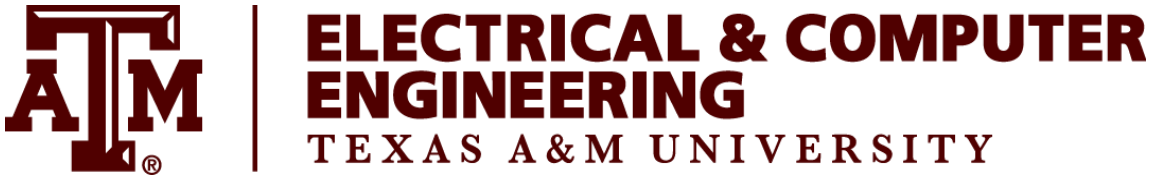

ECEN 449: Microprocessor System Design

Lab 1: Using Vivado

Kylan Lewis

UIN: 719001131

ECEN 449 -504

TA: Ashwin Ashokan

Date: 9/18/2020

Introduction:

The purpose of this lab is to learn and become familiar with the Xilinx FPGA via Vivado. This is so that we can run and synthesize Verilog code on the ZYBO Z7-10 board. I used Vivado to create two simple modules that control the onboard LEDs. The first module is a simple counter that increments or decrements every clock cycle while a button is pressed. Using the knowledge from the counter I then created a jackpot game. This module gives a jackpot in the event that a button press and the corresponding LED occur on the same. See more below.

Procedure:

Part 1:

1. Follow the lab manual instruction to compile pre-written source code and constraint files in Vivado.

2. Once the compilation is successful, generate a bitstream and program the target ZYBO Board in the Hardware Target Manager .

2. To make sure the board is programmed properly, toggle the switches and observe LEDs 0:3 .

Part 2:

1. Create a clock divider since the normal FGBA clock cycle is too fast and it needs to be slower. (Approx. 1 Hz)

2. Using the clock divider, create a 4-bit counter that counts up when button[0] gets pressed, counts down when button[1] is pressed, and resets the count when reset is pressed.

3. Changed the constraint file to include the reset button and button 1 and 2.

4. Generate bitstream and program the ZYBO board.

Part 3:

1. Import the same clock divider from Part 2.

2. Write a module for the jackpot that makes the LEDs from LED0 – LED3 flash one at a time, in order, each rising edge. When you flip the switch [0:3] corresponding to the LED on the same rising edge, you win the “jackpot”. (all LEDS will light up when won)

3. Changed constraint file so it included switches.

4. Generate bitstream and program the ZYBO board.

Results:

Part 1:

1. The LEDs and switches had the same value so whenever you flipped a switch the respective LED turned on.

Part 2:

2. I created a counter that assigns LEDs from 0-3 to the corresponding count value. When BUTTON0 is held down count decreases by 1 bit and when BUTTON1 is held, the count increases by 1 bit. The RESET button returned the count to 0.

Part 3:

3. I created a jackpot game that flashes LEDs 0-3 in order and if a switch is toggled while a LED is flashing, you win the "jackpot" (all LEDS will light up when won).

Post-Lab Questions :

1. 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)?

I used the push buttons on part 2 and 3. BUTTON1 was mapped to P16, BUTTON0 was mapped K18, and the RESET button was mapped to Y16. When going through the reference manual, it shows that the buttons are 3.3V when pushed.

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

Edge detection is used for always @ triggers. In my counter and jackpot, it was used on the positive edge of the reset button and switched and also in the output of the clock divider. Inside the clock divider the edge detection is used for the original clock.

Code:

```
`timescale 1ns / 1ps

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

endmodule
```

```
##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]}]
```

```
module divide_clock(output reg OutClk, input CLK, input RST);

   reg[31:0] count;
      always@(posedge CLK) begin
  if (RST == 1) begin
  count <= 0;
  end
      if(count == 150000000) begin   //divides clock to 1Hz
                  OutClk <= 1;              //when the count reaches 150,000,000  ,  OutClk becomes 1
         count <= 0;
      end
      else begin
         OutClk <= 0;
         count <= count + 1;
      end
      end
endmodule
```

```
`timescale 1ns / 1ps

module counter(output [3:0] LEDS, input CLK, input RST, input [0:1] BUTTON);

   wire newClock;
   reg [3:0] count;

   divide_clock newclock(newClock, CLK, RST); //declare clock divider


   always@(posedge newClock) begin
      if(RST) begin
            count <= 0;        //if RST is held, count restarts at zero.
      end

      else begin

         if(BUTTON[1])           //if BTN1 is held, count increases
            count <= count + 1;
         if(BUTTON[0])                        //if BTN0 is held, count decreases
            count <= count - 1;
      end

   end

   assign LEDS[3:0] = count[3:0]; //LEDS will display current count
endmodule
```

```
## 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]}]

set_property PACKAGE_PIN K17 [get_ports {CLK}]
set_property IOSTANDARD LVCMOS33 [get_ports {CLK}]


##BUTTONS
set_property PACKAGE_PIN K18 [get_ports {BUTTON[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {BUTTON[0]}]

set_property PACKAGE_PIN P16 [get_ports {BUTTON[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {BUTTON[1]}]

set_property PACKAGE_PIN Y16 [get_ports {RST}]
set_property IOSTANDARD LVCMOS33 [get_ports {RST}]
```

```
`timescale 1ns / 1ps

module jackpot(output [3:0] LEDS, input [3:0] SWITCHES, input CLK, input RST);

   wire newClock;
   reg [3:0] count;
   reg [3:0] switching; //extra variable to help with win condition logic

   divide_clock newclock(newClock, CLK, RST);  //declare new clock

   always@(posedge newClock or posedge RST or posedge SWITCHES[3:0]) begin
      if(RST) begin
            count <= 4'b0001;   //if RST button is pressed, first state happens.
            switching <= 4'b0000;
      end

      else begin  //The LEDS cycle around, changing every second.

                              if(count == 4'b0001) begin           //current LED
```

```
                        switching[3:0] <= SWITCHES[3:0];
                                count <= 4'b0010;               //next LED
                        end

                        else if(count == 4'b0010) begin
                           switching[3:0] <= SWITCHES[3:0];
                                count <= 4'b0100;
                        end

        else if(count == 4'b0100) begin


         switching[3:0] <= SWITCHES[3:0];
                                count <= 4'b1000 ;
                        end

                        else if(count == 4'b1000) begin
                                switching[3:0] <= SWITCHES[3:0];
                                count <= 4'b0001 ;
                        end

if(count[3:0] == SWITCHES[3:0] && switching[3:0] != SWITCHES[3:0]) begin // win condition:
                        count[3:0] <= 4'b1111;    //lights up every LED if win condition is achieved.

          end
      end
   end

      assign LEDS[3:0] = count[3:0]; // count = LEDS

endmodule
```

```
## 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]}]

## Clock


set_property PACKAGE_PIN K17 [get_ports {CLK}]
set_property IOSTANDARD LVCMOS33 [get_ports {CLK}]


## BUTTONS

set_property PACKAGE_PIN Y16 [get_ports {RST}]
set_property IOSTANDARD LVCMOS33 [get_ports {RST}]
```