ECEN 749- Microprocessor System Design

Section 603

TA: Mr. Kunal Bharathi

LAB 1

Using the Vivado

Date of Performance: 09/05/2019

Student Name: Dhiraj Dinesh Kudva

UIN: 829009538

Introduction

This lab is an introduction to Xilinx Vivado Software. This lab is focussed more on getting familiar with the Xilinx Vivado software and ZYBO Z7-10 board. In addition to this, it also provides an experience in Verilog programming. This lab is important as it provides a practical experience with the ZYBO Z7-10 board and Verilog programming. As a part of this lab, we will be performing three activities:

1. Assigning switches to LED
2. Up Down Counter
3. Jackpot Game.

1. Assigning switches to LED.

In this experiment, the switches on ZYBO Z7-10 board are linked with the LEDS in such a way that whenever a particular switch is switched on, that corresponding LED will glow.

Procedure:

1. Launch Vivado and open new project file.
2. Create new RTL project and skip for 'Add Existing Window' and 'Add Constraints'. The constraints would be added later as a separate file.
3. Then select the Zybo board from the boards tab.
   Set the device properties to the following:
   Device Family: Zynq-7000
   Sub-Family: Zynq-7000
   Package: clg400
   Speed Grade: -1
   And then select the first one among the two devices.
4. Connect the ZYBO board to the PC and switch on. Make sure it is in JTAG mode.
5. Then define inputs and outputs in the Define Tab based on the code requirements. Select okay.
6. The editor for writing the source code would open.
7. The constraints file can be written in a normal text editor like 'Notepad' and then save it with extension of " .xdc".
8. Right click on the constraints tab in the Vivado tool and add the created constraints file.
9. Write the source code in Verilog and generate the bitstream by clicking on the 'GENERATE BITSTREAM' button. Then check if the code is without any errors. If any errors are present, then it should be resolved first.
10. After bitstream generation, select the FPGA 'xc7z010 1 form the Hardware Manager and upload the code on board.
11. For the source code of assigning the switch to the LED, first check the terminals of the SWITCH and LEDS.
12. Mention these terminals in the module using assign statement. Refer the Verilog code and constraints file written below.

Verilog Code for assigning switches to LED.

```
      `timescale 1ns / 1ps

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

endmodule
```

Constraints file.

```
## 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]}]
```

2. Up Down Counter

The purpose of this experiment is to generate an Up Down Counter. The pushbutton 0 will acts as Up counter button and pushbutton 1 will act as down counter. As we press the button, based on whether it is up or down button, the LEDS will provide the output in the increasing or decreasing order.

Procedure:

1. Follow the steps from 1 to 10.
2. After selecting the device, the major challenge is to reduce the clock frequency. TheZYBO Z7-10 board has an internal clock of 125 MHz. At this rate, the change in output LEDs would not be visible.
3. So in order to see the output, we need to divide the clock frequency. This is done by using a counter and a new clkout parameter. The clkout will toggle at the positive edge of the internal clock only when the counter reaches the value of 124999999 (corresponding to 25 Mhz). This will give us a new clock of 1 Hz. At this rate, the output would be easily visible.
4. After this, select the up and down pushbuttons and cause the output to change as per the table below. The 4 LEDs will represent the decimal number from 0 to 15

Table

LEDS 3	LEDS 2	LEDS 1	LEDS 0	Decimal Count
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Verilog code

```
//input and output declaration
module lab1_1(LEDS, SWITCHES,CLK,RESET);
 input [1:0] SWITCHES;
        output [3:0] LEDS;
     //declaring output as register
       reg [3:0] LEDS;
        input CLK;
        input RESET;
   //clkout is the reduced frequency clock cycle and counter is used
//to reduce the clock frequency

        reg clkout;
       reg [27:0] counter;
   //block for reducing the clock frequency to 1GHz
    always@(posedge CLK)
       begin

                    counter <= counter + 1;
                    if ( counter == 125_000_000)
                        begin
                            counter <= 0;
                            clkout <= ~clkout;
//clkout will toggle only after reaching the 1250000000( indicating
//125 Mhz) thus resulting in a clock of 1GHz
                        end
               end
 //counter logic will trigger only at positive edge of clkout
 always @ (posedge clkout )
 begin

        if (RESET ==1)
        LEDS[3:0]=4'b0000; //resetting the LEDS to zero
        if (SWITCHES[0]==1)
        LEDS[3:0]= LEDS[3:0] + 1'b1;
//when switch 0 which is up counter switch is pressed, the output is
//incremented by 1
        if (SWITCHES[1]==1)
        LEDS[3:0]= LEDS[3:0] - 1'b1;
//when switch 1 which is down counter switch is pressed, the output
//is decremented by 1

 end
endmodule
```

Constraints.

```
## Switches

set_property PACKAGE_PIN K18 [get_ports {SWITCHES[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[0]}]

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



## 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}]

## RESET
set_property PACKAGE_PIN K19 [get_ports {RESET}]
set_property IOSTANDARD LVCMOS33 [get_ports {RESET}]
```

3. Jackpot Game.

In this experiment, the LEDS would be blinking continuously. In order to win the game, you must press the correct switch corresponding to the LED when it is glowing. If this is done, all the LEDS will be glowing at the same time indicating that you have got the jackpot.

Procedure:

1. Follow the steps from 1 to 10 of assigning switch to LED statement.
2. The internal clock must be divided in such a way that LEDs should blink one after another and only a single LED must be blinking at a time.
3. Whenever a switch is selected if that particular LED is blinking, then the jackpot condition is met. The below written Verilog code meets these two conditions.
4. If a wrong switch is switched ON, then the blinking stops and then the reset button must be pressed to restart it.

Verilog Code

```
//initialization of module
module jackpot(LEDS,SWITCHES,CLOCK,RESET);
 input [3:0] SWITCHES;
        output [3:0] LEDS;
        input CLOCK ;
        input RESET;

 reg [3:0] LEDS;

 reg [27:0] counter;//counter for reducing the internal clock
 reg clkout;//the reduceed clock frequency


//block for reducing the clock frequency
 always @(posedge CLOCK)
 begin
        counter <= counter + 1;
        if ( counter == 125_000_00)      //the number is choosen so
that LEDs blink at 10 Hz
              begin
                    counter <= 0;
                    clkout <= ~clkout;//clkout will toggle only
after reaching condition value thus resulting in a clock of 10Hz
              end
 end

//blinking and jackpot block
always @(posedge clkout)
begin
// the blinking of led will continue until a switch is pressed
if (SWITCHES ==4'b0000)
begin
```

```
if (LEDS==4'b0000)
begin
       LEDS<=4'b0001;   //initializing LED
       end
 else
 begin
    LEDS<=LEDS *2; //blinking the LED by shifting the bits towards
right//this is done by mutiplying by 2.
 end
 end
 //this following looop will check the jackpot condition
 //if a particular switch is pressed and the same LED is glowing,
then you hit a JACKPOT and all LEDS will glow.
 //if not the blinking will stop
 else
 begin
 //jackpot condition: All the LEDs will glow only when the switch
//corresponding to the LED which is glowing is selected
    if(SWITCHES==LEDS && LEDS != 4'b0000)
        begin
            LEDS<=4'b1111;

          end
 //the reset condition is to restart the game.
    if (RESET)
        LEDS<=4'b0000;
end
end
endmodule
```

Constraints

```
## 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 {CLOCK}]
set_property IOSTANDARD LVCMOS33 [get_ports {CLOCK}]

## RESET
set_property PACKAGE_PIN K19 [get_ports {RESET}]
set_property IOSTANDARD LVCMOS33 [get_ports {RESET}]
```

Result:

In the first experiment, the assign statement assigned the switches corresponding to the LEDS mentioned. In this way, each LED was controlled by its corresponding switch. In the second experiment, the push buttons functioned as the up down counter. The main challenge faced in this was reducing the frequency of the internal clock. This challenge was overcome by using an intermediate counter, that reduced the internal clock frequency. This made the change in output visible. In third experiment, initially I programmed it in such a way that if that particular switch is ON, then whenever the LED corresponding to that switch would glow, jackpot condition was met. But then I modified the code to ensure that only at the instance when the switch and LED glowing both matches, the jackpot condition will be met. The code was then successfully executed.

Conclusion:

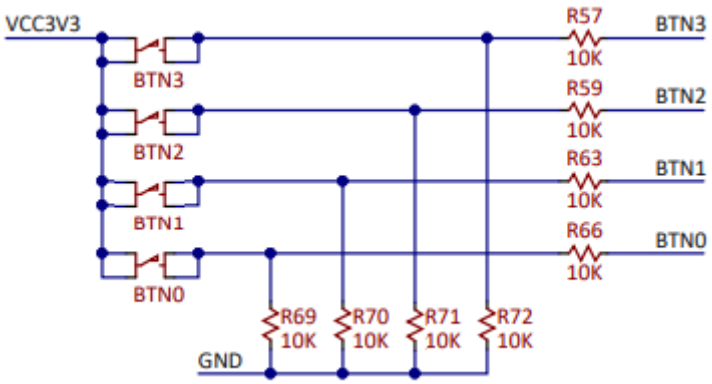
This lab served its purpose of introducing Verilog coding and ZYBO Z7-10 board. This lab provided hands on experience on ZYBO board. Errors, apart from syntax, that occur during synthesis were also learnt. Debugging of these errors further strengthened the basics.

Questions:

3a. How are the user pushbuttons 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)?

Ans:

```
#set_property -dict { PACKAGE_PIN K18   IOSTANDARD LVCMOS33 } [get_ports { btn[0] }];
#IO_L12N_T1_MRCC_35 Sch=btn[0]
```



As from the schematic, it is visible that the pushbuttons BTN0, BTN1, BTN2 and BTN3 are pulled low.

3b. What is the purpose of an edge detection circuit and how should it have been used in this lab?

Ans: The edge detection circuit would generate a trigger signal or be active only when there is an edge in the input fed to the circuit. A typical input would be clock. In this lab, we have used “posedge clock” to make the always block active only at the positive edge of the input clock. If edge detection circuit was used in our lab, then the output of the circuit would have been used to activate the always block.