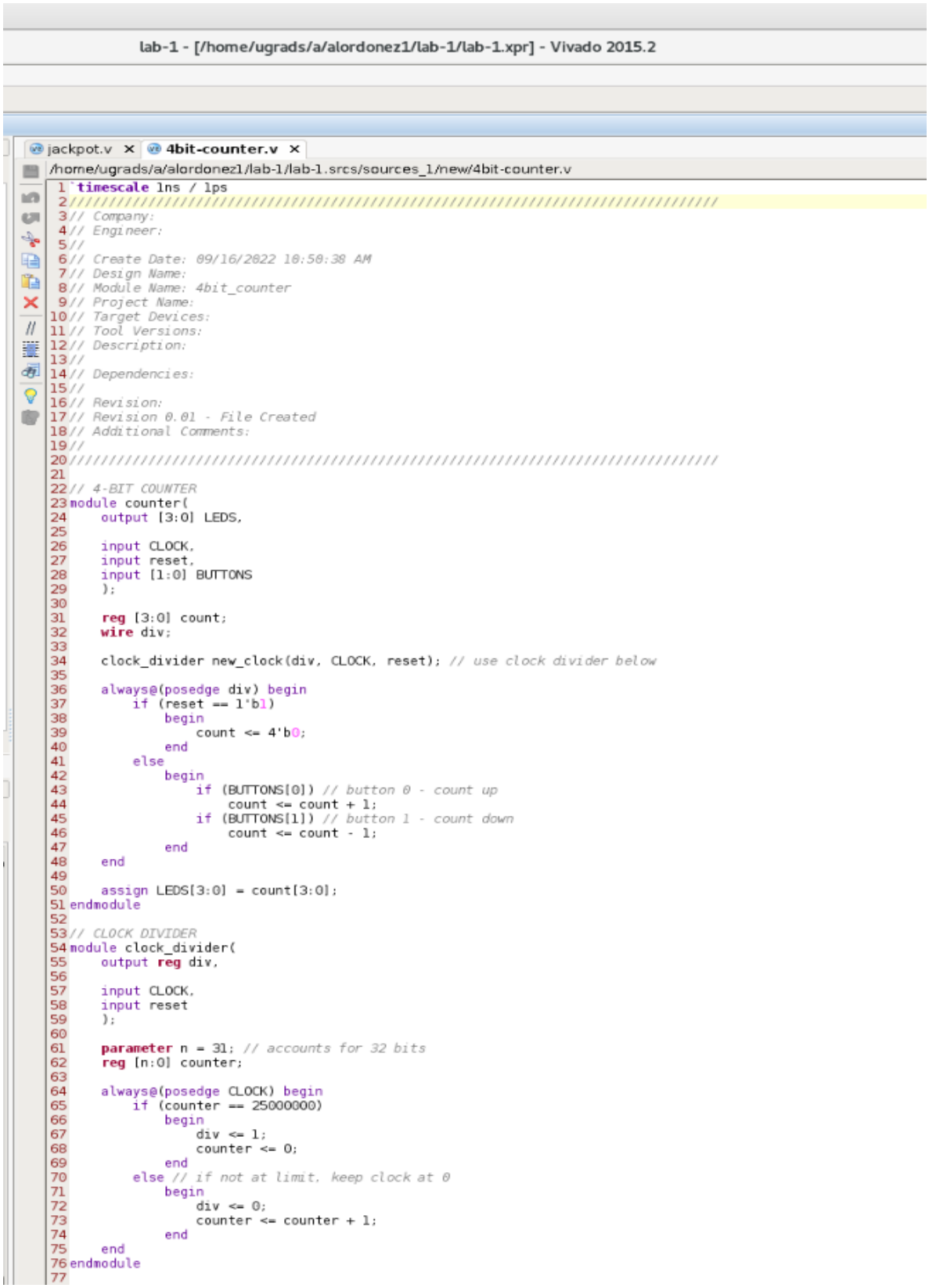
An edge detection circuit specifies exactly when, in relation to the signal, we want an event to occur. In this lab, setting the LEDs to change with every positive edge of the clock is using edge detection.

VI. Appendix

switch.v

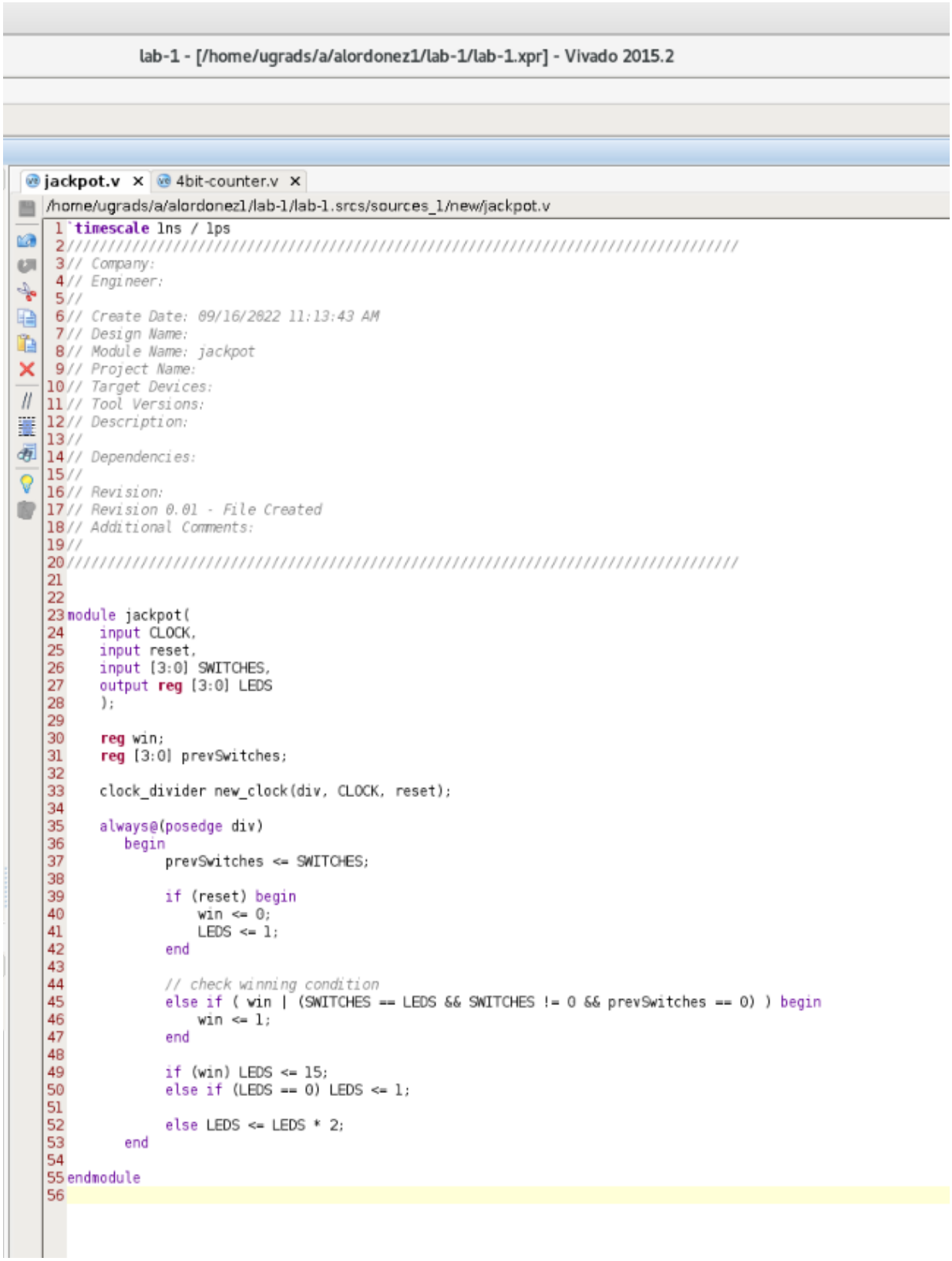


4bit-counter.v



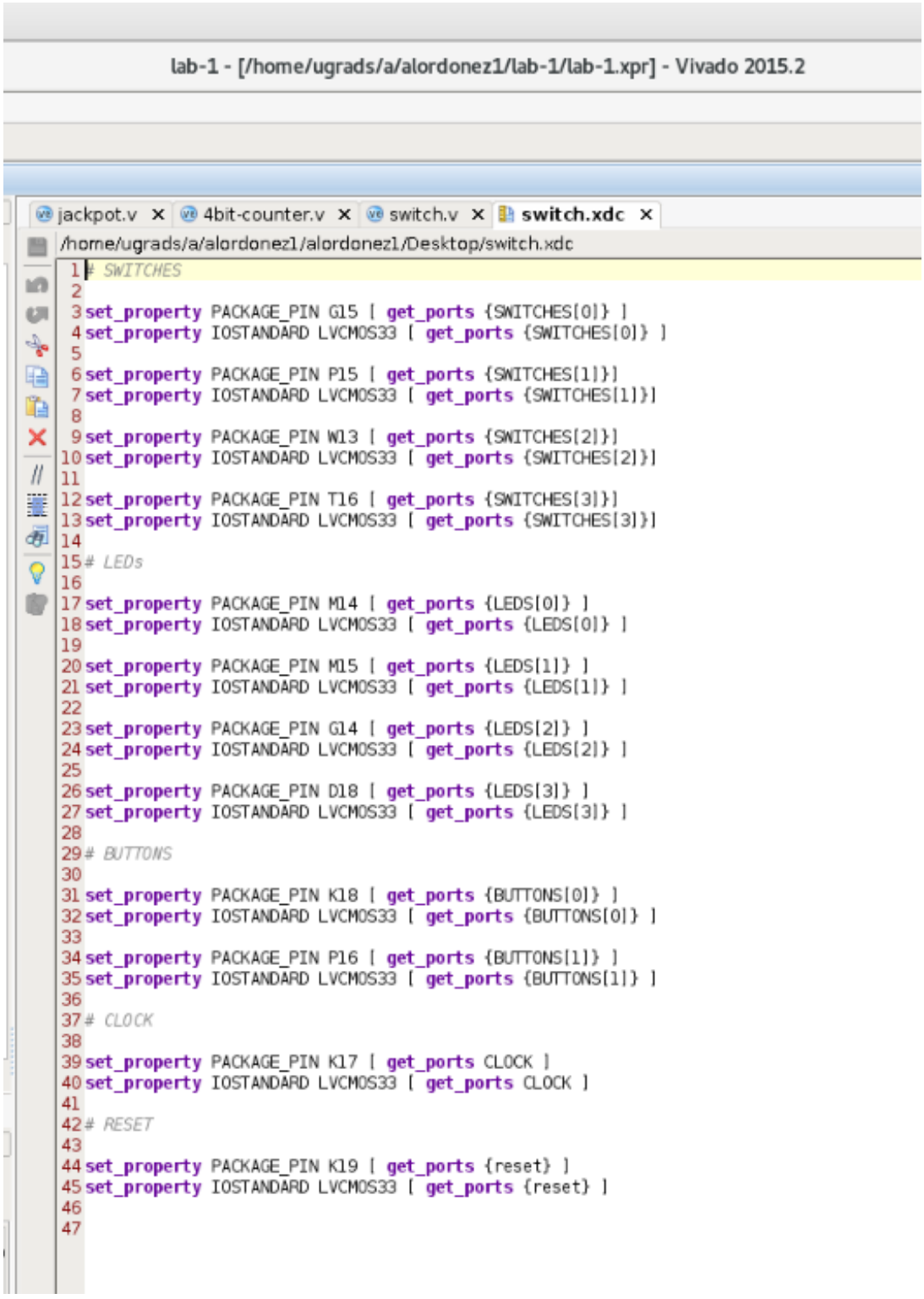

```
1 `timescale 1ns / 1ps
2 //////////////////////////////////////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 09/16/2022 10:50:38 AM
7 // Design Name:
8 // Module Name: 4bit_counter
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21
22 // 4-BIT COUNTER
23 module counter(
24     output [3:0] LEDS,
25
26     input CLOCK,
27     input reset,
28     input [1:0] BUTTONS
29     );
30
31     reg [3:0] count;
32     wire div;
33
34     clock_divider new_clock(div, CLOCK, reset); // use clock divider below
35
36     always@(posedge div) begin
37         if (reset == 1'b1)
38             begin
39                 count <= 4'b0;
40             end
41         else
42             begin
43                 if (BUTTONS[0]) // button 0 - count up
44                     count <= count + 1;
45                 if (BUTTONS[1]) // button 1 - count down
46                     count <= count - 1;
47             end
48     end
49
50     assign LEDS[3:0] = count[3:0];
51 endmodule
52
53 // CLOCK DIVIDER
54 module clock_divider(
55     output reg div,
56
57     input CLOCK,
58     input reset
59     );
60
61     parameter n = 31; // accounts for 32 bits
62     reg [n:0] counter;
63
64     always@(posedge CLOCK) begin
65         if (counter == 25000000)
66             begin
67                 div <= 1;
68                 counter <= 0;
69             end
70         else // if not at limit, keep clock at 0
71             begin
72                 div <= 0;
73                 counter <= counter + 1;
74             end
75     end
76 endmodule
77
```

jackpot.v




```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 09/16/2022 11:13:43 AM
// Design Name:
// Module Name: jackpot
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module jackpot(
    input CLOCK,
    input reset,
    input [3:0] SWITCHES,
    output reg [3:0] LEDS
    );

    reg win;
    reg [3:0] prevSwitches;

    clock_divider new_clock(div, CLOCK, reset);

    always@(posedge div)
       begin
            prevSwitches <= SWITCHES;

            if (reset) begin
                win <= 0;
                LEDS <= 1;
            end

            // check winning condition
            else if ( win | (SWITCHES == LEDS && SWITCHES != 0 && prevSwitches == 0) ) begin
                win <= 1;
            end

            if (win) LEDS <= 15;
            else if (LEDS == 0) LEDS <= 1;

            else LEDS <= LEDS * 2;
       end

endmodule
```

switch.xdc


lab-1 - [/home/ugrads/a/alordonez1/lab-1/lab-1.xpr] - Vivado 2015.2
jackpot.v
4bit-counter.v
switch.v
switch.xdc
/home/ugrads/a/alordonez1/alordonez1/Desktop/switch.xdc
1 # SWITCHES
2
3 set_property PACKAGE_PIN G15 [get_ports {SWITCHES[0]}]
4 set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[0]}]
5
6 set_property PACKAGE_PIN P15 [get_ports {SWITCHES[1]}]
7 set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[1]}]
8
9 set_property PACKAGE_PIN W13 [get_ports {SWITCHES[2]}]
10 set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[2]}]
11
12 set_property PACKAGE_PIN T16 [get_ports {SWITCHES[3]}]
13 set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[3]}]
14
15 # LEDs
16
17 set_property PACKAGE_PIN M14 [get_ports {LEDS[0]}]
18 set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[0]}]
19
20 set_property PACKAGE_PIN M15 [get_ports {LEDS[1]}]
21 set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[1]}]
22
23 set_property PACKAGE_PIN G14 [get_ports {LEDS[2]}]
24 set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[2]}]
25
26 set_property PACKAGE_PIN D18 [get_ports {LEDS[3]}]
27 set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[3]}]
28
29 # BUTTONS
30
31 set_property PACKAGE_PIN K18 [get_ports {BUTTONS[0]}]
32 set_property IOSTANDARD LVCMOS33 [get_ports {BUTTONS[0]}]
33
34 set_property PACKAGE_PIN P16 [get_ports {BUTTONS[1]}]
35 set_property IOSTANDARD LVCMOS33 [get_ports {BUTTONS[1]}]
36
37 # CLOCK
38
39 set_property PACKAGE_PIN K17 [get_ports CLOCK]
40 set_property IOSTANDARD LVCMOS33 [get_ports CLOCK]
41
42 # RESET
43
44 set_property PACKAGE_PIN K19 [get_ports {reset}]
45 set_property IOSTANDARD LVCMOS33 [get_ports {reset}]
46
47