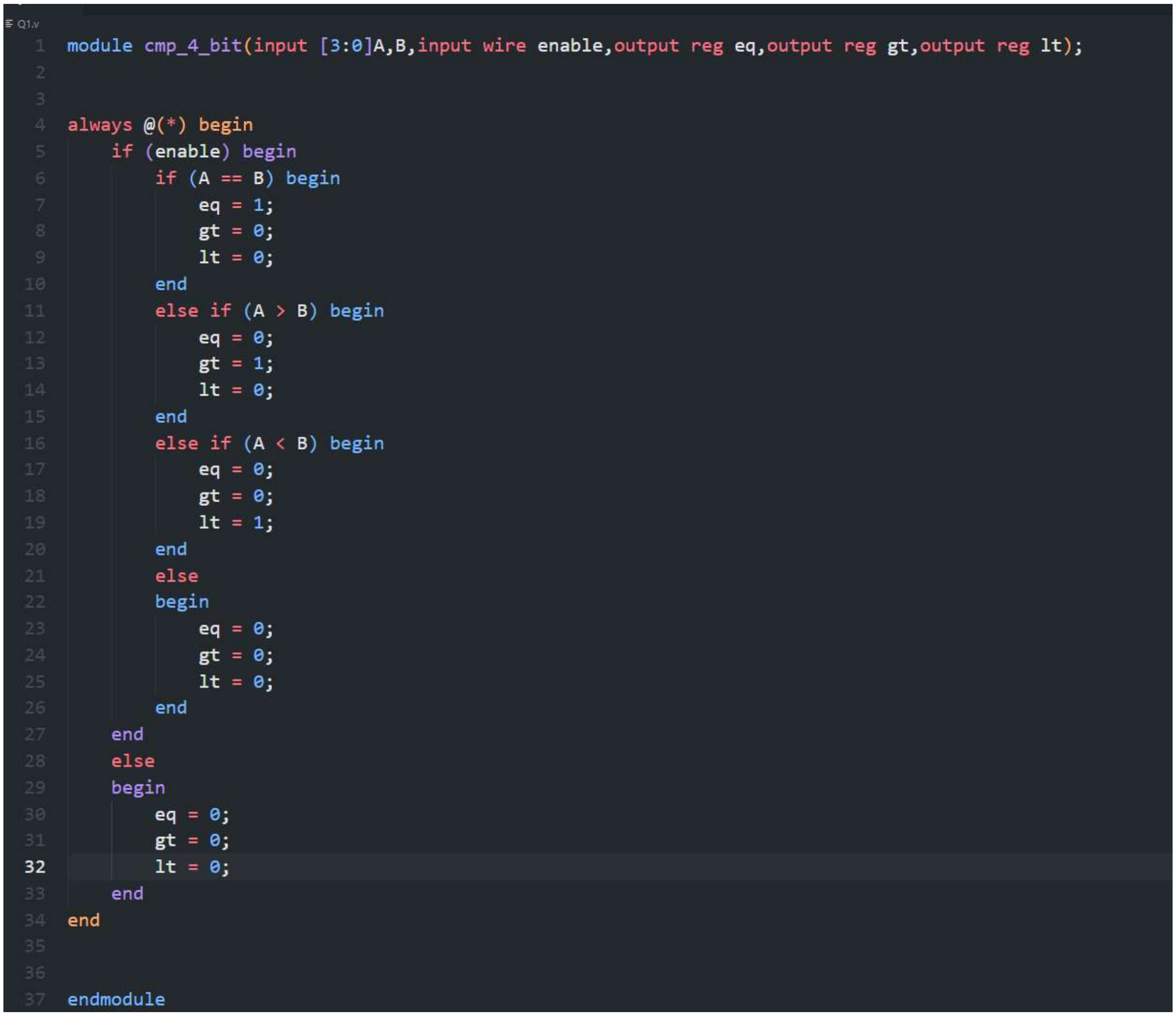
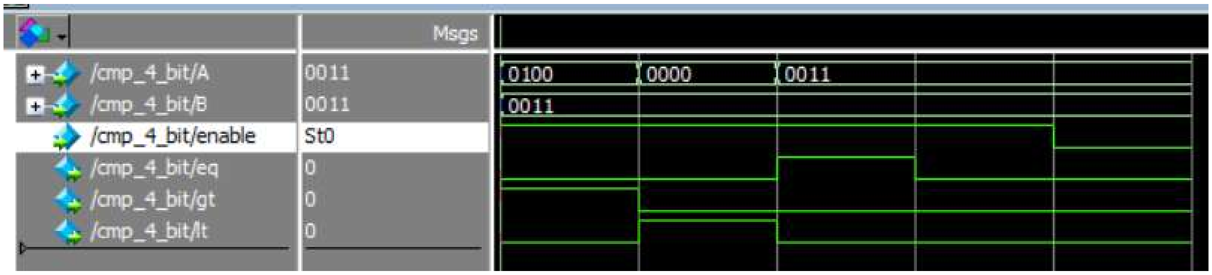
Assignment 2

Q1)
Code:



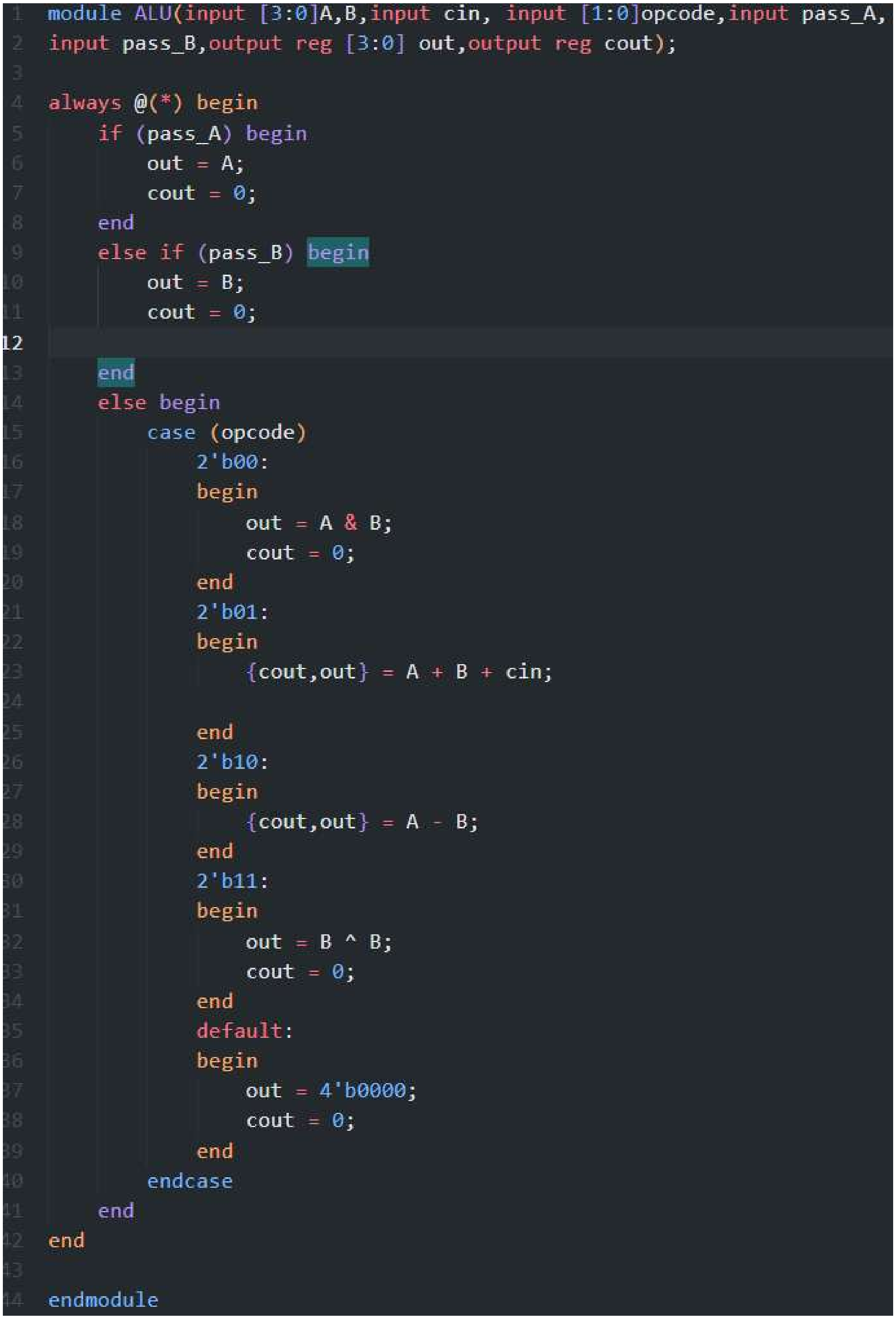
```
module cmp_4_bit(input [3:0]A,B,input wire enable,output reg eq,output reg gt,output reg lt);


always @(*) begin
    if (enable) begin
        if (A == B) begin
            eq = 1;
            gt = 0;
            lt = 0;
        end
        else if (A > B) begin
            eq = 0;
            gt = 1;
            lt = 0;
        end
        else if (A < B) begin
            eq = 0;
            gt = 0;
            lt = 1;
        end
        else
        begin
            eq = 0;
            gt = 0;
            lt = 0;
        end
    end
    else
    begin
        eq = 0;
        gt = 0;
        lt = 0;
    end
end


endmodule
```

Waveform:



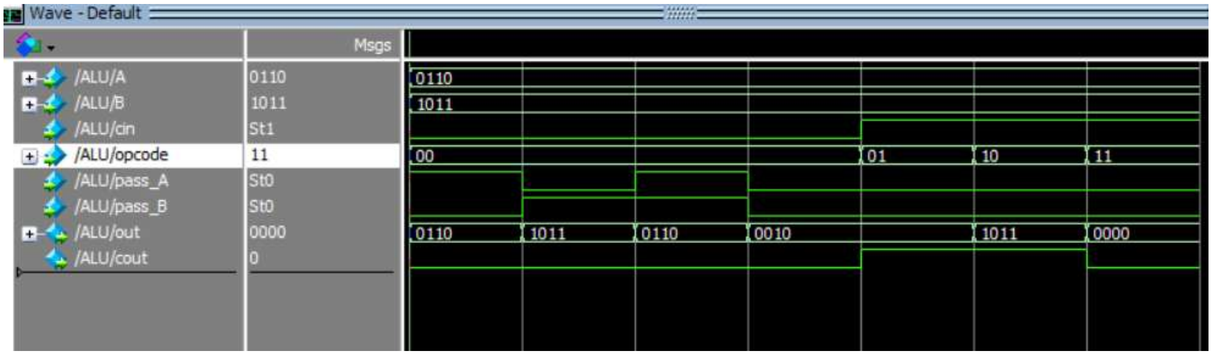
```
                     Msgs
/cmp_4_bit/A         0011
/cmp_4_bit/B         0011
/cmp_4_bit/enable    St0
/cmp_4_bit/eq        0
/cmp_4_bit/gt        0
/cmp_4_bit/lt        0
```

Q2)
Code:



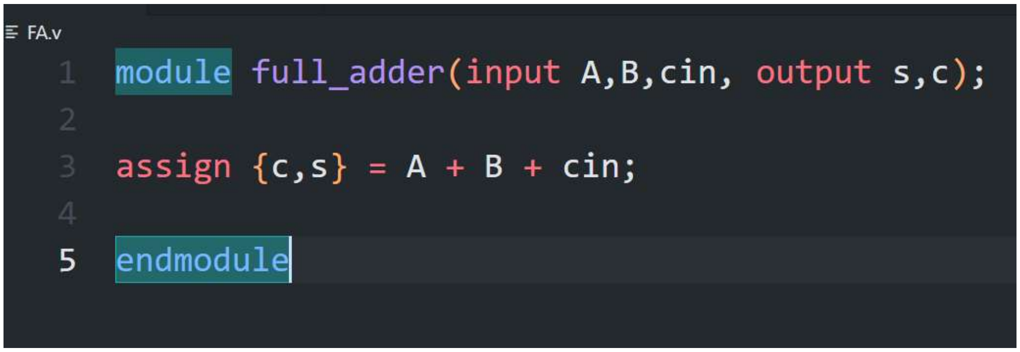
```
module ALU(input [3:0]A,B,input cin, input [1:0]opcode,input pass_A,
input pass_B,output reg [3:0] out,output reg cout);

always @(*) begin
    if (pass_A) begin
        out = A;
        cout = 0;
    end
    else if (pass_B) begin
        out = B;
        cout = 0;

    end
    else begin
        case (opcode)
            2'b00:
            begin
                out = A & B;
                cout = 0;
            end
            2'b01:
            begin
                {cout,out} = A + B + cin;

            end
            2'b10:
            begin
                {cout,out} = A - B;
            end
            2'b11:
            begin
                out = B ^ B;
                cout = 0;
            end
            default:
            begin
                out = 4'b0000;
                cout = 0;
            end
        endcase
    end
end

endmodule
```

Waveform:

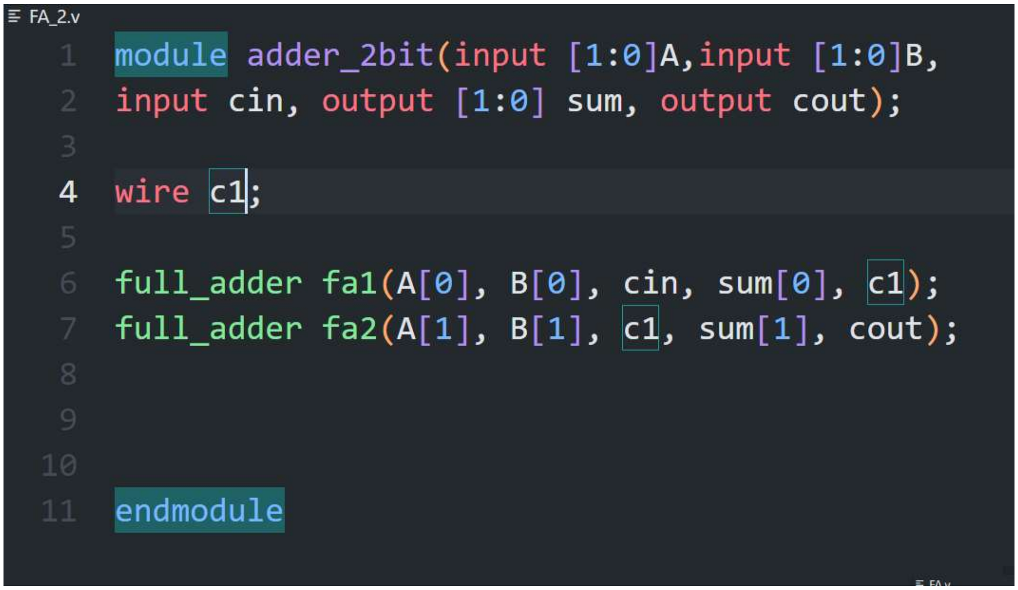


```
/ALU/A        0110
/ALU/B        1011
/ALU/cin      St1
/ALU/opcode   11
/ALU/pass_A   St0
/ALU/pass_B   St0
/ALU/out      0000
/ALU/cout     0
```

Q3)
Code:

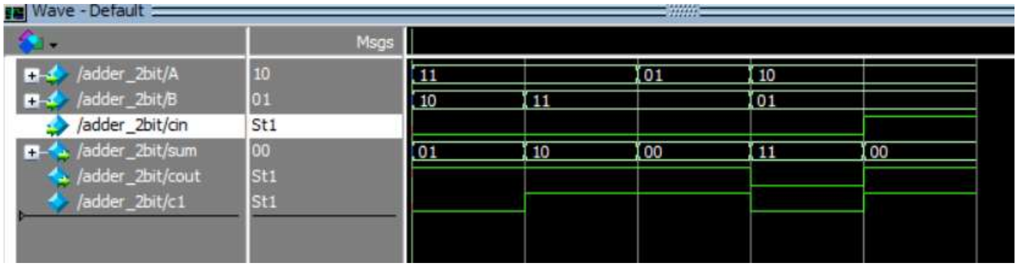


```
module full_adder(input A,B,cin, output s,c);

assign {c,s} = A + B + cin;

endmodule
```



```
module adder_2bit(input [1:0]A,input [1:0]B,
input cin, output [1:0] sum, output cout);

wire c1;

full_adder fa1(A[0], B[0], cin, sum[0], c1);
full_adder fa2(A[1], B[1], c1, sum[1], cout);



endmodule
```

Waveform:



```
/adder_2bit/A      10
/adder_2bit/B      01
/adder_2bit/cin    St1
/adder_2bit/sum    00
/adder_2bit/cout   St1
/adder_2bit/c1     St1
```