Lab Experiment 4
Time Multiplexed Display
October 2, 2018

By:
Calvin Truong
Tony Lee

Objective:

The objective of this lab was to create a clock using Verilog in order to control all of the seven segment display. The ultimate goal was to display different values on each of the display. Four new modules need to be created for this lab. The first one is a clock generator. It accepts a reset button and a 100MHz clock and generates a 1KHz clock enable signal. The next module is a data generation. We hardcoded what we wanted to display on the seven segment displays. In this lab we wanted to display "ECE3300" leaving the last one for us to decide. Another module we created was a counter module. It accepts both clocks to count from 0 to the number we input through the board. Lastly, a 8 to 1 mux was needed to allow us to control what is being displayed.

Procedure:

1. Created a clock generator to generate a 1KHz enable signal when it is 10 ns wide. A reset button was also hooked into this module. Set the MAX_COUNT local parameter to 9 for simulation testing and change the local parameter to 99999 for synthesist testing.
2. Made a module that generates the data of the inputs onto the seven segment display. "ECE3300" was hardcoded in the module.
3. A counter was created that made use of both clocks to count up to the inputted value.
4. An 8 to 1 mux needed to be created to display the inputs onto the seven segment display.
5. The 3 to 8 decoder and seven segment decoder was copied from the previous lab as well.
6. A large seven_seg_display module was created that incorporated all the modules needed to display our desired output, including: the counter, 8 to 1 mux, 3 to 8 decoder, and seven segment decoder. The counter's output is the input of the 3 to 8 decoder and 8 to 1 mux. The output of the mux is then linked to the seven segment decoder.
7. A large module was created which included all the modules mentioned. The clock generator, data generator and reset button is connected to the large seven_seg_display module.
8. Developed a test bench as well as connect all of the constraints onto the FPGA board to display the results.

Verilog:

lab4_top.v *

C:/Users/CalvinT/Desktop/ECE3300/Lab4/Lab4.srcs/sources_1/new/lab4_top.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Module:lab4_top
4  // Engineer: Calvin Truong and Tony Lee
5  //
6
7  module lab4_top(Clk100MHz, NumDigits, Sw, BTNC, CA, CB, CC, CD, CE, CF, CG, DP, AN);
8      input Clk100MHz, BTNC;
9      input [2:0] NumDigits;
10     input [5:0] Sw;
11     output CA,CB,CC,CD,CE,CF,CG,DP;
12     output [7:0] AN;
13     wire reset_n;
14     wire Clk1KHzEn;
15     wire [7:0] XDP;
16     wire [31:0] Data;
17
18     not (reset_n,BTNC);
19
20     clk_gen u_clk_gen(
21         .Clk100MHz(Clk100MHz),
22         .reset_n(reset_n),
23         .Clk1KHzEn(Clk1KHzEn)
24     );
25
26     data_gen u_data_gen(
27         .X(Sw[3:0]),
28         .Y(Sw[5:4]),
29         .Data(Data),
30         .XDP(XDP)
31     );
32
33     seven_seg_display u_seven_seg_display(
34         .Clk100MHz(Clk100MHz),
35         .Clk1KHzEn(Clk1KHzEn),
36         .NumDigits(NumDigits),
37         .Data(Data),
38         .XDP(XDP),
39         .reset_n(reset_n),
40         .CA(CA),
41         .CB(CB),
42         .CC(CC),
43         .CD(CD),
44         .CE(CE),
45         .CF(CF),
46         .CG(CG),
47         .DP(DP),
48         .AN(AN)
49     );
50     assign reset_n = ~BTNC;
51 endmodule
```

clk_gen.v *

C:/Users/CalvinT/Desktop/ECE3300/Lab4/Lab4.srcs/sources_1/new/clk_gen.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////
3  // Module:clk_gen
4  // Engineer: Calvin Truong and Tony Lee
5  //
6
7  module clk_gen(Clk100MHz, reset_n, Clk1KHzEn);
8      input Clk100MHz, reset_n;
9      output Clk1KHzEn;
10
11     //localparam MAX_COUNT = 9;        //Simulation
12     localparam MAX_COUNT = 99999;  //Synthesis
13
14     reg Clk1KHzEn;
15     reg [16:0] cnt;
16
17     always @(posedge Clk100MHz) begin
18         if (!reset_n)
19             cnt <= 0;
20         else if (cnt == MAX_COUNT)
21             cnt <= 0;
22         else
23             cnt <= cnt + 1;
24     end
25     always @(posedge Clk100MHz) begin
26         if(!reset_n)
27             Clk1KHzEn <= 0;
28         else if (cnt == MAX_COUNT)
29             Clk1KHzEn <= 1;
30         else
31             Clk1KHzEn <= 0;
32     end
33 endmodule
```

data_gen.v

C:/Users/CalvinT/Desktop/ECE3300/Lab4/Lab4.srcs/sources_1/new/data_gen.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Module:data_gen
4  // Engineer: Calvin Truong and Tony Lee
5  //
6
7  module data_gen(X, Y, Data, XDP);
8      input [3:0] X;
9      input [1:0] Y;
10     output reg [31:0] Data;
11     output reg [7:0] XDP;
12
13     always @ (X or Y)   begin
14         Data = {28'hECE3300,X};
15         XDP = {6'b010101, Y};
16     end
17 endmodule
```

counter.v

C:/Users/CalvinT/Desktop/ECE3300/Lab4/Lab4.srcs/sources_1/new/counter.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Module:counter
4  // Engineer: Calvin Truong and Tony Lee
5  //
6
7  module counter(Clk100MHz, Clk1KHzEn, reset_n, NumDigits, cnt);
8      input Clk100MHz, Clk1KHzEn, reset_n;
9      input [2:0] NumDigits;
10     output reg [2:0] cnt = 0;
11
12     always @ (posedge Clk100MHz)
13         begin
14             if(!reset_n)
15                 cnt <= 0;
16             else if(Clk1KHzEn == 1)
17                 begin
18                     if(NumDigits == 0)
19                         cnt <= 3'b000;
20                     else if(cnt == NumDigits)
21                         cnt <= 3'b000;
22                     else if (NumDigits > 1)
23                         cnt <= cnt + 1;
24                 end
25         end
26 endmodule
```

seven_seg_display.v *

C:/Users/CalvinT/Desktop/ECE3300/Lab4/Lab4.srcs/sources_1/new/seven_seg_display.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Module: seven_seg_display
4  // Engineer: Calvin Truong and Tony Lee
5  //
6
7  module seven_seg_display(Clk100MHz, Clk1KHzEn, NumDigits, Data, XDP, reset_n, CA, CB, CC, CD, CE, CF, CG, DP, AN);
8      input Clk100MHz, Clk1KHzEn, reset_n;
9      input [2:0] NumDigits;
10     input [31:0] Data;
11     input [7:0] XDP;
12     output CA, CB, CC, CD, CE, CF, CG;
13     output reg DP;
14     output [7:0] AN;
15
16     wire [2:0] DigitSelect;
17     wire[6:0] wireDec;
18     reg [3:0] D;
19
20     //module counter(clk100MHz, clk1KHzEn, reset_n, NumDigits, count);
21     counter u_counter(
22         .Clk100MHz(Clk100MHz),
23         .Clk1KHzEn(Clk1KHzEn),
24         .reset_n(reset_n),
25         .NumDigits(NumDigits),
26         .cnt(DigitSelect)
27         );
28
29     //module decoder_3_8(X, En, Yout);
30     decoder_3_8 u_decoder_3_8(
31         .X(DigitSelect),
32         .En(1'b1),
33         .Yout(AN)
34         );
35
36     //multiplexer with the block
37     always @ (posedge Clk100MHz)
38         begin
39             case (DigitSelect)
40                 3'b000: {DP,D} <= {XDP[0], Data[3:0]};
41                 3'b001: {DP,D} <= {XDP[1], Data[7:4]};
42                 3'b010: {DP,D} <= {XDP[2], Data[11:8]};
43                 3'b011: {DP,D} <= {XDP[3], Data[15:12]};
44                 3'b100: {DP,D} <= {XDP[4], Data[19:16]};
45                 3'b101: {DP,D} <= {XDP[5], Data[23:20]};
46                 3'b110: {DP,D} <= {XDP[6], Data[27:24]};
47                 3'b111: {DP,D} <= {XDP[7], Data[31:28]};
48             endcase
49         end
50
51     seven_segment_decoder u_seven_segment_decoder(
52         .X(D),
53         .Dec(wireDec)
54         );
55 endmodule
```

seven_segment_decoder.v

C:/Users/CalvinT/Desktop/ECE3300/Lab4/Lab4.srcs/sources_1/new/seven_segment_decoder.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Module: seven_segment_decoder
4  // Engineer: Calvin Truong and Tony Lee
5  //
6
7  module seven_segment_decoder(X, Dec);
8      input [3:0] X;
9      output reg [6:0] Dec;
10
11     /*
12     output wire CA,CB,CC,CD,CE,CF,CG;
13     assign CA = Dec[6];
14     assign CB = Dec[5];
15     assign CC = Dec[4];
16     assign CD = Dec[3];
17     assign CE = Dec[2];
18     assign CF = Dec[1];
19     assign CG = Dec[0];*/
20
21     always @(X)
22         begin
23             case (X)
24                 0: Dec = 7'b0000001;
25                 1: Dec = 7'b1001111;
26                 2: Dec = 7'b0010010;
27                 3: Dec = 7'b0000110;
28                 4: Dec = 7'b1001100;
29                 5: Dec = 7'b0100100;
30                 6: Dec = 7'b0100000;
31                 7: Dec = 7'b0001111;
32                 8: Dec = 7'b0000000;
33                 9: Dec = 7'b0000100;
34                 10: Dec = 7'b0001000;
35                 11: Dec = 7'b1100000;
36                 12: Dec = 7'b0110001;
37                 13: Dec = 7'b1000010;
38                 14: Dec = 7'b0110000;
39                 15: Dec = 7'b0111000;
40                 default: Dec = 7'b1111111;
41             endcase
42         end
43
44 endmodule
```

decoder_3_8.v

C:/Users/CalvinT/Desktop/ECE3300/Lab4/Lab4.srcs/sources_1/new/decoder_3_8.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Module: decoder_3_8
4  // Engineer: Calvin Truong and Tony Lee
5  //
6
7  module decoder_3_8(X, En, Yout);
8      input [2:0] X;
9      input En;
10     output reg [7:0] Yout;
11
12     always @(X, En)
13         if (~En)
14             Yout = 8'b1111_1111;
15         else
16             case (X)
17                 0: Yout = 8'b1111_1110;
18                 1: Yout = 8'b1111_1101;
19                 2: Yout = 8'b1111_1011;
20                 3: Yout = 8'b1111_0111;
21                 4: Yout = 8'b1110_1111;
22                 5: Yout = 8'b1101_1111;
23                 6: Yout = 8'b1011_1111;
24                 7: Yout = 8'b0111_1111;
25                 default: Yout = 8'b1111_1111;
26             endcase
27 endmodule
```

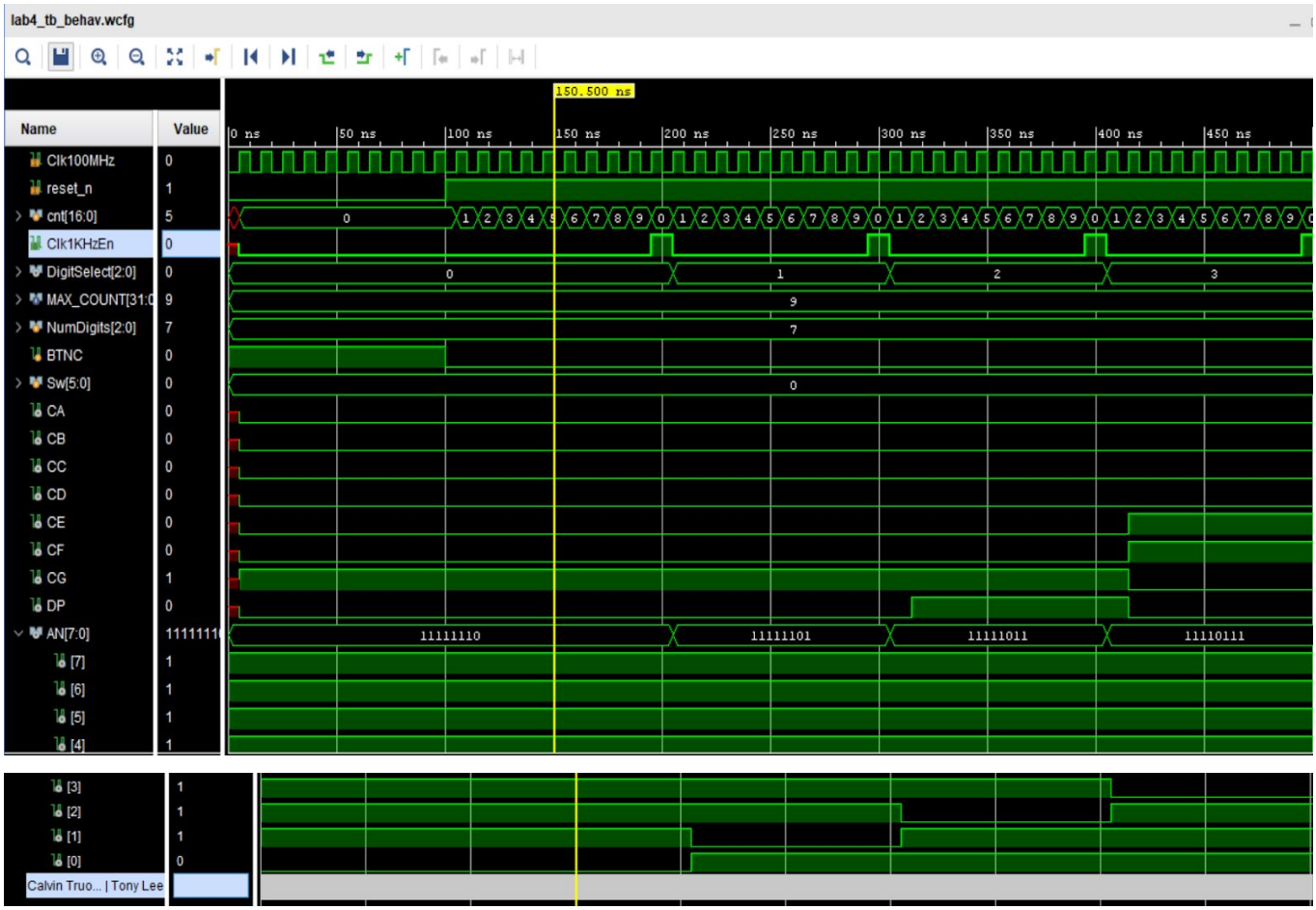
Testbench:

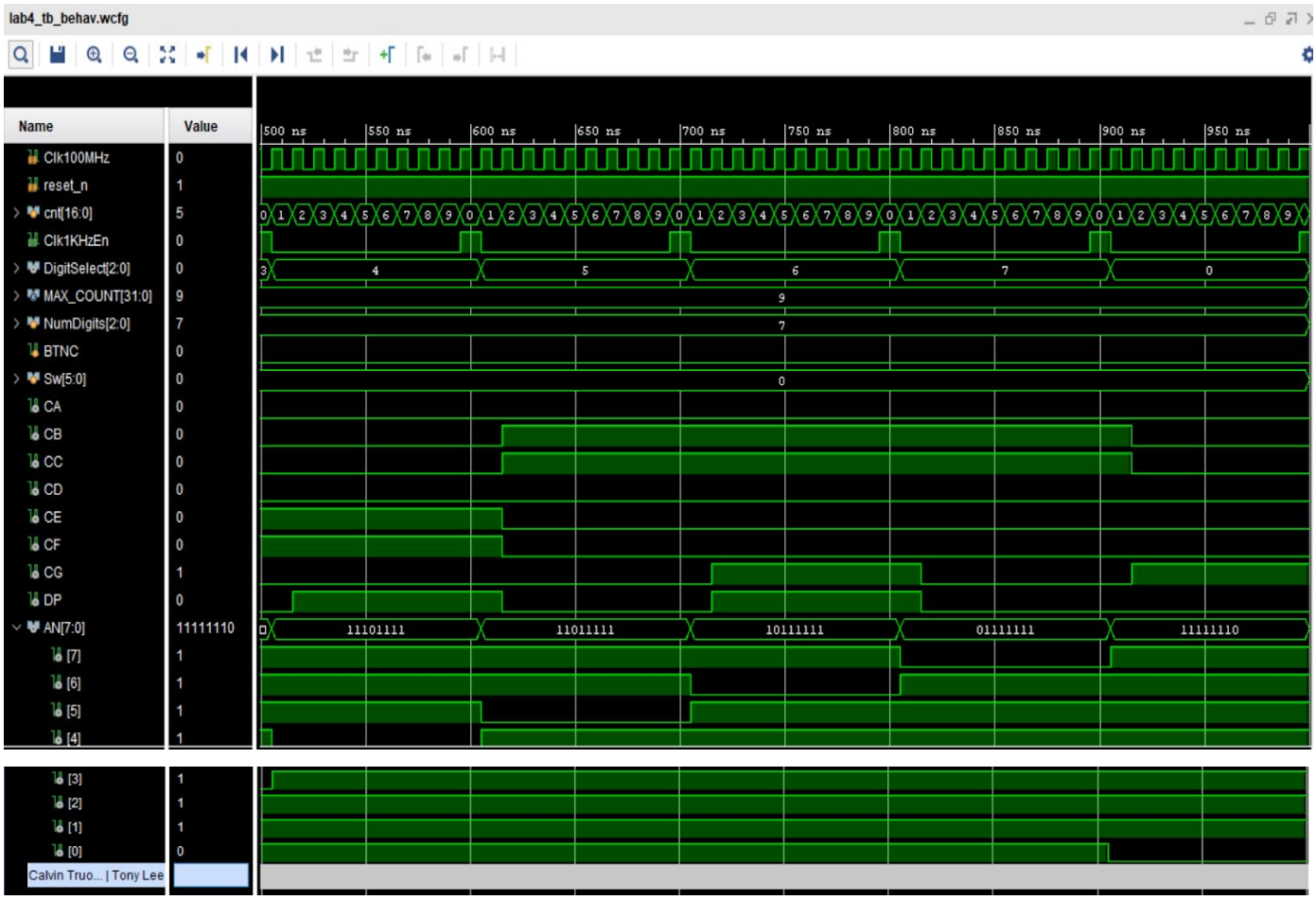
lab4_tb.v

C:/Users/CalvinT/Desktop/ECE3300/Lab4/Lab4.srcs/sim_1/new/lab4_tb.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Module: lab4_tb
4  // Engineer: Calvin Truong and Tony Lee
5  //
6
7  module lab4_tb();
8      reg Clk100MHz, BTNC;
9      reg [2:0] NumDigits;
10     reg [5:0] Sw;
11
12     wire CA,CB,CC,CD,CE,CF,CG,DP;
13     wire [7:0] AN;
14
15     lab4_top u_lab4_top (
16        .Clk100MHz(Clk100MHz),
17        .NumDigits(NumDigits),
18        .Sw(Sw),
19        .BTNC(BTNC),
20        .CA(CA),
21        .CB(CB),
22        .CC(CC),
23        .CD(CD),
24        .CE(CE),
25        .CF(CF),
26        .CG(CG),
27        .DP(DP),
28        .AN(AN)
29     );
30     initial Clk100MHz = 0;
31     always #5 Clk100MHz = ~Clk100MHz;
32
33     initial  begin
34        BTNC = 1;
35        NumDigits = 7;
36        Sw = 0;
37        #100;
38        BTNC = 0;
39        #10000;
40        $finish;
41     end
42
43 endmodule
```

Simulation:


lab4_tb_behav.wcfg
150.500 ns
Name
Value
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
400 ns
450 ns
Clk100MHz
reset_n
cnt[16:0]
Clk1KHzEn
DigitSelect[2:0]
MAX_COUNT[31:0]
NumDigits[2:0]
BTNC
Sw[5:0]
CA
CB
CC
CD
CE
CF
CG
DP
AN[7:0]
[7]
[6]
[5]
[4]
[3]
[2]
[1]
[0]
11111110
11111101
11111011
11110111
Calvin Truo... | Tony Lee

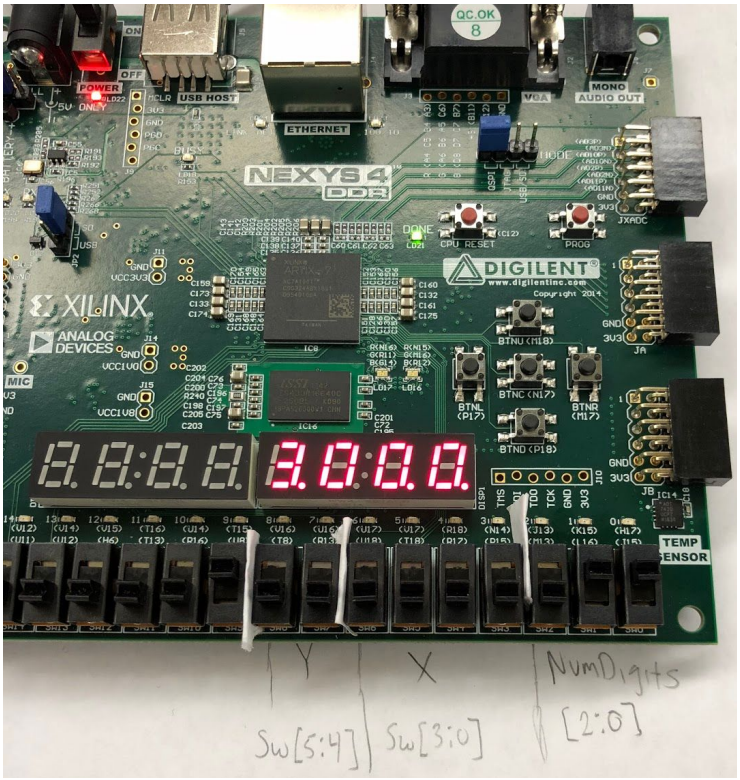

lab4_tb_behav.wcfg
Name
Value
Clk100MHz
0
reset_n
1
cnt[16:0]
5
Clk1KHzEn
0
DigitSelect[2:0]
0
MAX_COUNT[31:0]
9
NumDigits[2:0]
7
BTNC
0
Sw[5:0]
0
CA
0
CB
0
CC
0
CD
0
CE
0
CF
0
CG
1
DP
0
AN[7:0]
11111110
[7]
1
[6]
1
[5]
1
[4]
1
[3]
1
[2]
1
[1]
1
[0]
0
Calvin Truo... | Tony Lee
500 ns
550 ns
600 ns
650 ns
700 ns
750 ns
800 ns
850 ns
900 ns
950 ns
11101111
11011111
10111111
01111111
11111110

Constraint:

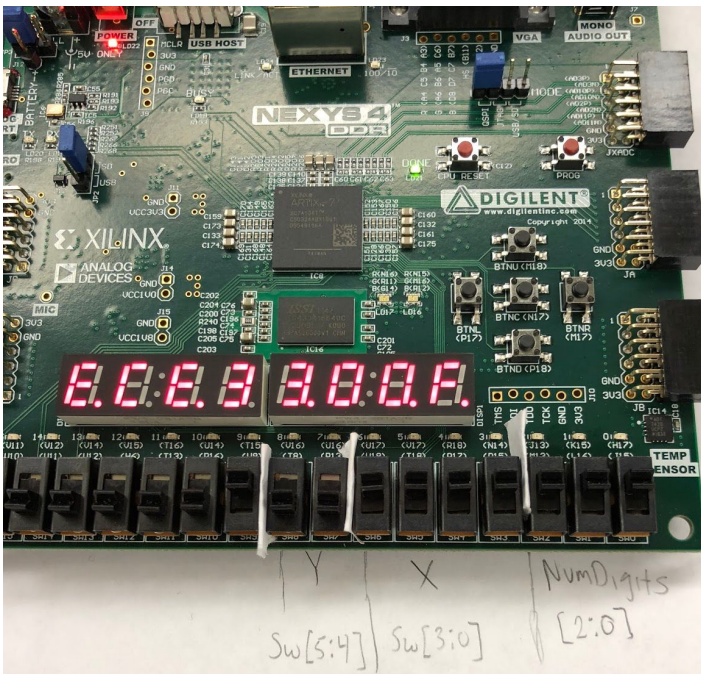
```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { Clk100MHz }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {Clk100MHz}];


##Switches

set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { NumDigits[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { NumDigits[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { NumDigits[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { Sw[0] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { Sw[1] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { Sw[2] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { Sw[3] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { Sw[4] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { Sw[5] }]; #IO_L24N_T3_34 Sch=sw[8]

set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { CA }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { CB }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { CC }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { CD }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { CE }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { CF }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { CG }]; #IO_L4P_T0_D04_14 Sch=cg

set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { DP }]; #IO_L19N_T3_A21_VREF_15 Sch=dp

set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { AN[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { AN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { AN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { AN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { AN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { AN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { AN[6] }]; #IO_L23P_T3_35 Sch=an[6]
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { AN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]


##Buttons

#set_property -dict { PACKAGE_PIN C12   IOSTANDARD LVCMOS33 } [get_ports { CPU_RESETN }]; #IO_L3P_T0_DQS_AD1P_15 Sch=cpu_resetn

set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { BTNC }]; #IO_L9P_T1_DQS_14 Sch=btnc
```

Conclusion:

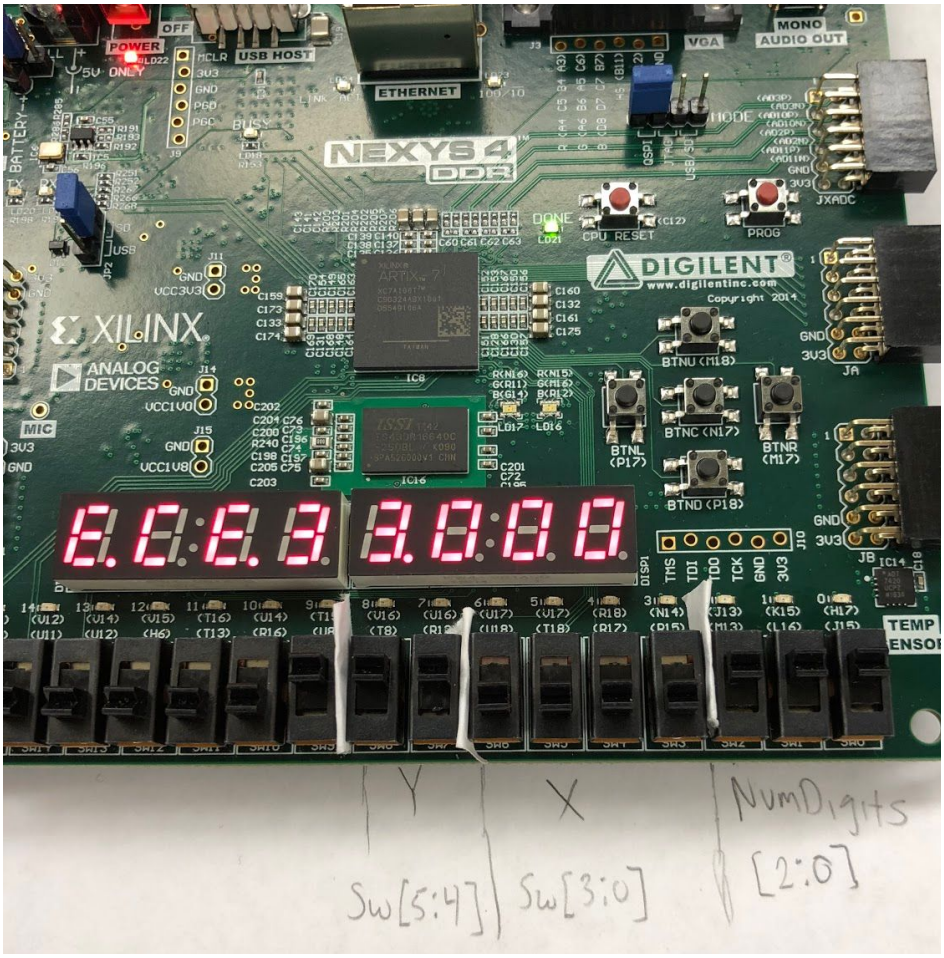
In conclusion, we were able to complete the objective with little error. We ran into one error because we mistakenly used a 1KHz clock in place of a 100MHz clock. We fixed that issue and ran into another error. In our code, our if-else statements were not nested properly, causing issues on the FPGA board. Once we fixed that one last issue, everything ran up to specifications.



Y = 0, X = 0, NumDigits = 3



Y = 0, X = 15 (F), NumDigits = 7



Y = 3, X = 0, NumDigits = 7