

# Computer Organization

## Assignment 1

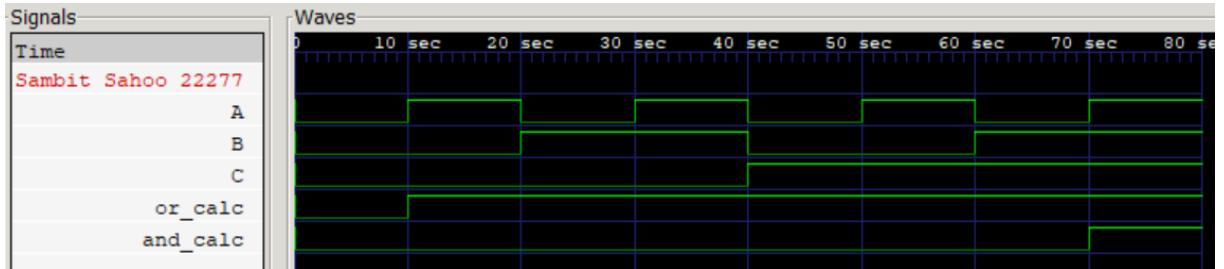
Sambit Sahoo (22277)

---

Q1 - 3-input AND gate, 3-input OR gate

```
Assignment_1 > q1 > ≡ a1.v
1  module myor (input a, input b, input c, output y);
2    wire w;
3    or (w, a, b);
4    or (y, c, w);
5  endmodule;
6
7  module myand (input a, input b, input c, output y);
8    wire w;
9    and (w, a, b);
10   and (y, c, w);
11 endmodule;
12
```

```
Assignment_1 > q1 > ≡ a1.v
12
13  module a1_tb();
14
15    reg A, B, C;
16    wire or_calc, and_calc;
17
18    myor G1(A, B, C, or_calc);
19    myand G2(A, B, C, and_calc);
20
21    initial begin
22      $dumpfile("a1_tb.vcd");
23      $dumpvars(0, a1_tb);
24
25      A = 0; B = 0; C = 0;
26      #10; A = 1; B = 0; C = 0;
27      #10; A = 0; B = 1; C = 0;
28      #10; A = 1; B = 1; C = 0;
29      #10; A = 0; B = 0; C = 1;
30      #10; A = 1; B = 0; C = 1;
31      #10; A = 0; B = 1; C = 1;
32      #10; A = 1; B = 1; C = 1;
33      #10;
34    end
```



## Q2 - 4-bit magnitude comparator

```
Assignment_1 > q2 > a2.v
1  module mycomparator (input a, input b, input c, input d, input e, input f, input g, input h, output y);
2    wire n1, n2, n3, n4, w1, w2;
3
4    xnor(n1, a, b);
5    xnor(n2, c, d);
6    xnor(n3, e, f);
7    xnor(n4, g, h);
8    and(w1, n1, n2);
9    and(w2, n3, w1);
10   and(y, n4, w2);
11
12 endmodule;
13
```

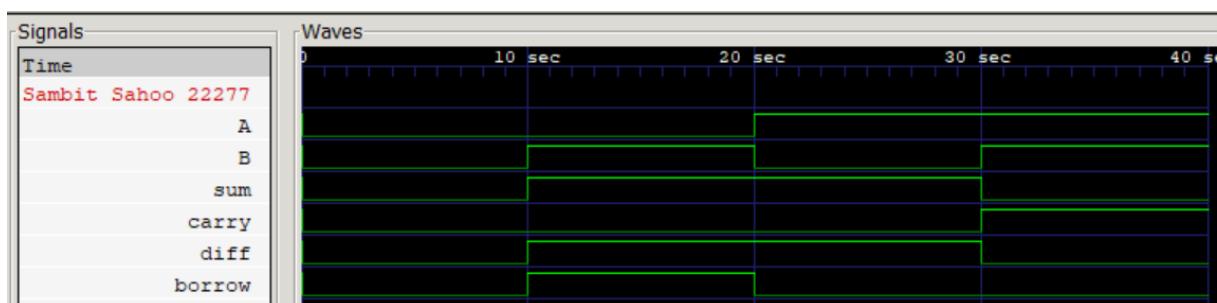
```
Assignment_1 > q2 > a2.v
13
14 module a2_tb();
15
16   reg A, B, C, D, E, F, G, H;
17   wire comp;
18
19   mycomparator G1(A, B, C, D, E, F, G, H, comp);
20
21
22   initial begin
23     $dumpfile("a2_tb.vcd");
24     $dumpvars(0, a2_tb);
25
26     A = 0; B = 0; C = 0; D = 1; E = 0; F = 0; G = 0; H = 0;
27     #10; A = 1; B = 0; C = 0; D = 0; E = 0; F = 0; G = 0; H = 0;
28     #10; A = 0; B = 1; C = 0; D = 0; E = 0; F = 0; G = 0; H = 0;
29     #10; A = 1; B = 1; C = 0; D = 0; E = 0; F = 0; G = 0; H = 0;
30     #10; A = 0; B = 0; C = 1; D = 0; E = 0; F = 0; G = 0; H = 0;
31     #10; A = 1; B = 0; C = 1; D = 1; E = 0; F = 0; G = 0; H = 0;
32     #10; A = 0; B = 1; C = 1; D = 1; E = 0; F = 0; G = 0; H = 0;
33     #10; A = 1; B = 1; C = 1; D = 1; E = 0; F = 0; G = 0; H = 0;
34     #10;
35   end
36
37 endmodule;
```



### Q3- Half adder and Half subtractor & Full adder and Full subtractor

```
Assignment_1 > q3 >  ≡ a3.v
1  module myhalfadder (input a, input b, output sum, output carry);
2      xor (sum, a, b);
3      and (carry, a, b);
4  endmodule
5
6  module myhalfsubtractor (input a, input b, output diff, output borrow);
7      wire na;
8      not (na , a);
9      xor (diff, a, b);
10     and (borrow, na, b);
11 endmodule
12
```

```
Assignment_1 > q3 >  ≡ a3.v
13 module a3_tb();
14
15     reg A, B;
16     wire sum, carry;
17     wire diff, borrow;
18     myhalfadder G1 (A, B, sum, carry);
19     myhalfsubtractor G2 (A, B, diff, borrow);
20
21     initial begin
22         $dumpfile("a3_tb.vcd");
23         $dumpvars(0, a3_tb);
24
25         A = 0; B = 0; #10;
26         A = 0; B = 1; #10;
27         A = 1; B = 0; #10;
28         A = 1; B = 1; #10;
29
30     end
31
32 endmodule
```



```

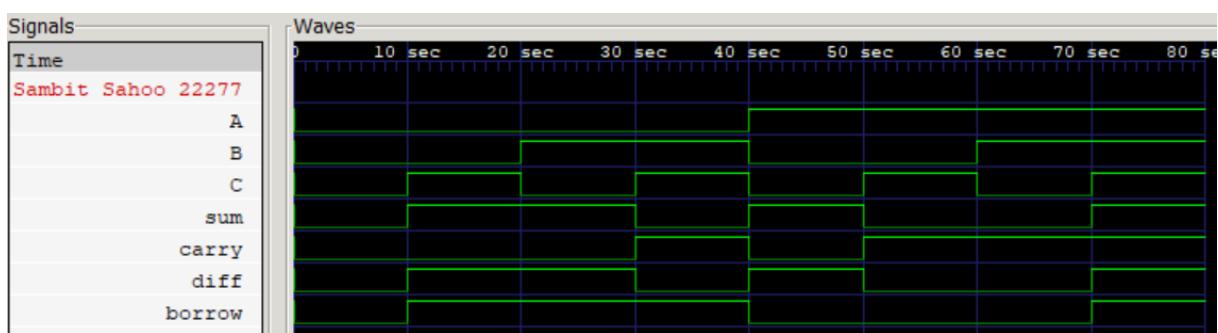
Assignment_1 > q3 > a3_FAFS.v
1  module myhalfadder (input a, input b, output sum, output carry);
2    xor (sum, a, b);
3    and (carry, a, b);
4  endmodule
5
6  module myhalfsubtractor (input a, input b, output diff, output borrow);
7    wire na;
8    not (na , a);
9    xor (diff, a, b);
10   and (borrow, na, b);
11 endmodule
12
13 module myfulladder (input a, input b, input c, output sum, output carry);
14   wire w1, w2, w3;
15   myhalfadder HA1 (.a(a), .b(b), .sum(w2), .carry(w1));
16   myhalfadder HA2 (.a(w2), .b(c), .sum(sum), .carry(w3));
17   or(carry, w1, w3);
18 endmodule
19
20 module myfullsubtractor (input a, input b, input c, output diff, output borrow);
21   wire w1, w2, w3;
22   myhalfsubtractor HS1 (.a(a), .b(b), .diff(w2), .borrow(w1));
23   myhalfsubtractor HS2 (.a(w2), .b(c), .diff(diff), .borrow(w3));
24   or(borrow, w1, w3);
25 endmodule
26

```

```

Assignment_1 > q3 > a3_FAFS.v
26
27 module a3_FAFS_tb();
28   reg A, B, C;
29   wire sum, carry;
30   wire diff, borrow;
31   myfulladder FA1 (A, B, C, sum, carry);
32   myfullsubtractor FS2 (A, B, C, diff, borrow);
33
34   initial begin
35     $dumpfile("a3_FAFS_tb.vcd");
36     $dumpvars(0, a3_FAFS_tb);
37
38     A = 0; B = 0; C = 0; #10;
39     A = 0; B = 0; C = 1; #10;
40     A = 0; B = 1; C = 0; #10;
41     A = 0; B = 1; C = 1; #10;
42     A = 1; B = 0; C = 0; #10;
43     A = 1; B = 0; C = 1; #10;
44     A = 1; B = 1; C = 0; #10;
45     A = 1; B = 1; C = 1; #10;
46   end
47 endmodule

```



## Q4 - 4-to-1 multiplexer and 3-to-8 decoder

```
Assignment_1 > q4 > ≡ a4_decoder.v
1  module my3inputand (input a, input b, input c, output y);
2    wire w;
3    and(w, a, b);
4    and(y, c, w);|
5  endmodule
6
7  module mydecoder3to8 (input a, input b, input c, output s, output t, output u, output v, output w, output x, output y, output z);
8    wire na, nb, nc;
9    not(na, a);
10   not(nb, b);
11   not(nc, c);
12   my3inputand and1 (.a(na), .b(nb), .c(nc), .y(s));
13   my3inputand and2 (.a(na), .b(nb), .c(c), .y(t));
14   my3inputand and3 (.a(na), .b(b), .c(nc), .y(u));
15   my3inputand and4 (.a(na), .b(b), .c(c), .y(v));
16   my3inputand and5 (.a(a), .b(nb), .c(nc), .y(w));
17   my3inputand and6 (.a(a), .b(nb), .c(c), .y(x));
18   my3inputand and7 (.a(a), .b(b), .c(nc), .y(y));
19   my3inputand and8 (.a(a), .b(b), .c(c), .y(z));
20 endmodule
21
```

```
Assignment_1 > q4 > ≡ a4_decoder_tb.v
1
2  module a4_decoder_tb();
3
4    reg A, B, C;
5    wire D, E, F, G, H, I, J, K;
6
7    mydecoder3to8 D1(A, B, C, D, E, F, G, H, I, J, K);
8
9
10   initial begin
11     $dumpfile("a4_decoder_tb.vcd");
12     $dumpvars(0, a4_decoder_tb);
13
14     A = 0; B = 0; C = 0;
15     #10; A = 0; B = 0; C = 1;
16     #10; A = 0; B = 1; C = 0;
17     #10; A = 0; B = 1; C = 1;
18     #10; A = 1; B = 0; C = 0;
19     #10; A = 1; B = 0; C = 1;
20     #10; A = 1; B = 1; C = 0;
21     #10; A = 1; B = 1; C = 1;
22     #10;
23   end
24
25 endmodule
26
```



Assignment\_1 > q4 > a4\_mux.v

```

1  module mymux2to1 (input a, input b, input s, output y);
2    wire w1, w2, ns;
3    not(ns, s);
4    and(w1, a, ns);
5    and(w2, b, s);
6    or(y, w1, w2);
7  endmodule
8
9  module mymux4to1 (input a, input b, input c, input d, input s, input t, output y);
10   wire w3, w4;
11   mymux2to1 M1 (.a(a), .b(b), .s(t), .y(w3));
12   mymux2to1 M2 (.a(c), .b(d), .s(t), .y(w4));
13   mymux2to1 M3 (.a(w3), .b(w4), .s(s), .y(y));
14 endmodule
15

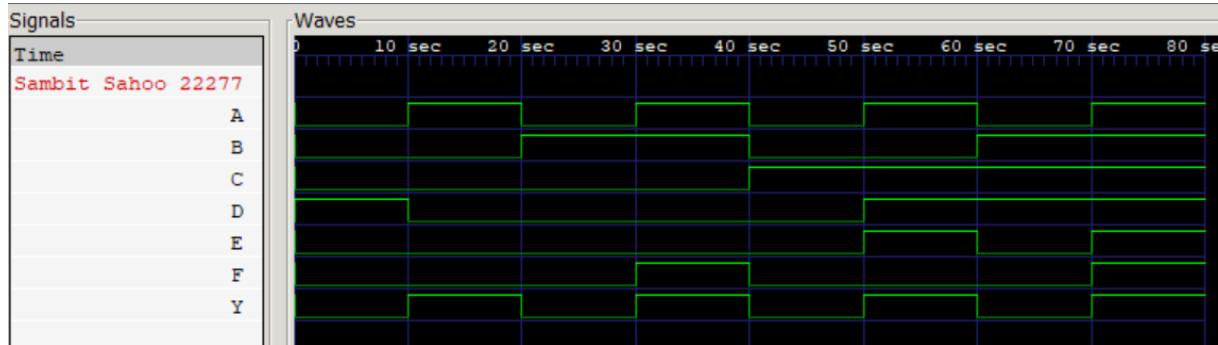
```

Assignment\_1 > q4 > a4\_mux.v

```

16  module a4_mux_tb();
17
18    reg A, B, C, D, E, F;
19    wire Y;
20
21    mymux4to1 G1(A, B, C, D, E, F, Y);
22
23
24    initial begin
25      $dumpfile("a4_mux_tb.vcd");
26      $dumpvars(0, a4_mux_tb);
27
28      A = 0; B = 0; C = 0; D = 1; E = 0; F = 0;
29      #10; A = 1; B = 0; C = 0; D = 0; E = 0; F = 0;
30      #10; A = 0; B = 1; C = 0; D = 0; E = 0; F = 0;
31      #10; A = 1; B = 1; C = 0; D = 0; E = 0; F = 1;
32      #10; A = 0; B = 0; C = 1; D = 0; E = 0; F = 0;
33      #10; A = 1; B = 0; C = 1; D = 1; E = 1; F = 0;
34      #10; A = 0; B = 1; C = 1; D = 1; E = 0; F = 0;
35      #10; A = 1; B = 1; C = 1; D = 1; E = 1; F = 1;
36      #10;
37    end
38
39  endmodule;

```



### Q5 - 4-bit barrel shifter with (1-bit and 2-bit) left and right shift

```

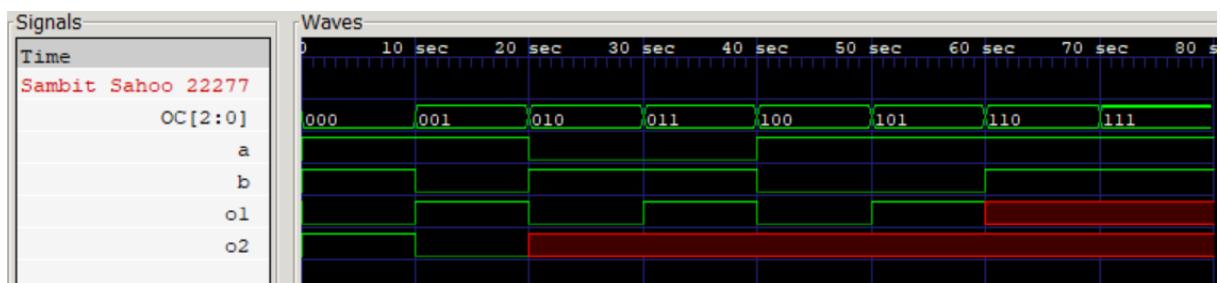
Assignment_1 > q5 > ≡ a5.v
1  module my1BLshifter (input A, input B, input C, input D, output a, output b, output c, output d);
2    buf(a, B);
3    buf(b, C);
4    buf(c, D);
5    buf(d, A);
6  endmodule
7
8  module my1BRshifter (input A, input B, input C, input D, output e, output f, output g, output h);
9    buf(e, D);
10   buf(f, A);
11   buf(g, B);
12   buf(h, C);
13 endmodule
14
15 module my2BLshifter (input A, input B, input C, input D, output i, output j, output k, output l);
16   buf(i, C);
17   buf(j, D);
18   buf(k, A);
19   buf(l, B);
20 endmodule
21
22 module my2BRshifter (input A, input B, input C, input D, output m, output n, output o, output p);
23   buf(m, C);
24   buf(n, D);
25   buf(o, A);
26   buf(p, B);
27 endmodule
28

```

```

Assignment_1 > q5 > a5.v
29 module a5_tb();
30   reg A,B,C,D;
31   wire a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p;
32
33   my1BLshifter BS1 (A, B, C, D, a, b, c, d);
34   my1BRshifter BS2 (A, B, C, D, e, f, g, h);
35   my2BLshifter BS3 (A, B, C, D, i, j, k, l);
36   my2BRshifter BS4 (A, B, C, D, m, n, o, p);
37
38   initial begin
39     $dumpfile("a5_tb.vcd");
40     $dumpvars(0, a5_tb);
41
42     A=0; B=0; C=0; D=0; #10;
43     A=0; B=0; C=0; D=1; #10;
44     A=0; B=0; C=1; D=0; #10;
45     A=0; B=0; C=1; D=1; #10;
46
47     A=0; B=1; C=0; D=0; #10;
48     A=0; B=1; C=0; D=1; #10;
49     A=0; B=1; C=1; D=0; #10;
50     A=0; B=1; C=1; D=1; #10;
51
52     A=1; B=0; C=0; D=0; #10;
53     A=1; B=0; C=0; D=1; #10;
54     A=1; B=0; C=1; D=0; #10;
55     A=1; B=0; C=1; D=1; #10;
56
57     A=1; B=1; C=0; D=0; #10;
58     A=1; B=1; C=0; D=1; #10;
59     A=1; B=1; C=1; D=0; #10;
60     A=1; B=1; C=1; D=1; #10;
61
62   end
63 endmodule

```



## Q6 - Single-bit ALU

```
Assignment_1 > q6 > ≡ a6.v
1  module mymux2to1 (input a, input b, input s, output y);
2    wire w1, w2, s;
3    not(ns, s);
4    and(w1, a, ns);
5    and(w2, b, s);
6    or(y, w1, w2);
7  endmodule
8
9  module mymux4to1 (input a, input b, input c, input d, input s, input t, output y);
10   wire w1, w2;
11   mymux2to1 M1 (.a(a), .b(b), .s(t), .y(w1));
12   mymux2to1 M2 (.a(c), .b(d), .s(t), .y(w2));
13   mymux2to1 M3 (.a(w1), .b(w2), .s(s), .y(y));
14 endmodule
15
16 module mymux8to1 (input a, input b, input c, input d, input e, input f, input g, input h, input [2:0] s, output y);
17   wire w1, w2;
18   mymux4to1 M1 (.a(a), .b(b), .c(c), .d(d), .s(s[1]), .t(s[0]), .y(w1));
19   mymux4to1 M2 (.a(e), .b(f), .c(g), .d(h), .s(s[1]), .t(s[0]), .y(w2));
20   mymux2to1 M3 (.a(w1), .b(w2), .s(s[2]), .y(y));
21 endmodule
22
23 module myhalfadder (input a, input b, output sum, output carry);
24   xor(sum, a, b);
25   and(carry, a, b);
26 endmodule
```

```
Assignment_1 > q6 > ≡ a6.v
27
28 module myhalfsubtractor (input a, input b, output diff, output borrow);
29   wire na;
30   not (na, a);
31   xor(diff, a, b);
32   and(borrow, na, b);
33 endmodule
34
35 module mymultiplier (input a, input b, output c);
36   and(c, a, b);
37 endmodule
38
39
40 module myALU (input a, input b, input [2:0] OC, output o1, output o2);
41   wire y1_sum, y1_carry, y2_diff, y2_borrow, y3, y4, y5, y6;
42   myhalfadder addition (.a(a), .b(b), .sum(y1_sum), .carry(y1_carry));
43   myhalfsubtractor subtraction (.a(a), .b(b), .diff(y2_diff), .borrow(y2_borrow));
44   mymultiplier multiplication (.a(a), .b(b), .c(y3));
45   or(y4, a, b);
46   and(y5, a, b);
47   xor(y6, a, b);
48
49   mymux8to1 MUX1 (.a(y1_sum), .b(y2_diff), .c(y3), .d(y4), .e(y5), .f(y6), .g(1'bz), .h(1'bz), .s(OC), .y(o1));
50   mymux8to1 MUX2 (.a(y1_carry), .b(y2_borrow), .c(1'bz), .d(1'bz), .e(1'bz), .f(1'bz), .g(1'bz), .h(1'bz), .s(OC), .y(o2));
51 endmodule
52
```

```
Assignment_1 > q6 > ≡ a6.v
53 module a6_tb();
54
55   reg a, b;
56   reg [2:0] OC;
57   wire o1, o2;
58
59   myALU ALU (a, b, OC, o1, o2);
60
61   initial begin
62     $dumpfile("a6_tb.vcd");
63     $dumpvars(0,a6_tb);
64
65     a = 1; b = 1; OC = 3'b000; #10;
66     a = 1; b = 0; OC = 3'b001; #10;
67     a = 0; b = 1; OC = 3'b010; #10;
68     a = 0; b = 1; OC = 3'b011; #10;
69     a = 1; b = 0; OC = 3'b100; #10;
70     a = 1; b = 0; OC = 3'b101; #10;
71     a = 1; b = 1; OC = 3'b110; #10;
72     a = 1; b = 1; OC = 3'b111; #10;
73   end
74
75 endmodule
```

