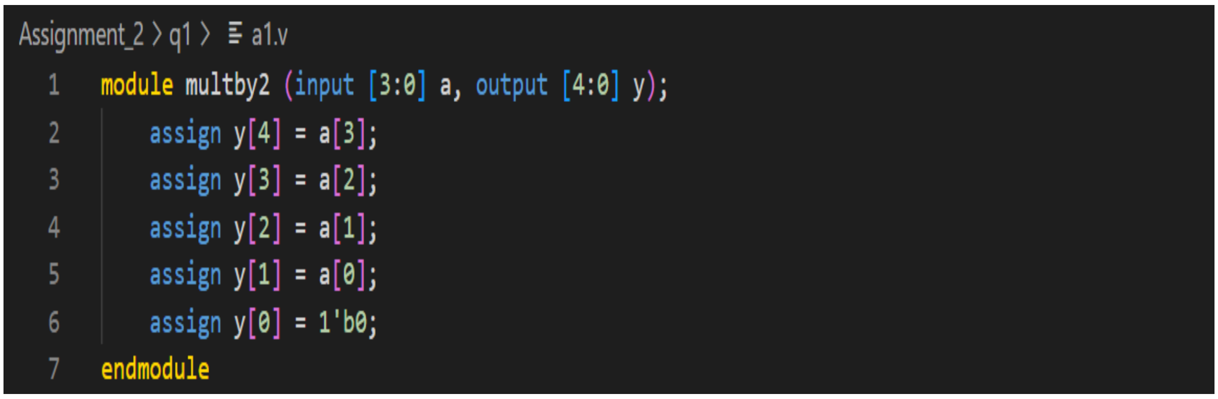
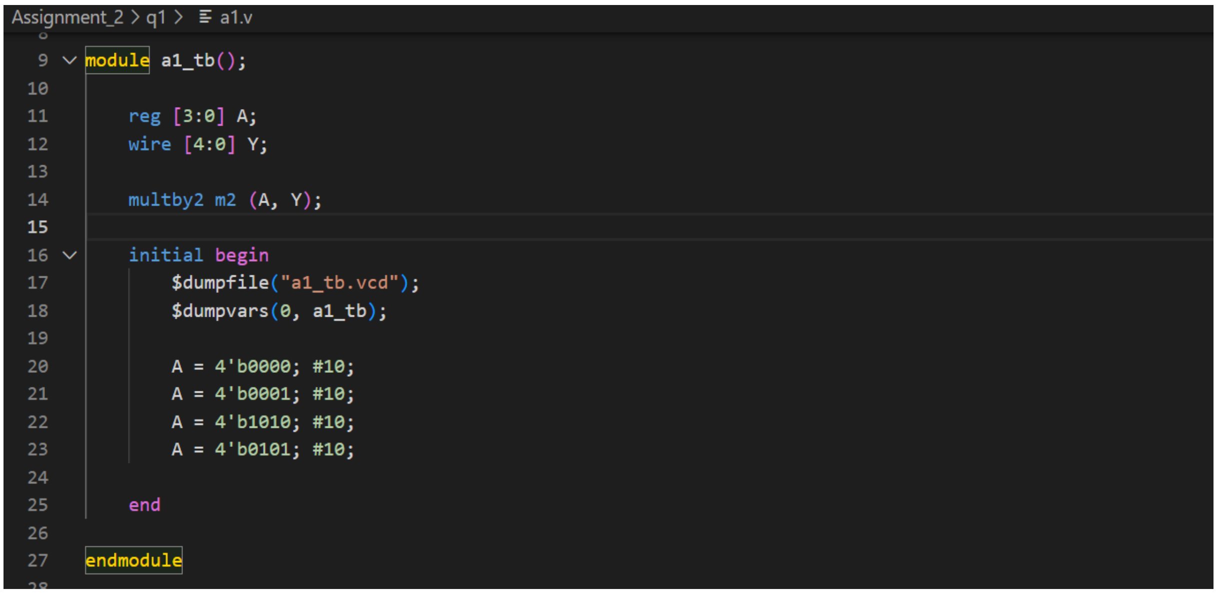
Computer Organization
Assignment 2
Sambit Sahoo (22277)

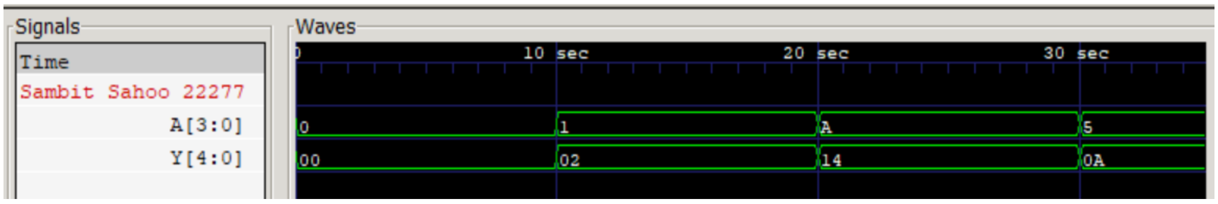
Q1 - Multiply a 4-bit input by 2



```
Assignment_2 > q1 > a1.v
module multby2 (input [3:0] a, output [4:0] y);
    assign y[4] = a[3];
    assign y[3] = a[2];
    assign y[2] = a[1];
    assign y[1] = a[0];
    assign y[0] = 1'b0;
endmodule
```

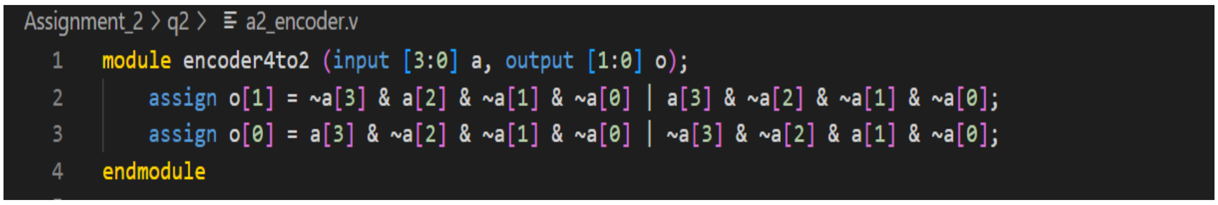


```
Assignment_2 > q1 > a1.v
module a1_tb();

    reg [3:0] A;
    wire [4:0] Y;

    multby2 m2 (A, Y);

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

        A = 4'b0000; #10;
        A = 4'b0001; #10;
        A = 4'b1010; #10;
        A = 4'b0101; #10;

    end

endmodule
```

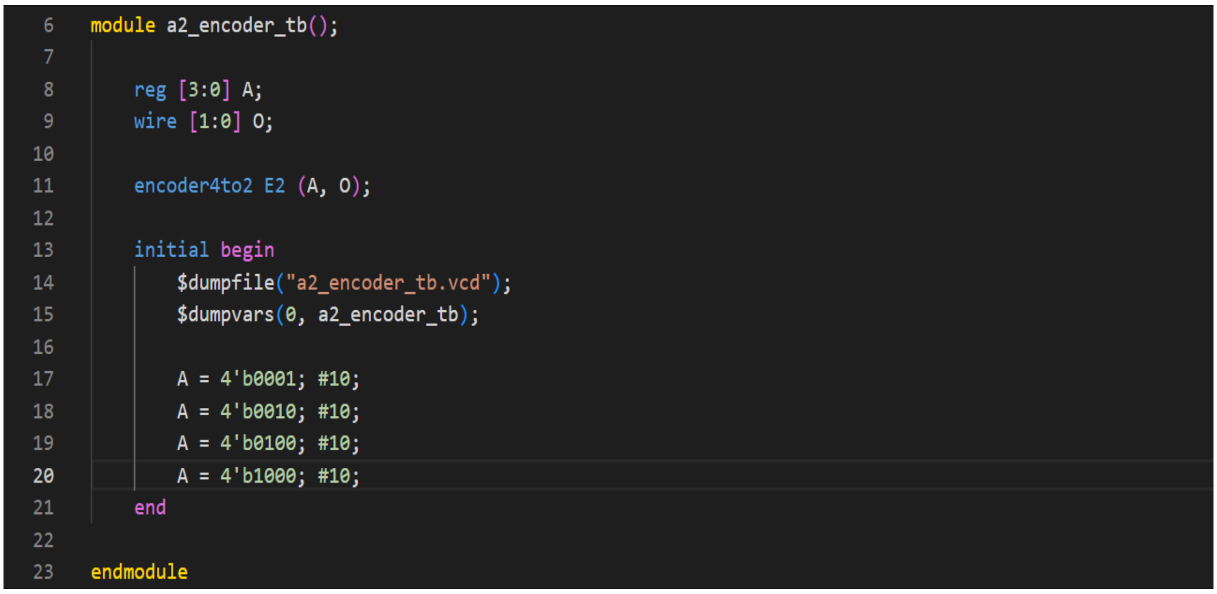


```
Signals              Waves
Time                 10 sec      20 sec      30 sec
Sambit Sahoo 22277
A[3:0]     0         1           A           5
Y[4:0]     00        02          14          0A
```

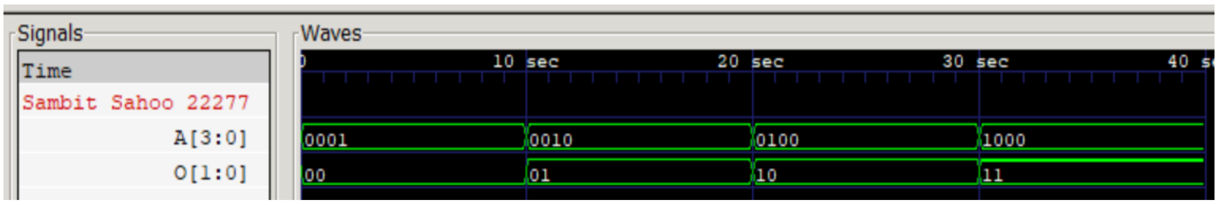
Q2 - 4-to-2 encoder and 2- to-4 decoder

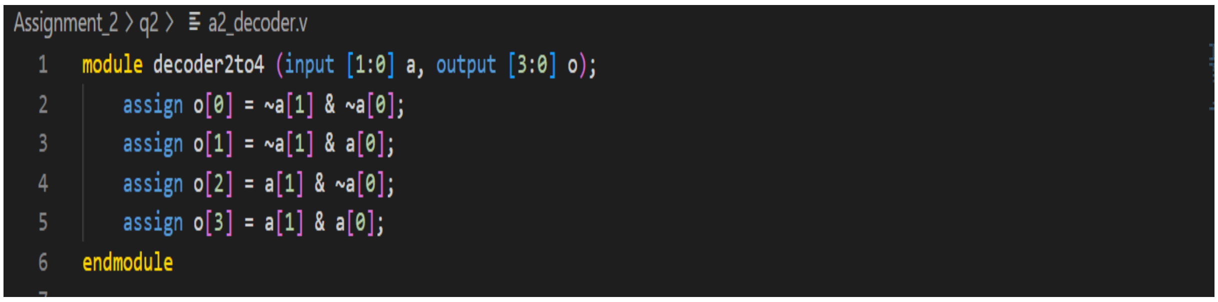


```
Assignment_2 > q2 > a2_encoder.v
module encoder4to2 (input [3:0] a, output [1:0] o);
    assign o[1] = ~a[3] & a[2] & ~a[1] & ~a[0] | a[3] & ~a[2] & ~a[1] & ~a[0];
    assign o[0] = a[3] & ~a[2] & ~a[1] & ~a[0] | ~a[3] & ~a[2] & a[1] & ~a[0];
endmodule
```



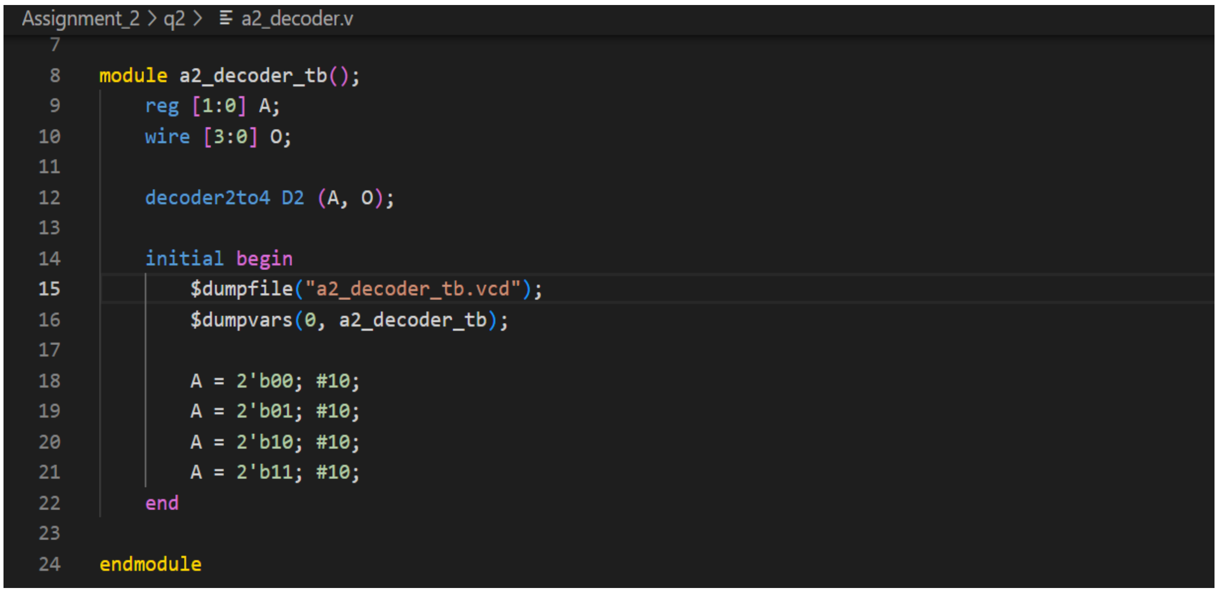
```verilog
module a2_encoder_tb();

    reg [3:0] A;
    wire [1:0] O;

    encoder4to2 E2 (A, O);

    initial begin
        $dumpfile("a2_encoder_tb.vcd");
        $dumpvars(0, a2_encoder_tb);

        A = 4'b0001; #10;
        A = 4'b0010; #10;
        A = 4'b0100; #10;
        A = 4'b1000; #10;
    end

endmodule
```





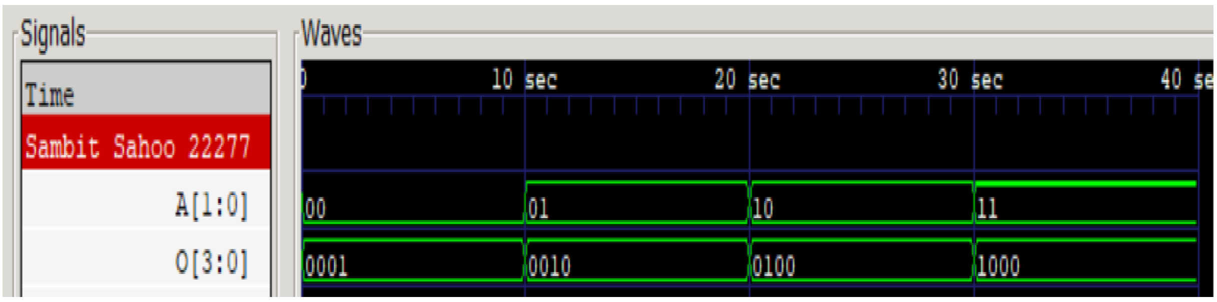
Assignment_2 > q2 > a2_decoder.v

```verilog
module decoder2to4 (input [1:0] a, output [3:0] o);
    assign o[0] = ~a[1] & ~a[0];
    assign o[1] = ~a[1] & a[0];
    assign o[2] = a[1] & ~a[0];
    assign o[3] = a[1] & a[0];
endmodule
```

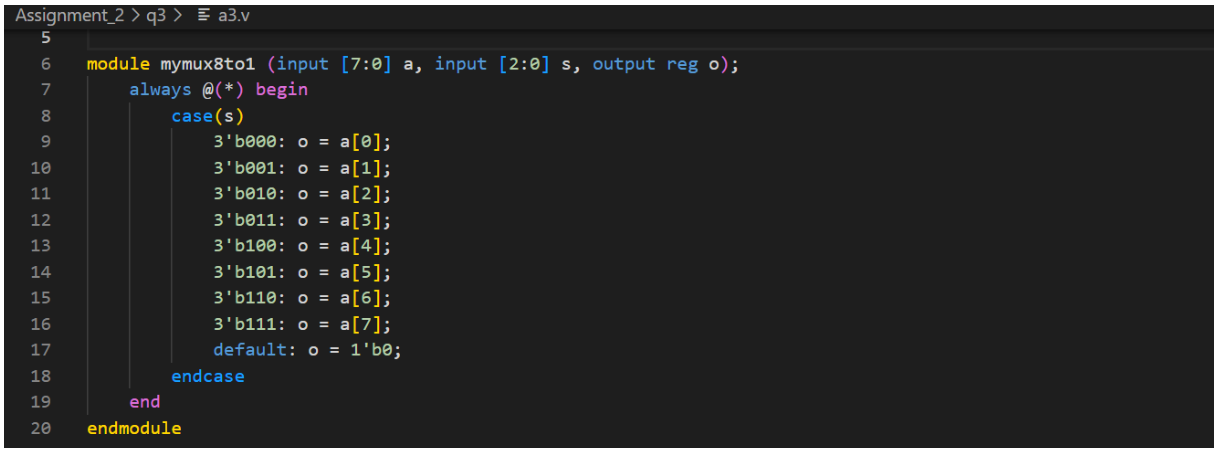


Assignment_2 > q2 > a2_decoder.v

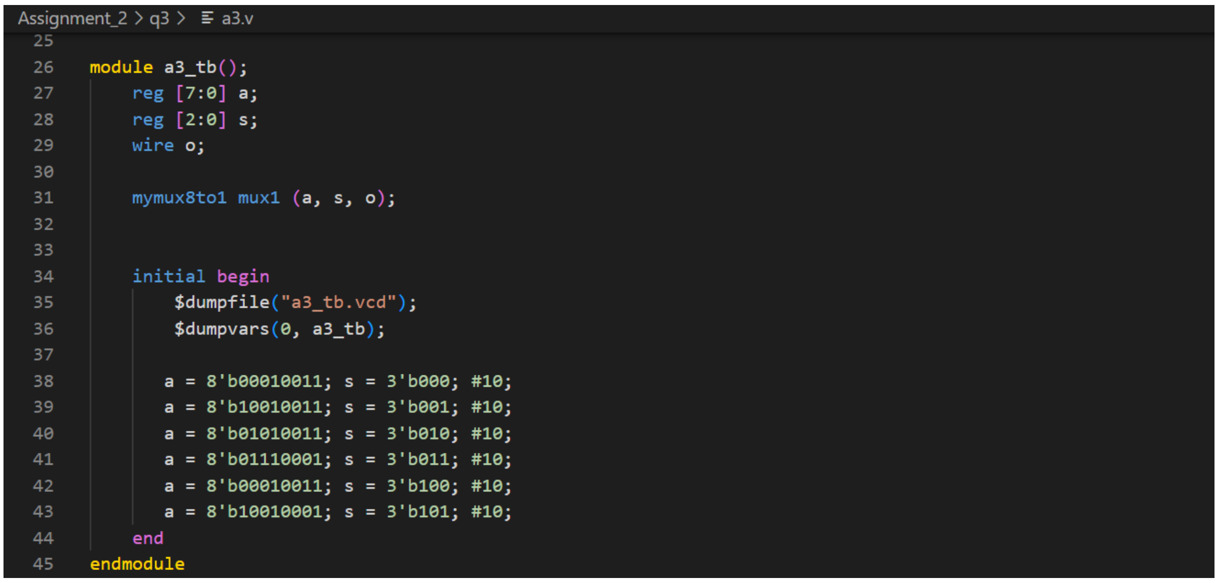
```verilog

module a2_decoder_tb();
    reg [1:0] A;
    wire [3:0] O;

    decoder2to4 D2 (A, O);

    initial begin
        $dumpfile("a2_decoder_tb.vcd");
        $dumpvars(0, a2_decoder_tb);

        A = 2'b00; #10;
        A = 2'b01; #10;
        A = 2'b10; #10;
        A = 2'b11; #10;
    end

endmodule
```



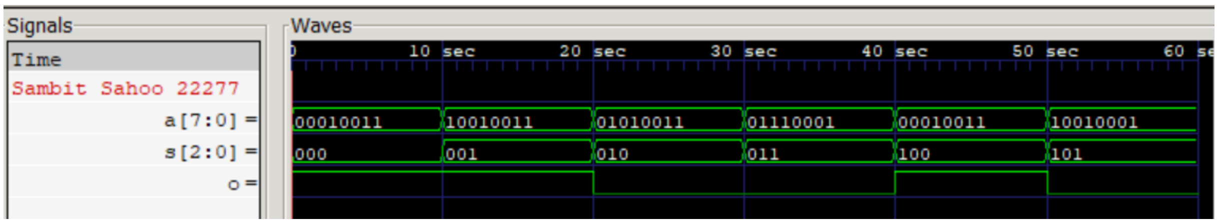
Q3- 8-to-1 multiplexer using a case statement



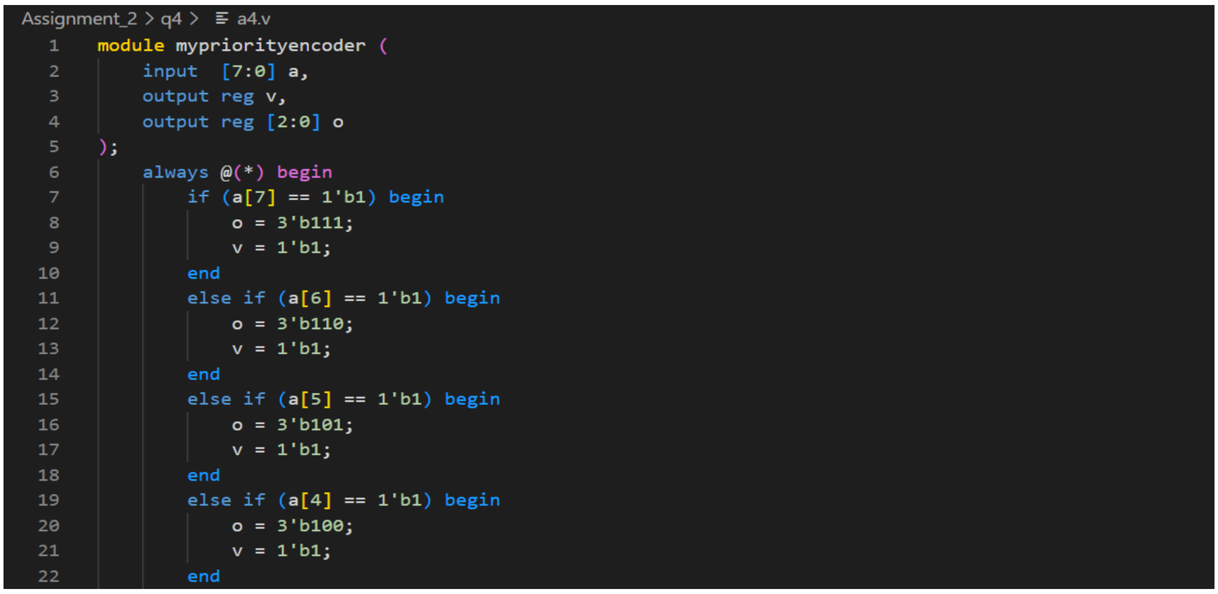
```verilog

module mymux8to1 (input [7:0] a, input [2:0] s, output reg o);
    always @(*) begin
        case(s)
            3'b000: o = a[0];
            3'b001: o = a[1];
            3'b010: o = a[2];
            3'b011: o = a[3];
            3'b100: o = a[4];
            3'b101: o = a[5];
            3'b110: o = a[6];
            3'b111: o = a[7];
            default: o = 1'b0;
        endcase
    end
endmodule
```



```verilog

module a3_tb();
    reg [7:0] a;
    reg [2:0] s;
    wire o;

    mymux8to1 mux1 (a, s, o);


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

        a = 8'b00010011; s = 3'b000; #10;
        a = 8'b10010011; s = 3'b001; #10;
        a = 8'b01010011; s = 3'b010; #10;
        a = 8'b01110001; s = 3'b011; #10;
        a = 8'b00010011; s = 3'b100; #10;
        a = 8'b10010001; s = 3'b101; #10;
    end
endmodule
```

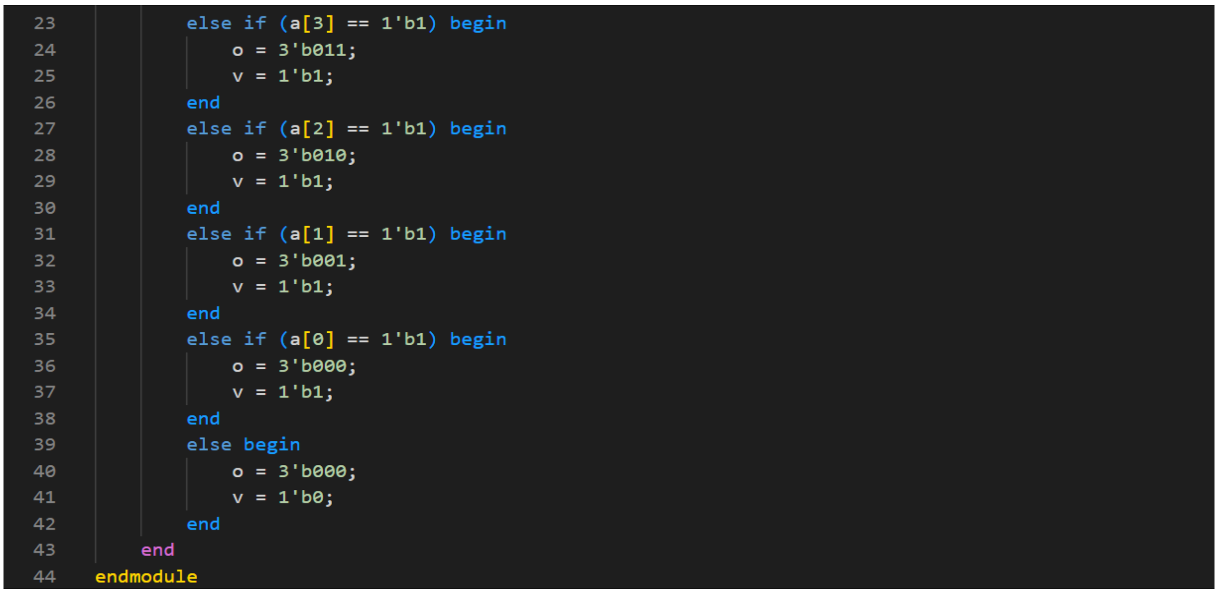


Q4 - 8-to-3 priority encoder using the if else statement

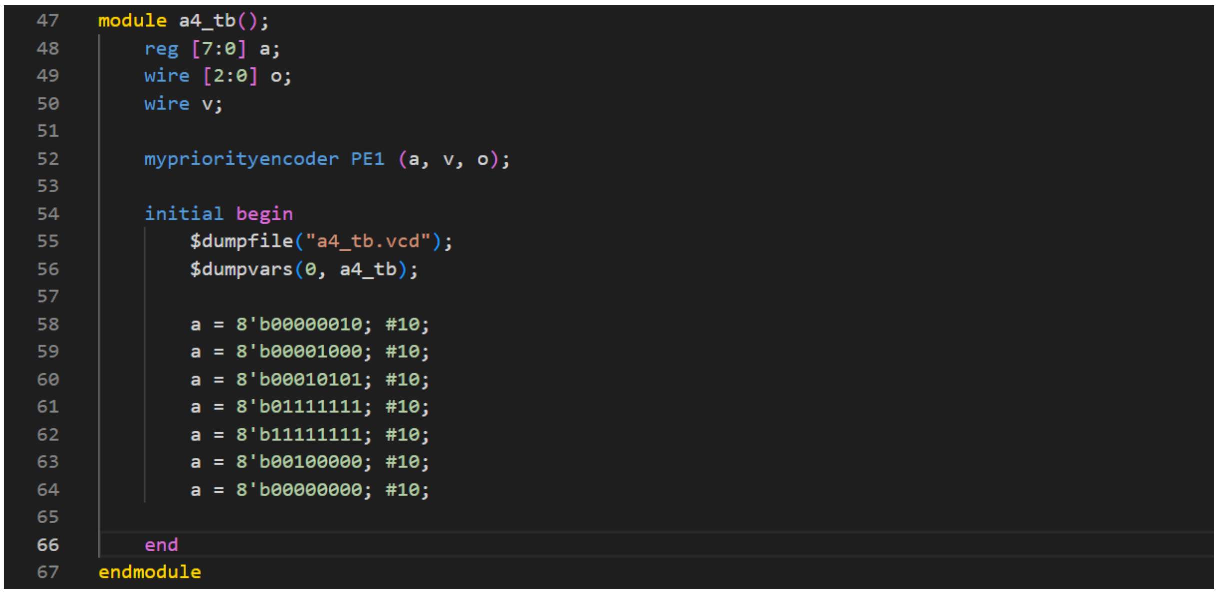


Assignment_2 > q4 > a4.v

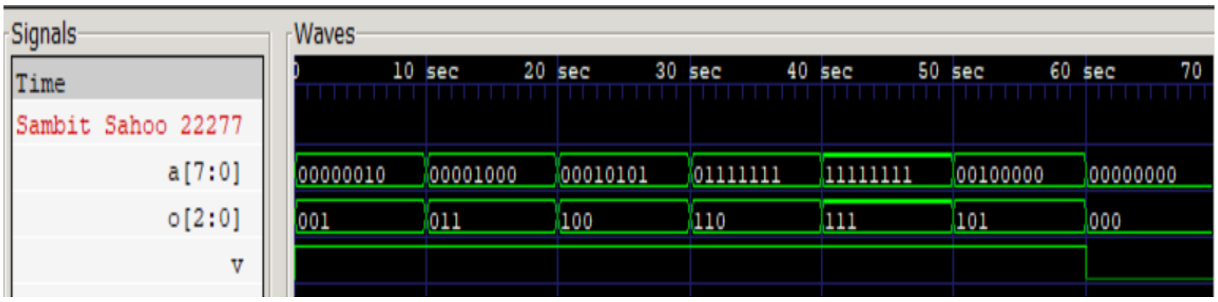
```verilog
module mypriorityencoder (
    input  [7:0] a,
    output reg v,
    output reg [2:0] o
);
    always @(*) begin
        if (a[7] == 1'b1) begin
            o = 3'b111;
            v = 1'b1;
        end
        else if (a[6] == 1'b1) begin
            o = 3'b110;
            v = 1'b1;
        end
        else if (a[5] == 1'b1) begin
            o = 3'b101;
            v = 1'b1;
        end
        else if (a[4] == 1'b1) begin
            o = 3'b100;
            v = 1'b1;
        end
```



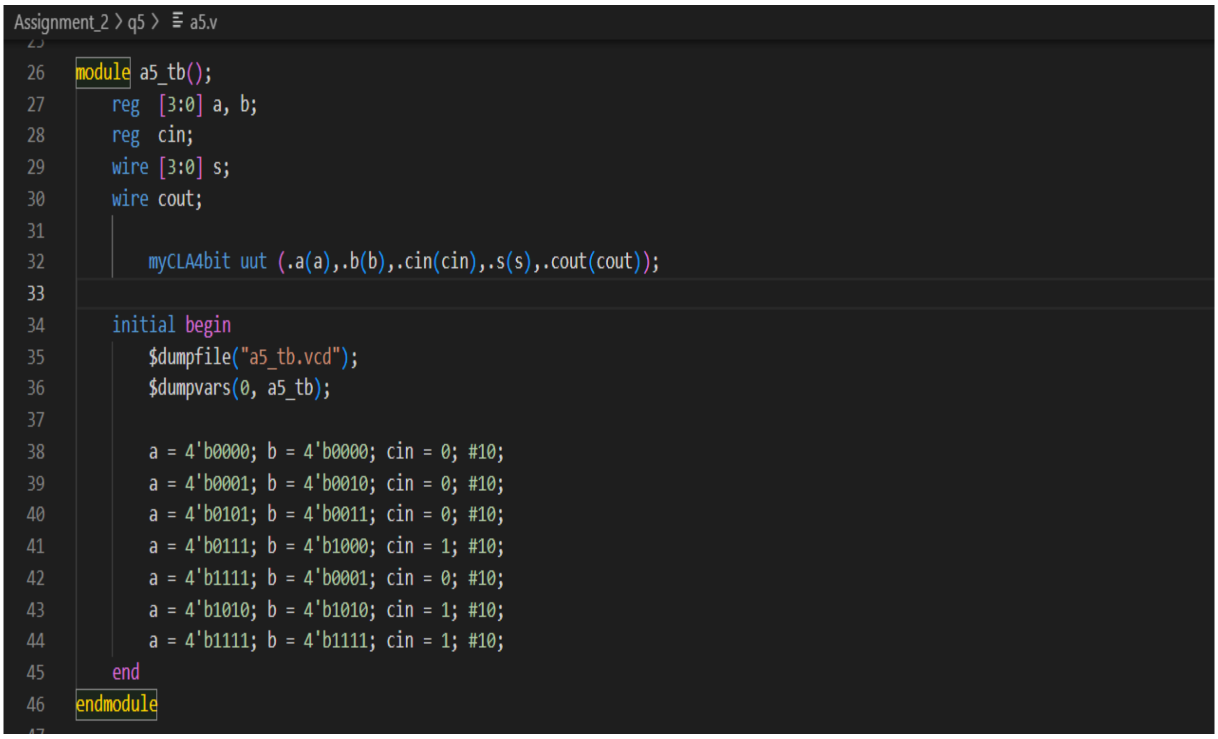
```verilog
        else if (a[3] == 1'b1) begin
            o = 3'b011;
            v = 1'b1;
        end
        else if (a[2] == 1'b1) begin
            o = 3'b010;
            v = 1'b1;
        end
        else if (a[1] == 1'b1) begin
            o = 3'b001;
            v = 1'b1;
        end
        else if (a[0] == 1'b1) begin
            o = 3'b000;
            v = 1'b1;
        end
        else begin
            o = 3'b000;
            v = 1'b0;
        end
    end
endmodule
```



```verilog
module a4_tb();
    reg [7:0] a;
    wire [2:0] o;
    wire v;

    mypriorityencoder PE1 (a, v, o);

    initial begin
        $dumpfile("a4_tb.vcd");
        $dumpvars(0, a4_tb);

        a = 8'b00000010; #10;
        a = 8'b00001000; #10;
        a = 8'b00010101; #10;
        a = 8'b01111111; #10;
        a = 8'b11111111; #10;
        a = 8'b00100000; #10;
        a = 8'b00000000; #10;

    end
endmodule
```

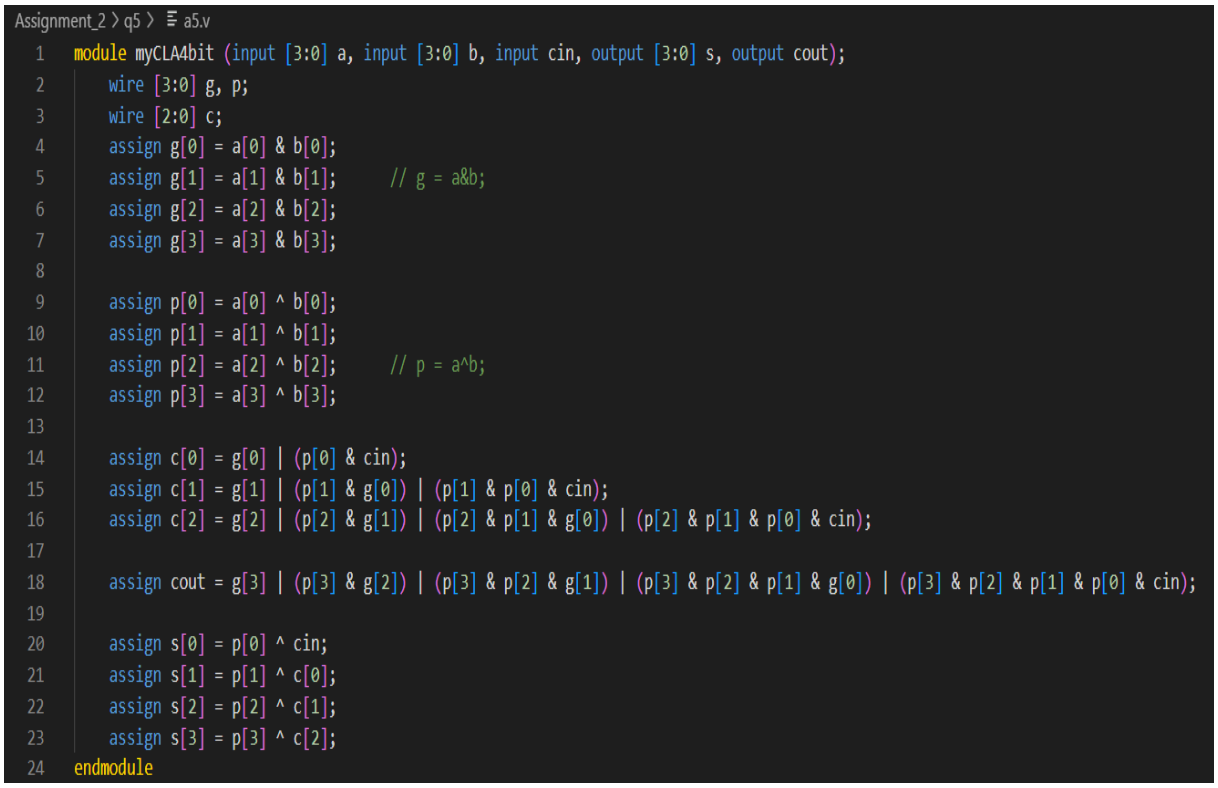


Q5- 4-bit Carry Look-Ahead Adder



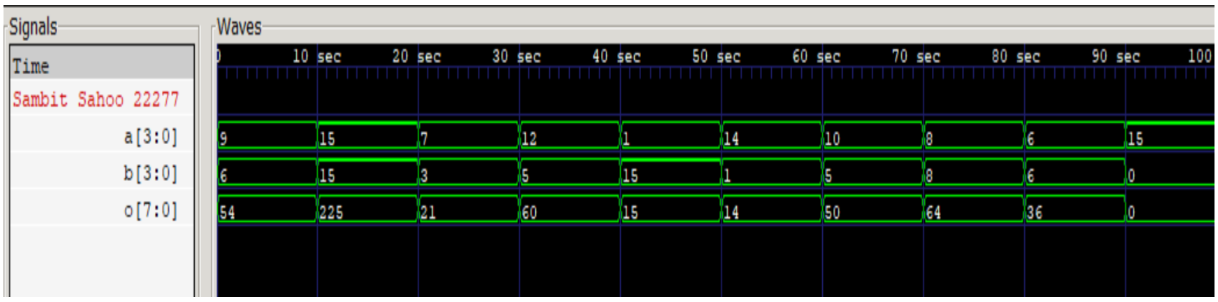
Assignment_2 > q5 > a5.v

```verilog
module a5_tb();
    reg  [3:0] a, b;
    reg  cin;
    wire [3:0] s;
    wire cout;

        myCLA4bit uut (.a(a),.b(b),.cin(cin),.s(s),.cout(cout));

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

        a = 4'b0000; b = 4'b0000; cin = 0; #10;
        a = 4'b0001; b = 4'b0010; cin = 0; #10;
        a = 4'b0101; b = 4'b0011; cin = 0; #10;
        a = 4'b0111; b = 4'b1000; cin = 1; #10;
        a = 4'b1111; b = 4'b0001; cin = 0; #10;
        a = 4'b1010; b = 4'b1010; cin = 1; #10;
        a = 4'b1111; b = 4'b1111; cin = 1; #10;
    end
endmodule
```



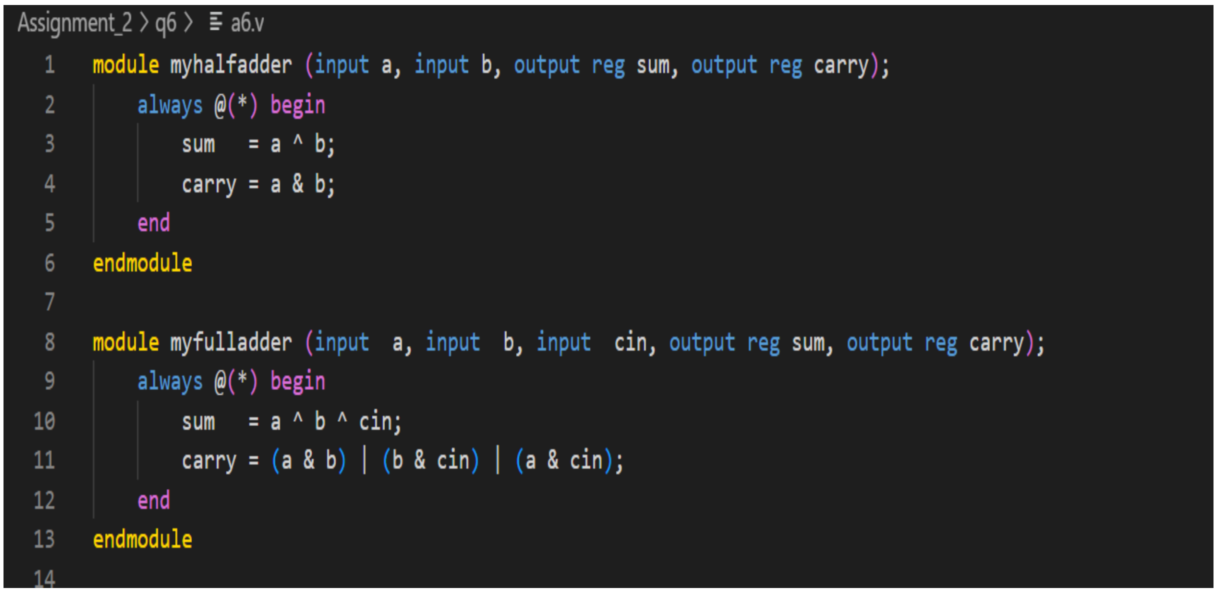
Assignment_2 > q5 > a5.v

```verilog
module myCLA4bit (input [3:0] a, input [3:0] b, input cin, output [3:0] s, output cout);
    wire [3:0] g, p;
    wire [2:0] c;
    assign g[0] = a[0] & b[0];
    assign g[1] = a[1] & b[1];      // g = a&b;
    assign g[2] = a[2] & b[2];
    assign g[3] = a[3] & b[3];

    assign p[0] = a[0] ^ b[0];
    assign p[1] = a[1] ^ b[1];
    assign p[2] = a[2] ^ b[2];      // p = a^b;
    assign p[3] = a[3] ^ b[3];

    assign c[0] = g[0] | (p[0] & cin);
    assign c[1] = g[1] | (p[1] & g[0]) | (p[1] & p[0] & cin);
    assign c[2] = g[2] | (p[2] & g[1]) | (p[2] & p[1] & g[0]) | (p[2] & p[1] & p[0] & cin);

    assign cout = g[3] | (p[3] & g[2]) | (p[3] & p[2] & g[1]) | (p[3] & p[2] & p[1] & g[0]) | (p[3] & p[2] & p[1] & p[0] & cin);

    assign s[0] = p[0] ^ cin;
    assign s[1] = p[1] ^ c[0];
    assign s[2] = p[2] ^ c[1];
    assign s[3] = p[3] ^ c[2];
endmodule
```

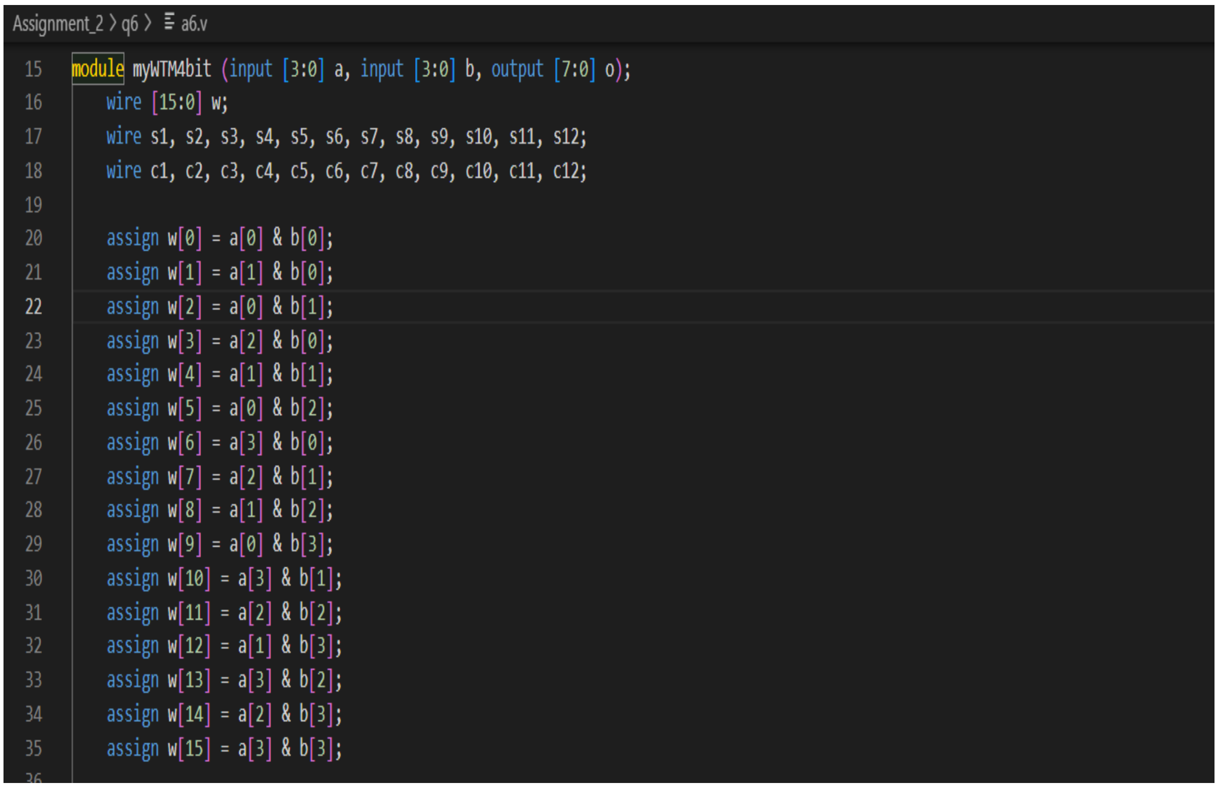


```
a[3:0]  9   15   7   12  1   14  10  8   6   15
b[3:0]  6   15   3   5   15  1   5   8   6   0
o[7:0]  54  225  21  60  15  14  50  64  36  0
```

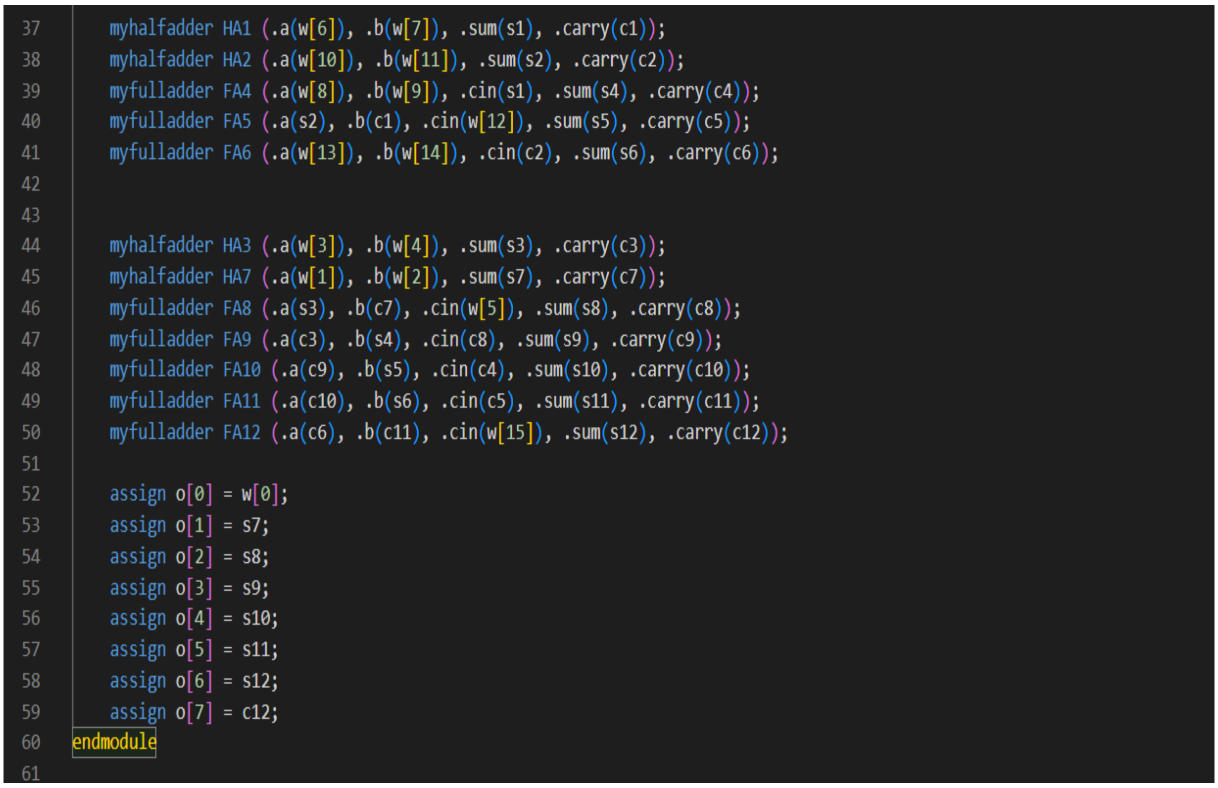
Q6- 4-bit Wallace Tree Multiplier


Assignment_2 > q6 > a6.v

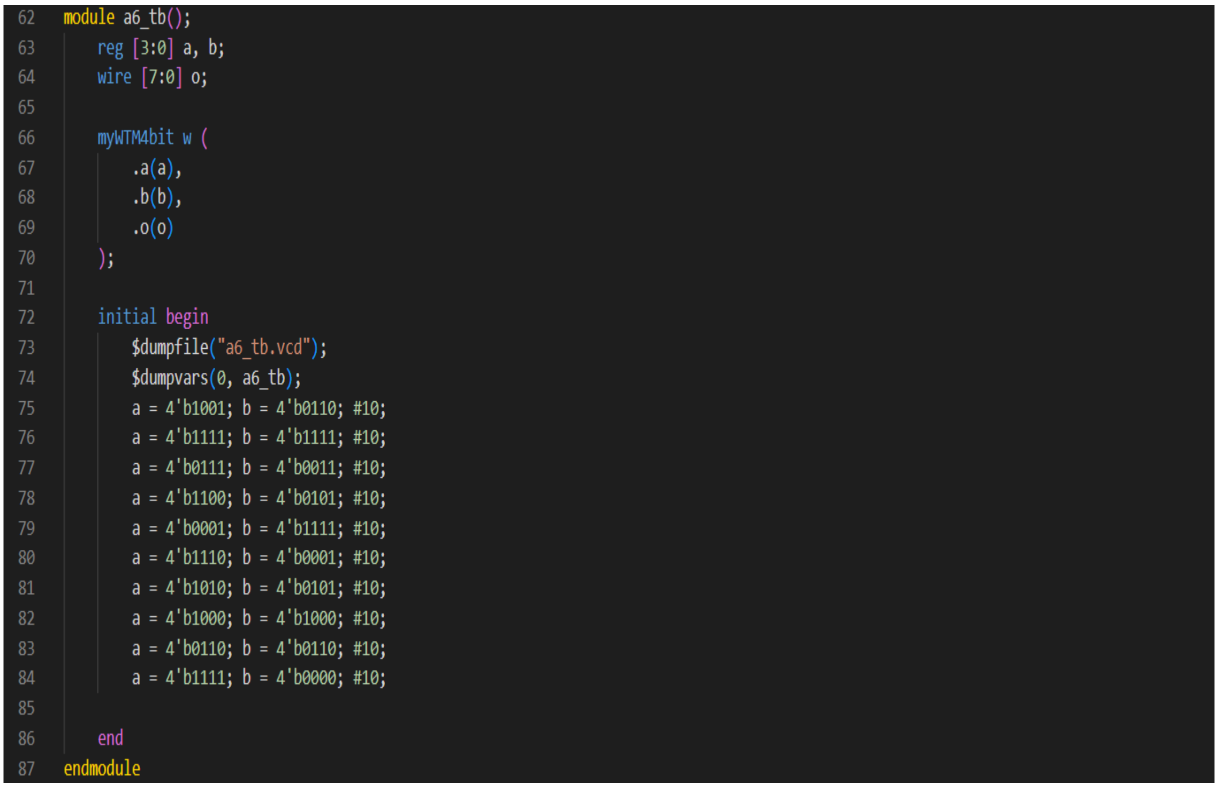
```verilog
module myhalfadder (input a, input b, output reg sum, output reg carry);
    always @(*) begin
        sum   = a ^ b;
        carry = a & b;
    end
endmodule

module myfulladder (input  a, input  b, input  cin, output reg sum, output reg carry);
    always @(*) begin
        sum   = a ^ b ^ cin;
        carry = (a & b) | (b & cin) | (a & cin);
    end
endmodule

```


Assignment_2 > q6 > a6.v

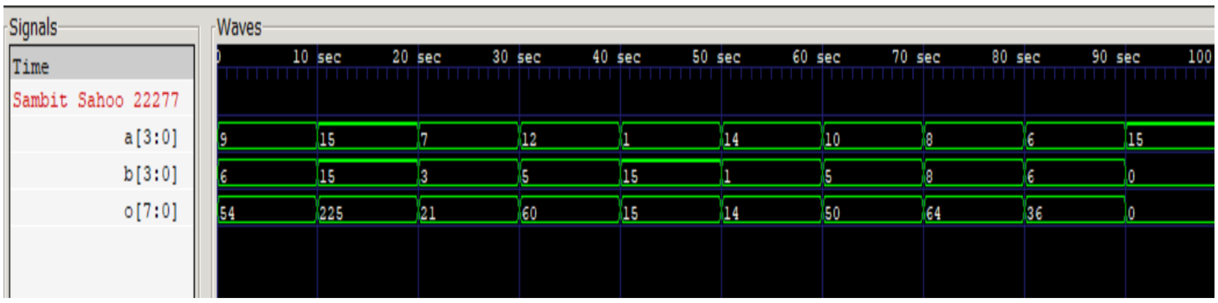
```verilog
module myWTM4bit (input [3:0] a, input [3:0] b, output [7:0] o);
    wire [15:0] w;
    wire s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12;
    wire c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12;

    assign w[0] = a[0] & b[0];
    assign w[1] = a[1] & b[0];
    assign w[2] = a[0] & b[1];
    assign w[3] = a[2] & b[0];
    assign w[4] = a[1] & b[1];
    assign w[5] = a[0] & b[2];
    assign w[6] = a[3] & b[0];
    assign w[7] = a[2] & b[1];
    assign w[8] = a[1] & b[2];
    assign w[9] = a[0] & b[3];
    assign w[10] = a[3] & b[1];
    assign w[11] = a[2] & b[2];
    assign w[12] = a[1] & b[3];
    assign w[13] = a[3] & b[2];
    assign w[14] = a[2] & b[3];
    assign w[15] = a[3] & b[3];
```



```verilog
    myhalfadder HA1 (.a(w[6]), .b(w[7]), .sum(s1), .carry(c1));
    myhalfadder HA2 (.a(w[10]), .b(w[11]), .sum(s2), .carry(c2));
    myfulladder FA4 (.a(w[8]), .b(w[9]), .cin(s1), .sum(s4), .carry(c4));
    myfulladder FA5 (.a(s2), .b(c1), .cin(w[12]), .sum(s5), .carry(c5));
    myfulladder FA6 (.a(w[13]), .b(w[14]), .cin(c2), .sum(s6), .carry(c6));


    myhalfadder HA3 (.a(w[3]), .b(w[4]), .sum(s3), .carry(c3));
    myhalfadder HA7 (.a(w[1]), .b(w[2]), .sum(s7), .carry(c7));
    myfulladder FA8 (.a(s3), .b(c7), .cin(w[5]), .sum(s8), .carry(c8));
    myfulladder FA9 (.a(c3), .b(s4), .cin(c8), .sum(s9), .carry(c9));
    myfulladder FA10 (.a(c9), .b(s5), .cin(c4), .sum(s10), .carry(c10));
    myfulladder FA11 (.a(c10), .b(s6), .cin(c5), .sum(s11), .carry(c11));
    myfulladder FA12 (.a(c6), .b(c11), .cin(w[15]), .sum(s12), .carry(c12));

    assign o[0] = w[0];
    assign o[1] = s7;
    assign o[2] = s8;
    assign o[3] = s9;
    assign o[4] = s10;
    assign o[5] = s11;
    assign o[6] = s12;
    assign o[7] = c12;
endmodule

```



```verilog
module a6_tb();
    reg [3:0] a, b;
    wire [7:0] o;

    myWTM4bit w (
        .a(a),
        .b(b),
        .o(o)
    );

    initial begin
        $dumpfile("a6_tb.vcd");
        $dumpvars(0, a6_tb);
        a = 4'b1001; b = 4'b0110; #10;
        a = 4'b1111; b = 4'b1111; #10;
        a = 4'b0111; b = 4'b0011; #10;
        a = 4'b1100; b = 4'b0101; #10;
        a = 4'b0001; b = 4'b1111; #10;
        a = 4'b1110; b = 4'b0001; #10;
        a = 4'b1010; b = 4'b0101; #10;
        a = 4'b1000; b = 4'b1000; #10;
        a = 4'b0110; b = 4'b0110; #10;
        a = 4'b1111; b = 4'b0000; #10;

    end
endmodule
```


Signals
Waves
Time
Sambit Sahoo 22277
a[3:0]
b[3:0]
o[7:0]
10 sec
20 sec
30 sec
40 sec
50 sec
60 sec
70 sec
80 sec
90 sec