Description: Description: Description: Description: Description: Description: Description: Description: Description: Description: Texas A&M University Logo - aTm

Lab #1: Vivado

Khanh Nguyen

UIN# 525000335

ECEN 449– 503

Date: January 23, 2017

**INTRODUCTION:**

The purpose of the lab is to get us the first touch to get used with Vivado and Verilog. I also learned how to implement Verilog into the FPGA board (Zybo board). By using dataflow and behavioral Verilog, we were able to divide the clock rate then created a simple counter and a jackpot game.

**PROCEDURE:**

1/ An example code and constraints code for the switches was provided. I used Vivado to implement the Verilog code into the FPGA board. The LEDs should be on and off according to the switches.

2/ We designed a 4-bit counter that can count up or count down whenever the assigned button is pressed. The counter will have 4 inputs: UP, DOWN, RESET, and CLOCK. The output will be the LEDs on the board. I also had to create the constraints file for the according buttons and LEDs.

3/ The last exercise is to use the divided clock to create a jackpot game with 4 switches and 4 LEDs. I had the LEDs changing in sequential order. The player must turn on the right switch where the LED is on in order to win. I wrote the Verilog code with the constraints and debug it until it successfully implemented on hardware.

**RESULT:**

1/ I learned how to create project, create constraint and implement onto FPGA with Vivado.

2/ I learned how to divide the clock rate and create simple counter. I approached by dividing the clock rate since the default clock rate is 125MHz. I wanted to reduce the clock rate to 1Hz (1s/cycle) so I multiply the current clock period with 40,000,000. It resulted in 0.5s for half cycle. After 0.5s, the clock will be reversed. I used that clock for my counter to get updated every the UP, DOWN, or RESET button is pressed.

3/ The Jackpot game was successfully implemented onto the FPGA board. I used the same divided clock from previous exercise to get the LEDs updated. When a switch is on, a memory variable will be assigned to have the same value of that switch after 40e6 unit delays (0.5s). When the LED and the switch are the same, it will check for another condition if the memory is assigned or not. Player only wins if the switch is on and memory is **not** equal to the switch, and switch matches with LED. If the switch is on and memory is assigned, and switch doesn’t match with LED, the LED will go pass the on switch.

**VERILOG CODES:**

**counter.v**

module counter(

input UP,

input DOWN,

input CLOCK,

input RESET,

output [3:0] LEDS

);

reg [3:0] COUNT;

reg divided\_clk = 1'b0; // divided clock starts at 0

reg [31:0] counter = 1'b0;// use counter to increment the clock

always@(posedge divided\_clk) //three buttons to control leds

begin

if(RESET == 1) begin

COUNT <=1'b0;

end else if (UP == 1) begin

COUNT <= COUNT + 1; //count up

end else if (DOWN == 1) begin

COUNT <= COUNT - 1; //count down

end

end

always@ ( posedge CLOCK ) //divide the clock to 1HZ (1s for one period)

begin

counter <= counter + 1'b1;

if ( counter == 40000000) // overtun the clk1 after 0.5s

// clock = 125MHz=0.8E-8 (s)

//=>(0.8E-8)\*40E6=0.5s

begin

divided\_clk <= ~divided\_clk; //reverse the clock

counter <= 1'b0; //reset the counter

end

end

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

endmodule

**counter.xdc**

**##INPUTS**

**set\_property PACKAGE\_PIN R18 [get\_ports {UP}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {UP}]**

**set\_property PACKAGE\_PIN P16 [get\_ports {DOWN}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {DOWN}]**

**set\_property PACKAGE\_PIN V16 [get\_ports {RESET}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {RESET}]**

**set\_property PACKAGE\_PIN L16 [get\_ports {CLOCK}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {CLOCK}]**

**##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]}]**

**jackpot.v**

module Jackpot(

input CLOCK,BUTTON,

input [3:0] SWITCHES,

input RESET,

output [3:0] LEDS

);

reg [3:0] memmory;

reg [3:0] COUNT;

reg divided\_clk = 0; //clock starts at 0

reg [31:0] counter = 1'b0; // use to increment the clock

always@ ( posedge CLOCK ) //divide the clock to 1HZ

begin

counter <= counter + 1'b1;

if ( counter == 40000000) // overtun the clk1 after 0.5s

// clock = 125MHz=1.25E-8 (s)

//=>(1.25E-8)\*40E6=0.5s

begin

divided\_clk <= ~divided\_clk; // reverse the clock

counter <= 1'b0;// reset clock

end

end

always@(posedge divided\_clk)

begin

if ( RESET ) //reset the LEDS

begin

COUNT = 4'b1000;

end

else if ( COUNT == 4'b0001 ) //let the LEDS glowing in cycle

begin

COUNT = 4'b1000;

end

else if (SWITCHES)

begin

#40000000 memmory<=SWITCHES; // save the position of the switch after 0.5s

if ( (COUNT == SWITCHES) && (SWITCHES!=memmory) ) //detect the switch, if memmory== switch it doesnt work

begin

COUNT = 4'b1111; //win the jackpot

end

else

begin

COUNT = {COUNT[0], COUNT[3:1]};

end

end

else

begin

COUNT = {COUNT[0], COUNT[3:1]}; //move the 1 in cycle,keep the LEDS glowing sequentially

end

end

assign LEDS[3:0] = COUNT[3:0]; //assign COUNT to LEDS

endmodule

**jackpot.xdc**

**##SWITCHES INPUTS**

**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]}]**

**set\_property PACKAGE\_PIN L16 [get\_ports {CLOCK}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {CLOCK}]**

**##LEDs OUTPUTS**

**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 R18 [get\_ports {RESET}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {RESET}]**

**QUESTIONS:**

1/The use push buttons are connected like below:

BTN[0]=R18 (up button in counter and reset button in jackpot)

BTN[1]=P16 (down button in counter)

BTN[2]=V16 (reset button in jackpot)

BTN[3]=Y16

2/ The purpose of an edge detection circuit is to know when a signal gets updated. Therefore, all the components can be implemented synchronously. In this lab, it is used as a clock to have the LEDs to be on every equal clock cycle.