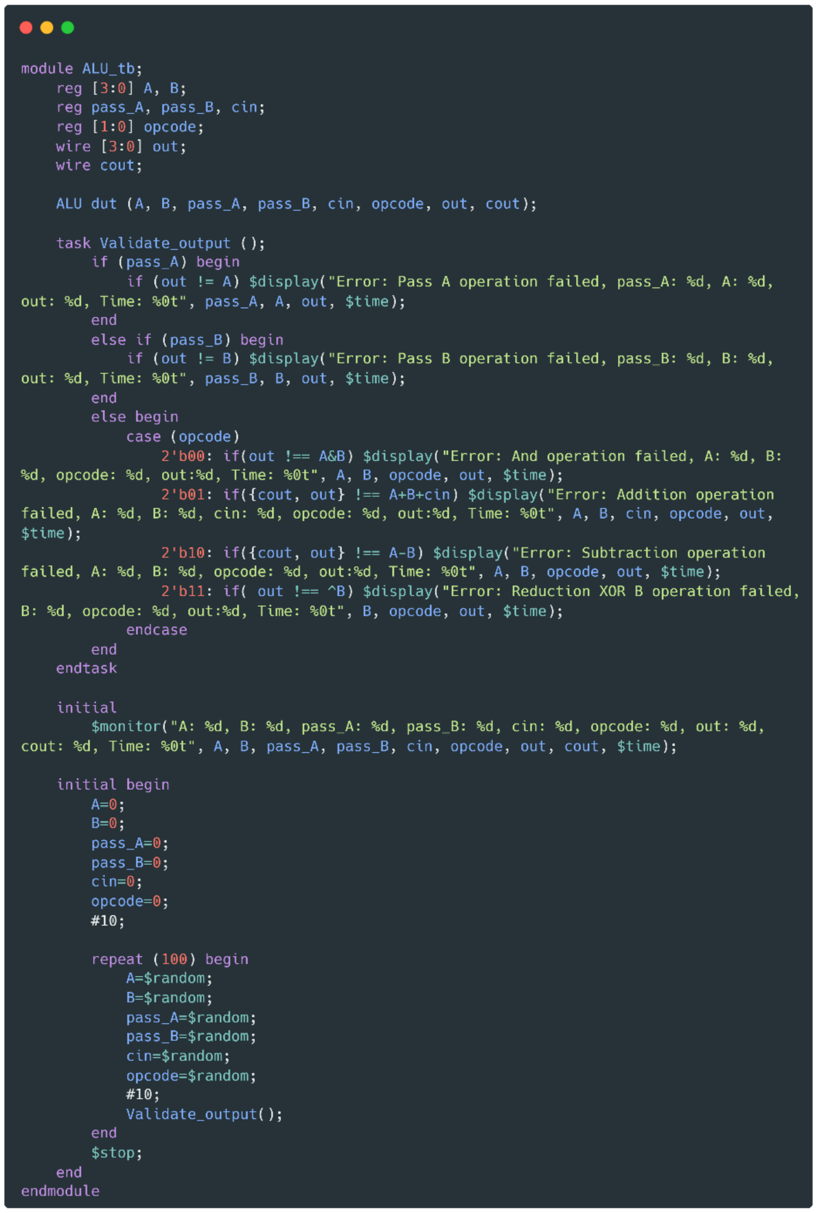
Assignment 3 – Model Answer

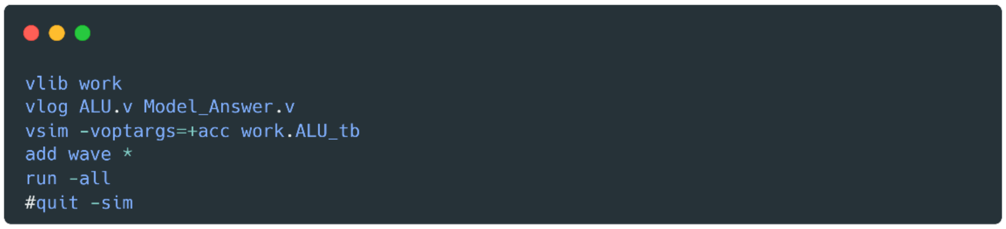
Q1)

ALU_tb code:



```
module ALU_tb;
    reg [3:0] A, B;
    reg pass_A, pass_B, cin;
    reg [1:0] opcode;
    wire [3:0] out;
    wire cout;

    ALU dut (A, B, pass_A, pass_B, cin, opcode, out, cout);

    task Validate_output ();
        if (pass_A) begin
            if (out != A) $display("Error: Pass A operation failed, pass_A: %d, A: %d,
out: %d, Time: %0t", pass_A, A, out, $time);
        end
        else if (pass_B) begin
            if (out != B) $display("Error: Pass B operation failed, pass_B: %d, B: %d,
out: %d, Time: %0t", pass_B, B, out, $time);
        end
        else begin
            case (opcode)
                2'b00: if(out !== A&B) $display("Error: And operation failed, A: %d, B:
%d, opcode: %d, out:%d, Time: %0t", A, B, opcode, out, $time);
                2'b01: if({cout, out} !== A+B+cin) $display("Error: Addition operation
failed, A: %d, B: %d, cin: %d, opcode: %d, out:%d, Time: %0t", A, B, cin, opcode, out,
$time);
                2'b10: if({cout, out} !== A-B) $display("Error: Subtraction operation
failed, A: %d, B: %d, opcode: %d, out:%d, Time: %0t", A, B, opcode, out, $time);
                2'b11: if( out !== ^B) $display("Error: Reduction XOR B operation failed,
B: %d, opcode: %d, out:%d, Time: %0t", B, opcode, out, $time);
            endcase
        end
    endtask

    initial
        $monitor("A: %d, B: %d, pass_A: %d, pass_B: %d, cin: %d, opcode: %d, out: %d,
cout: %d, Time: %0t", A, B, pass_A, pass_B, cin, opcode, out, cout, $time);

    initial begin
        A=0;
        B=0;
        pass_A=0;
        pass_B=0;
        cin=0;
        opcode=0;
        #10;

        repeat (100) begin
            A=$random;
            B=$random;
            pass_A=$random;
            pass_B=$random;
            cin=$random;
            opcode=$random;
            #10;
            Validate_output();
        end
        $stop;
    end
endmodule
```

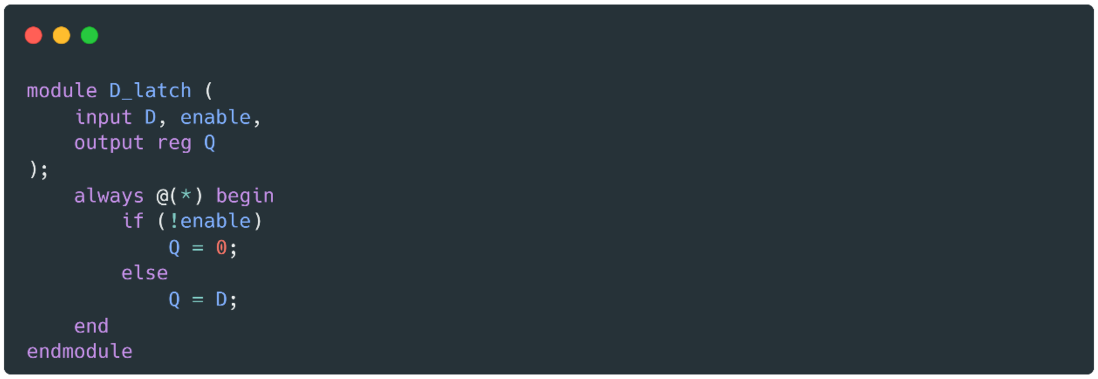
Do file code:



```
vlib work
vlog ALU.v Model_Answer.v
vsim -voptargs=+acc work.ALU_tb
add wave *
run -all
#quit -sim
```

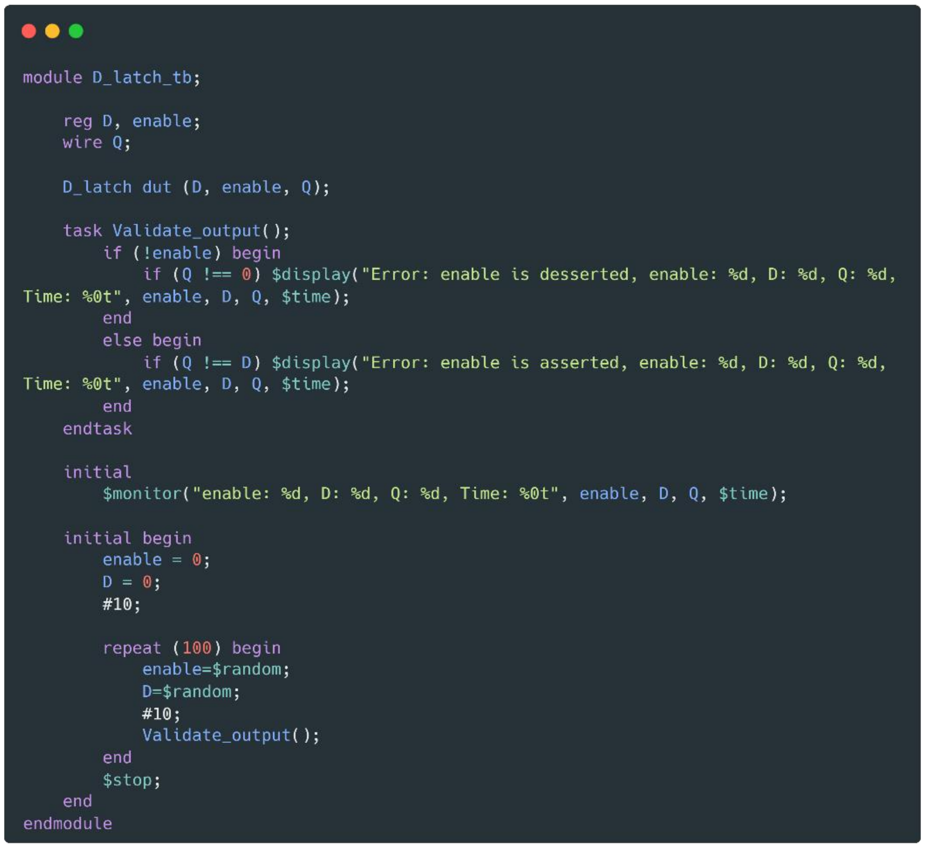
Q2)

D_latch code:



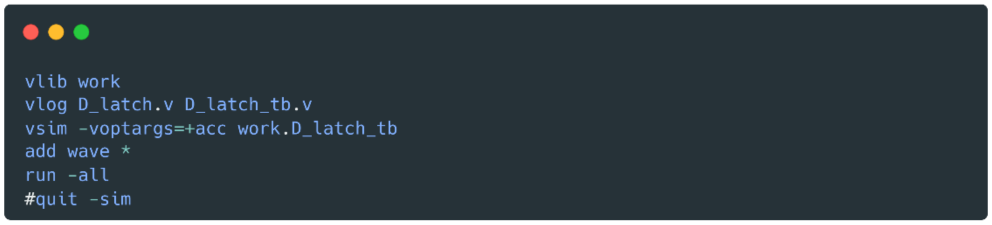
```
module D_latch (
    input D, enable,
    output reg Q
);
    always @(*) begin
        if (!enable)
            Q = 0;
        else
            Q = D;
    end
endmodule
```

D_latch_tb code:



```
module D_latch_tb;

    reg D, enable;
    wire Q;

    D_latch dut (D, enable, Q);

    task Validate_output();
        if (!enable) begin
            if (Q !== 0) $display("Error: enable is desserted, enable: %d, D: %d, Q: %d,
Time: %0t", enable, D, Q, $time);
        end
        else begin
            if (Q !== D) $display("Error: enable is asserted, enable: %d, D: %d, Q: %d,
Time: %0t", enable, D, Q, $time);
        end
    endtask

    initial
        $monitor("enable: %d, D: %d, Q: %d, Time: %0t", enable, D, Q, $time);

    initial begin
        enable = 0;
        D = 0;
        #10;

        repeat (100) begin
            enable=$random;
            D=$random;
            #10;
            Validate_output();
        end
        $stop;
    end
endmodule
```

Do file code:



```
vlib work
vlog D_latch.v D_latch_tb.v
vsim -voptargs=+acc work.D_latch_tb
add wave *
run -all
#quit -sim
```

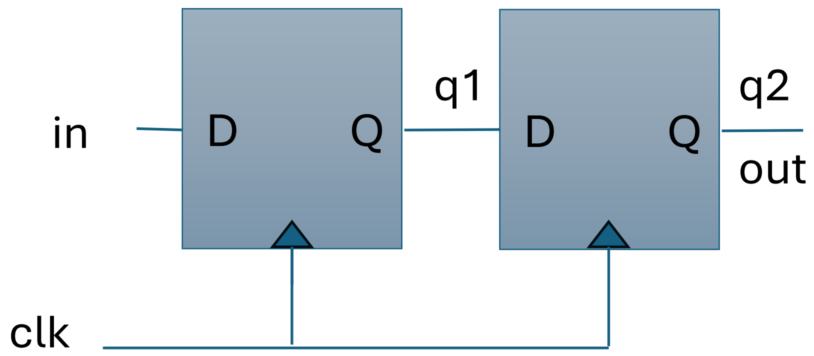
Q3)

D_t_ff:

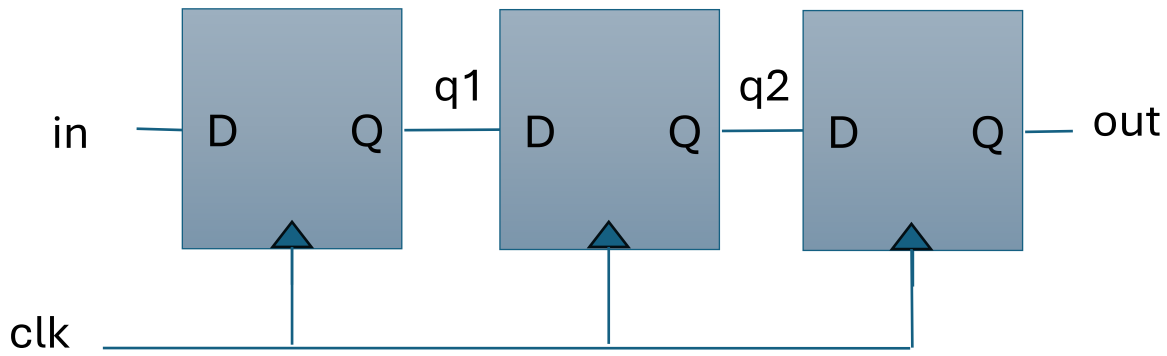
```
module d_t_flipflop (
    D, rst_n, clk, Q, Qbar
);
    parameter FF_TYPE = "DFF";
    input D, rst_n, clk;
    output reg Q;
    output Qbar;
    assign Qbar = ~Q;
    always @(posedge clk or negedge rst_n) begin
        if (~rst_n)
            Q <= 1'b0;
        else begin
            if ("DFF" == FF_TYPE)
                Q <= D;
            else if (D)
                Q <= ~Q;
        end
    end
endmodule
```

Q4)

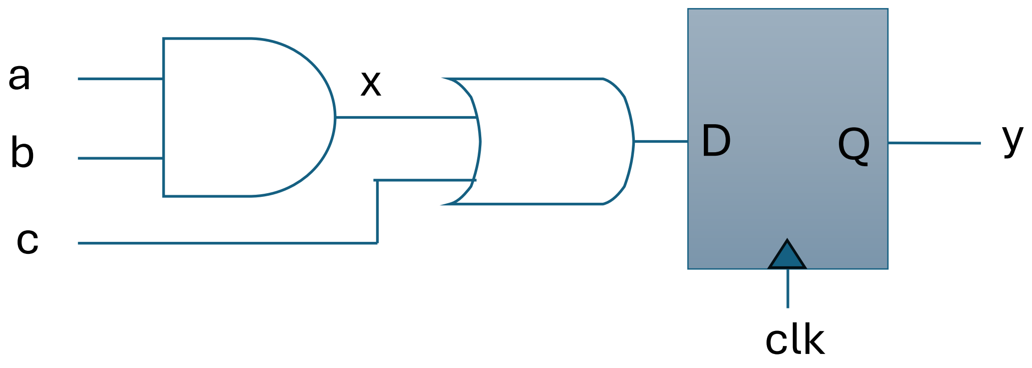
reg_blocking:



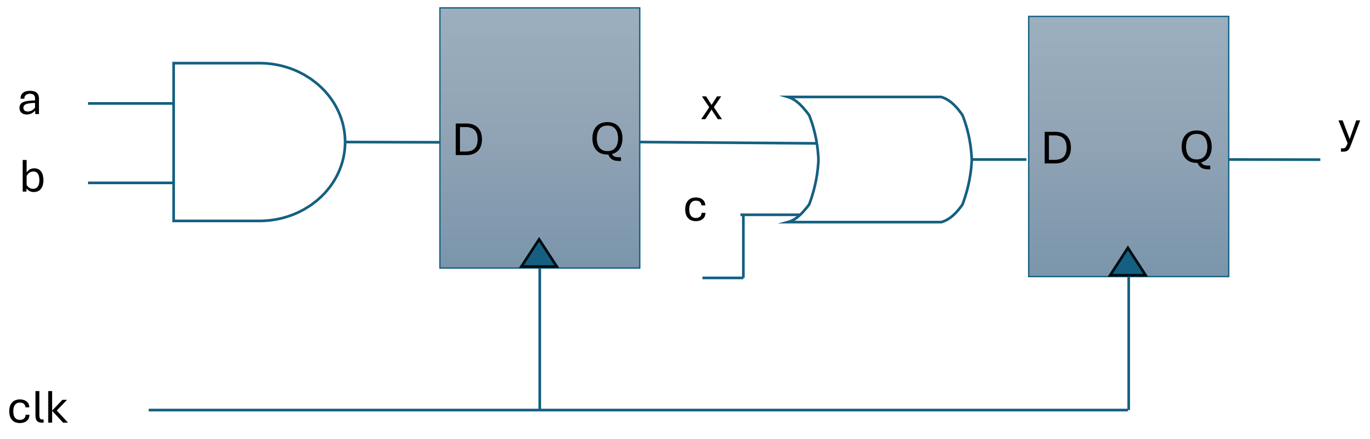
reg_nonblocking:



comb_blocking:



comb_nonblocking::


a
b
D
Q
x
c
D
Q
y
clk