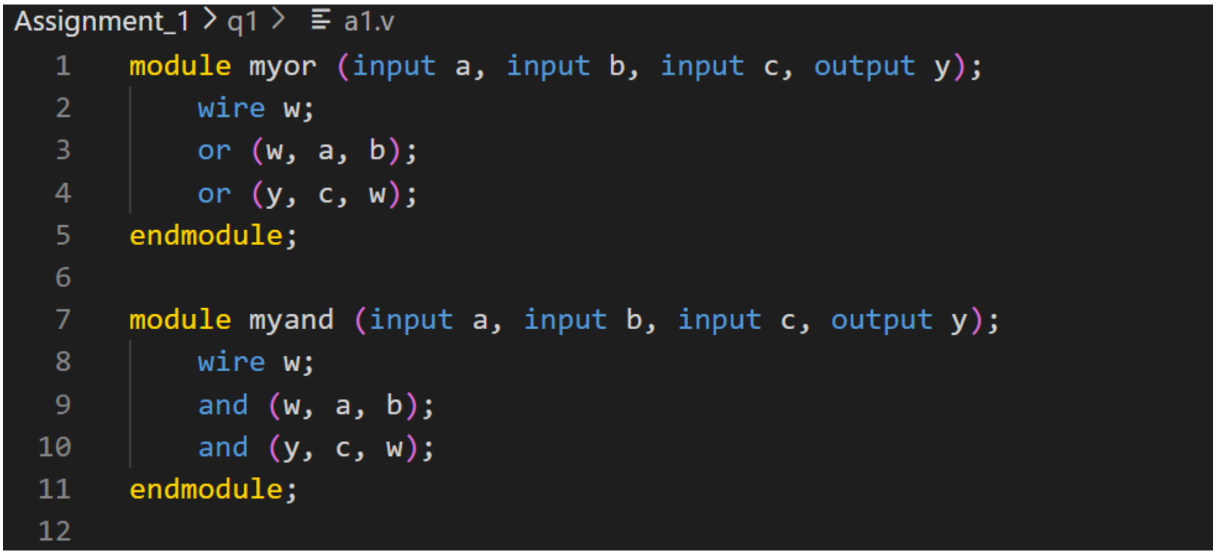
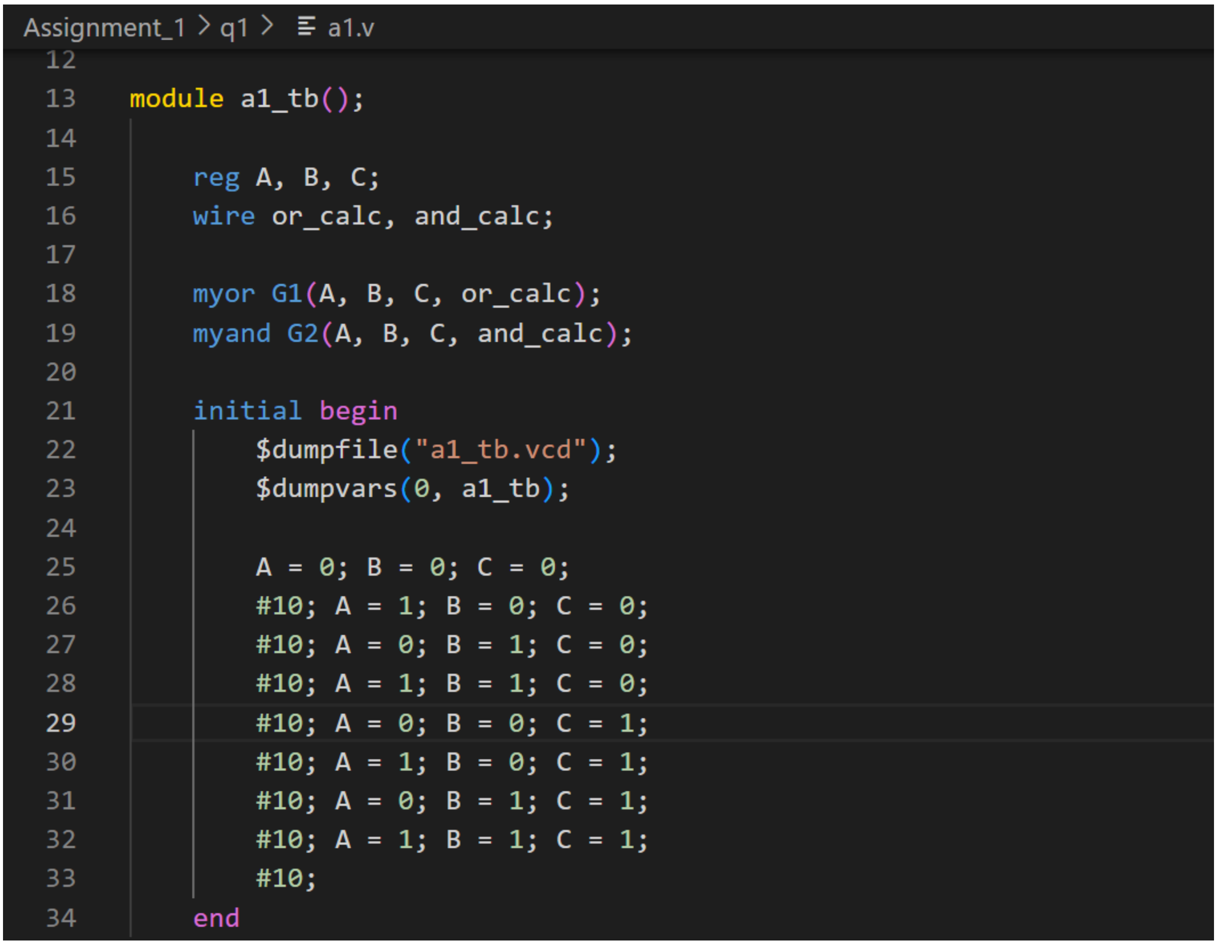
Computer Organization
Assignment 1
Sambit Sahoo (22277)

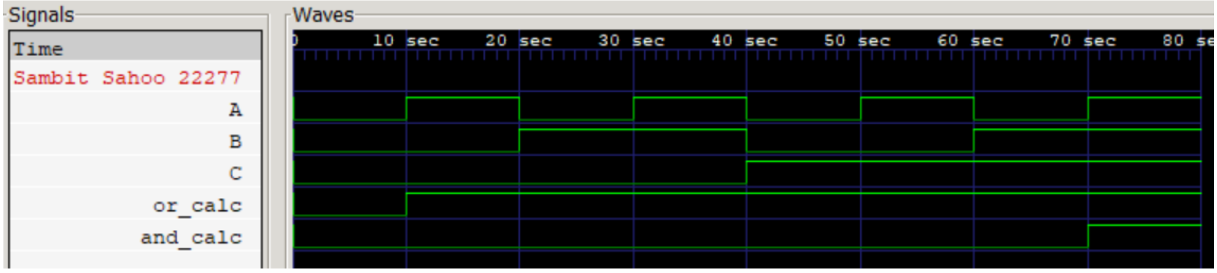
Q1 - 3-input AND gate, 3-input OR gate



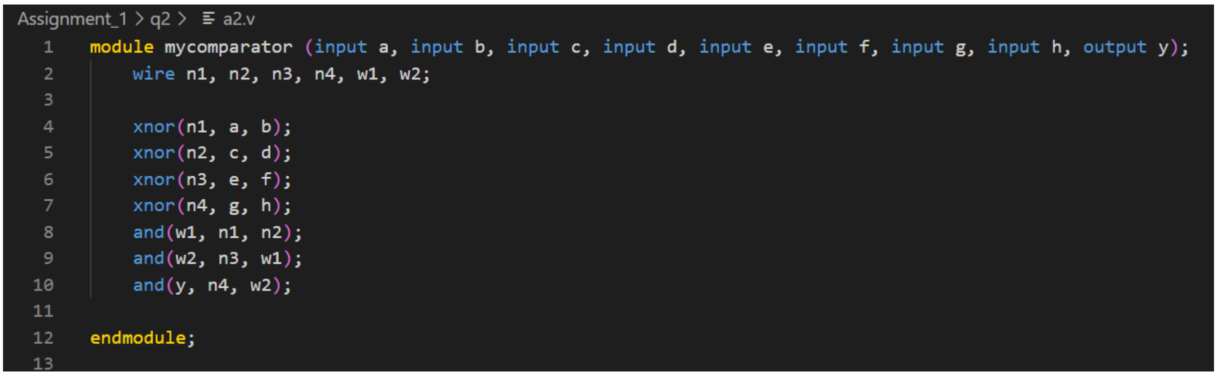
```
module myor (input a, input b, input c, output y);
    wire w;
    or (w, a, b);
    or (y, c, w);
endmodule;

module myand (input a, input b, input c, output y);
    wire w;
    and (w, a, b);
    and (y, c, w);
endmodule;

```



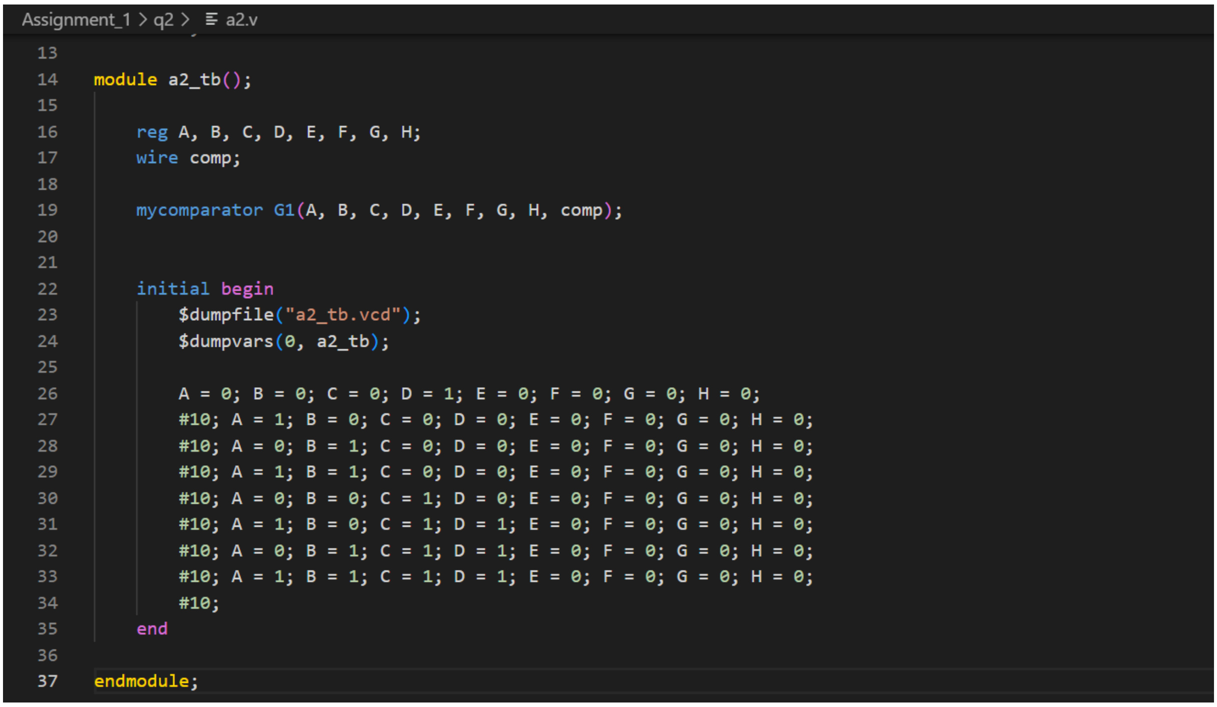
```

module a1_tb();

    reg A, B, C;
    wire or_calc, and_calc;

    myor G1(A, B, C, or_calc);
    myand G2(A, B, C, and_calc);

    initial begin
        $dumpfile("a1_tb.vcd");
        $dumpvars(0, a1_tb);

        A = 0; B = 0; C = 0;
        #10; A = 1; B = 0; C = 0;
        #10; A = 0; B = 1; C = 0;
        #10; A = 1; B = 1; C = 0;
        #10; A = 0; B = 0; C = 1;
        #10; A = 1; B = 0; C = 1;
        #10; A = 0; B = 1; C = 1;
        #10; A = 1; B = 1; C = 1;
        #10;
    end
```



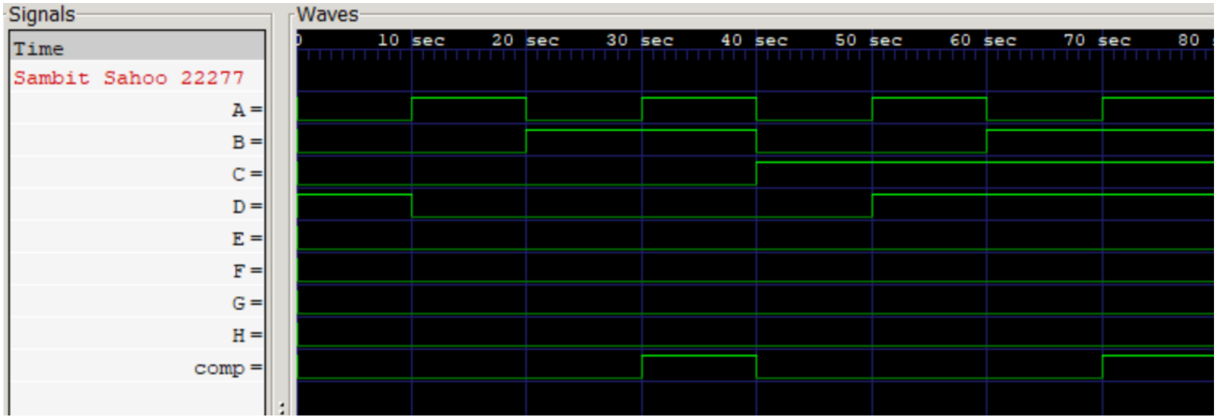
Q2 - 4-bit magnitude comparator



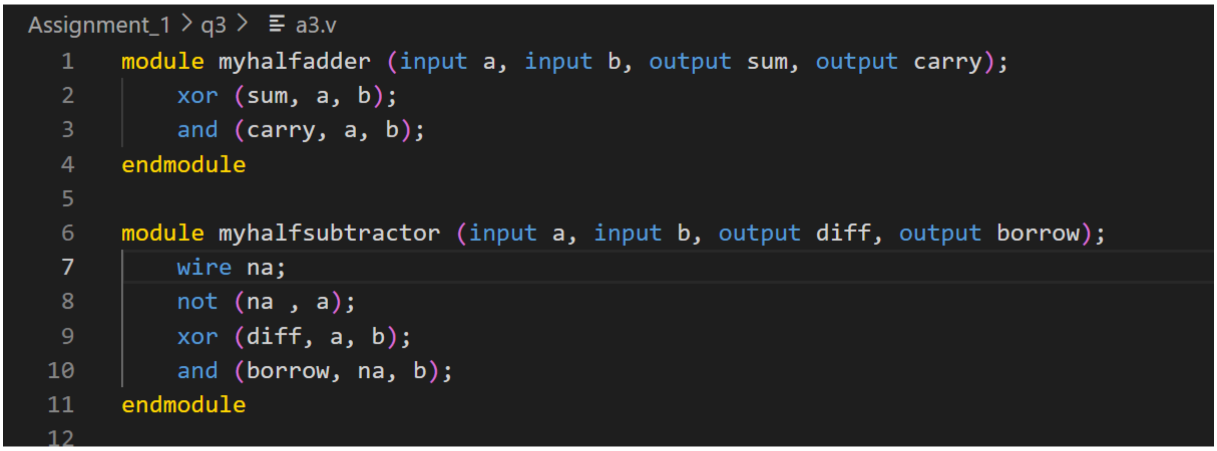
```verilog
module mycomparator (input a, input b, input c, input d, input e, input f, input g, input h, output y);
    wire n1, n2, n3, n4, w1, w2;

    xnor(n1, a, b);
    xnor(n2, c, d);
    xnor(n3, e, f);
    xnor(n4, g, h);
    and(w1, n1, n2);
    and(w2, n3, w1);
    and(y, n4, w2);

endmodule;

```



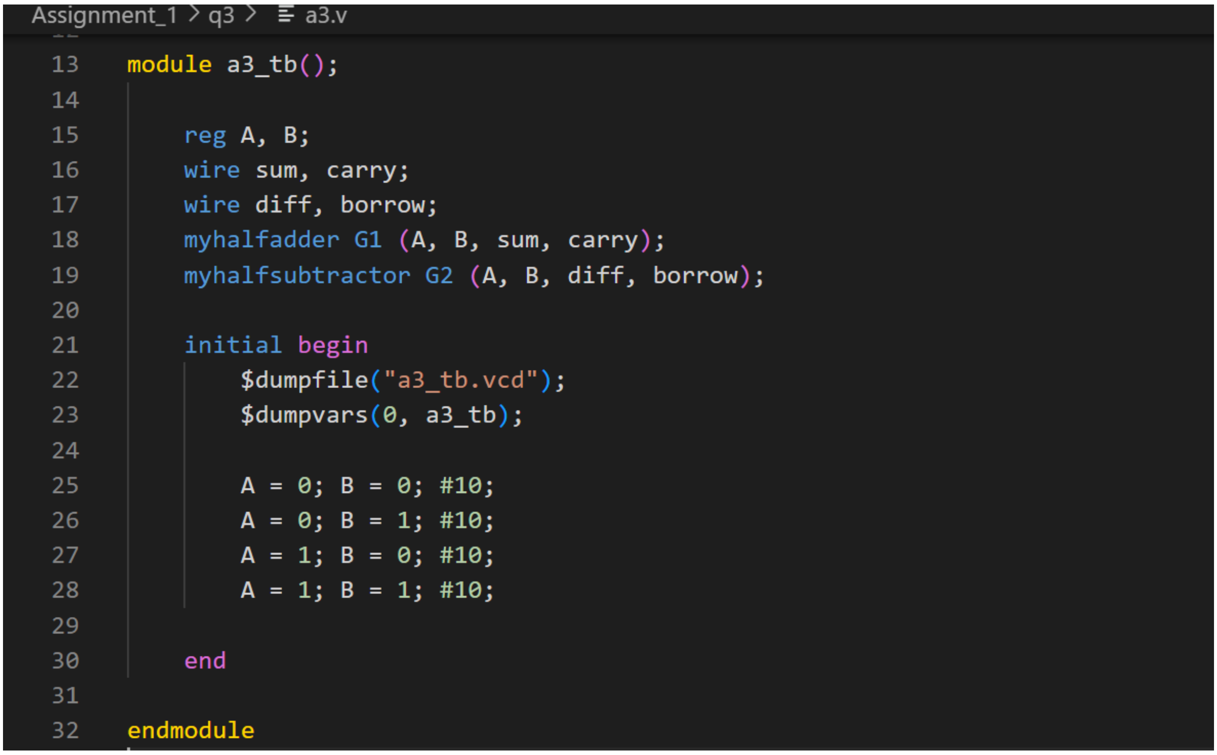
```verilog

module a2_tb();

    reg A, B, C, D, E, F, G, H;
    wire comp;

    mycomparator G1(A, B, C, D, E, F, G, H, comp);


    initial begin
        $dumpfile("a2_tb.vcd");
        $dumpvars(0, a2_tb);

        A = 0; B = 0; C = 0; D = 1; E = 0; F = 0; G = 0; H = 0;
        #10; A = 1; B = 0; C = 0; D = 0; E = 0; F = 0; G = 0; H = 0;
        #10; A = 0; B = 1; C = 0; D = 0; E = 0; F = 0; G = 0; H = 0;
        #10; A = 1; B = 1; C = 0; D = 0; E = 0; F = 0; G = 0; H = 0;
        #10; A = 0; B = 0; C = 1; D = 0; E = 0; F = 0; G = 0; H = 0;
        #10; A = 1; B = 0; C = 1; D = 1; E = 0; F = 0; G = 0; H = 0;
        #10; A = 0; B = 1; C = 1; D = 1; E = 0; F = 0; G = 0; H = 0;
        #10; A = 1; B = 1; C = 1; D = 1; E = 0; F = 0; G = 0; H = 0;
        #10;
    end

endmodule;
```



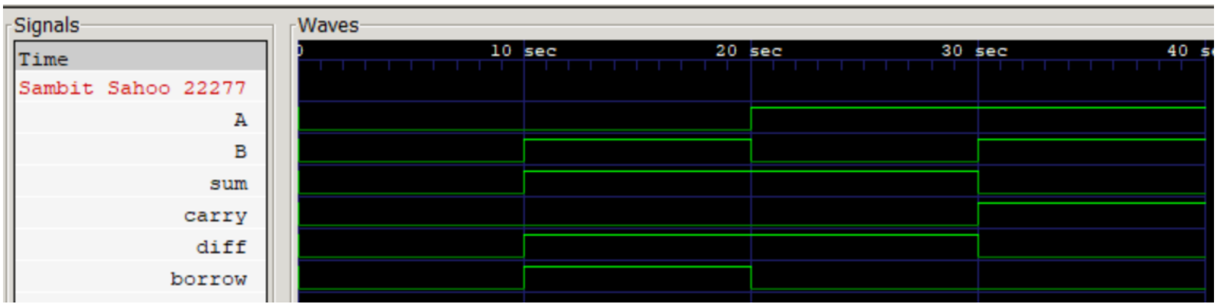
Q3- Half adder and Half subtractor & Full adder and Full subtractor

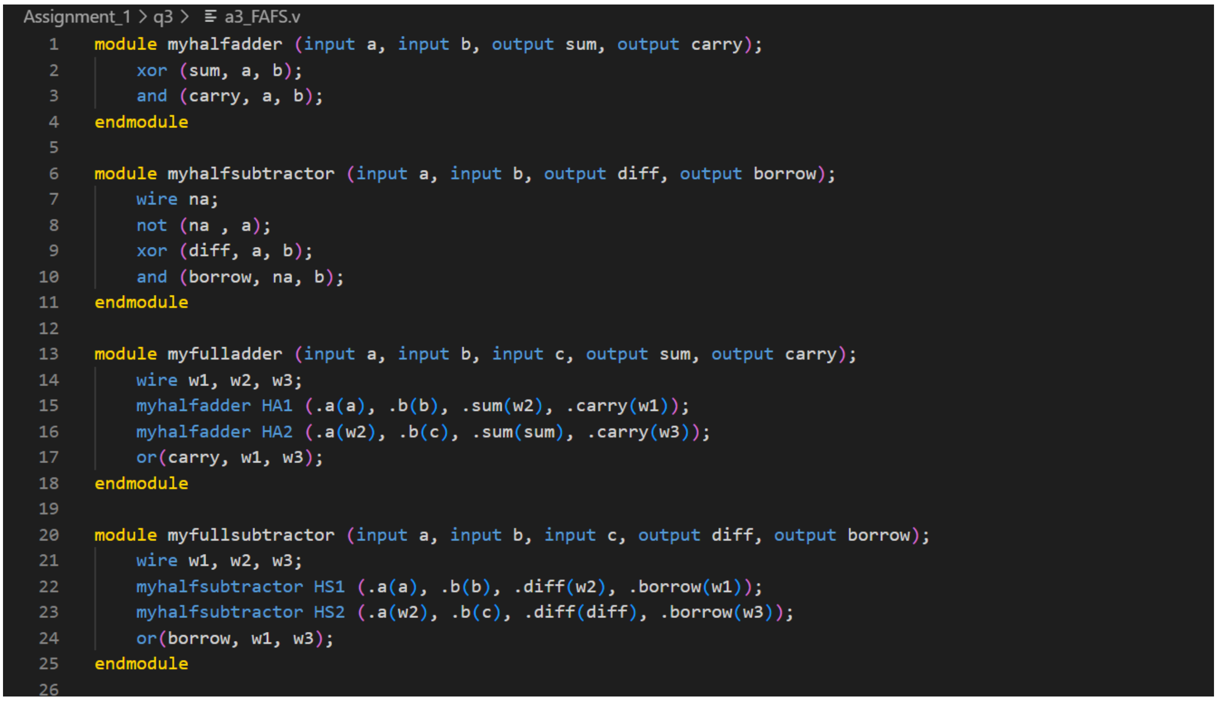


```
module myhalfadder (input a, input b, output sum, output carry);
    xor (sum, a, b);
    and (carry, a, b);
endmodule

module myhalfsubtractor (input a, input b, output diff, output borrow);
    wire na;
    not (na , a);
    xor (diff, a, b);
    and (borrow, na, b);
endmodule

```

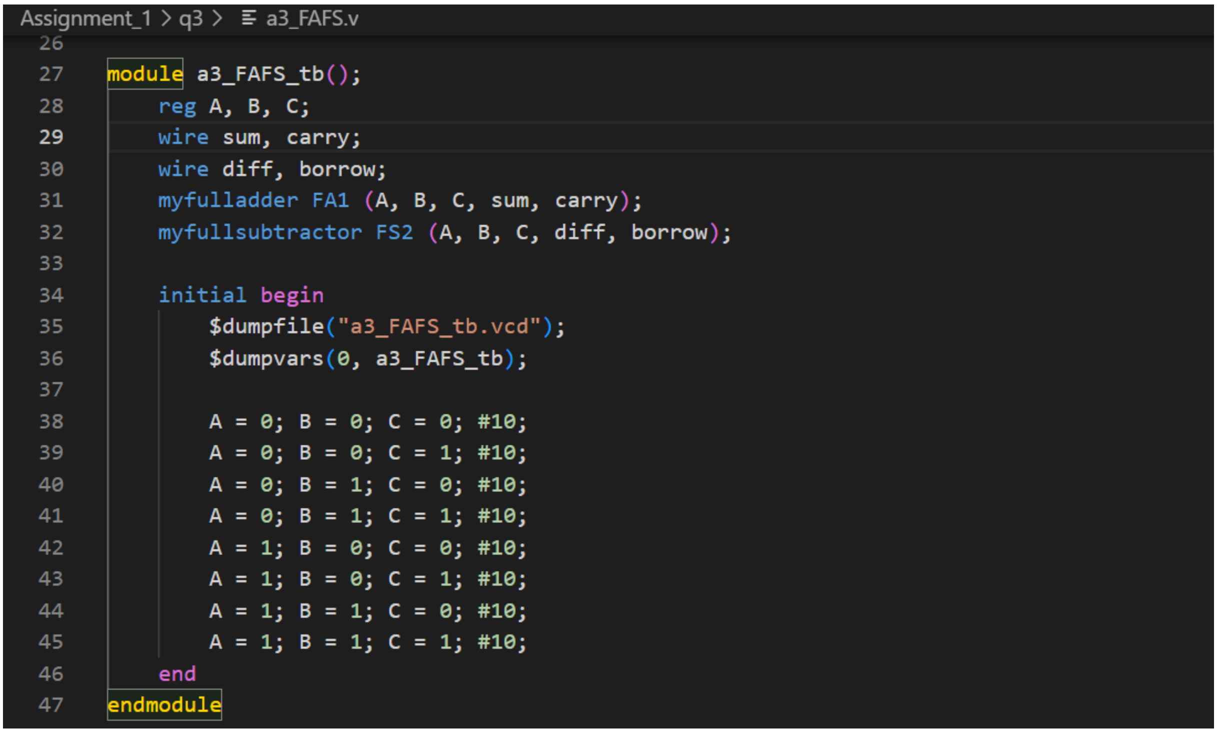


```
module a3_tb();

    reg A, B;
    wire sum, carry;
    wire diff, borrow;
    myhalfadder G1 (A, B, sum, carry);
    myhalfsubtractor G2 (A, B, diff, borrow);

    initial begin
        $dumpfile("a3_tb.vcd");
        $dumpvars(0, a3_tb);

        A = 0; B = 0; #10;
        A = 0; B = 1; #10;
        A = 1; B = 0; #10;
        A = 1; B = 1; #10;

    end

endmodule
```

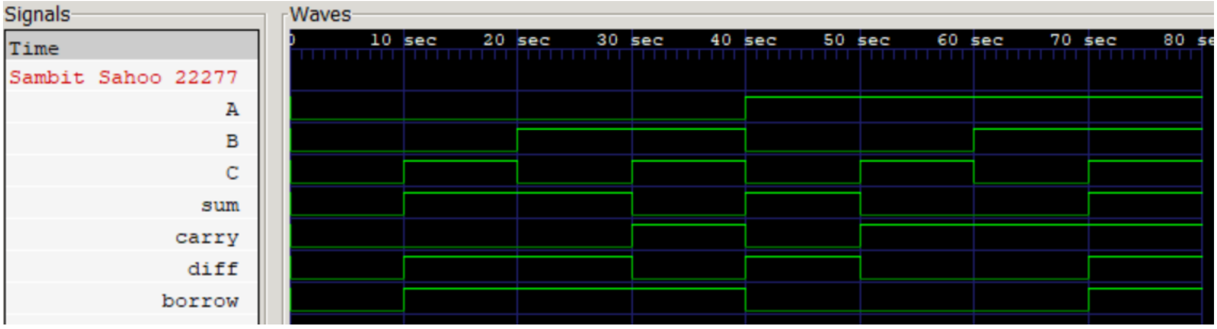




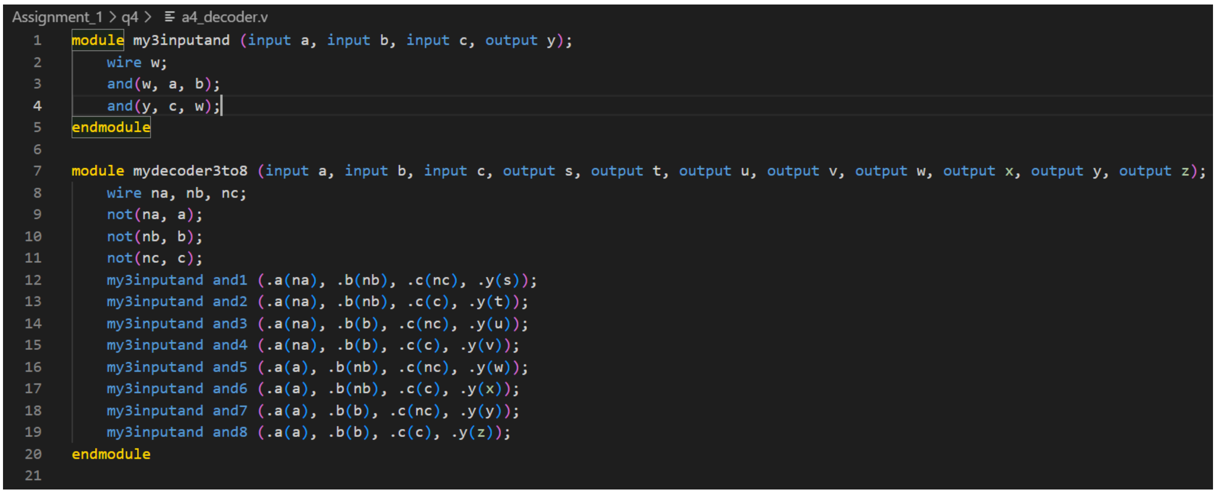
```
module myhalfadder (input a, input b, output sum, output carry);
    xor (sum, a, b);
    and (carry, a, b);
endmodule

module myhalfsubtractor (input a, input b, output diff, output borrow);
    wire na;
    not (na , a);
    xor (diff, a, b);
    and (borrow, na, b);
endmodule

module myfulladder (input a, input b, input c, output sum, output carry);
    wire w1, w2, w3;
    myhalfadder HA1 (.a(a), .b(b), .sum(w2), .carry(w1));
    myhalfadder HA2 (.a(w2), .b(c), .sum(sum), .carry(w3));
    or(carry, w1, w3);
endmodule

module myfullsubtractor (input a, input b, input c, output diff, output borrow);
    wire w1, w2, w3;
    myhalfsubtractor HS1 (.a(a), .b(b), .diff(w2), .borrow(w1));
    myhalfsubtractor HS2 (.a(w2), .b(c), .diff(diff), .borrow(w3));
    or(borrow, w1, w3);
endmodule

```



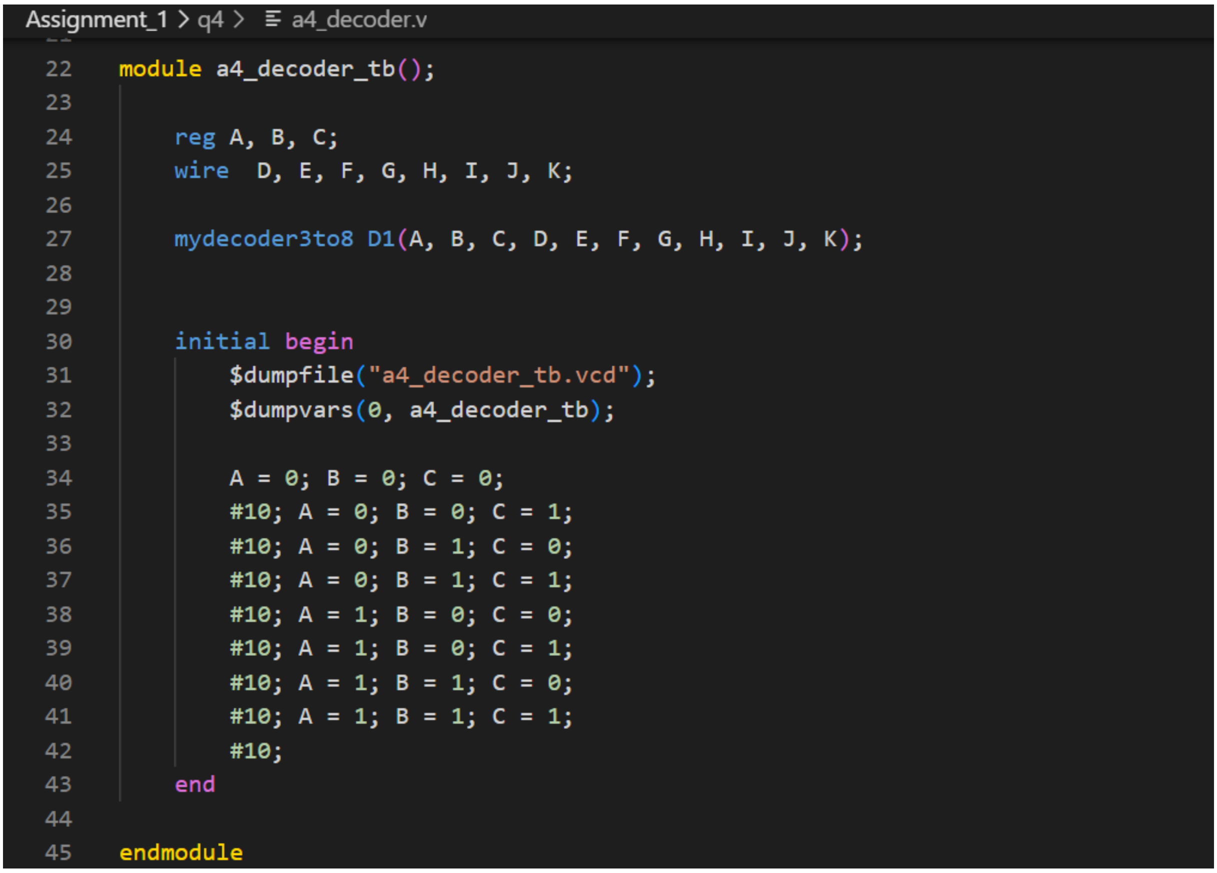
```

module a3_FAFS_tb();
    reg A, B, C;
    wire sum, carry;
    wire diff, borrow;
    myfulladder FA1 (A, B, C, sum, carry);
    myfullsubtractor FS2 (A, B, C, diff, borrow);

    initial begin
        $dumpfile("a3_FAFS_tb.vcd");
        $dumpvars(0, a3_FAFS_tb);

        A = 0; B = 0; C = 0; #10;
        A = 0; B = 0; C = 1; #10;
        A = 0; B = 1; C = 0; #10;
        A = 0; B = 1; C = 1; #10;
        A = 1; B = 0; C = 0; #10;
        A = 1; B = 0; C = 1; #10;
        A = 1; B = 1; C = 0; #10;
        A = 1; B = 1; C = 1; #10;
    end
endmodule
```



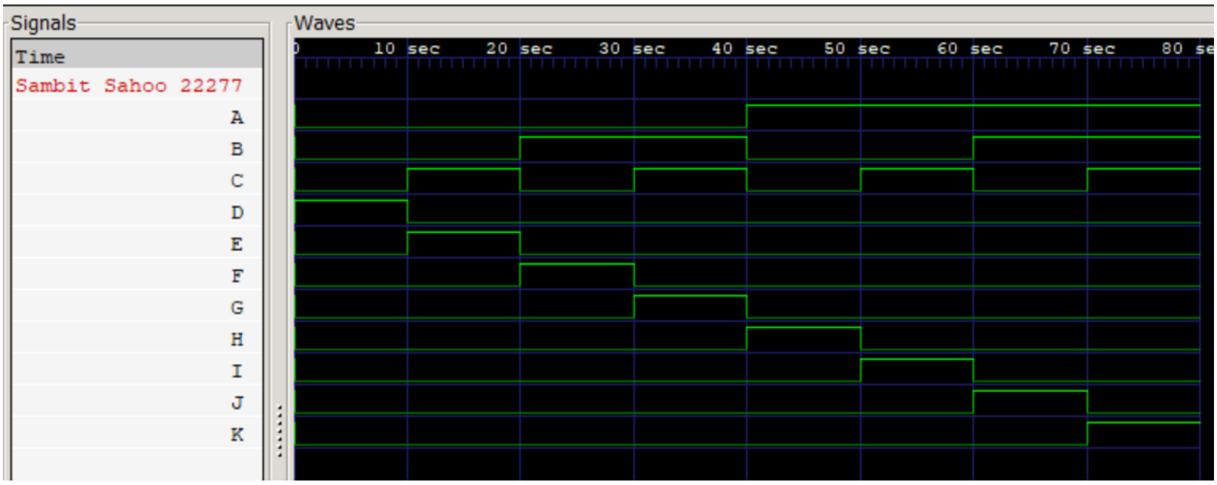
Q4 - 4-to-1 multiplexer and 3-to-8 decoder

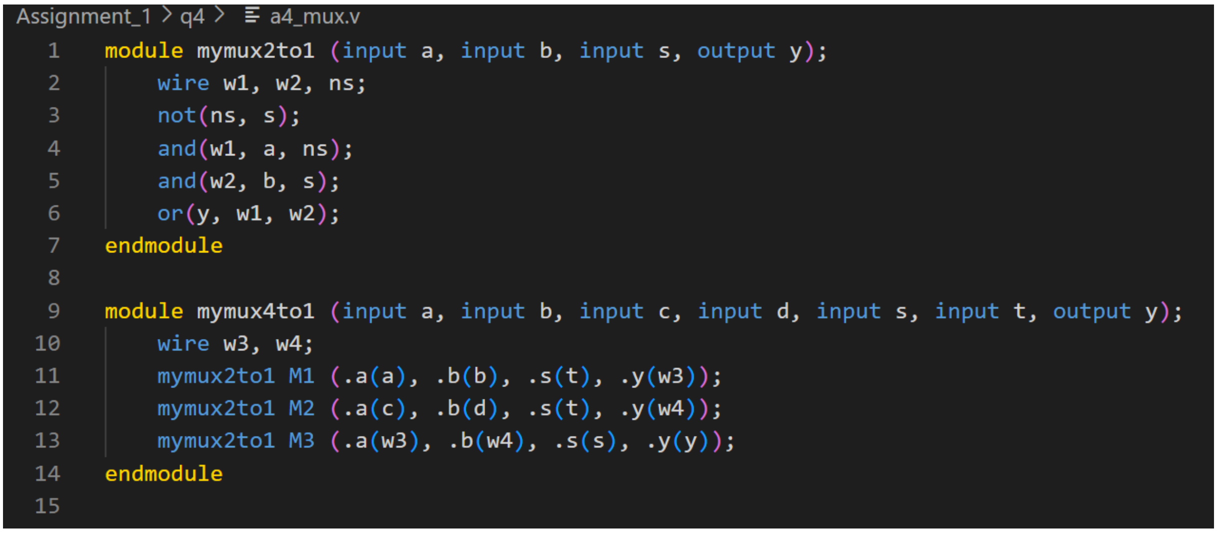


```verilog
module my3inputand (input a, input b, input c, output y);
    wire w;
    and(w, a, b);
    and(y, c, w);
endmodule

module mydecoder3to8 (input a, input b, input c, output s, output t, output u, output v, output w, output x, output y, output z);
    wire na, nb, nc;
    not(na, a);
    not(nb, b);
    not(nc, c);
    my3inputand and1 (.a(na), .b(nb), .c(nc), .y(s));
    my3inputand and2 (.a(na), .b(nb), .c(c), .y(t));
    my3inputand and3 (.a(na), .b(b), .c(nc), .y(u));
    my3inputand and4 (.a(na), .b(b), .c(c), .y(v));
    my3inputand and5 (.a(a), .b(nb), .c(nc), .y(w));
    my3inputand and6 (.a(a), .b(nb), .c(c), .y(x));
    my3inputand and7 (.a(a), .b(b), .c(nc), .y(y));
    my3inputand and8 (.a(a), .b(b), .c(c), .y(z));
endmodule

```

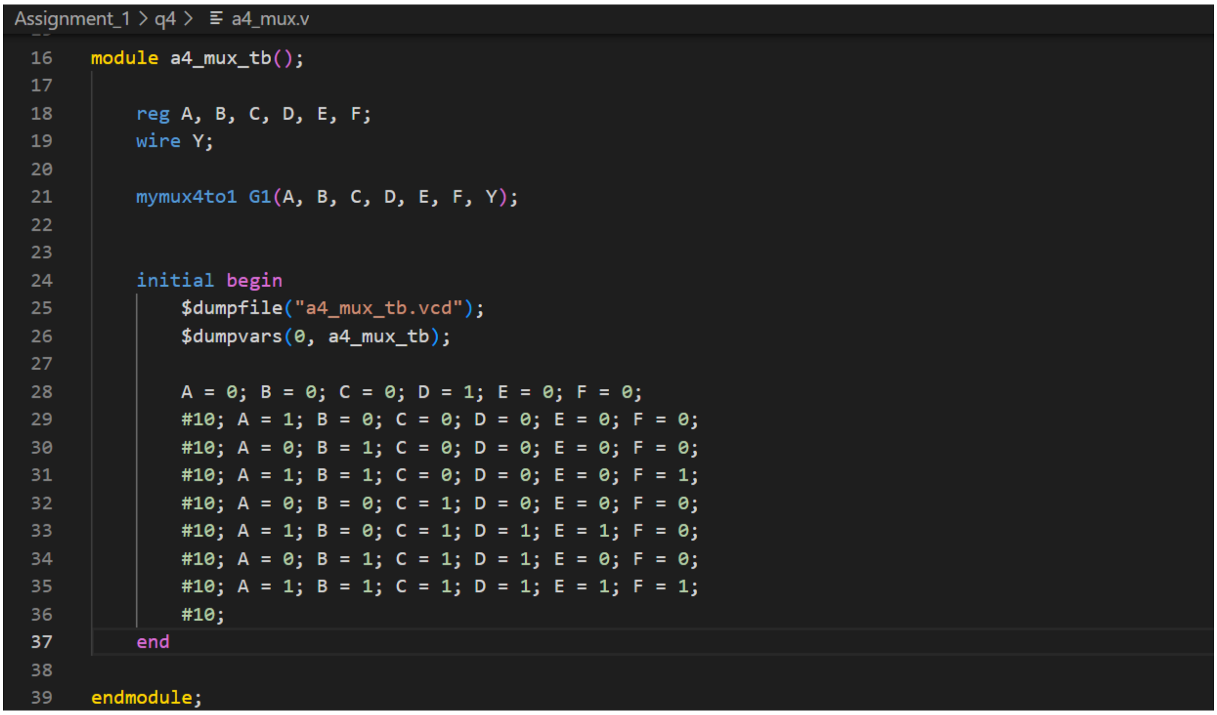


```verilog
module a4_decoder_tb();

    reg A, B, C;
    wire  D, E, F, G, H, I, J, K;

    mydecoder3to8 D1(A, B, C, D, E, F, G, H, I, J, K);


    initial begin
        $dumpfile("a4_decoder_tb.vcd");
        $dumpvars(0, a4_decoder_tb);

        A = 0; B = 0; C = 0;
        #10; A = 0; B = 0; C = 1;
        #10; A = 0; B = 1; C = 0;
        #10; A = 0; B = 1; C = 1;
        #10; A = 1; B = 0; C = 0;
        #10; A = 1; B = 0; C = 1;
        #10; A = 1; B = 1; C = 0;
        #10; A = 1; B = 1; C = 1;
        #10;
    end

endmodule
```

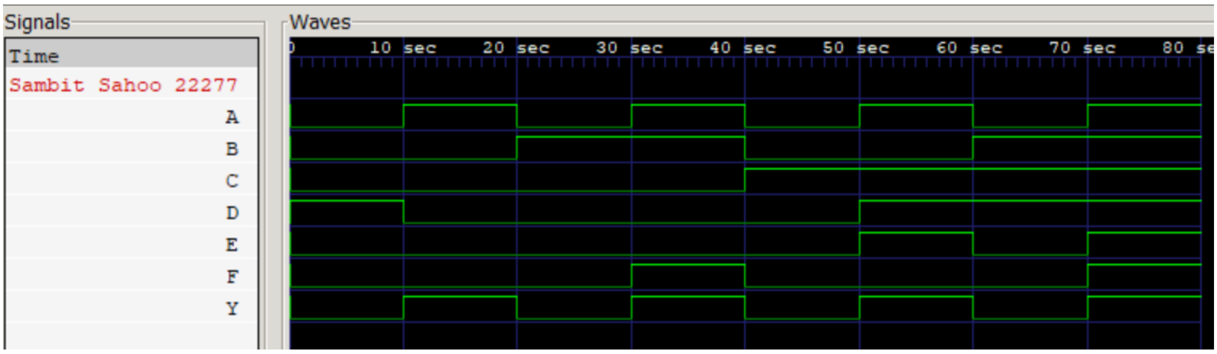




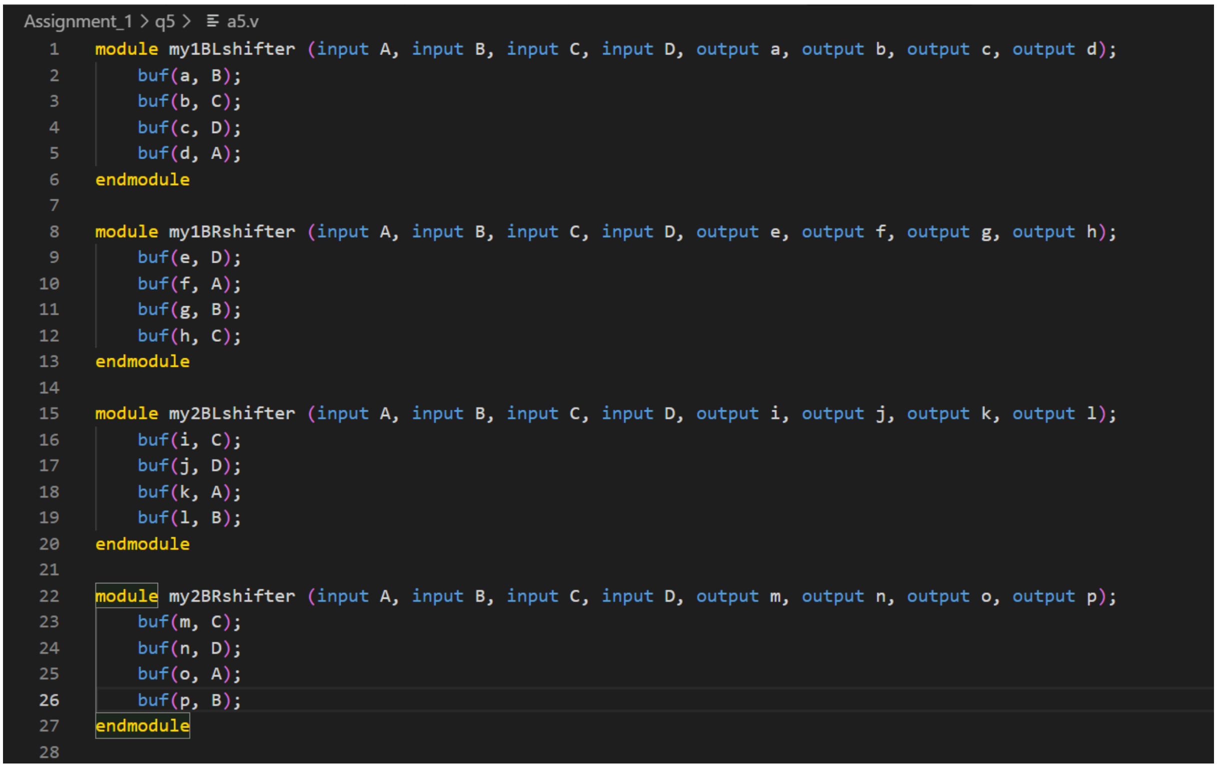
```verilog
module mymux2to1 (input a, input b, input s, output y);
    wire w1, w2, ns;
    not(ns, s);
    and(w1, a, ns);
    and(w2, b, s);
    or(y, w1, w2);
endmodule

module mymux4to1 (input a, input b, input c, input d, input s, input t, output y);
    wire w3, w4;
    mymux2to1 M1 (.a(a), .b(b), .s(t), .y(w3));
    mymux2to1 M2 (.a(c), .b(d), .s(t), .y(w4));
    mymux2to1 M3 (.a(w3), .b(w4), .s(s), .y(y));
endmodule

```



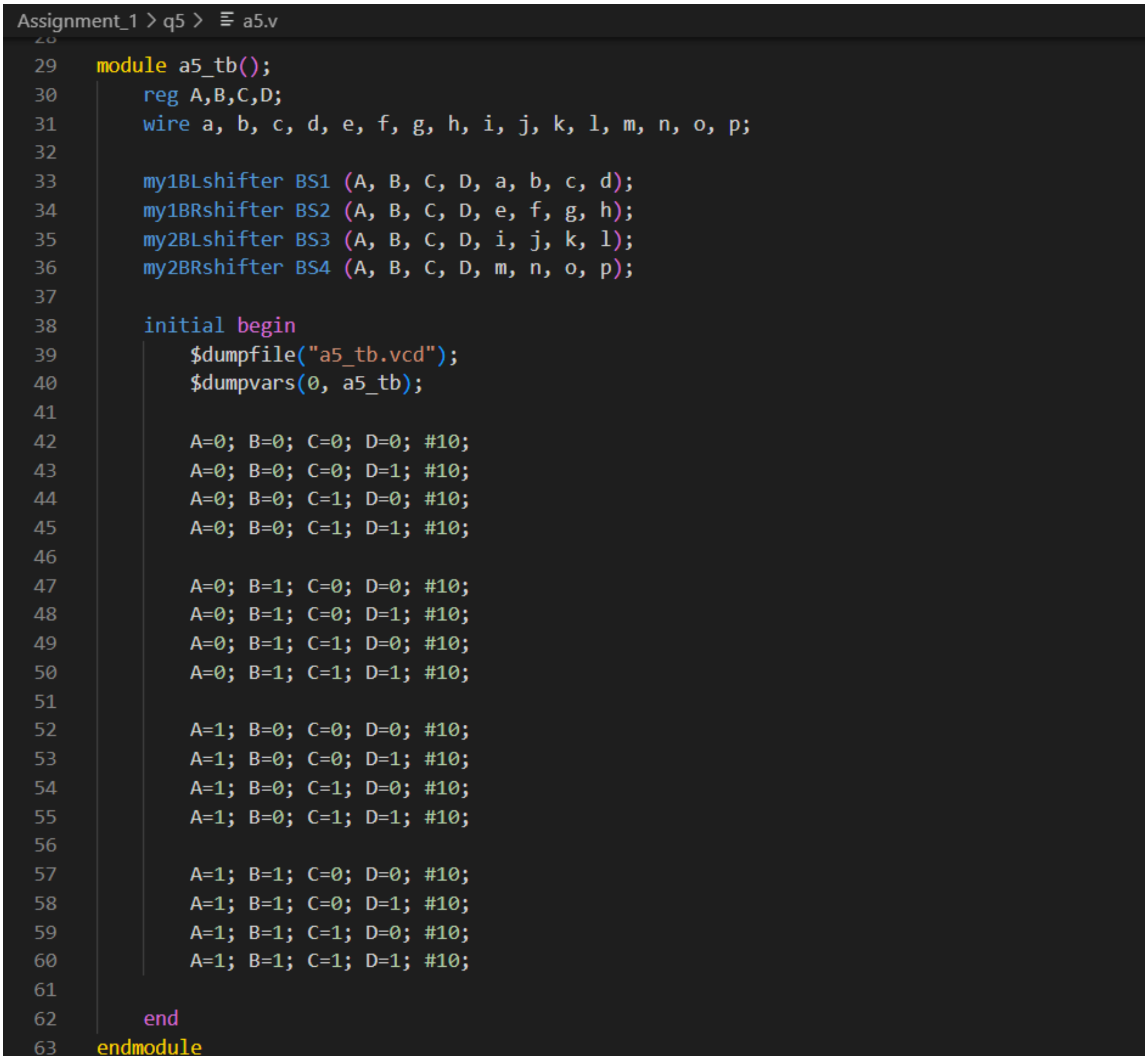
```verilog
module a4_mux_tb();

    reg A, B, C, D, E, F;
    wire Y;

    mymux4to1 G1(A, B, C, D, E, F, Y);


    initial begin
        $dumpfile("a4_mux_tb.vcd");
        $dumpvars(0, a4_mux_tb);

        A = 0; B = 0; C = 0; D = 1; E = 0; F = 0;
        #10; A = 1; B = 0; C = 0; D = 0; E = 0; F = 0;
        #10; A = 0; B = 1; C = 0; D = 0; E = 0; F = 0;
        #10; A = 1; B = 1; C = 0; D = 0; E = 0; F = 1;
        #10; A = 0; B = 0; C = 1; D = 0; E = 0; F = 0;
        #10; A = 1; B = 0; C = 1; D = 1; E = 1; F = 0;
        #10; A = 0; B = 1; C = 1; D = 1; E = 0; F = 0;
        #10; A = 1; B = 1; C = 1; D = 1; E = 1; F = 1;
        #10;
    end

endmodule;
```



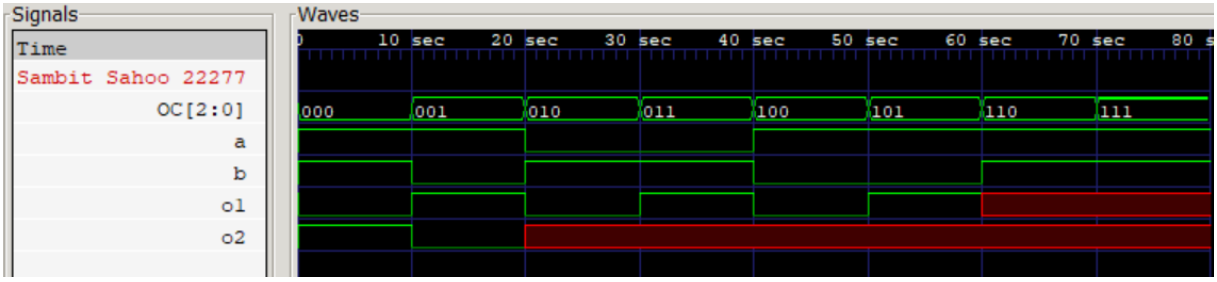
Q5 - 4-bit barrel shifter with (1-bit and 2-bit) left and right shift



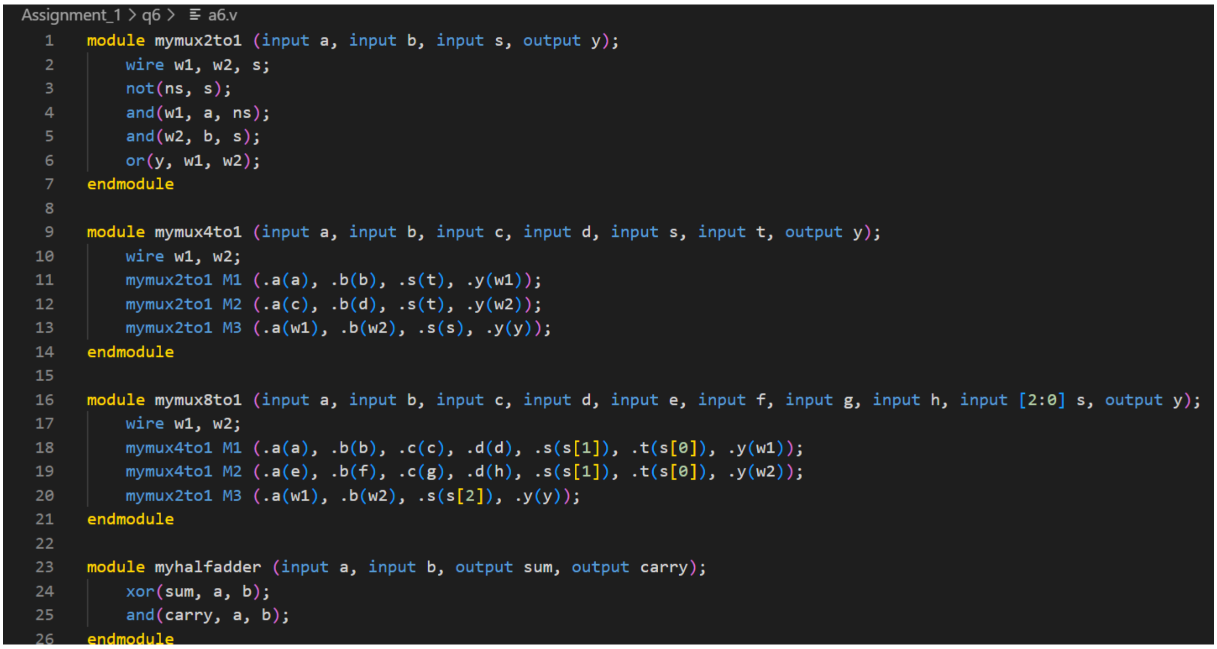
```verilog
module my1BLshifter (input A, input B, input C, input D, output a, output b, output c, output d);
    buf(a, B);
    buf(b, C);
    buf(c, D);
    buf(d, A);
endmodule

module my1BRshifter (input A, input B, input C, input D, output e, output f, output g, output h);
    buf(e, D);
    buf(f, A);
    buf(g, B);
    buf(h, C);
endmodule

module my2BLshifter (input A, input B, input C, input D, output i, output j, output k, output l);
    buf(i, C);
    buf(j, D);
    buf(k, A);
    buf(l, B);
endmodule

module my2BRshifter (input A, input B, input C, input D, output m, output n, output o, output p);
    buf(m, C);
    buf(n, D);
    buf(o, A);
    buf(p, B);
endmodule

```



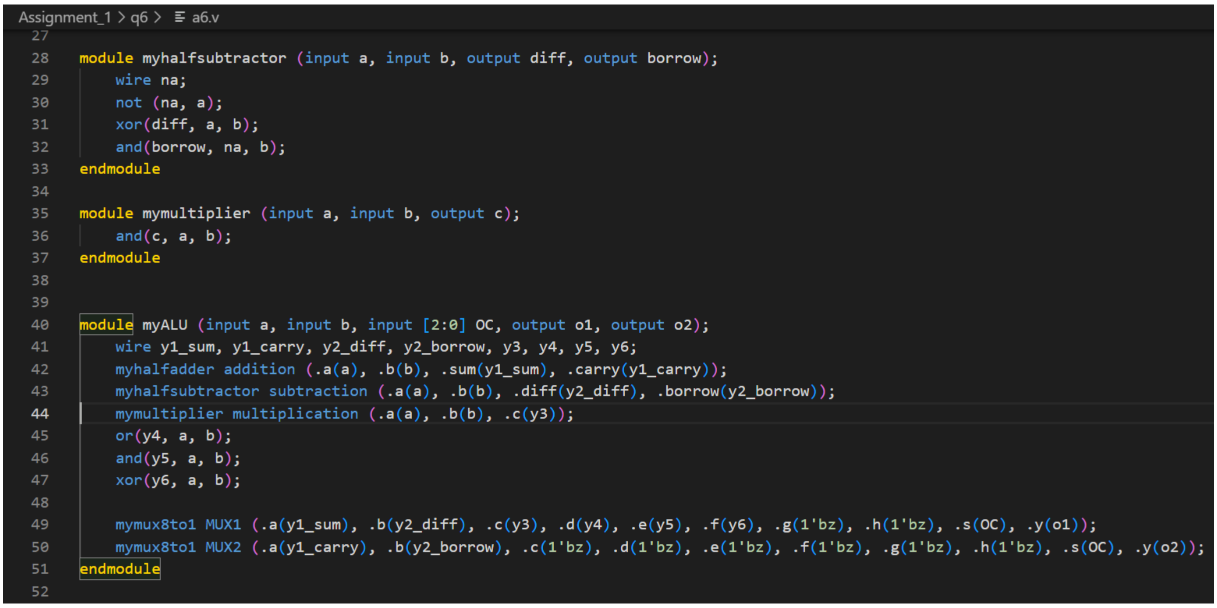
```
module a5_tb();
    reg A,B,C,D;
    wire a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p;

    my1BLshifter BS1 (A, B, C, D, a, b, c, d);
    my1BRshifter BS2 (A, B, C, D, e, f, g, h);
    my2BLshifter BS3 (A, B, C, D, i, j, k, l);
    my2BRshifter BS4 (A, B, C, D, m, n, o, p);

    initial begin
        $dumpfile("a5_tb.vcd");
        $dumpvars(0, a5_tb);

        A=0; B=0; C=0; D=0; #10;
        A=0; B=0; C=0; D=1; #10;
        A=0; B=0; C=1; D=0; #10;
        A=0; B=0; C=1; D=1; #10;

        A=0; B=1; C=0; D=0; #10;
        A=0; B=1; C=0; D=1; #10;
        A=0; B=1; C=1; D=0; #10;
        A=0; B=1; C=1; D=1; #10;

        A=1; B=0; C=0; D=0; #10;
        A=1; B=0; C=0; D=1; #10;
        A=1; B=0; C=1; D=0; #10;
        A=1; B=0; C=1; D=1; #10;

        A=1; B=1; C=0; D=0; #10;
        A=1; B=1; C=0; D=1; #10;
        A=1; B=1; C=1; D=0; #10;
        A=1; B=1; C=1; D=1; #10;

    end
endmodule
```



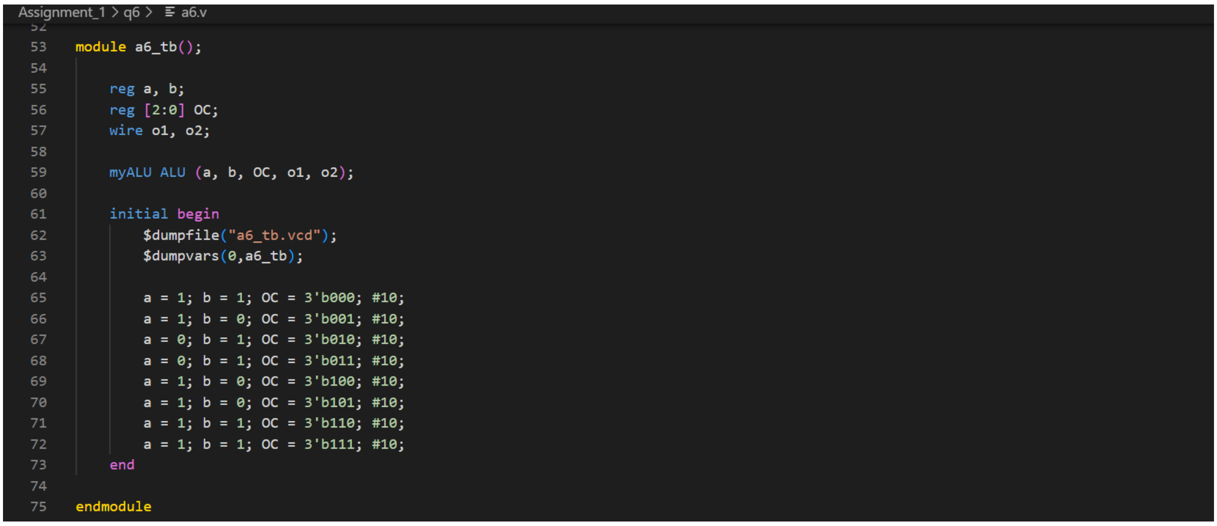
Q6 - Single-bit ALU



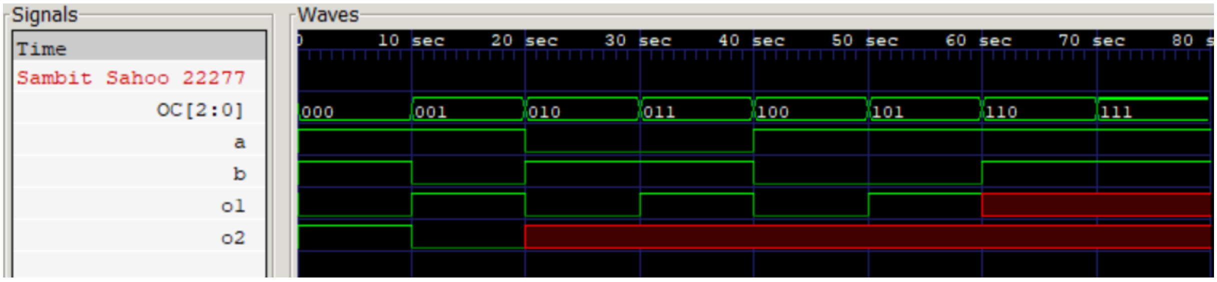
```verilog
module mymux2to1 (input a, input b, input s, output y);
    wire w1, w2, s;
    not(ns, s);
    and(w1, a, ns);
    and(w2, b, s);
    or(y, w1, w2);
endmodule

module mymux4to1 (input a, input b, input c, input d, input s, input t, output y);
    wire w1, w2;
    mymux2to1 M1 (.a(a), .b(b), .s(t), .y(w1));
    mymux2to1 M2 (.a(c), .b(d), .s(t), .y(w2));
    mymux2to1 M3 (.a(w1), .b(w2), .s(s), .y(y));
endmodule

module mymux8to1 (input a, input b, input c, input d, input e, input f, input g, input h, input [2:0] s, output y);
    wire w1, w2;
    mymux4to1 M1 (.a(a), .b(b), .c(c), .d(d), .s(s[1]), .t(s[0]), .y(w1));
    mymux4to1 M2 (.a(e), .b(f), .c(g), .d(h), .s(s[1]), .t(s[0]), .y(w2));
    mymux2to1 M3 (.a(w1), .b(w2), .s(s[2]), .y(y));
endmodule

module myhalfadder (input a, input b, output sum, output carry);
    xor(sum, a, b);
    and(carry, a, b);
endmodule
```



```verilog

module myhalfsubtractor (input a, input b, output diff, output borrow);
    wire na;
    not (na, a);
    xor(diff, a, b);
    and(borrow, na, b);
endmodule

module mymultiplier (input a, input b, output c);
    and(c, a, b);
endmodule


module myALU (input a, input b, input [2:0] OC, output o1, output o2);
    wire y1_sum, y1_carry, y2_diff, y2_borrow, y3, y4, y5, y6;
    myhalfadder addition (.a(a), .b(b), .sum(y1_sum), .carry(y1_carry));
    myhalfsubtractor subtraction (.a(a), .b(b), .diff(y2_diff), .borrow(y2_borrow));
    mymultiplier multiplication (.a(a), .b(b), .c(y3));
    or(y4, a, b);
    and(y5, a, b);
    xor(y6, a, b);

    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));
    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));
endmodule

```



```verilog
module a6_tb();

    reg a, b;
    reg [2:0] OC;
    wire o1, o2;

    myALU ALU (a, b, OC, o1, o2);

    initial begin
        $dumpfile("a6_tb.vcd");
        $dumpvars(0,a6_tb);

        a = 1; b = 1; OC = 3'b000; #10;
        a = 1; b = 0; OC = 3'b001; #10;
        a = 0; b = 1; OC = 3'b010; #10;
        a = 0; b = 1; OC = 3'b011; #10;
        a = 1; b = 0; OC = 3'b100; #10;
        a = 1; b = 0; OC = 3'b101; #10;
        a = 1; b = 1; OC = 3'b110; #10;
        a = 1; b = 1; OC = 3'b111; #10;
    end

endmodule
```



Signals
Waves
Time
Sambit Sahoo 22277
OC[2:0]
a
b
o1
o2
0
10 sec
20 sec
30 sec
40 sec
50 sec
60 sec
70 sec
80 s
000
001
010
011
100
101
110
111