Combinational Logic Blocks

Calvin Reese
cjreese@fortlewis.edu

2/16/22

1 Introduction

This HW, we worked on designing out own tests for logic.

2 Materials and Methods

The tutorial for making these examples are in `http://www.yilectronics.com/Courses/CE433/s2022/lectures/week3_combinationalBlocks/week3_combinationalBlocks.html`

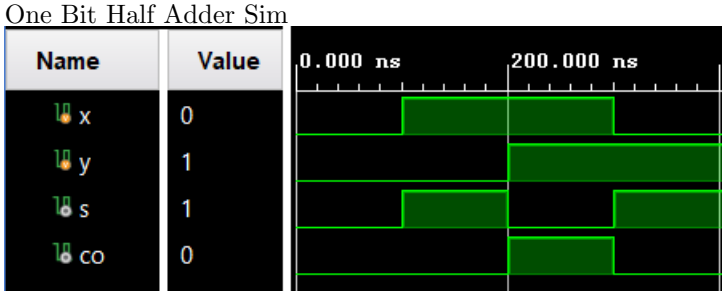
3 Results

3.1 Task 1

One Bit Half Adder Code

```
23 module one_bit_half_adder(s,co,x,y);
24 input x,y;
25 output s,co;
26
27 assign co=x&y;
28 assign s=x^y;
29
30 endmodule
31
32 module onebithalfadder_tb;
33 reg x,y;
34 wire s,co;
35 initial begin
36     x=0;y=0;
37     #100
38     x=1;
39     #100
40     y=1;
41     #100
42     x=0;
43 end
44
45 one_bit_half_adder UUT(.x(x),.y(y),.co(co),.s(s));
46 endmodule
```

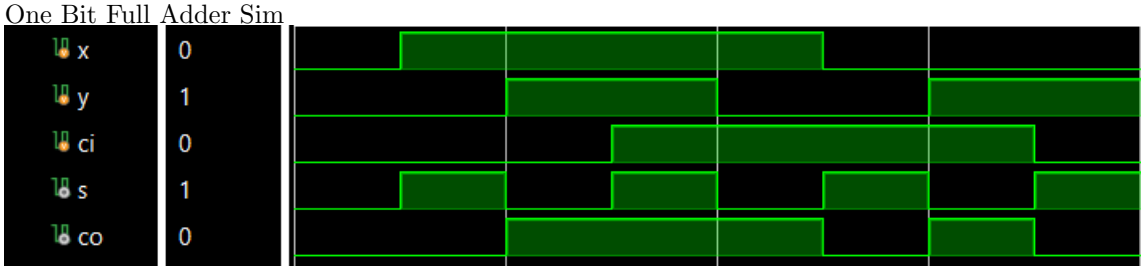
One Bit Half Adder Sim



One Bit Full Adder Code

```
23 module onebitfulladder(s,co,x,y,ci);
24 input x,y,ci;
25 output s,co;
26
27 assign co=(x&y) | (ci&(x^y));
28 assign s=x^y^ci;
29
30 endmodule
31
32 module onebithalfadder_tb;
33 reg x,y,ci;
34 wire s,co;
35 initial begin
36     x=0;y=0;ci=0;
37     #100
38     x=1;
39     #100
40     y=1;
41     #100
42     ci=1;
43     #100
44     y=0;
45     #100
46
47     x=0;
48     #100
49     y=1;
50     #100
51     ci=0;
52 end
53
54 onebitfulladder UUT(.x(x),.y(y),.co(co),.s(s),.ci(ci));
55 endmodule
```

One Bit Full Adder Sim

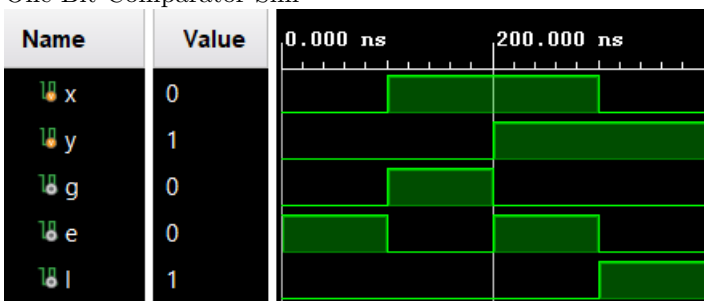


3.2 Task 2

One Bit Comparator Code

```
23 module onebitcomparitor(g,e,l,x,y);
24 input x,y;
25 output g,e,l;
26
27 assign g=x&~y;
28 assign e=~(x^y);
29 assign l=~x&y;
30 endmodule
31
32 module onebitcomparitor_tb;
33 reg x,y;
34 wire g,e,l;
35 initial begin
36     x=0;y=0;
37     #100
38     x=1;
39     #100
40     y=1;
41     #100
42     x=0;
43 end
44
45 onebitcomparitor UUT(.x(x),.y(y),.g(g),.e(e),.l(l));
46 endmodule
```

One Bit Comparator Sim

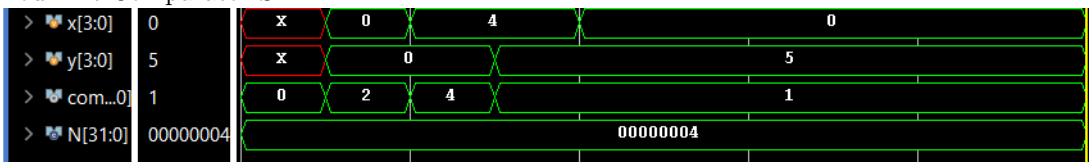


3.3 Task 3

Four Bit Comparator Code

```
23 module fourbitcomparitor(comp,x,y);
24 parameter N=4;
25
26 input [N-1:0] x,y;
27 output reg [2:0] comp;
28
29 initial
30 comp = 3'b0;
31 always @(x or y)
32 if (x > y) comp = 3'b100;
33 else if (x==y) comp = 3'b010;
34 else if  (x < y) comp = 3'b001;
35 else comp =3'b111;
36
37 endmodule
38
39 module fourbitcomparitor_tb;
40 parameter N=4;
41 reg [N-1:0] x,y;
42 wire [2:0] comp;
43 initial begin
44     #100
45     x=3'b0;y=3'b0;
46     #100
47     x=3'b100;
48     #100
49     y=3'b101;
50     #100
51     x=3'b000;
52 end
53
54 fourbitcomparitor UUT(.x(x),.y(y),.comp(comp));
55 endmodule
```

Four Bit Comparator Sim

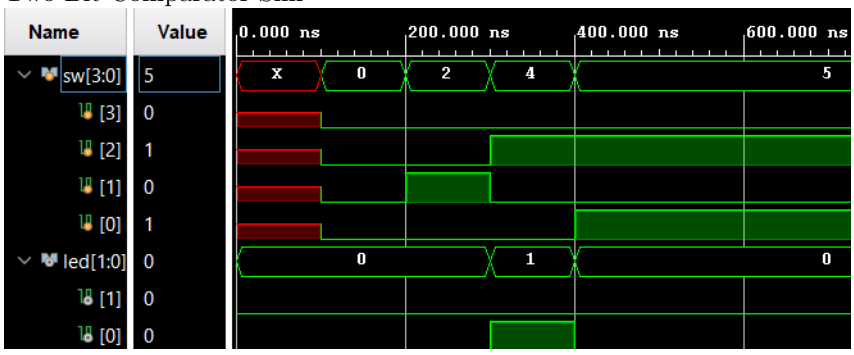


3.4 Task 4

Two Bit Comparator Code

```
23 module twobitcomparitor(comp,sw);
24 input [3:0] sw;
25 output reg [1:0] comp;
26 assign x = sw[1:0];
27 assign y = sw[3:2];
28 initial
29
30 comp = 3'b0;
31 always @(x or y)
32 if (x > y) comp = 2'b10;
33 else if (x==y) comp = 2'b00;
34 else if  (x < y) comp = 2'b01;
35 else comp =2'b11;
36
37 endmodule
38
39 module twobitcomparitor_top(sw,led);
40 input [3:0] sw;
41 output [1:0] led;
42 twobitcomparitor UUT(.sw(sw),.comp(led));
43 endmodule
44
45 module twobitcomparitor_tb;
46 reg [3:0] sw;
47 wire [1:0] led;
48 initial begin
49     #100
50     sw=4'b0000;
51     #100
52     sw=4'b0010;
53     #100
54     sw=4'b0100;
55     #100
56     sw=4'b0101;
57 end
58 twobitcomparitor UUT(.sw(sw),.comp(led));
59
60 endmodule
```

Two Bit Comparator Sim

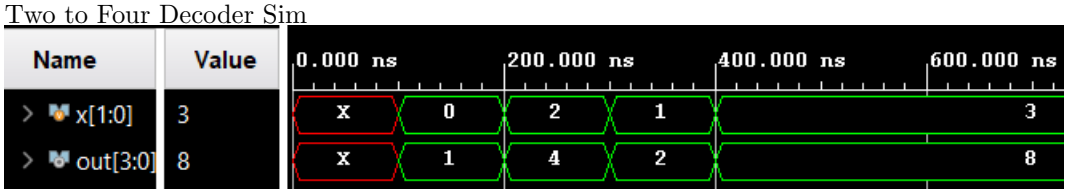


3.5 Task 5

Two to Four Decoder Code

```
23 module twotofourdecoder(out,x);
24 input [1:0] x;
25 output [3:0] out;
26
27 assign out[0] = ~x[0] &~x[1];
28 assign out[1] = x[0] & ~x[1];
29 assign out[2] = ~x[0] & x[1];
30 assign out[3] = x[0] & x[1];
31
32 endmodule
33
34 module twotofourdecoder_tb;
35 reg [1:0] x;
36 wire [3:0] out;
37 initial begin
38     #100
39     x=2'b0;
40     #100
41     x=2'b10;
42     #100
43     x=2'b01;
44     #100
45     x=2'b11;
46 end
47
48 twotofourdecoder UUT(.x(x),.out(out));
49 endmodule
```

Two to Four Decoder Sim

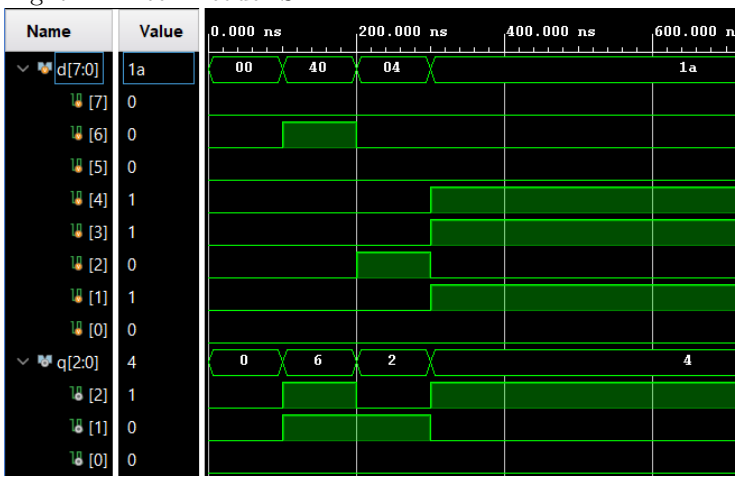


3.6 Task 6

Eight x Three Encoder Code

```
23 module eightxthreeencoder(d,q);
24 input [7:0] d;
25 output [2:0] q;
26 assign q[0] = ~d[6]&(~d[4]&~d[2]&d[1]|~d[4]&d[3]|d[5])|d[7];
27 assign q[1] = ((~d[7]&~d[6]&~d[5]&~d[4])&(d[3]|d[2]))|d[6]|d[7];
28 assign q[2] = d[7]|d[6]|d[5]|d[4];
29 endmodule
30
31
32 module eightxthreeencoder_tb;
33
34 reg [7:0] d;
35 wire [2:0] q;
36 initial begin
37     d=8'b00000000;
38     #100
39     d=8'b01000000;
40     #100
41     d=8'b00000100;
42     #100
43     d=8'b00011010;
44 end
45
46 eightxthreeencoder UUT(.d(d),.q(q));
47 endmodule
```

Eight x Three Encoder Sim



3.7 Task 7

Eight x Three Encoder Code

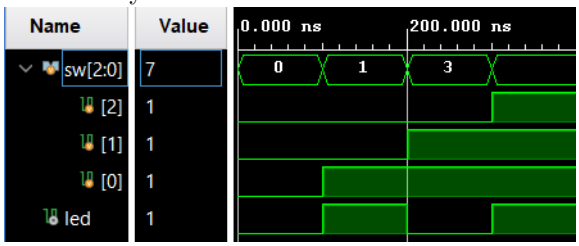
```
23 module fourtoonemultiplexer(y,x,s);
24 input [1:0]  s;
25 input [3:0] x;
26 output y;
27 assign y=(x[0]&~s[0]&~s[1])|(x[1]&s[0]&~s[1])|(x[2]&~s[0]&s[1])|(x[3]&s[0]&s[1]);
28 endmodule
29
30 module fourtoonemultiplexer_top(sw,led);
31 input [5:0] sw;
32 output [0:0] led;
33 fourtoonemultiplexer UUT(.s(sw[1:0]),.x(sw[5:2]),.y(led));
34 endmodule
```

3.8 Task 8

Even Parity Code

```
23 module threebitevenparity(a,b,c,p);
24 input a,b,c;
25 output p;
26 assign p=a^b^c;
27 endmodule
28
29 module threebitevenparity_top(sw,led);
30 input [2:0] sw;
31 output led;
32 threebitevenparity UUT(.a(sw[0]),.b(sw[1]),.c(sw[2]),.p(led));
33 endmodule
34
35 module threebitevenparity_tb;
36 reg [2:0] sw;
37 wire led;
38 initial begin
```

Even Parity Sim



3.9 All FPGA Outputs

https://youtu.be/2vE-nKPsRK4
https://youtu.be/eJyV4ZKMO5Y
https://youtu.be/m6zkERHIkyA

4 Discussion

All of this was fairly simple just to follow your tutorial. I was able to get the 2-bit Comparator to create a sim, but it didn't like the synthesis. I understand how to make simulations and upload code to the FPGA better now that I have had the chance to make the code mostly on my own.