Seven-Segment Display on An FPGA

Calvin Reese
cjreese@fortlewis.edu

2/20/22

1 Introduction

This Lab, we started to make our own code to complete various simple tasks

2 Materials and Methods

The tutorial for making these examples are in `http://www.yilectronics.com/Courses/CE433_Labs/s2022/Lab3_MoreFPGA/Lab3.html`

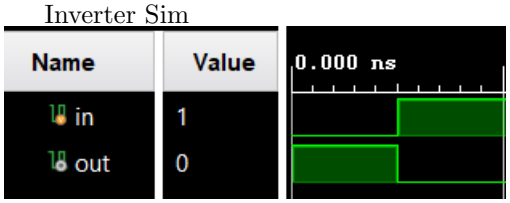
3 Results

3.1 Task 1

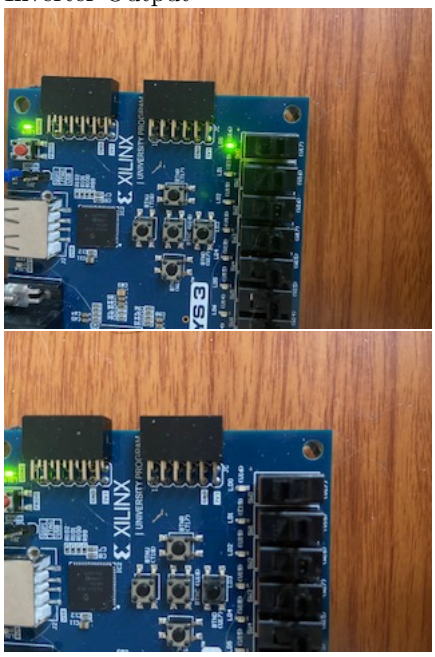
Inverter Code

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
   //////
3
4  module inverter(sw,led);
5  input [0:0]sw;
6  output [0:0]led;
7  assign led[0] = ~sw[0];
8  endmodule
9
10 module inverter_tb;
11 reg in;
12 wire out;
13
14 inverter UUT(.sw(in),.led(out));
15
16 initial begin
17 in = 1'b0;
18 #100;
19 in = 1'b1;
20 end
21 endmodule
```

Inverter Sim



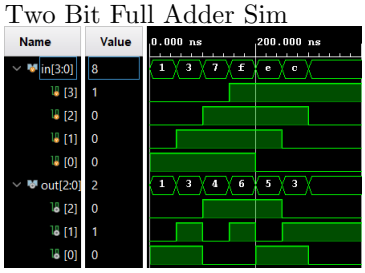
Inverter Output



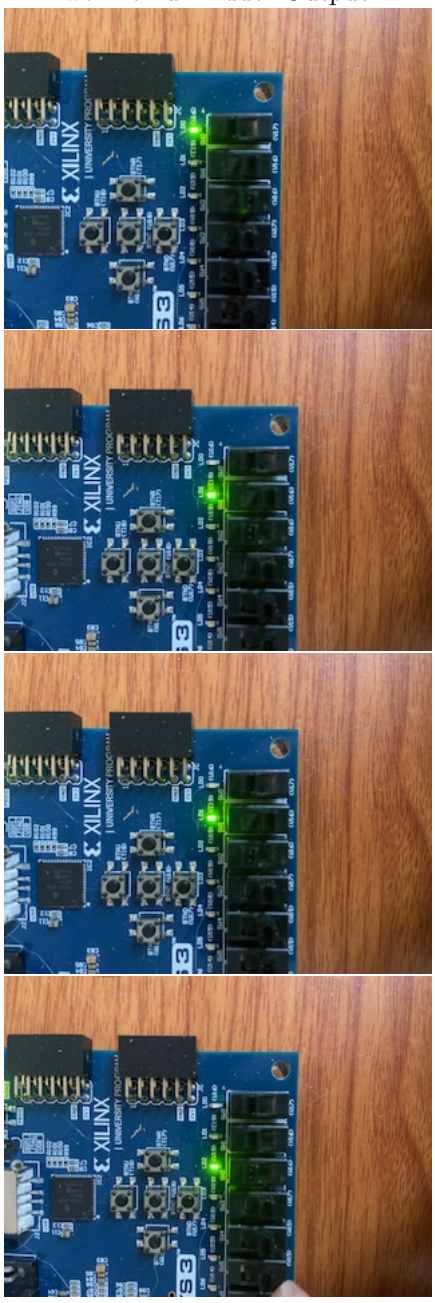
Two Bit Full Adder Code

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3
4  module twoBitFullAdder(sw,led);
5  input [3:0]sw;
6  output [2:0]led;
7  assign led[0] = (sw[0]^sw[2]);
8  assign led[1] = (~(sw[0]&sw[2])&(sw[1]^sw[3]))|((sw[0]&sw[2])&~(sw[1]^sw[3]));
9  assign led[2] = (sw[1]&sw[3])|((sw[0]&sw[2])&(sw[1]^sw[3]));
10 endmodule
11
12 module twoBitFullAdder_tb;
13 reg [3:0]in;
14 wire [2:0]out;
15
16 twoBitFullAdder UUT(.sw(in),.led(out));
17
18 initial begin
19 in = 4'b0001;
20 #50;
21 in = 4'b0011;
22 #50;
23 in = 4'b0111;
24 #50;
25 in = 4'b1111;
26 #50;
27 in = 4'b1110;
28 #50;
29 in = 4'b1100;
30 #50;
31 in = 4'b1000;
32 end
33 endmodule
```

Two Bit Full Adder Sim



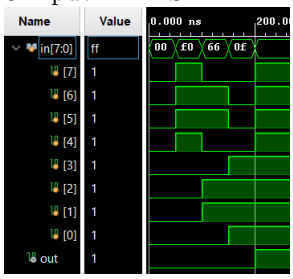
Two Bit Full Adder Output



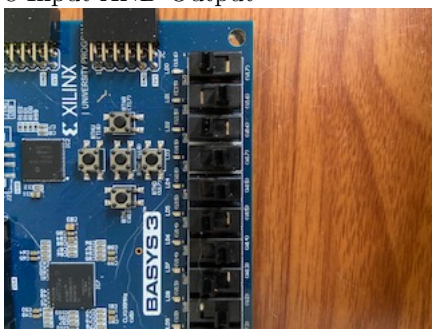
8 Input AND Code

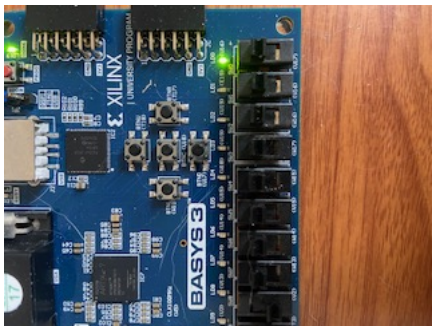
```
23 module eightAND(sw,led);
24 input [7:0]sw;
25 output [0:0] led;
26 assign led[0] = sw[0]&sw[1]&sw[2]&sw[3]&sw[4]&sw[5]&sw[6]&sw[7];
27 endmodule
28
29 module eightAND_tb;
30 reg [7:0] in;
31 wire out;
32 eightAND UUT(.sw(in),.led(out));
33 initial begin
34 in = 2'h00;
35 #50;
36 in = 2'hF0;
37 #50;
38 in = 2'h66;
39 #50;
40 in = 2'h0F;
41 #50;
42 in = 2'hFF;
43 #50;
44 end
45 endmodule
```

8 Input AND Sim



8 Input AND Output

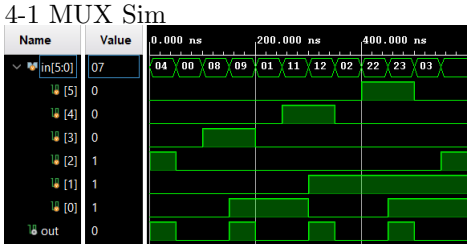




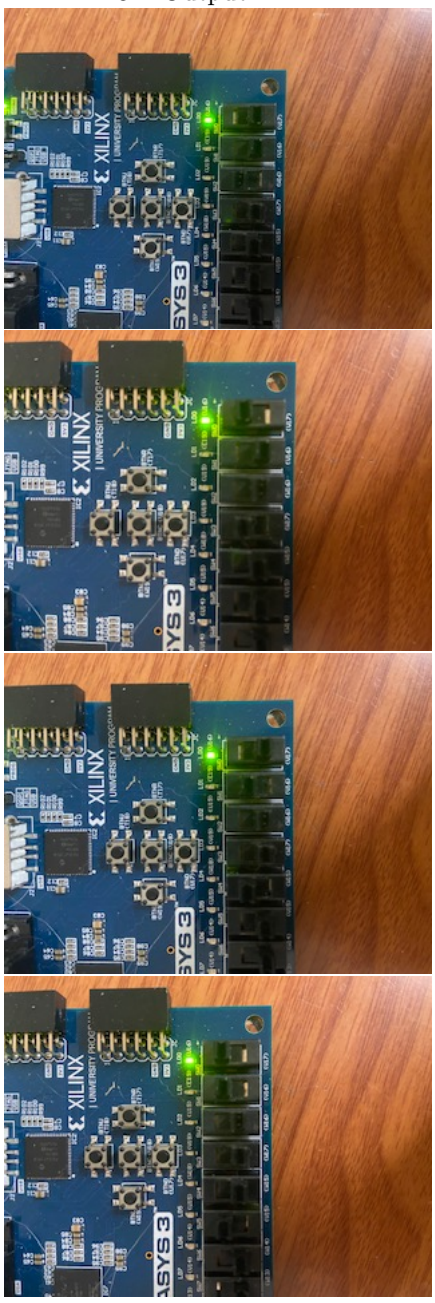
4-1 MUX Code

```
23 module fourtooneMUX(sw,led);
24 input [5:0] sw;
25 output [0:0] led;
26 assign led[0] = (~sw[0]&~sw[1]&sw[2])|(sw[0]&~sw[1]&sw[3])|(~sw[0]&sw[1]&sw[4])|(sw[0]&sw[1]&sw[5]);
27 endmodule
28
29 module fourtooneMUX_tb;
30 reg [5:0] in;
31 wire out;
32 fourtooneMUX UUT(.sw(in),.led(out));
33 initial begin
34 in = 6'b000100;
35 #50;
36 in = 6'b000000;
37 #50;
38 in = 6'b001000;
39 #50;
40 in = 6'b001001;
41 #50;
42 in = 6'b000001;
43 #50;
44 in = 6'b010001;
45 #50;
46 in = 6'b010010;
47 #50;
48 in = 6'b000010;
49 #50;
50 in = 6'b100010;
51 #50;
52 in = 6'b100011;
53 #50;
54 in = 6'b000011;
55 #50;
56 in = 6'b000111;
57 end
58 endmodule
```

4-1 MUX Sim



4-1 MUX Output



3.2 Task 2

Running LED Code

```
23 module runningLED(clk,sw,led);
24 input clk;
25 input[1:0] sw;
26 output reg [3:0] led = 4'b0001;
27 parameter cntmax = 32'd100000000;
28 reg [31:0] cnt;
29 always @(posedge clk or negedge sw[0])
30 if(~sw[0])
31 begin
32     cnt <=32'd0;
33     led<=4'b0001;
34 end
35 else if (cnt==cntmax)
36 begin
37     cnt<=32'd0;
38     led<={led[2:0],led[3]};
39 end
40 else
41 begin
42     cnt<=cnt+32'd1;
43     led=led;
44 end
45 endmodule
```

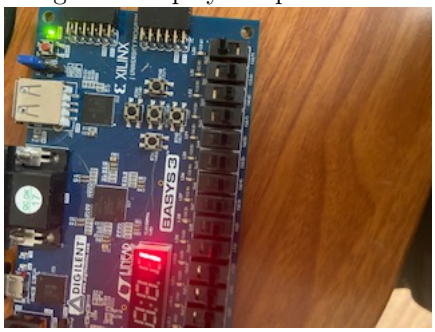
Running LED Output: `https://youtu.be/Sh14Sl2qcLU`

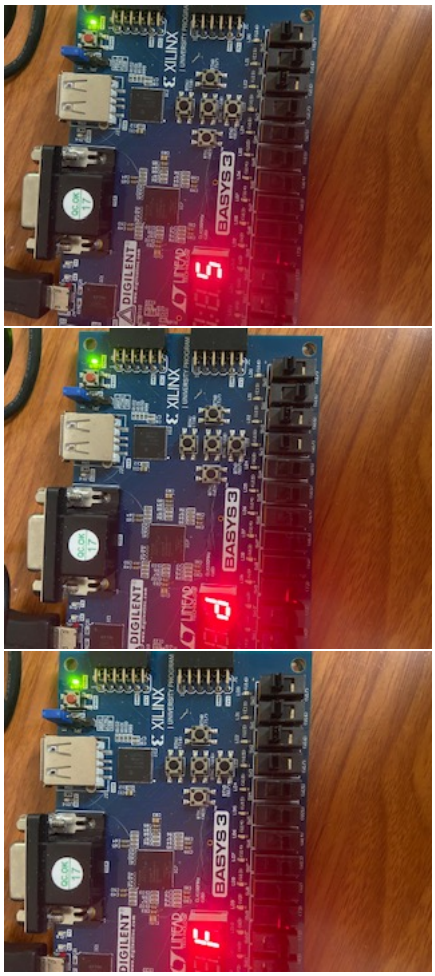
3.3 Task 3

7 Segment Display Code

```
21 module decoder_7seg(in1,out1);
22 input[3:0] in1;
23 output reg [6:0]out1;
24 always @(in1)
25 case(in1)
26 4'b0000 : out1=7'b1000000;
27 4'b0001 : out1=7'b1111001;
28 4'b0010 : out1=7'b0100100;
29 4'b0011 : out1=7'b0110000;
30 4'b0100 : out1=7'b0110001;
31 4'b0101 : out1=7'b0010010;
32 4'b0110 : out1=7'b0000010;
33 4'b0111 : out1=7'b1111000;
34 4'b1000 : out1=7'b0000000;
35 4'b1001 : out1=7'b0010000;
36 4'b1010 : out1=7'b0001000;
37 4'b1011 : out1=7'b0000011;
38 4'b1100 : out1=7'b1000110;
39 4'b1101 : out1=7'b0100001;
40 4'b1110 : out1=7'b0000110;
41 4'b1111 : out1=7'b0001110;
42 endcase
43 endmodule
44
45 module sevenseg(sw,seg,an);
46 input [3:0] sw;
47 output [6:0] seg;
48 output [3:0] an;
49
50
51 assign an =4'b1110;
52 decoder_7seg ss(sw,seg);
53 endmodule
```

7 Segment Display Output





4 Discussion

All of this was fairly simple just to follow your tutorial. All the outputs works exactly as intended, and only had some small issues getting the compiler to like my reset switch for the running LED