San Jose State University
Department of Computer Engineering

CMPE 125 Lab Report

Lab 2 Report

Title FPGA Implementation and Hardware Validation

Semester FALL 2019 **Date** 09/09/2019

by

Name Phat Le (typed) **SID** 012067666 (typed)

Lab Checkup Record

Week	Performed By (signature)	Checked By (signature)	Tasks Successfully Completed*	Tasks Partially Completed*	Tasks Failed or Not Performed*
1			✓		

* Detailed descriptions must be given in the report.

Introduction

The purpose of this lab is to go through the FPGA implementation procedure through Xilinx's Vivado software, and to follow the hardware validation method using Digilent's Basys3 board.

Design Methodology

Given the Voting Rule modules in *Figure 1* and list of files in Table 1, this lab task is to simulate the design and upload it in the Basys3 device through Vivado software. Specifically, we need to simulate the given files and verify the output by looking at the waveform and through FPGA validation process using the Basys3 board. Therefore, we use *Table 2* to verify the result input and output.

Table 1: List of Modules and Files Used

Module/File Name	**Comments**
voting_machine_fpga.v	Top-level module for the system shown in Figure 1
voting_rule.v	Designed module with the function shown in Table 1
clk_gen.v	Utility module for converting the on-board clock to 5KHz
bcd_to_7seg.v	Utility module with the function shown in Table 2
led_mux.v	Utility module for multiplexing signals to be displayed on the eight 7-segment LEDs on the Basys3 board
tb_voting_machine.v	Testbench file for the designed module voting_rule.v
voting_machine_fpga.xdc	Design constraint file for prototyping the voting machine

Table 2: Function table for Module *voting_rule.v*

Inputs (with weights)			**Outputs (in BCD code)**				**Expected Decimal Display**
w(2)	n(3)	o(4)	a	b	c	d	On 7-Seg LED
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	4
0	1	0	0	0	1	1	3
0	1	1	0	1	1	1	7
1	0	0	0	0	1	0	2
1	0	1	0	1	1	0	6
1	1	0	0	1	0	1	5
1	1	1	1	0	0	1	9

Table 3: Function table for Module *bcd_to_7seg.v*

Inputs (in BCD code)	**Outputs (active low)**	**For Decimal Display**
BCD [3:0]	S6 S5 S4 S3 S2 S1 S0	On 7-Seg LED
0 0 0 0	1 0 0 0 0 0 0	0
0 1 0 0	0 0 1 1 0 0 1	4
0 0 1 1	0 1 1 0 0 0 0	3
0 1 1 1	1 1 1 1 0 0 0	7
0 0 1 0	0 1 0 0 1 0 0	2
0 1 1 0	0 0 0 0 0 1 0	6
0 1 0 1	0 0 1 0 0 1 0	5
1 0 0 1	0 0 1 0 0 0 0	9

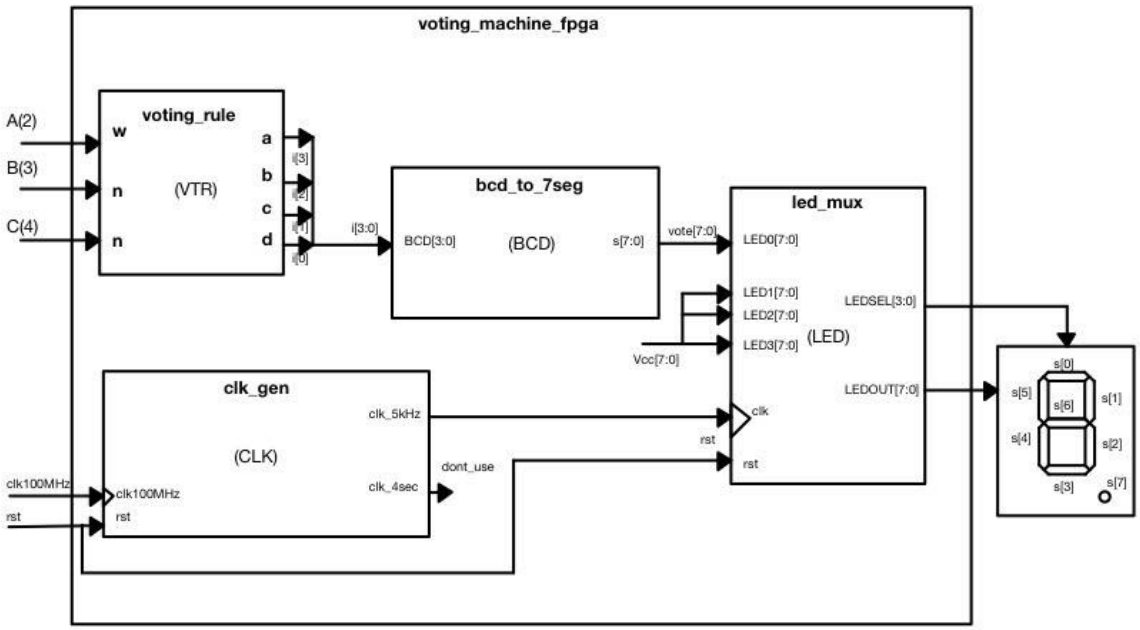

Figure 1: Block diagram of module *voting_machine_fpga*

Simulation Results

According to figure 2 below, signal w, n o is the input from the three dip switches. When w, n, o is 1,1,1, we can observe that a, b, c, d is 1001 in binary which is equal to 9 in decimal. For this reason, we can tell that the simulation process has been succeeded since it matches the result in *Table 2* below.

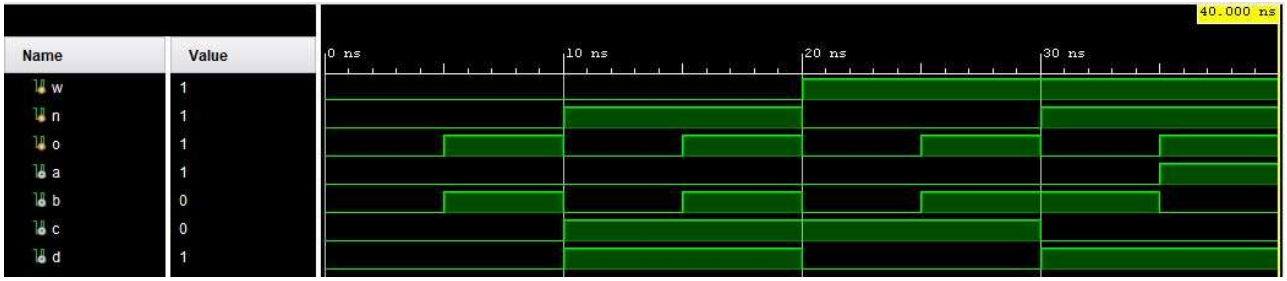

Figure 2: Simulation waveforms produced using Vivado

FPGA Validation

According to figure 4, when we turn on all switches (111), the LED 7 segments output 9. When it is 001, the output is 4. Thus, the FPGA validation has been succeeded as the output matches *Table 2.* .

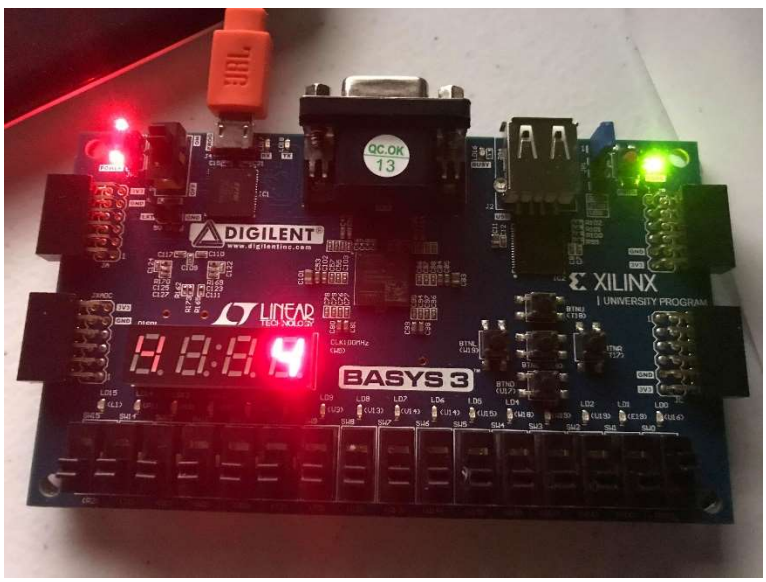

Figure 3: Photo of operation being (insert operation here is 001) tested on FPGA board.

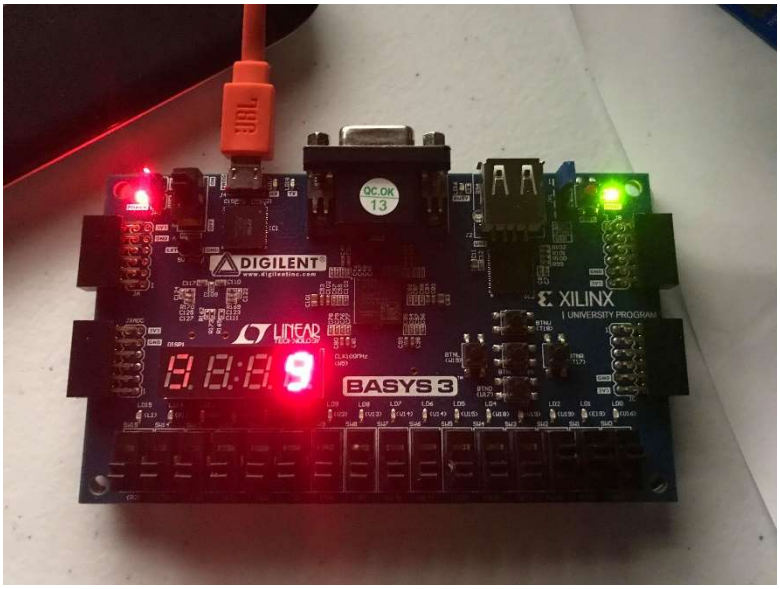

Figure 4: Photo of operation being (insert operation here is 111) tested on FPGA board.

Conclusion

The result waveform from simulation as well as the output display matches the expected result from the function table in *Figure 2.* For this reason, lab tasks have successfully completed. During the lab, one thing that I noticed when doing the FPGA validation process was having difficulty detecting the hardware device on the computer. The reason for this is because the Basys3 only accept certain type of connecting wires. I have tried out five different wires but only one that works.

Appendix

Voting_machine_fpga.v

```
module voting_machine_fpga (
          input    wire          clk100MHz,
          input    wire          rst,
          input    wire          A,
          input    wire          B,
          input    wire          C,
          output wire            A_led,
          output wire            B_led,
          output wire            C_led,
          output wire [3:0] LEDSEL,
          output wire [7:0] LEDOUT
     );

     assign A_led = A;
     assign B_led = B;
     assign C_led = C;

     supply1 [7:0] vcc;

     wire DONT_USE;
     wire clk_5KHz;

     wire [3:0] i;
     wire [7:0] vote;

     voting_rule VTR (
               .w                  (A),
               .n                  (B),
               .o                  (C),
               .a                  (i[3]),
               .b                  (i[2]),
               .c                  (i[1]),
               .d                  (i[0])
          );

     clk_gen CLK (
               .clk100MHz          (clk100MHz),
               .rst                (rst),
               .clk_4sec           (DONT_USE),
               .clk_5KHz           (clk_5KHz)
          );

     bcd_to_7seg BCD (
               .BCD                (i),
               .s                  (vote)
          );

     led_mux LED (
               .clk                (clk_5KHz),
               .rst                (rst),
               .LED3               (vcc),
               .LED2               (vcc),
               .LED1               (vcc),
               .LED0               (vote),
               .LEDSEL             (LEDSEL),
               .LEDOUT             (LEDOUT)
          };
 endmodule
```

voting_rule.v

```
module voting_rule (
         input   wire   w,
         input   wire   n,
         input   wire   o,
         output   wire   a,
         output   wire   b,
         output   wire   c,
         output wire d
    );

    assign a = w&n&o;
    assign b = ~w&~n&o | ~w&n&o | w&~n&o | w&n&~o; assign c =
    ~w&n&~o | ~w&n&o | w&~n&o | w&~n&~o; assign d = ~w&n&~o |
    ~w&n&o | w&n&~o | w&n&o;

endmodule
```

clk_gen.v

```
module clk_gen (
        input  wire clk100MHz,
        input  wire rst,
        output reg  clk_4sec,
        output reg  clk_5KHz
    );

    integer count1, count2;

    always @ (posedge clk100MHz) begin
        if (rst) begin
            count1 = 0; clk_4sec = 0;
            count2 = 0; clk_5KHz = 0;
        end
        else begin
            if (count1 == 200000000) begin
                clk_4sec = ~clk_4sec;
                count1 = 0;
            end

            if (count2 == 10000) begin
                clk_5KHz = ~clk_5KHz;
                count2 = 0;
            end
            count1 = count1 + 1;
            count2 = count2 + 1;
        end
        end
endmodule
```

bcd_to_7seg.v

```
module bcd_to_7seg (
        input  wire [3:0] BCD,
        output reg  [7:0] s
    );

    always @ (BCD) begin
        case (BCD)
               4'd0: s = 8'b11000000;
               4'd1: s = 8'b11111001;
               4'd2: s = 8'b10100100;
               4'd3: s = 8'b10110000;
               4'd4: s = 8'b10011001;
               4'd5: s = 8'b10010010;
               4'd6: s = 8'b10000010;
               4'd7: s = 8'b11111000;
               4'd8: s = 8'b10000000;
               4'd9: s = 8'b10010000;
            default: s = 8'b01111111;
        endcase
    end

endmodule
```

Led_mux.v

```
module led_mux (
          input    wire          clk,
          input    wire          rst,
          input    wire [7:0] LED3, input
                   wire [7:0] LED2, input
                   wire [7:0] LED1, input
                   wire [7:0] LED0,
          output wire [3:0] LEDSEL,
          output wire [7:0] LEDOUT
     );

     reg [1:0]     index;
     reg [11:0] led_ctrl;

     assign {LEDSEL, LEDOUT} = led_ctrl;

     always @ (posedge clk) index <= (rst) ? 2'b0 : (index + 2'd1); always @

     (index, LED0, LED1, LED2, LED3) begin
          case (index)
                   4'd0:  led_ctrl  <=  {4'b1110,  LED0};  4'd1:
                   led_ctrl   <=   {4'b1101,   LED1};   4'd2:
                   led_ctrl   <=   {4'b1011,   LED2};   4'd3:
                   led_ctrl <= {4'b0111, LED3};
             default: led_ctrl <= {8'b1111, 8'hFF};
        endcase
    end

endmodule
```

tb_voting_rule.v

```
module tb_voting_rule;

     reg   w;
     reg   n;
     reg   o;
     wire  a;
     wire  b;
     wire  c;
     wire d;

     voting_rule DUT (
               .w            (w),
               .n            (n),
               .o            (o),
               .a            (a),
               .b            (b),
               .c            (c),
               .d            (d)
          );

     initial begin o
          = 0;
          forever #5 o = ~o;
     end

     initial begin n
          = 0;
          forever #10 n = ~n;
     end

     initial begin w
          = 0;
          forever #20 w = ~w;
     end

     initial #40 $stop;

     initial $monitor(
               $time,
               " A=%b, B=%b, C=%b : a=%b, b=%b, c=%b, d=%b",
               w, n, o, a, b, c, d
          );

endmodule
```