Laboratory Exercise #1
Using Vivado

Name: Daniel Horan
UIN: 527005307
ECEN 449-504

I. Introduction
In this lab, we'll learn how to use the Vivado software to design for the Xilinx FPGA. Our main task is to set up the ZYBO Z7-10 board so that its LEDs light up based on the settings of its DIP switches. We'll use Verilog in Vivado for this. After we get this working, we'll move on to creating a simple counter and a jackpot game using what we've learned.

II. Procedure
The lab consisted of three distinct parts. Initially, we followed specific instructions to establish a project. Subsequently, we created two files: switch.v and switch.xdc. After reprogramming the Zybo Z7-10, we tested the functionality of the switches.

We wrote a 4-bit counter using the ZYBO Z7-10 board's LEDs, updating approximately every second. BTN0 increased the count, while BTN1 decreased it; without pressing, the count remained unchanged. We added clock and reset pins to our Verilog module and checked the ZYBO Z7-10 manual for pin assignments. After modifying the XDC file, we added text for Vivado's timing constraints. Considering the 125MHz onboard clock, we divided the clock for discernible LED changes.

We wrote a "Jackpot" game where LEDs on the ZYBO Z7-10 board light up sequentially. The goal is to activate the DIP switch corresponding to the currently lit LED. A correct switch results in all LEDs lighting up, signaling a win.

III. Results
The ZYBO Z7-10 board's LEDs lit up as expected based on the DIP switch settings, thanks to the Verilog code in switch.v and the pin mappings in switch.xdc.

The 4-bit counter, outlined in BitCounter.v, mostly worked well. However, there was a hiccup: BTN0 occasionally needed 2-3 presses to increment the count, while BTN1 worked smoothly for decrementing. The counter updated roughly every second, making the LED changes easily noticeable.

The "Jackpot" game, derived from jackpot.v, presented an engaging task. LEDs sequentially illuminated, and the correct DIP switch would light them all, signaling victory. However, a flaw was noted: if the wrong switch was hit, the LED would eventually stop and falsely indicate a win. Despite this, the game's rhythm allowed players to easily spot the active LED.

Both the counter and the game relied on their respective XDC files for pin assignments and timing, ensuring smooth interaction with the ZYBO board.

IV. Conclusion
This lab immersed us in FPGA design using the ZYBO Z7-10 board, Vivado, and Verilog. While the switch functioned and the "Jackpot" game mostly delivered its promise, the BTN0 issue in the counter and the game's flaw highlighted the intricacies of hardware design. This practical venture emphasized the importance of accurate pin assignments and the excitement of transforming basic logic into tangible applications. It was a profound insight into the potential and hurdles of microprocessor design!

V. Questions

a) BTN0 is wired to pin K18.
BTN1 is wired to pin P16.
The RESET button is wired to pin K19.

b) The purpose of an edge detection circuit is to identify transitions in a signal, specifically from a low to high (rising edge) or high to low (falling edge). In the context of this lab, an edge detection circuit could have been beneficial for the "Jackpot" game. Specifically, it would ensure that the LEDs only respond to a single press of a switch, preventing the LED from halting and falsely indicating a win when the wrong switch is pressed. By detecting the precise moment a switch is toggled, the game's logic could be refined to function more accurately and consistently.

VI. Appendix

switch.v:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Daniel Horan
//
// Create Date: 09/04/2023 10:31:51 AM
 // Design Name:
// Module Name: switch
 // Project Name: Lab 1
 // Target Devices:
// Tool Versions:
 // Description:
//
// Dependencies:
//
// Revision:
 // Revision 0.01 - File Created
 // Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module switch(SWITCHES,LEDS);
        input [3:0] SWITCHES;
        output [3:0] LEDS;

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

switch.xdc:

```
##Switches

set_property PACKAGE_PIN G15 [get_ports {SWITCHES[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[0]}]

set_property PACKAGE_PIN P15 [get_ports {SWITCHES[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[1]}]

set_property PACKAGE_PIN W13 [get_ports {SWITCHES[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[2]}]

set_property PACKAGE_PIN T16 [get_ports {SWITCHES[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[3]}]

##LEDs

set_property PACKAGE_PIN M14 [get_ports {LEDS[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[0]}]

set_property PACKAGE_PIN M15 [get_ports {LEDS[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[1]}]

set_property PACKAGE_PIN G14 [get_ports {LEDS[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[2]}]

set_property PACKAGE_PIN D18 [get_ports {LEDS[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[3]}]
```

BitCounter.v:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Daniel Horan
//
// Create Date: 09/07/2023 12:30:06 PM
// Design Name:
// Module Name: BitCounter
// Project Name: Lab 1
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
```

```
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module BitCounter(
    input CLOCK,
    input RESET,
    input BTN0,
    input BTN1,
    output [3:0] LED
    );
    // Internal signals
    wire newClock;        // Slowed down clock signal
    reg [3:0] count;      // 4-bit register to store the count value

    // Instantiate the clock divider module
    divider newclock_inst(
        .out_clk(newClock),   // Output of the slowed down clock
        .clock(CLOCK),        // Input clock signal
        .reset(RESET)         // Input reset signal
    );

    // Logic to update the counter value
    always@(posedge newClock) begin
        if(RESET) begin       // If reset is pressed
            count <= 0;       // Reset the counter to 0
        end
        else begin
            if(BTN0)          // If button 0 is pressed
                count <= count + 1;  // Increment the counter
            if(BTN1)          // If button 1 is pressed
                count <= count - 1;  // Decrement the counter
        end
    end

    // Assign the counter value to the LEDs
    assign LED[3:0] = count[3:0];

endmodule

// Module to divide the clock frequency
```

```
module divider(
   output reg out_clk,   // Output of the slowed down clock
   input clock,          // Input clock signal
   input reset           // Input reset signal
);

   parameter n = 31;             // Parameter to define the size of the counter
   reg[n:0] counter;             // n-bit register to store the counter value

   // Logic to divide the clock frequency
   always@(posedge clock) begin
      if(counter == 25000000) begin  // If counter reaches 25,000,000
         out_clk <= 1;              // Set the output clock high
         counter <= 0;              // Reset the counter
      end
      else begin
         out_clk <= 0;              // Set the output clock low
         counter <= counter + 1;    // Increment the counter
      end
   end

endmodule
```

BitCounter.xdc:

```
##LEDs (Already defined, kept as is)
set_property PACKAGE_PIN M14 [get_ports {LED[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[0]}]
set_property PACKAGE_PIN M15 [get_ports {LED[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[1]}]
set_property PACKAGE_PIN G14 [get_ports {LED[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[2]}]
set_property PACKAGE_PIN D18 [get_ports {LED[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[3]}]

##Clock (Added)
set_property PACKAGE_PIN K17 [get_ports CLOCK]
set_property IOSTANDARD LVCMOS33 [get_ports CLOCK]

##Buttons (Added)
set_property PACKAGE_PIN K18 [get_ports BTN0]
set_property IOSTANDARD LVCMOS33 [get_ports BTN0]
set_property PACKAGE_PIN P16 [get_ports BTN1]
set_property IOSTANDARD LVCMOS33 [get_ports BTN1]
set_property PACKAGE_PIN K19 [get_ports RESET]
```

set_property IOSTANDARD LVCMOS33 [get_ports RESET]

jackpot.v:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Daniel Horan
//
// Create Date: 09/07/2023 06:06:19 PM
// Design Name:
// Module Name: jackpot
// Project Name: Lab 1
// 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 [3:0] LEDS
    );
    // Slowed down version of the main CLOCK
    wire NEWCLOCK;

    // Initial state of the LEDS
    reg [3:0] lights = 4'b1000;

    // Clock divider logic
    divider clock_divider(NEWCLOCK, CLOCK, RESET);

    // Logic to control the state of the LEDs based on SWITCHES and RESET
always @(posedge NEWCLOCK) begin
    if (RESET) begin
        lights <= 4'b1000;
```

```
    end else begin
        case (lights)
            4'b1000:
                if (~SWITCHES[3]) lights <= 4'b0100;
                else lights <= 4'b1111; // All LEDs ON if SWITCHES[3] is pressed
            4'b0100:
                if (~SWITCHES[2]) lights <= 4'b0010;
                else lights <= 4'b1111; // All LEDs ON if SWITCHES[2] is pressed
            4'b0010:
                if (~SWITCHES[1]) lights <= 4'b0001;
                else lights <= 4'b1111; // All LEDs ON if SWITCHES[1] is pressed
            4'b0001:
                if (~SWITCHES[0]) lights <= 4'b1000;
                else lights <= 4'b1111; // All LEDs ON if SWITCHES[0] is pressed
            default: lights <= 4'b1111;
        endcase
    end
end

    // Assign the 'lights' state to the LEDS output
    assign LEDS = lights;

endmodule
module divider(
    output reg out_clk,   // Output of the slowed down clock
    input clock,          // Input clock signal
    input reset           // Input reset signal
);

    parameter n = 31;             // Parameter to define the size of the counter
    reg[n:0] counter;             // n-bit register to store the counter value

    // Logic to divide the clock frequency
    always@(posedge clock) begin
        if(counter == 25000000) begin  // If counter reaches 25,000,000
            out_clk <= 1;              // Set the output clock high
            counter <= 0;              // Reset the counter
        end
        else begin
            out_clk <= 0;              // Set the output clock low
            counter <= counter + 1;    // Increment the counter
        end
    end
```

endmodule

jackpot.xdc:

```
##Switches
set_property PACKAGE_PIN G15 [get_ports {SWITCHES[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[0]}]
set_property PACKAGE_PIN P15 [get_ports {SWITCHES[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[1]}]
set_property PACKAGE_PIN W13 [get_ports {SWITCHES[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[2]}]
set_property PACKAGE_PIN T16 [get_ports {SWITCHES[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[3]}]

##LEDs (Already defined, kept as is)
set_property PACKAGE_PIN M14 [get_ports {LEDS[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[0]}]
set_property PACKAGE_PIN M15 [get_ports {LEDS[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[1]}]
set_property PACKAGE_PIN G14 [get_ports {LEDS[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[2]}]
set_property PACKAGE_PIN D18 [get_ports {LEDS[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[3]}]

##Clock (Added)
set_property PACKAGE_PIN K17 [get_ports CLOCK]
set_property IOSTANDARD LVCMOS33 [get_ports CLOCK]

##Buttons (Added)
set_property PACKAGE_PIN K19 [get_ports RESET]
set_property IOSTANDARD LVCMOS33 [get_ports RESET]
```