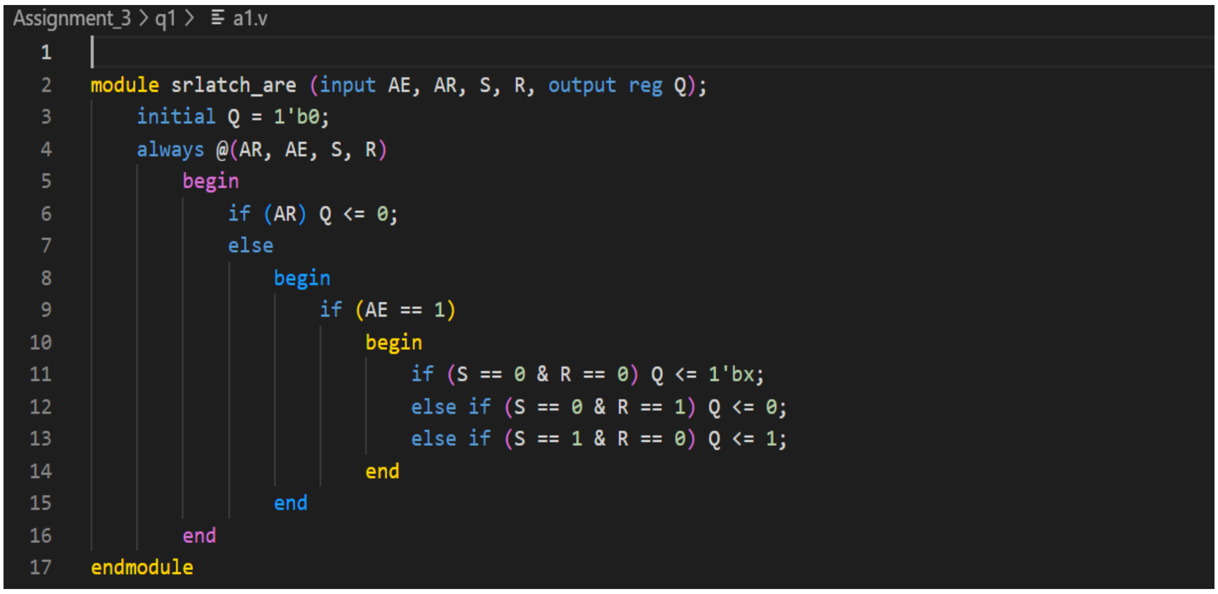
# Computer Organization
## Assignment 3
Sambit Sahoo (22277)

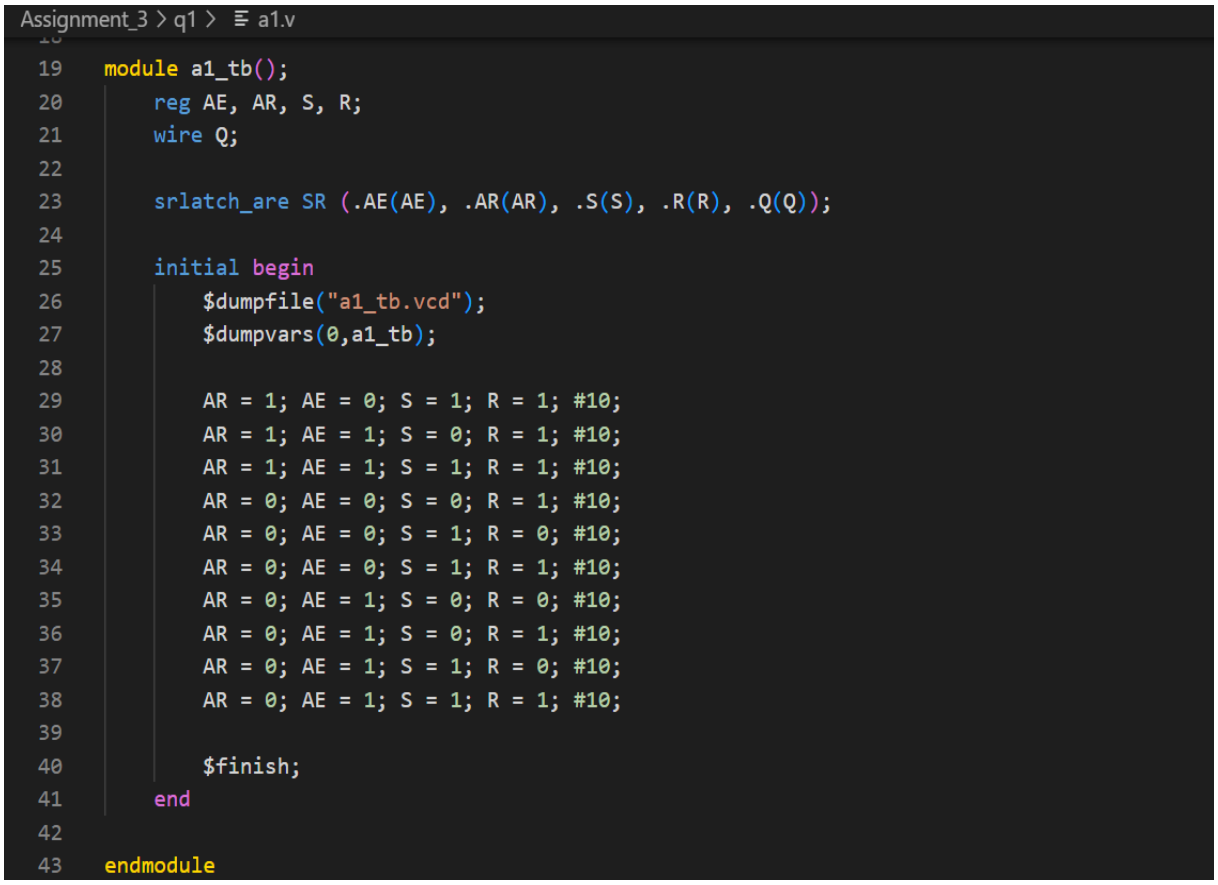
---

## Q1 - SR latch with asynchronous enable and reset



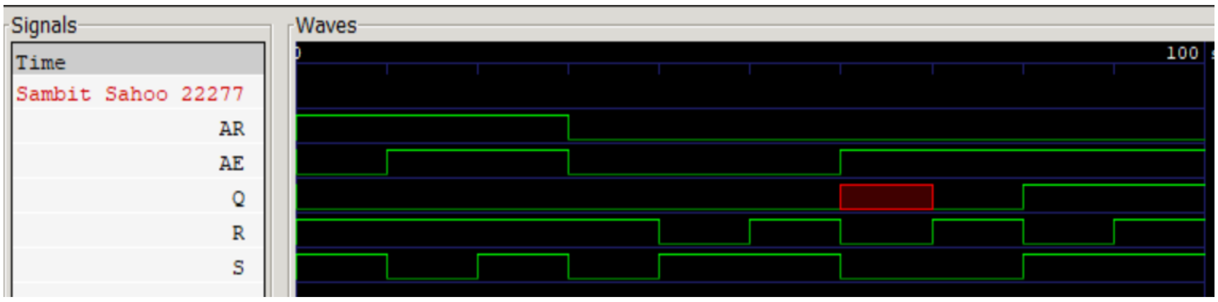
Assignment_3 > q1 > a1.v

```verilog

module srlatch_are (input AE, AR, S, R, output reg Q);
    initial Q = 1'b0;
    always @(AR, AE, S, R)
        begin
            if (AR) Q <= 0;
            else
                begin
                    if (AE == 1)
                        begin
                            if (S == 0 & R == 0) Q <= 1'bx;
                            else if (S == 0 & R == 1) Q <= 0;
                            else if (S == 1 & R == 0) Q <= 1;
                        end
                end
        end
endmodule
```

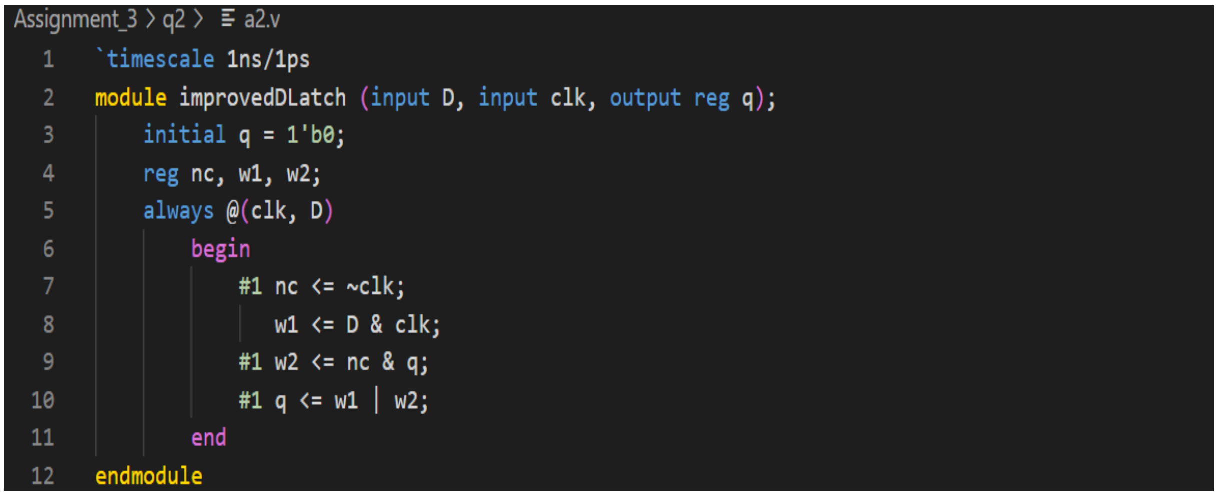


Assignment_3 > q1 > a1.v

```verilog
module a1_tb();
    reg AE, AR, S, R;
    wire Q;

    srlatch_are SR (.AE(AE), .AR(AR), .S(S), .R(R), .Q(Q));

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

        AR = 1; AE = 0; S = 1; R = 1; #10;
        AR = 1; AE = 1; S = 0; R = 1; #10;
        AR = 1; AE = 1; S = 1; R = 1; #10;
        AR = 0; AE = 0; S = 0; R = 1; #10;
        AR = 0; AE = 0; S = 1; R = 0; #10;
        AR = 0; AE = 0; S = 1; R = 1; #10;
        AR = 0; AE = 1; S = 0; R = 0; #10;
        AR = 0; AE = 1; S = 0; R = 1; #10;
        AR = 0; AE = 1; S = 1; R = 0; #10;
        AR = 0; AE = 1; S = 1; R = 1; #10;

        $finish;
    end

endmodule
```

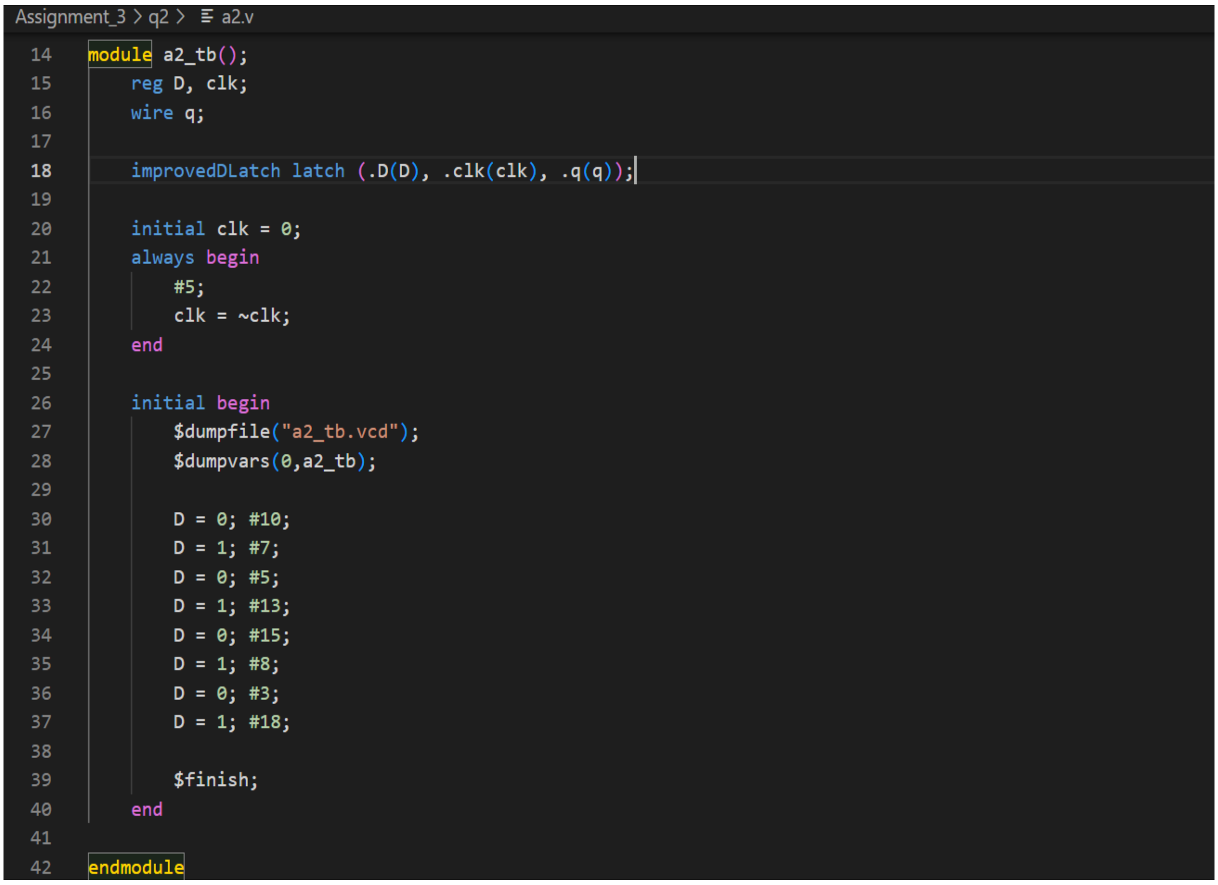


## Q2 - Improved version of D latch with delays of 1 ns to each gate



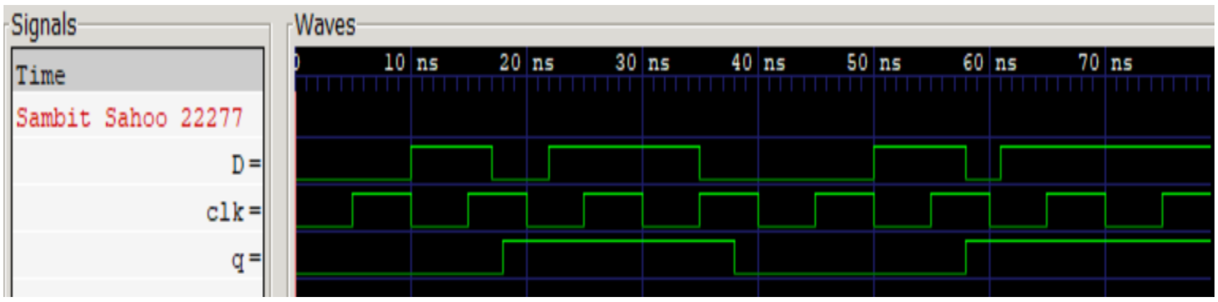
Assignment_3 > q2 > a2.v

```verilog
`timescale 1ns/1ps
module improvedDLatch (input D, input clk, output reg q);
    initial q = 1'b0;
    reg nc, w1, w2;
    always @(clk, D)
        begin
            #1 nc <= ~clk;
               w1 <= D & clk;
            #1 w2 <= nc & q;
            #1 q <= w1 | w2;
        end
endmodule
```

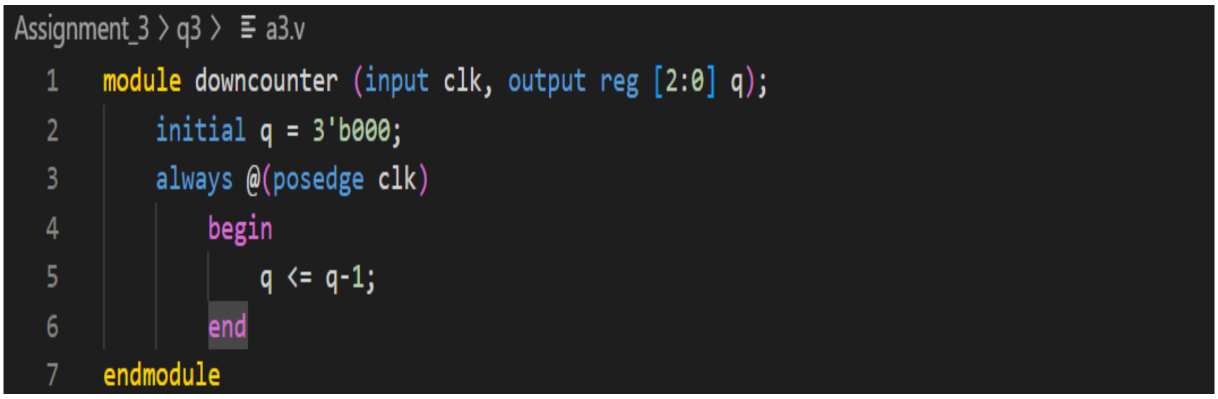


Assignment_3 > q2 > a2.v

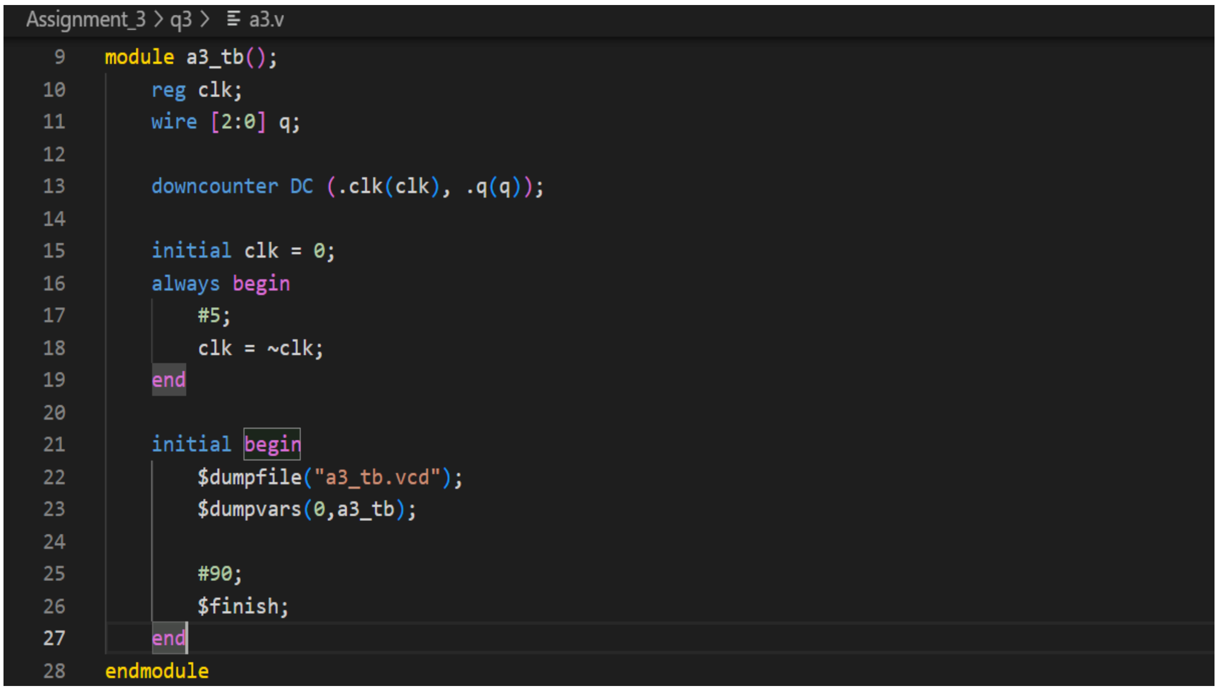
```verilog
module a2_tb();
    reg D, clk;
    wire q;

    improvedDLatch latch (.D(D), .clk(clk), .q(q));

    initial clk = 0;
    always begin
        #5;
        clk = ~clk;
    end

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

        D = 0; #10;
        D = 1; #7;
        D = 0; #5;
        D = 1; #13;
        D = 0; #15;
        D = 1; #8;
        D = 0; #3;
        D = 1; #18;

        $finish;
    end

endmodule
```



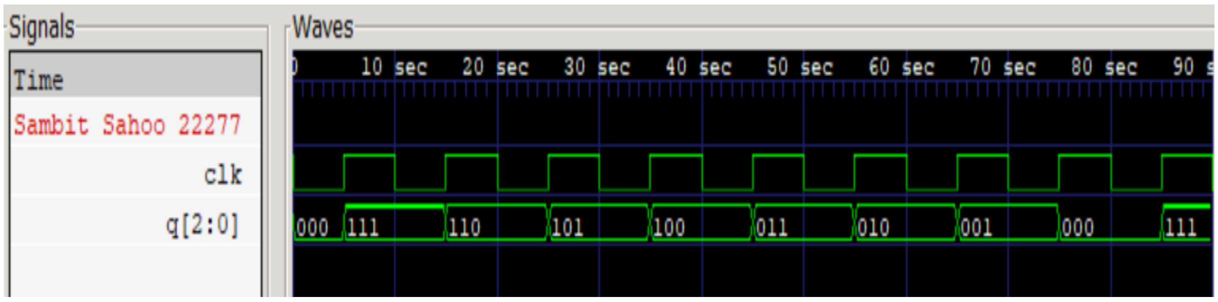
Q3- Counter that counts from 7 to 0 (e.g., 7, 6, 5, 4, 3, 2, 1, 0)


Assignment_3 > q3 > a3.v

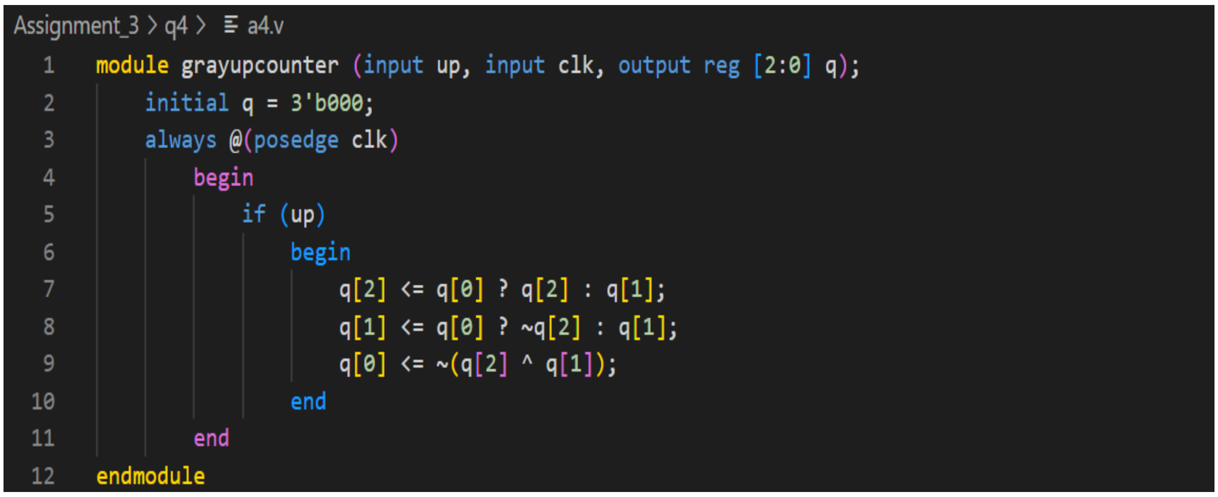
```verilog
module downcounter (input clk, output reg [2:0] q);
    initial q = 3'b000;
    always @(posedge clk)
        begin
            q <= q-1;
        end
endmodule
```


Assignment_3 > q3 > a3.v

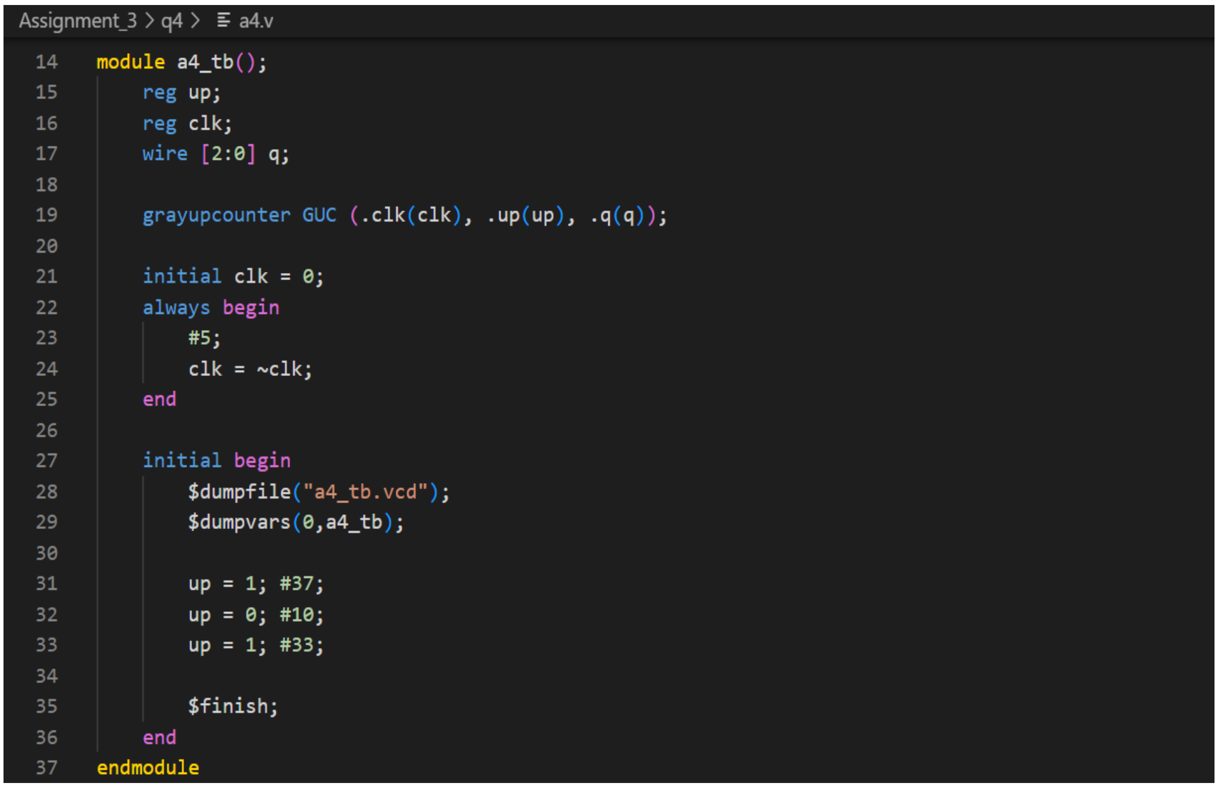
```verilog
module a3_tb();
    reg clk;
    wire [2:0] q;

    downcounter DC (.clk(clk), .q(q));

    initial clk = 0;
    always begin
        #5;
        clk = ~clk;
    end

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

        #90;
        $finish;
    end
endmodule
```



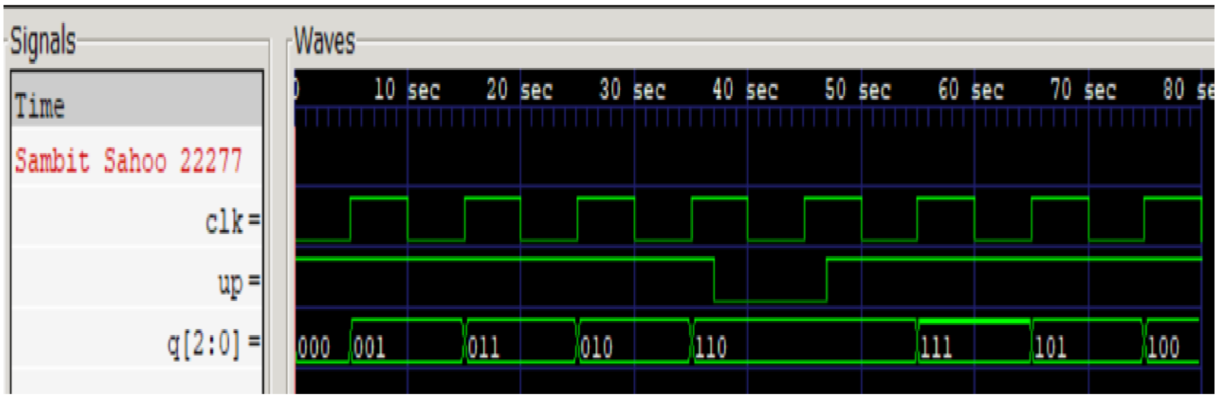
## Q4 - UP/DOWN modulo 8 Gray Code Counter



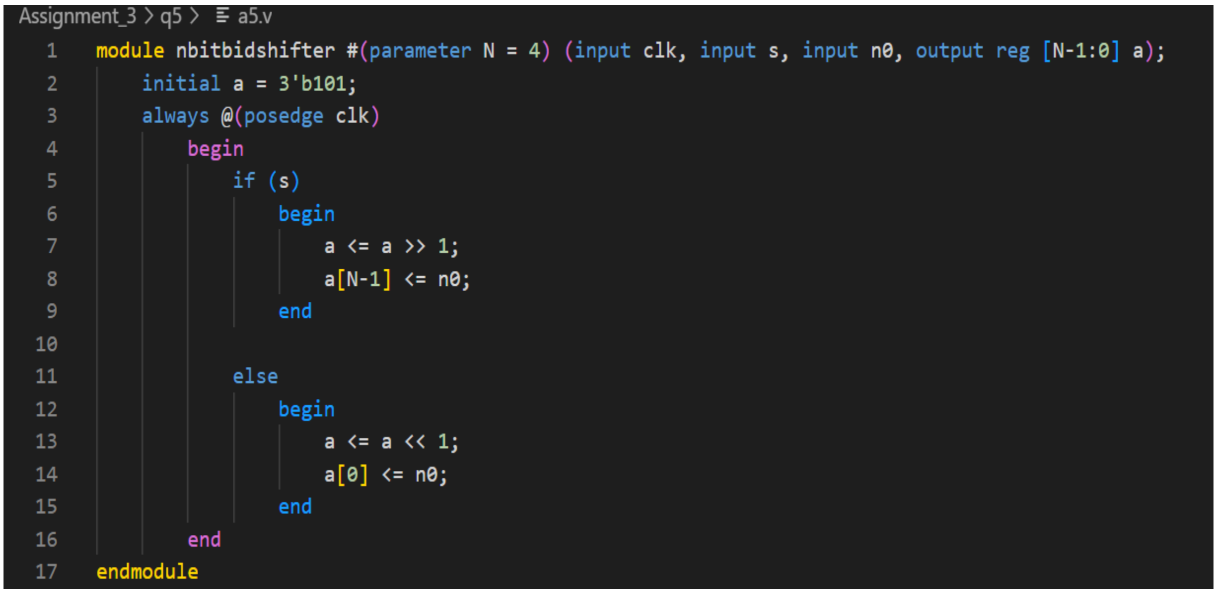
```
module grayupcounter (input up, input clk, output reg [2:0] q);
    initial q = 3'b000;
    always @(posedge clk)
        begin
            if (up)
                begin
                    q[2] <= q[0] ? q[2] : q[1];
                    q[1] <= q[0] ? ~q[2] : q[1];
                    q[0] <= ~(q[2] ^ q[1]);
                end
        end
endmodule
```



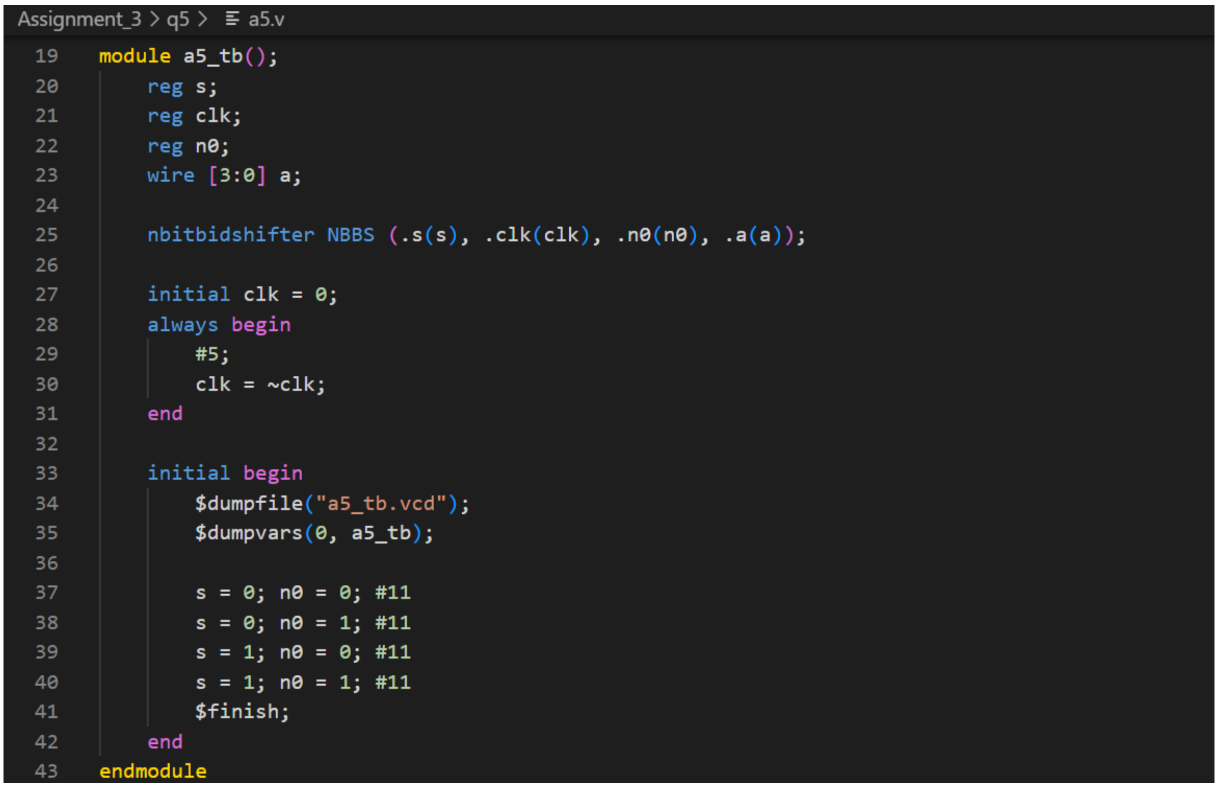
```
module a4_tb();
    reg up;
    reg clk;
    wire [2:0] q;

    grayupcounter GUC (.clk(clk), .up(up), .q(q));

    initial clk = 0;
    always begin
        #5;
        clk = ~clk;
    end

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

        up = 1; #37;
        up = 0; #10;
        up = 1; #33;

        $finish;
    end
endmodule
```



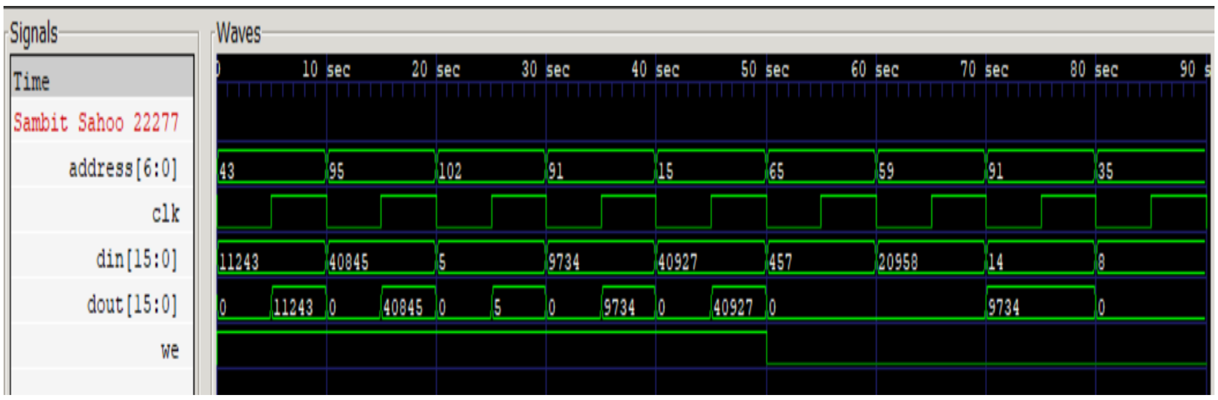
## Q5- N-bit bidirectional shift register


Assignment_3 > q5 > a5.v

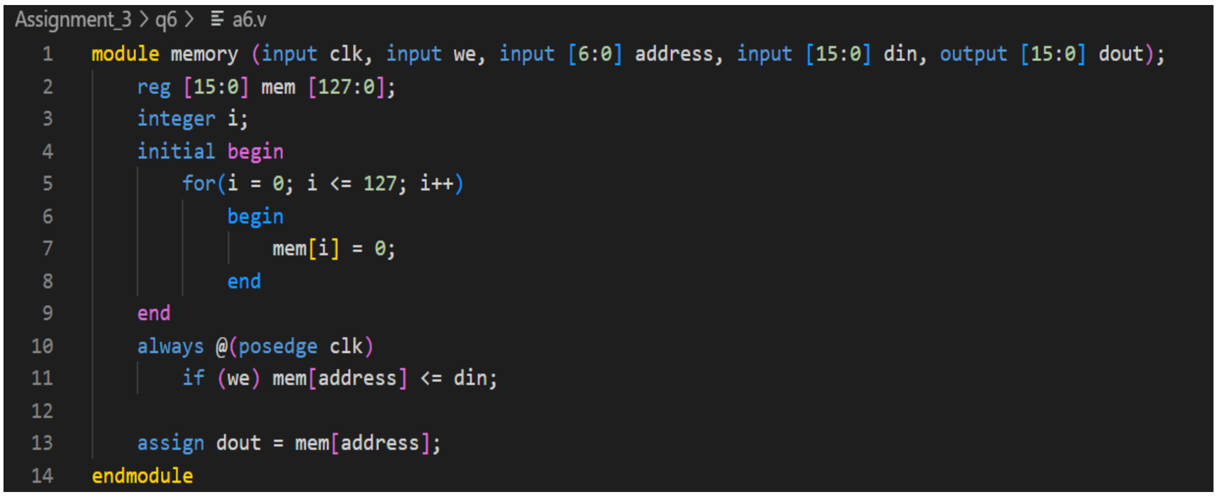
```verilog
module nbitbidshifter #(parameter N = 4) (input clk, input s, input n0, output reg [N-1:0] a);
    initial a = 3'b101;
    always @(posedge clk)
        begin
            if (s)
                begin
                    a <= a >> 1;
                    a[N-1] <= n0;
                end

            else
                begin
                    a <= a << 1;
                    a[0] <= n0;
                end
        end
endmodule
```


Assignment_3 > q5 > a5.v

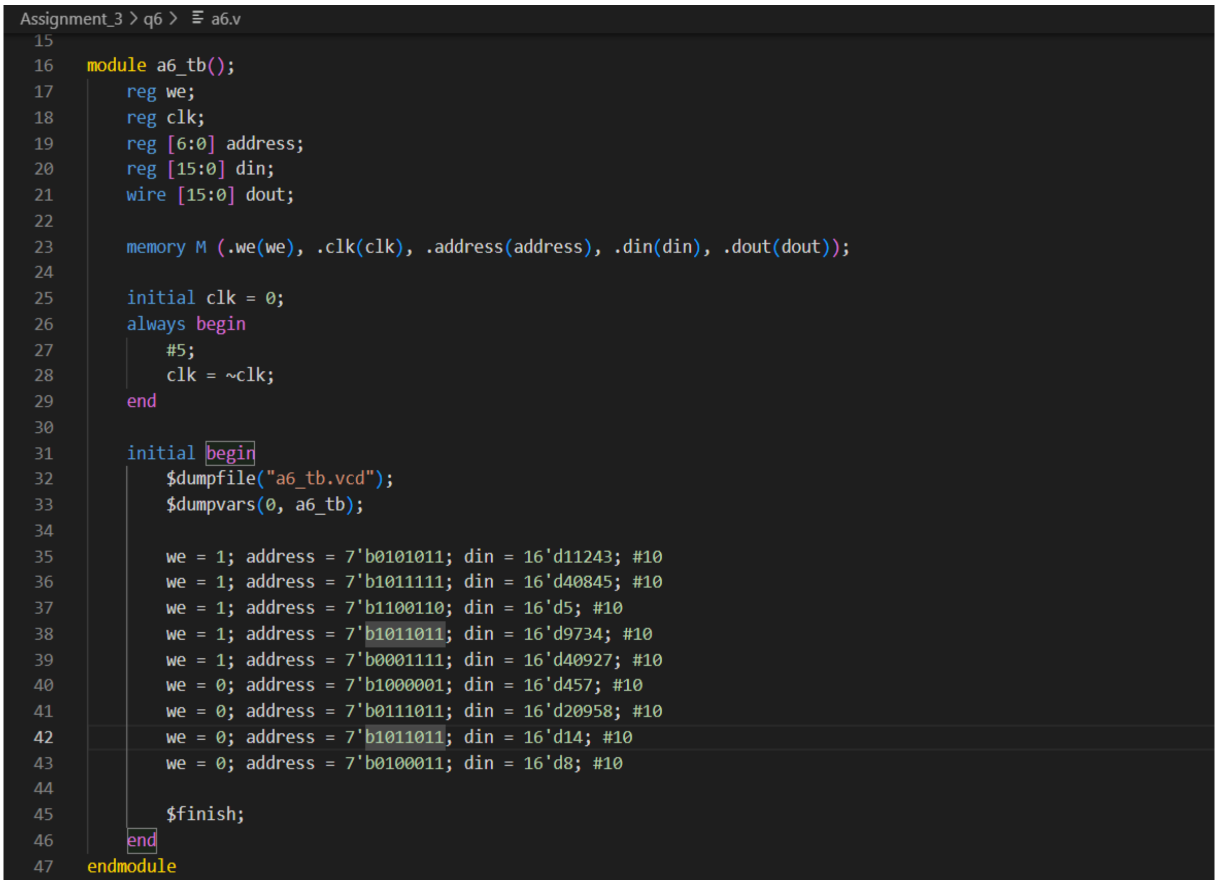
```verilog
module a5_tb();
    reg s;
    reg clk;
    reg n0;
    wire [3:0] a;

    nbitbidshifter NBBS (.s(s), .clk(clk), .n0(n0), .a(a));

    initial clk = 0;
    always begin
        #5;
        clk = ~clk;
    end

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

        s = 0; n0 = 0; #11
        s = 0; n0 = 1; #11
        s = 1; n0 = 0; #11
        s = 1; n0 = 1; #11
        $finish;
    end
endmodule
```



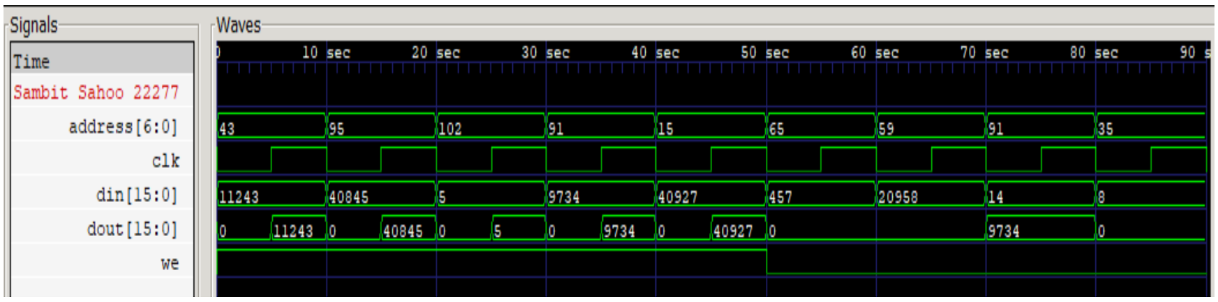
## Q6- A single port memory of address space 128 with 16-bit addressability


Assignment_3 > q6 > a6.v

```verilog
module memory (input clk, input we, input [6:0] address, input [15:0] din, output [15:0] dout);
    reg [15:0] mem [127:0];
    integer i;
    initial begin
        for(i = 0; i <= 127; i++)
            begin
                mem[i] = 0;
            end
    end
    always @(posedge clk)
        if (we) mem[address] <= din;

    assign dout = mem[address];
endmodule
```


Assignment_3 > q6 > a6.v

```verilog

module a6_tb();
    reg we;
    reg clk;
    reg [6:0] address;
    reg [15:0] din;
    wire [15:0] dout;

    memory M (.we(we), .clk(clk), .address(address), .din(din), .dout(dout));

    initial clk = 0;
    always begin
        #5;
        clk = ~clk;
    end

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

        we = 1; address = 7'b0101011; din = 16'd11243; #10
        we = 1; address = 7'b1011111; din = 16'd40845; #10
        we = 1; address = 7'b1100110; din = 16'd5; #10
        we = 1; address = 7'b1011011; din = 16'd9734; #10
        we = 1; address = 7'b0001111; din = 16'd40927; #10
        we = 0; address = 7'b1000001; din = 16'd457; #10
        we = 0; address = 7'b0111011; din = 16'd20958; #10
        we = 0; address = 7'b1011011; din = 16'd14; #10
        we = 0; address = 7'b0100011; din = 16'd8; #10

        $finish;
    end
endmodule
```


Signals
Waves
Time
Sambit Sahoo 22277
address[6:0]
clk
din[15:0]
dout[15:0]
we
10 sec
20 sec
30 sec
40 sec
50 sec
60 sec
70 sec
80 sec
90 s