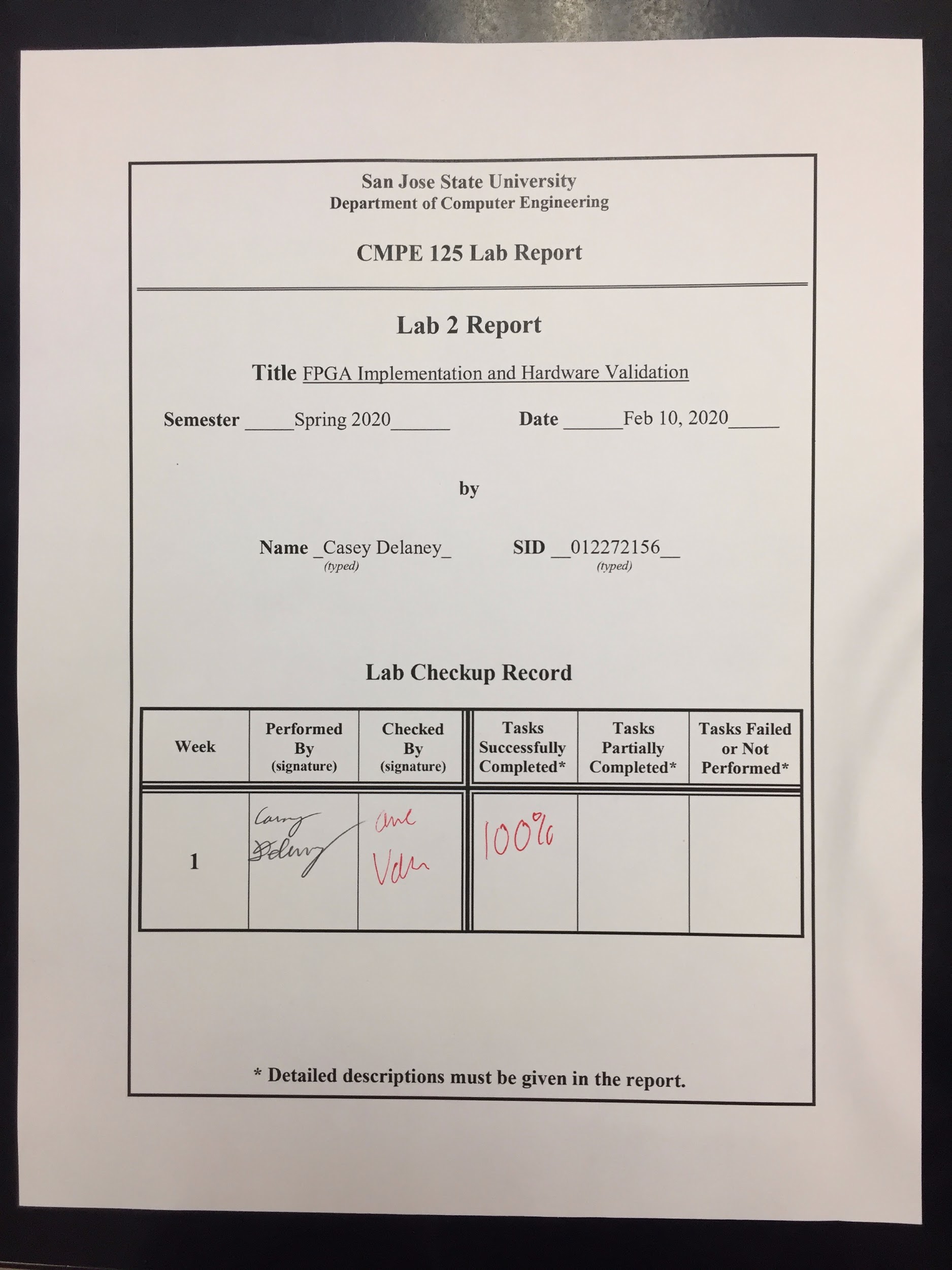
****

**Introduction**

In this lab, we will be using the Bassy3 board to leard about FPGA implementation. By using the Vivoda EDA tool, we will be able to flash a program onto the board to verify that it works correctly. The program is a voting machine with weighted inputs. One of the voters will receive a weight of 2, the second will receive a weight of 3 and the last will receive a weight of 4.

**Design Methodology**

For the design of this lab, I referred to the given material in the Lab Tutorial document from the CMPE 125 class. There are a total of 5 modules that were used in this lab, as described below in the Module Table. The use of the modules is seen in the top level design as shown in figure 1.

In addition to the 5 modules, there was a testbench file to compare the simulation waveform to.

The last item is the constraint file, *voting\_machine\_fpga.xdc*, which was used to set certain pins on the Bassy3 FPGA board to be linked to certain items in the modules.

*Table 1:* Module Table

|  |  |
| --- | --- |
| Module | Function |
| voting\_machine\_fpga.v | Top level module which combines all of the lower level modules to create a voting machine |
| voting\_rule.v | Converts inputs to BCD (binary coded decimal) |
| clk\_gen.v | Converts clock to 5KHz |
| bcd\_to\_7seg.v | Converts BCD number to match appropriate 7seg display value |
| led\_mux.v | Multiplexer which choose a 7seg display |

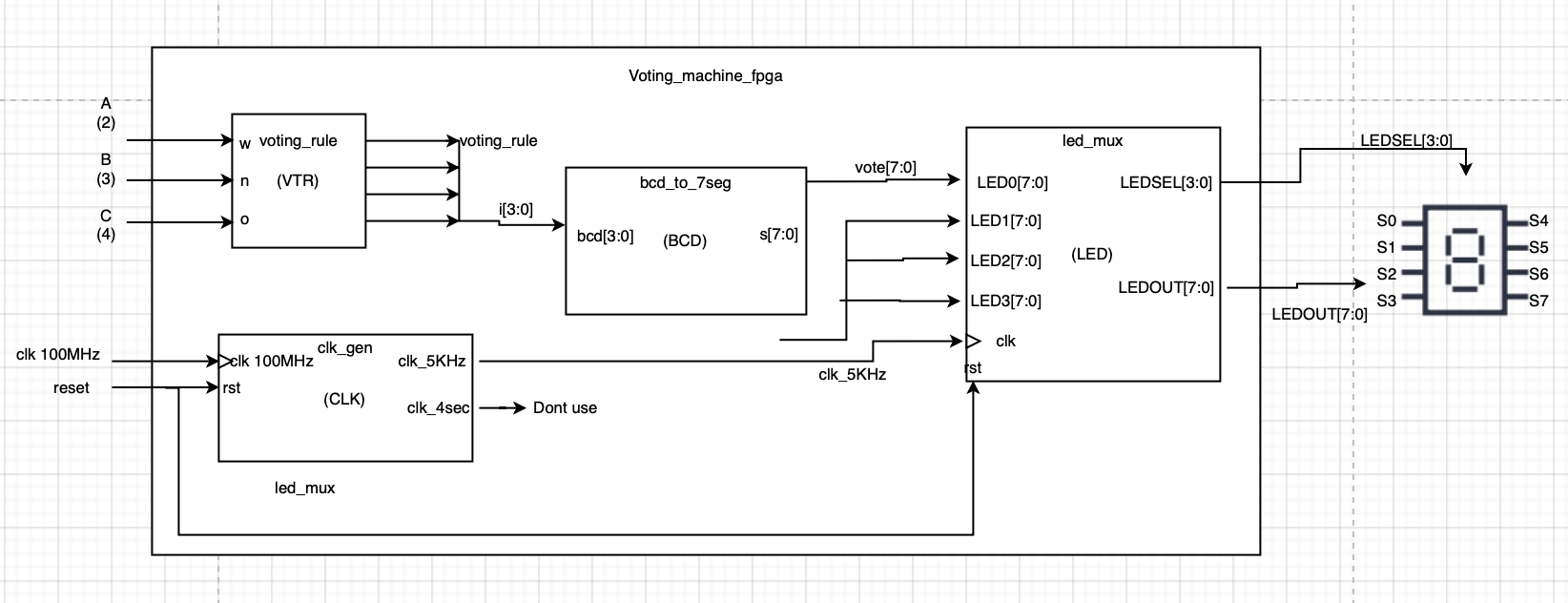
The functions of *voting\_rule.v* and *bcd\_to\_7seg.v* can be seen in table 2 and table 3 below.

*Table 2.* Function table for Module*voting\_rule.v*

|  |  |  |
| --- | --- | --- |
| Inputs(with weights) | Outputs (in BCD code) | Decimal Display |
| (w1) (w2) (w3) | a b c d | 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) | Outputs (active low) | Decimal Display |
| BCD[3:0] | S6 S5 S4 S3 S2 S1 S0 | 7-seg LED |
| 0 0 0 0 | 1 0 0 0 0 0 0 | 0 |
| 0 1 0 0 | 0 0 0 1 1 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 | 1 0 0 0 0 1 0 | 6 |
| 0 1 0 1 | 0 0 1 0 0 1 0 | 5 |
| 0 0 0 0 | 0 0 1 0 0 0 0 | 9 |

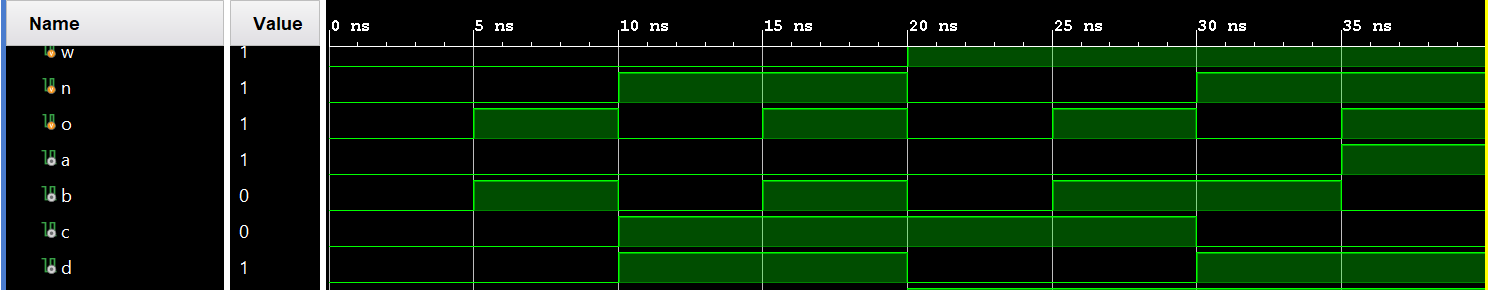


*Figure 1.* Top level module of *voting\_machine\_fpga.v*

**Simulation Results**

For this lab, we had three weighted inputs, labeled w, n and o in the figure 2 simulation. These three weighted bits were converted to BCD by the *voting\_rule* module and then converted to outputs for the 7 segment displays by the *bcd\_to\_7seg* module. If we look at the case of wno = 100, we see that abcd = 0010 which results in the number 2 showing up on the 7 segment display. By looking at the function tables in table 1 and table 2, we see that the simulation results are correct.

Also, a file called *tb\_voting\_rule* was created to verify that the simulation was correct, and both waveforms were the same.

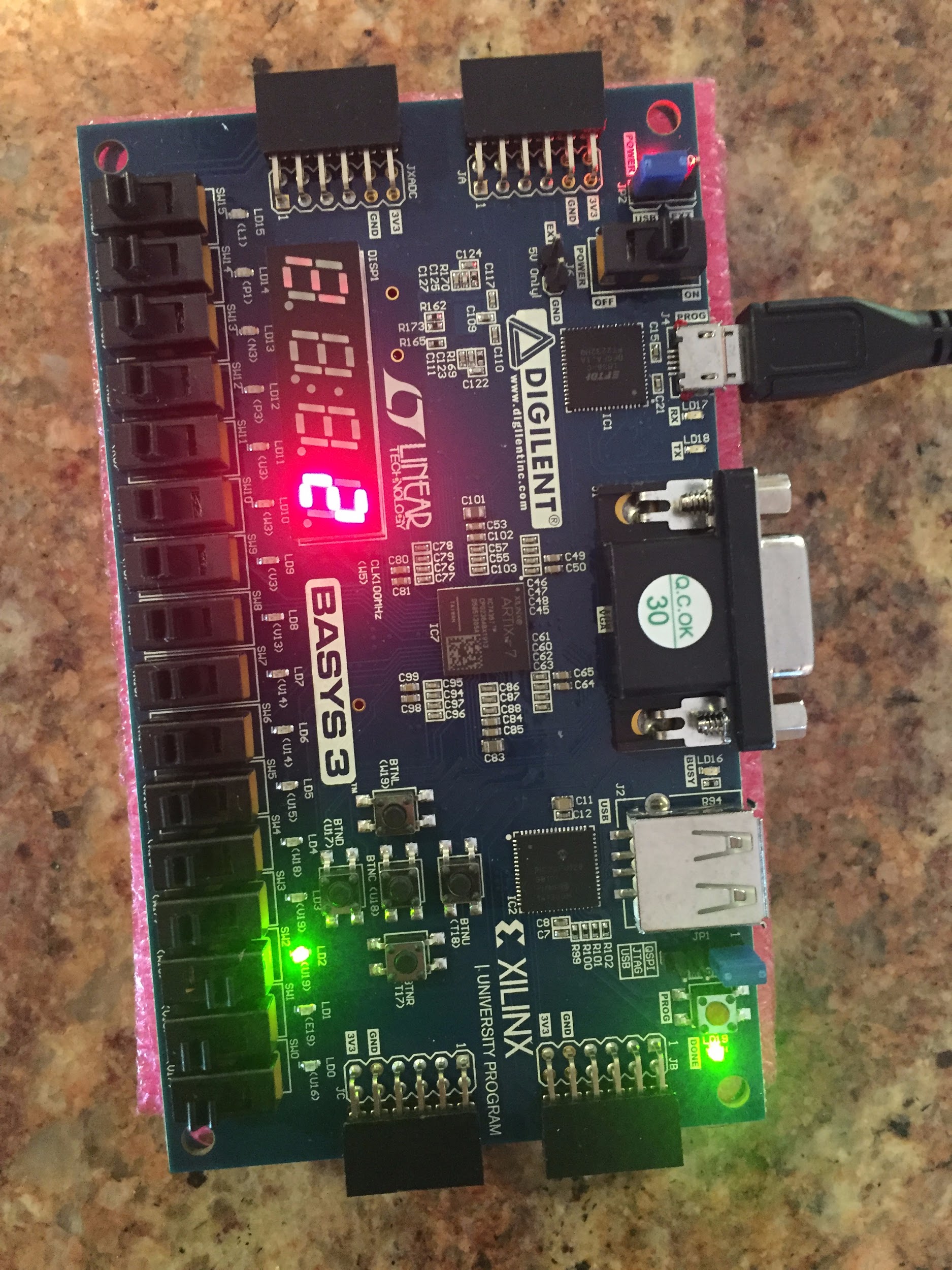


*Figure 2.* Simulation waveform

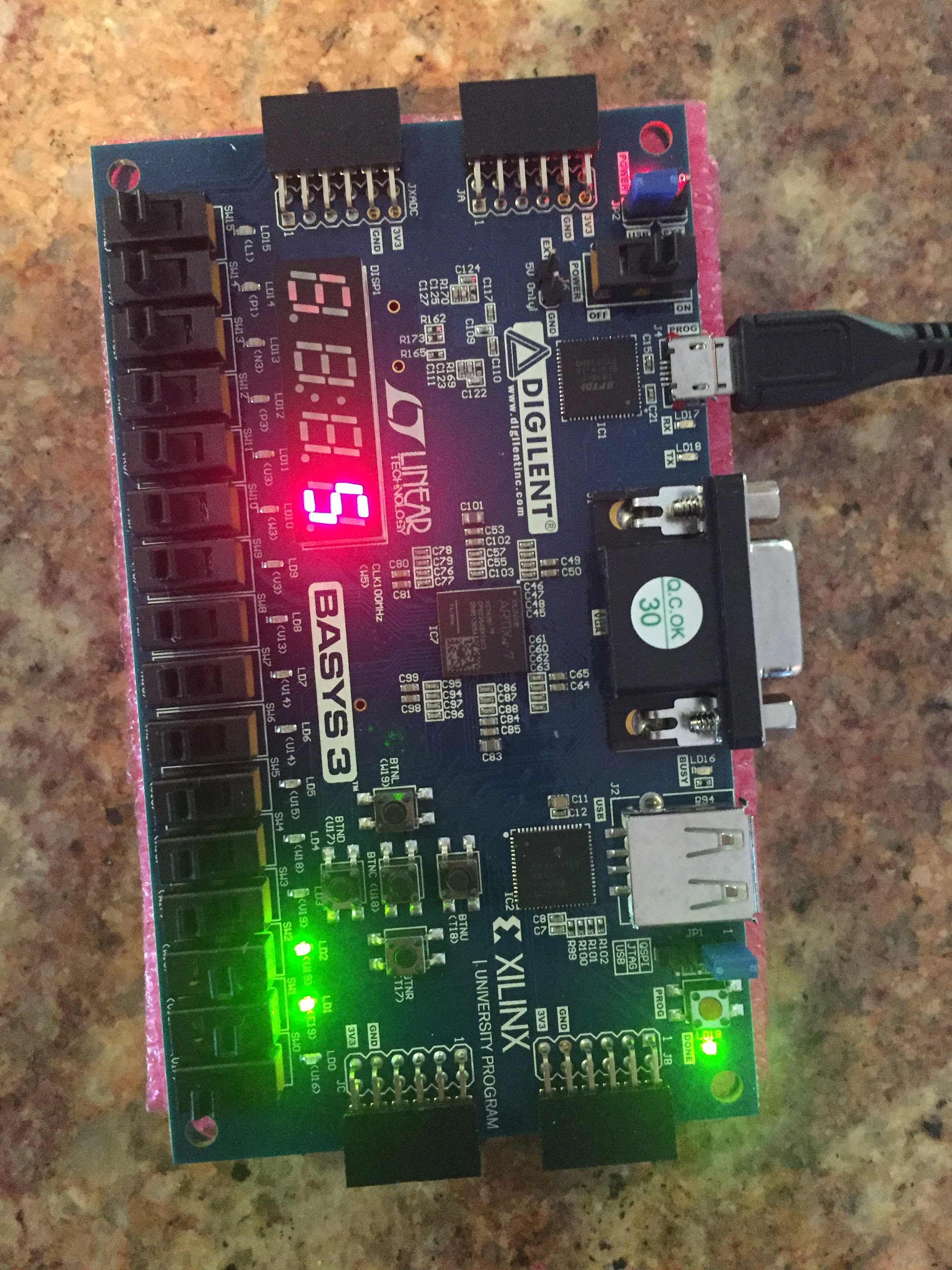
**FPGA Validation**

To test the FPGA board, I used a variety of inputs to test the voting machine. Some of the inputs tested are shown in figures 3, 4 and 5.

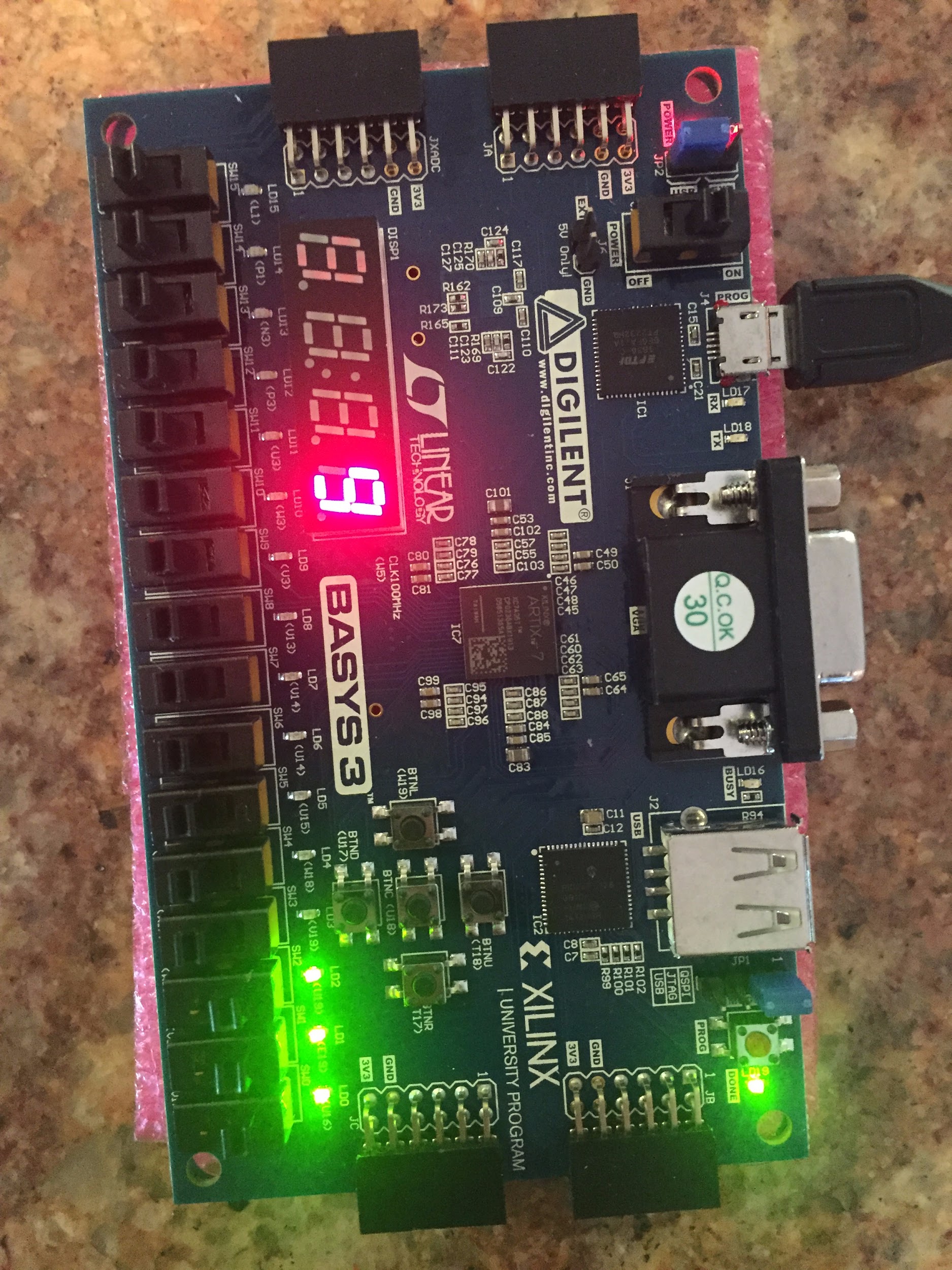
First off, I tested an input of 1 0 0. Referring to table 2, we see that the output should be 2. This is seen in figure 3. I proceeded to turn the other switches on to test for cases of 1 1 0 and 1 1 1 respectively, and all testing scenarios proved to be correct.



*Figure 3.* Input of 1 0 0 on the FPGA board (2 + 0 + 0) results in 2



*Figure 4.* Input of 1 1 0 on the FPGA board (2 + 3 + 0) results in 5



*Figure 5.* Input of 1 1 1 on the FPGA board (2 + 3 + 4) results in 9

**Conclusion**

In this lab, I learned how to create a project in the Vivado Design Suite EDA and successfully turn the project into a bitstream which can then be loaded onto my Bassy3 FPGA board. One of the important takeaways is that I must validate my waveform with a testbench waveform to verify that all of the modules are working correctly. Once this has been verified to be true, the next step is to flash the board with the bitstream. After successfully flashing the board, I was able to verify that the board works by comparing it to the tables in tables 2 and 3. I was able to successfully complete the lab during the allotted time with a complete mark.

**Appendix**

*Table 4:* Source Code

|  |
| --- |
| 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 |  |  | | --- | | .LED1 (vcc),  .LED0 (vote),  .LEDSEL (LEDSEL),  .LEDOUT (LEDOUT)  );  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 | |

|  |
| --- |
| voting\_machine\_fpga.xdc |
| |  | | --- | | # Clock signal  set\_property -dict {PACKAGE\_PIN W5 IOSTANDARD LVCMOS33} [get\_ports {clk100MHz}];  create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports {clk100MHz}];  # Buttons  set\_property -dict {PACKAGE\_PIN U18 IOSTANDARD LVCMOS33} [get\_ports {rst}]; # Center Button  # Switches  set\_property -dict {PACKAGE\_PIN V17 IOSTANDARD LVCMOS33} [get\_ports {C}]; # Switch 0  set\_property -dict {PACKAGE\_PIN V16 IOSTANDARD LVCMOS33} [get\_ports {B}]; # Switch 1  set\_property -dict {PACKAGE\_PIN W16 IOSTANDARD LVCMOS33} [get\_ports {A}]; # Switch 2  # LEDs  set\_property -dict {PACKAGE\_PIN U16 IOSTANDARD LVCMOS33} [get\_ports {C\_led}]; # LED 0  set\_property -dict {PACKAGE\_PIN E19 IOSTANDARD LVCMOS33} [get\_ports {B\_led}]; # LED 1  set\_property -dict {PACKAGE\_PIN U19 IOSTANDARD LVCMOS33} [get\_ports {A\_led}]; # LED 2  # 7 Segment Display  set\_property -dict {PACKAGE\_PIN W7 IOSTANDARD LVCMOS33} [get\_ports {LEDOUT[0]}]; # CA  set\_property -dict {PACKAGE\_PIN W6 IOSTANDARD LVCMOS33} [get\_ports {LEDOUT[1]}]; # CB  set\_property -dict {PACKAGE\_PIN U8 IOSTANDARD LVCMOS33} [get\_ports {LEDOUT[2]}]; # CC  set\_property -dict {PACKAGE\_PIN V8 IOSTANDARD LVCMOS33} [get\_ports {LEDOUT[3]}]; # CD  set\_property -dict {PACKAGE\_PIN U5 IOSTANDARD LVCMOS33} [get\_ports {LEDOUT[4]}]; # CE  set\_property -dict {PACKAGE\_PIN V5 IOSTANDARD LVCMOS33} [get\_ports {LEDOUT[5]}]; # CF  set\_property -dict {PACKAGE\_PIN U7 IOSTANDARD LVCMOS33} [get\_ports {LEDOUT[6]}]; # CG  set\_property -dict {PACKAGE\_PIN V7 IOSTANDARD LVCMOS33} [get\_ports {LEDOUT[7]}]; # DP  set\_property -dict {PACKAGE\_PIN U2 IOSTANDARD LVCMOS33} [get\_ports {LEDSEL[0]}]; # AN0  set\_property -dict {PACKAGE\_PIN U4 IOSTANDARD LVCMOS33} [get\_ports {LEDSEL[1]}]; # AN1  set\_property -dict {PACKAGE\_PIN V4 IOSTANDARD LVCMOS33} [get\_ports {LEDSEL[2]}]; # AN2 | |