Assignment_4_extra

==

Name: Ahmed Nour

Sheet ID: 16

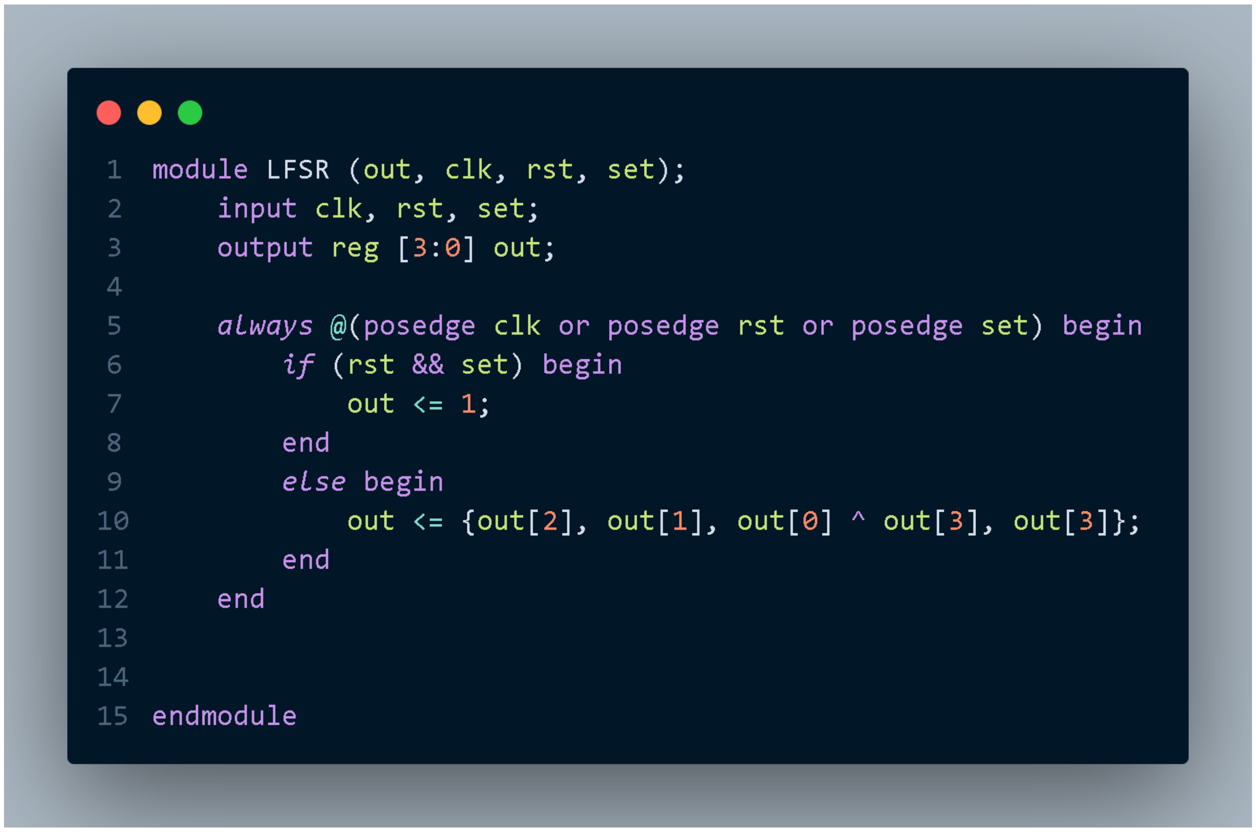
Email: ahmedmohamednoour@gmail.com

==

Q1)

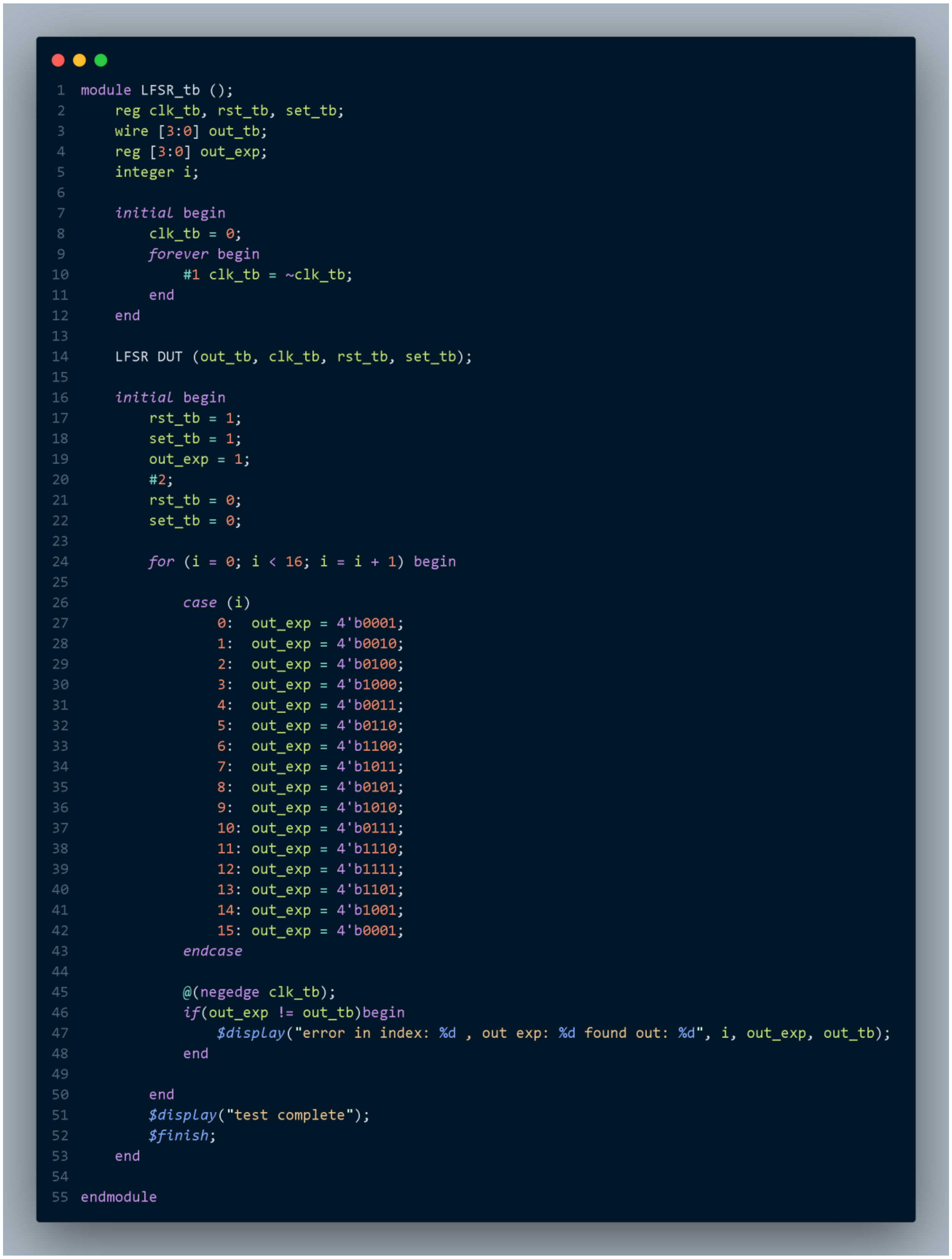
Verilog code:

a) **RTL design:**



```verilog
module LFSR (out, clk, rst, set);
    input clk, rst, set;
    output reg [3:0] out;

    always @(posedge clk or posedge rst or posedge set) begin
        if (rst && set) begin
            out <= 1;
        end
        else begin
            out <= {out[2], out[1], out[0] ^ out[3], out[3]};
        end
    end


endmodule
```

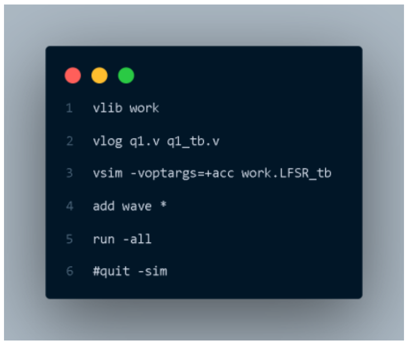
b) Testbench code:



```
module LFSR_tb ();
    reg clk_tb, rst_tb, set_tb;
    wire [3:0] out_tb;
    reg [3:0] out_exp;
    integer i;

    initial begin
        clk_tb = 0;
        forever begin
            #1 clk_tb = ~clk_tb;
        end
    end

    LFSR DUT (out_tb, clk_tb, rst_tb, set_tb);

    initial begin
        rst_tb = 1;
        set_tb = 1;
        out_exp = 1;
        #2;
        rst_tb = 0;
        set_tb = 0;

        for (i = 0; i < 16; i = i + 1) begin

            case (i)
                0:  out_exp = 4'b0001;
                1:  out_exp = 4'b0010;
                2:  out_exp = 4'b0100;
                3:  out_exp = 4'b1000;
                4:  out_exp = 4'b0011;
                5:  out_exp = 4'b0110;
                6:  out_exp = 4'b1100;
                7:  out_exp = 4'b1011;
                8:  out_exp = 4'b0101;
                9:  out_exp = 4'b1010;
                10: out_exp = 4'b0111;
                11: out_exp = 4'b1110;
                12: out_exp = 4'b1111;
                13: out_exp = 4'b1101;
                14: out_exp = 4'b1001;
                15: out_exp = 4'b0001;
            endcase

            @(negedge clk_tb);
            if(out_exp != out_tb)begin
                $display("error in index: %d , out exp: %d found out: %d", i, out_exp, out_tb);
            end

        end
        $display("test complete");
        $finish;
    end

endmodule
```

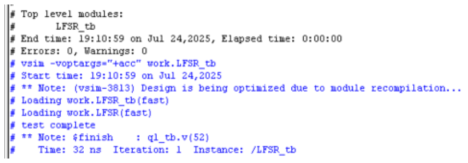
Simulation Tool

a) Do file

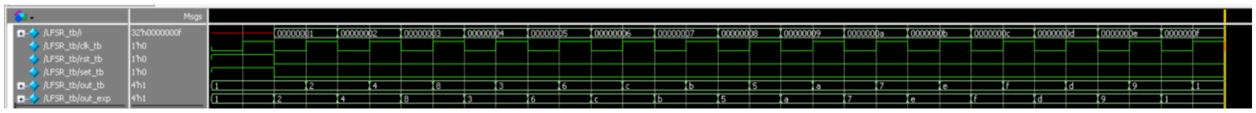


```
vlib work
vlog q1.v q1_tb.v
vsim -voptargs=+acc work.LFSR_tb
add wave *
run -all
#quit -sim
```

b) Questasim



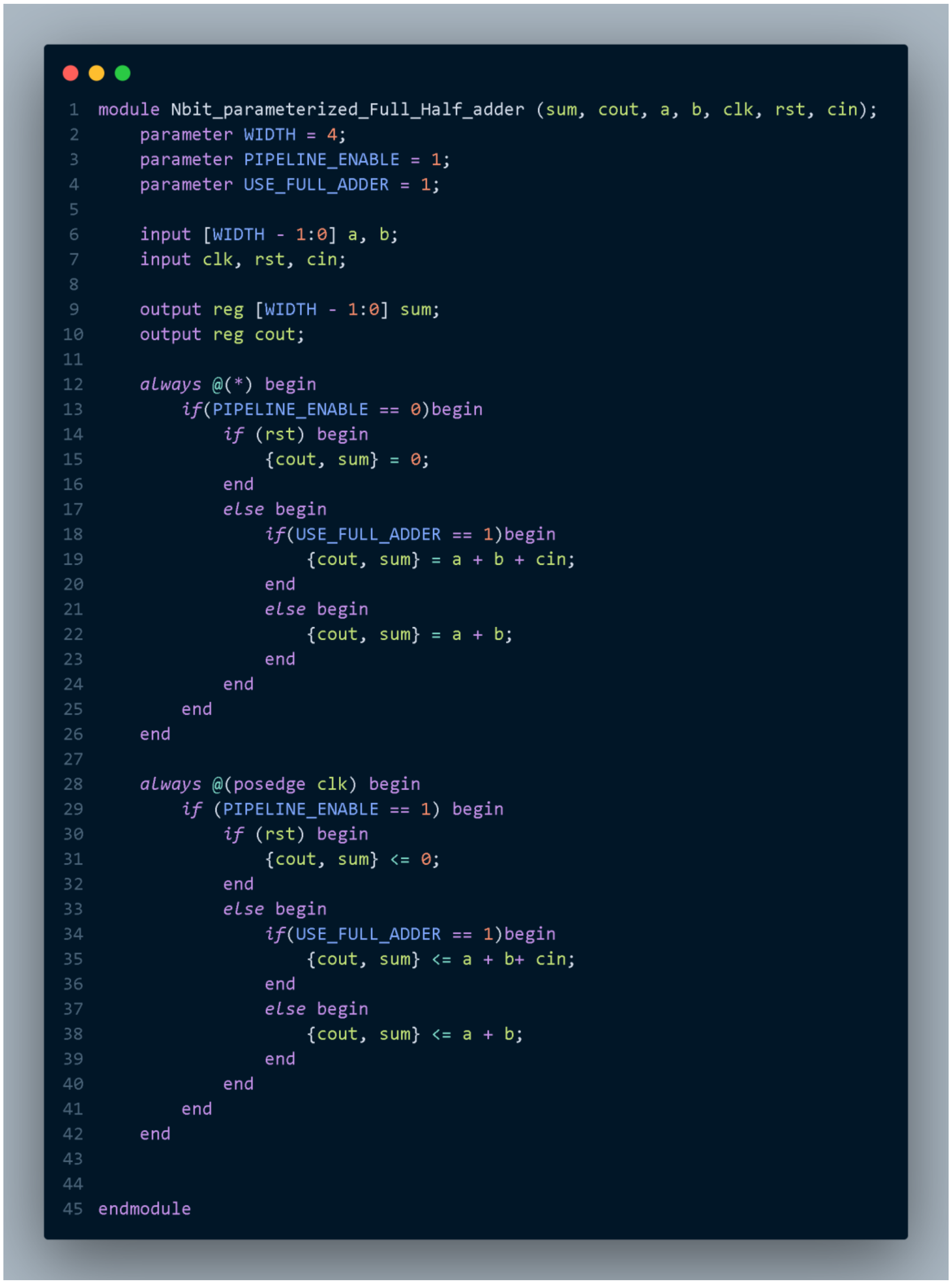
```
# Top level modules:
#       LFSR_tb
# End time: 19:10:59 on Jul 24,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.LFSR_tb
# Start time: 19:10:59 on Jul 24,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading work.LFSR_tb(fast)
# Loading work.LFSR(fast)
# test complete
# ** Note: $finish    : q1_tb.v(52)
#    Time: 32 ns  Iteration: 1  Instance: /LFSR_tb
```


Msgs
/LFSR_tb/i 32'h0000000f
/LFSR_tb/clk_tb 1'h0
/LFSR_tb/rst_tb 1'h0
/LFSR_tb/set_tb 1'h0
/LFSR_tb/out_tb 4'h1
/LFSR_tb/out_exp 4'h1
00000001 00000002 00000003 00000004 00000005 00000006 00000007 00000008 00000009 0000000a 0000000b 0000000c 0000000d 0000000e 0000000f
1 2 4 8 3 6 c b 5 a 7 e f d 9 1
1 2 4 8 3 6 c b 5 a 7 e f d 9 1

Q2)

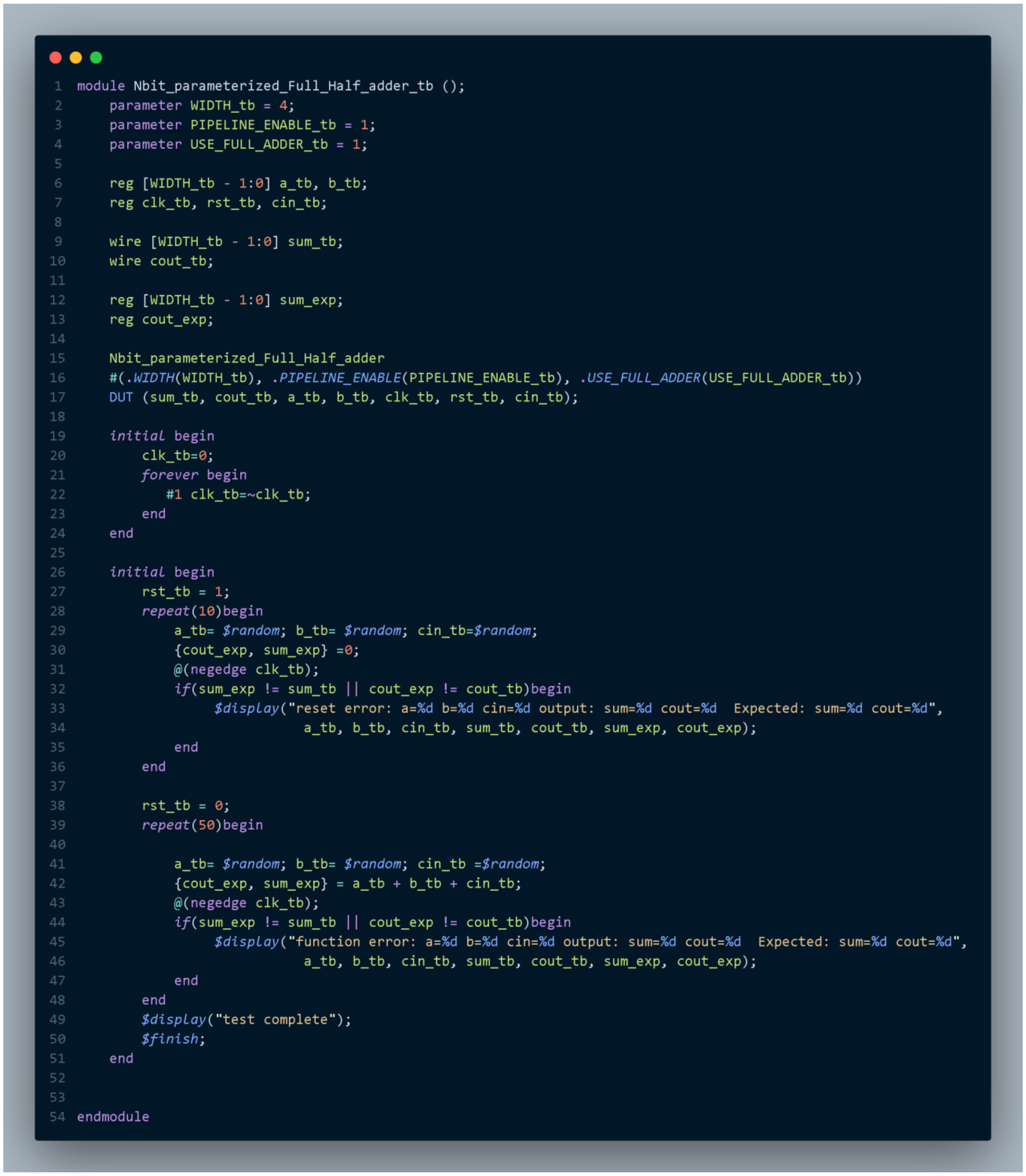
Verilog code:

a) RTL design:



```verilog
module Nbit_parameterized_Full_Half_adder (sum, cout, a, b, clk, rst, cin);
    parameter WIDTH = 4;
    parameter PIPELINE_ENABLE = 1;
    parameter USE_FULL_ADDER = 1;

    input [WIDTH - 1:0] a, b;
    input clk, rst, cin;

    output reg [WIDTH - 1:0] sum;
    output reg cout;

    always @(*) begin
        if(PIPELINE_ENABLE == 0)begin
            if (rst) begin
                {cout, sum} = 0;
            end
            else begin
                if(USE_FULL_ADDER == 1)begin
                    {cout, sum} = a + b + cin;
                end
                else begin
                    {cout, sum} = a + b;
                end
            end
        end
    end

    always @(posedge clk) begin
        if (PIPELINE_ENABLE == 1) begin
            if (rst) begin
                {cout, sum} <= 0;
            end
            else begin
                if(USE_FULL_ADDER == 1)begin
                    {cout, sum} <= a + b+ cin;
                end
                else begin
                    {cout, sum} <= a + b;
                end
            end
        end
    end


endmodule
```

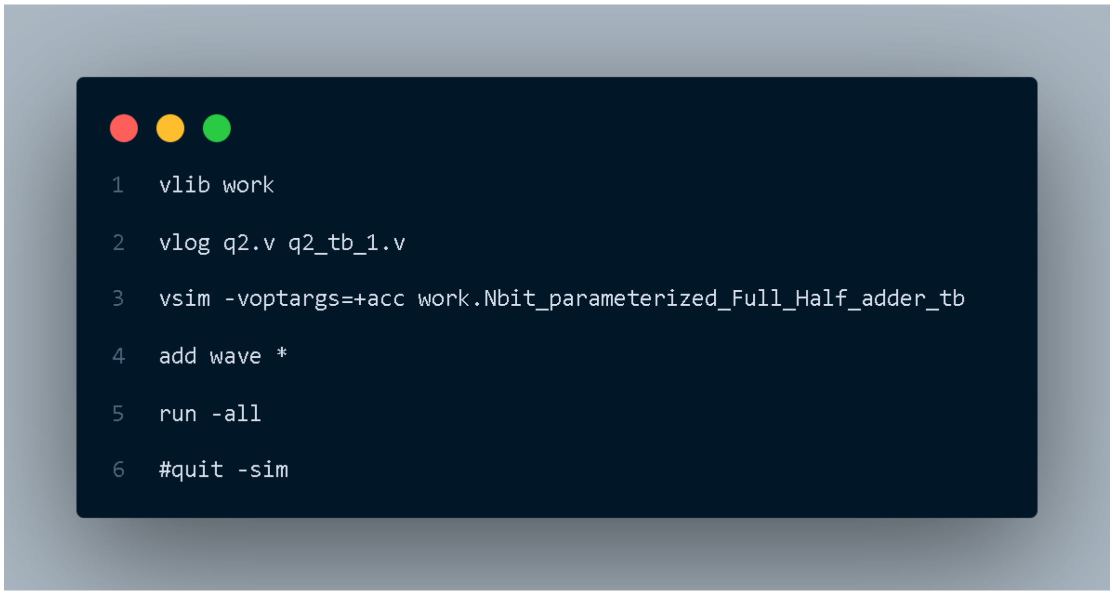
Testbench 1:



```verilog
module Nbit_parameterized_Full_Half_adder_tb ();
    parameter WIDTH_tb = 4;
    parameter PIPELINE_ENABLE_tb = 1;
    parameter USE_FULL_ADDER_tb = 1;

    reg [WIDTH_tb - 1:0] a_tb, b_tb;
    reg clk_tb, rst_tb, cin_tb;

    wire [WIDTH_tb - 1:0] sum_tb;
    wire cout_tb;

    reg [WIDTH_tb - 1:0] sum_exp;
    reg cout_exp;

    Nbit_parameterized_Full_Half_adder
    #(.WIDTH(WIDTH_tb), .PIPELINE_ENABLE(PIPELINE_ENABLE_tb), .USE_FULL_ADDER(USE_FULL_ADDER_tb))
    DUT (sum_tb, cout_tb, a_tb, b_tb, clk_tb, rst_tb, cin_tb);

    initial begin
        clk_tb=0;
        forever begin
           #1 clk_tb=~clk_tb;
        end
    end

    initial begin
        rst_tb = 1;
        repeat(10)begin
            a_tb= $random; b_tb= $random; cin_tb=$random;
            {cout_exp, sum_exp} =0;
            @(negedge clk_tb);
            if(sum_exp != sum_tb || cout_exp != cout_tb)begin
                 $display("reset error: a=%d b=%d cin=%d output: sum=%d cout=%d  Expected: sum=%d cout=%d",
                            a_tb, b_tb, cin_tb, sum_tb, cout_tb, sum_exp, cout_exp);
            end
        end

        rst_tb = 0;
        repeat(50)begin

            a_tb= $random; b_tb= $random; cin_tb =$random;
            {cout_exp, sum_exp} = a_tb + b_tb + cin_tb;
            @(negedge clk_tb);
            if(sum_exp != sum_tb || cout_exp != cout_tb)begin
                 $display("function error: a=%d b=%d cin=%d output: sum=%d cout=%d  Expected: sum=%d cout=%d",
                            a_tb, b_tb, cin_tb, sum_tb, cout_tb, sum_exp, cout_exp);
            end
        end
        $display("test complete");
        $finish;
    end


endmodule
```

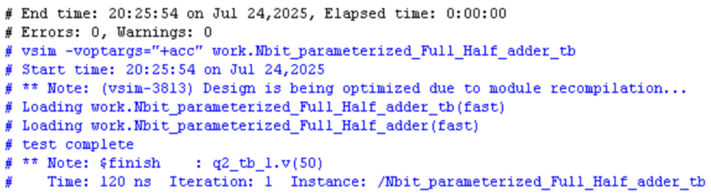
Simulation Tool for testbench 1:

a) Do file

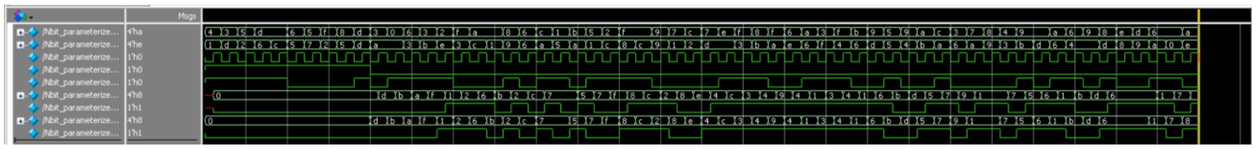


```
vlib work
vlog q2.v q2_tb_1.v
vsim -voptargs=+acc work.Nbit_parameterized_Full_Half_adder_tb
add wave *
run -all
#quit -sim
```

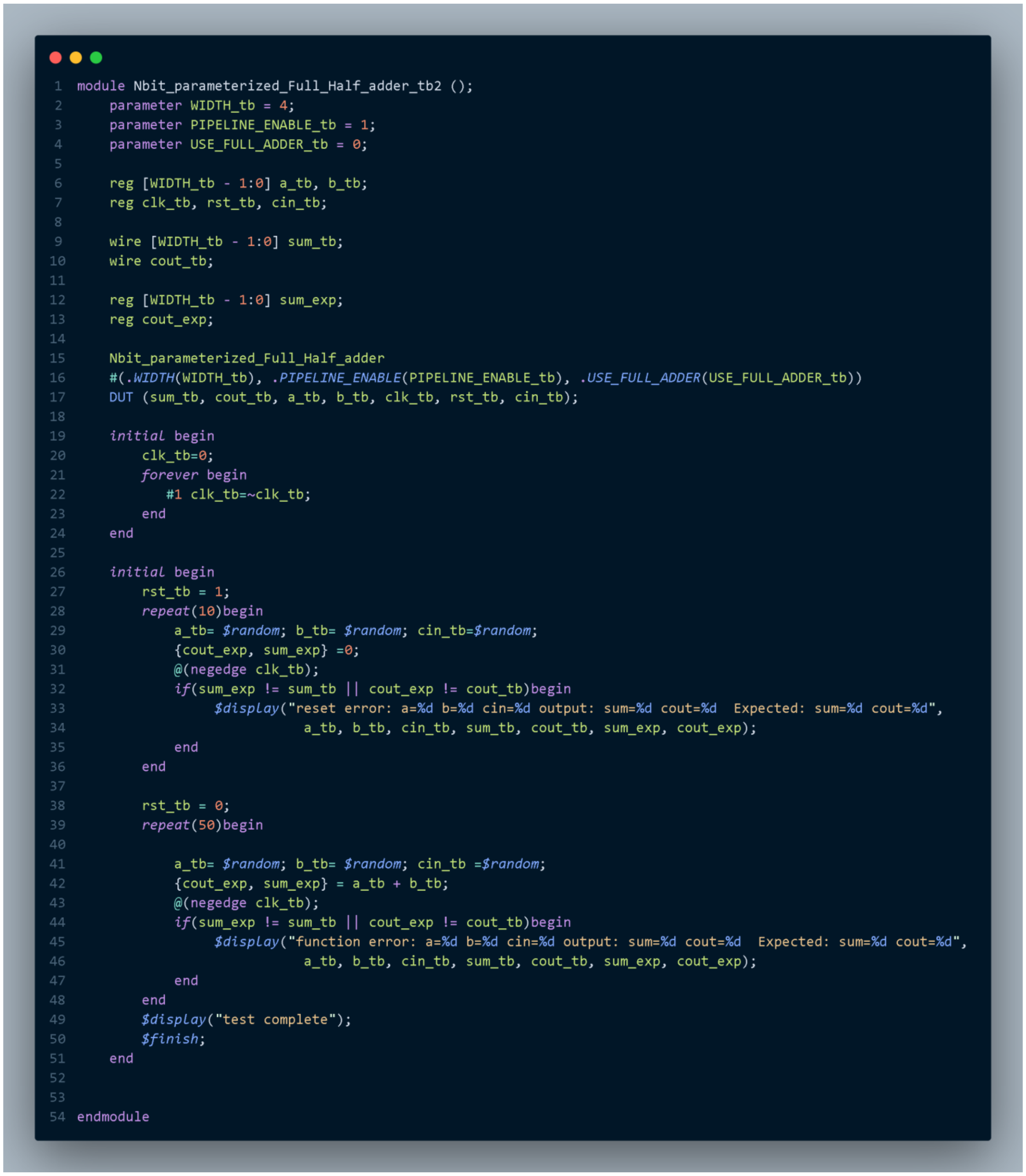
b) Questasim



```
# End time: 20:25:54 on Jul 24,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.Nbit_parameterized_Full_Half_adder_tb
# Start time: 20:25:54 on Jul 24,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading work.Nbit_parameterized_Full_Half_adder_tb(fast)
# Loading work.Nbit_parameterized_Full_Half_adder(fast)
# test complete
# ** Note: $finish    : q2_tb_1.v(50)
#    Time: 120 ns  Iteration: 1  Instance: /Nbit_parameterized_Full_Half_adder_tb
```



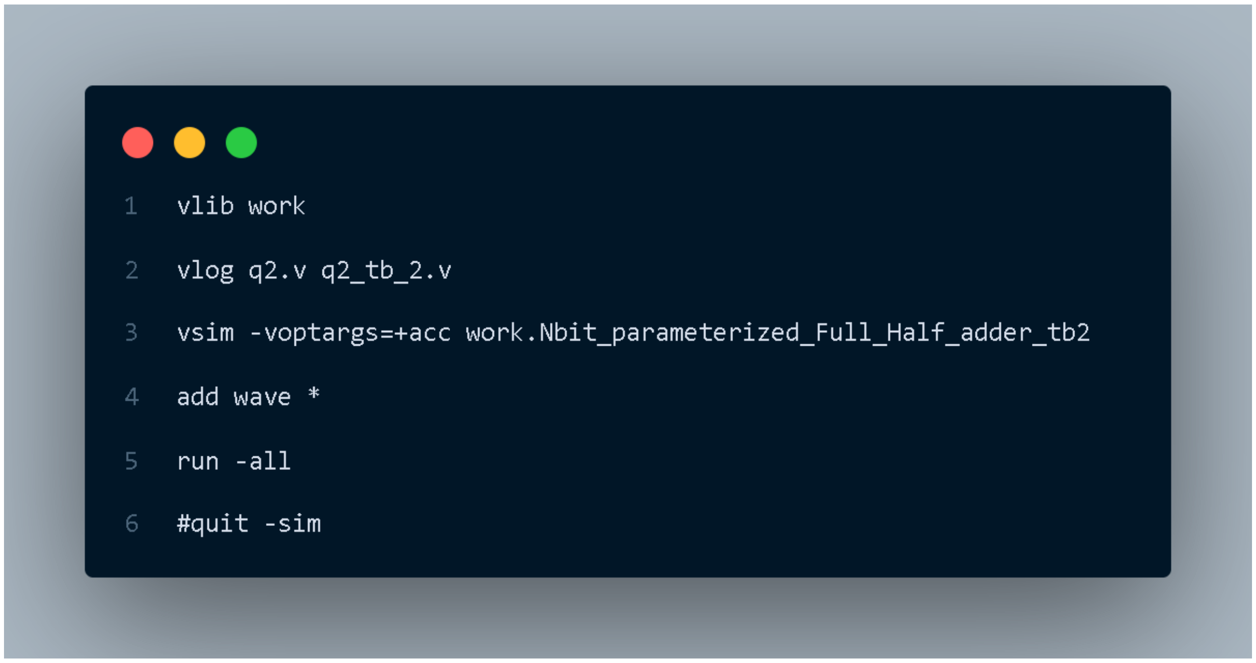
Testbench 2:



```verilog
module Nbit_parameterized_Full_Half_adder_tb2 ();
    parameter WIDTH_tb = 4;
    parameter PIPELINE_ENABLE_tb = 1;
    parameter USE_FULL_ADDER_tb = 0;

    reg [WIDTH_tb - 1:0] a_tb, b_tb;
    reg clk_tb, rst_tb, cin_tb;

    wire [WIDTH_tb - 1:0] sum_tb;
    wire cout_tb;

    reg [WIDTH_tb - 1:0] sum_exp;
    reg cout_exp;

    Nbit_parameterized_Full_Half_adder
    #(.WIDTH(WIDTH_tb), .PIPELINE_ENABLE(PIPELINE_ENABLE_tb), .USE_FULL_ADDER(USE_FULL_ADDER_tb))
    DUT (sum_tb, cout_tb, a_tb, b_tb, clk_tb, rst_tb, cin_tb);

    initial begin
        clk_tb=0;
        forever begin
           #1 clk_tb=~clk_tb;
        end
    end

    initial begin
        rst_tb = 1;
        repeat(10)begin
            a_tb= $random; b_tb= $random; cin_tb=$random;
            {cout_exp, sum_exp} =0;
            @(negedge clk_tb);
            if(sum_exp != sum_tb || cout_exp != cout_tb)begin
                 $display("reset error: a=%d b=%d cin=%d output: sum=%d cout=%d  Expected: sum=%d cout=%d",
                            a_tb, b_tb, cin_tb, sum_tb, cout_tb, sum_exp, cout_exp);
            end
        end

        rst_tb = 0;
        repeat(50)begin

            a_tb= $random; b_tb= $random; cin_tb =$random;
            {cout_exp, sum_exp} = a_tb + b_tb;
            @(negedge clk_tb);
            if(sum_exp != sum_tb || cout_exp != cout_tb)begin
                 $display("function error: a=%d b=%d cin=%d output: sum=%d cout=%d  Expected: sum=%d cout=%d",
                            a_tb, b_tb, cin_tb, sum_tb, cout_tb, sum_exp, cout_exp);
            end
        end
        $display("test complete");
        $finish;
    end


endmodule
```

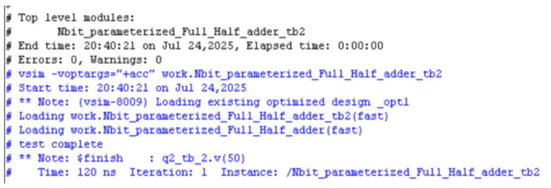
Simulation Tool for testbench 2:

a) Do file

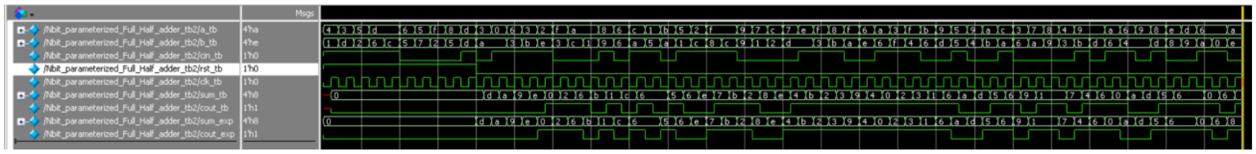


```
vlib work
vlog q2.v q2_tb_2.v
vsim -voptargs=+acc work.Nbit_parameterized_Full_Half_adder_tb2
add wave *
run -all
#quit -sim
```

b) Questasim

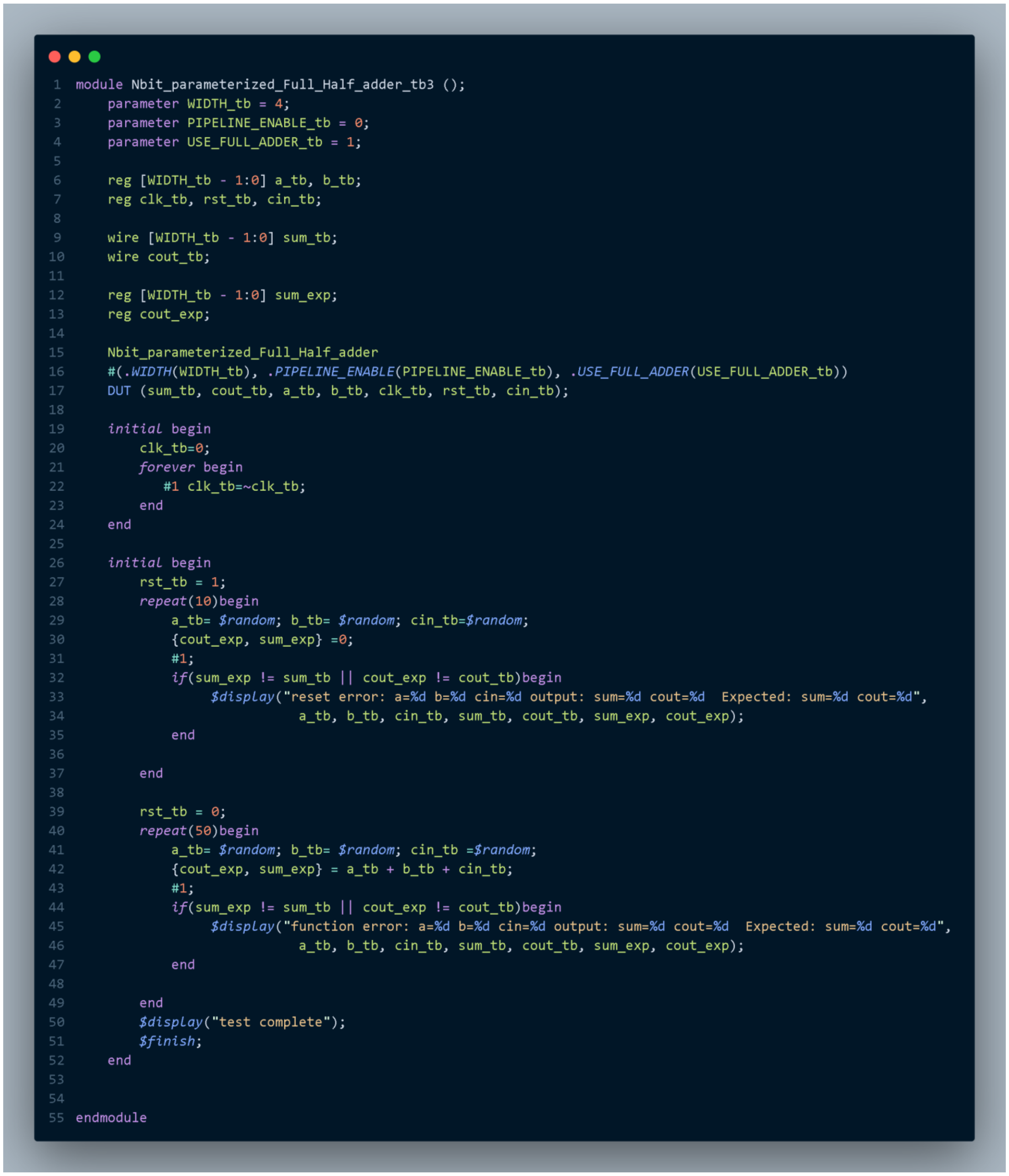


```
# Top level modules:
#       Nbit_parameterized_Full_Half_adder_tb2
# End time: 20:40:21 on Jul 24,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.Nbit_parameterized_Full_Half_adder_tb2
# Start time: 20:40:21 on Jul 24,2025
# ** Note: (vsim-8009) Loading existing optimized design _opt1
# Loading work.Nbit_parameterized_Full_Half_adder_tb2(fast)
# Loading work.Nbit_parameterized_Full_Half_adder(fast)
# test complete
# ** Note: $finish    : q2_tb_2.v(50)
#    Time: 120 ns  Iteration: 1  Instance: /Nbit_parameterized_Full_Half_adder_tb2
```



```
                                                      Msgs
/Nbit_parameterized_Full_Half_adder_tb2/a_tb          4'ha
/Nbit_parameterized_Full_Half_adder_tb2/b_tb          4'he
/Nbit_parameterized_Full_Half_adder_tb2/cin_tb        1'h0
/Nbit_parameterized_Full_Half_adder_tb2/rst_tb        1'h0
/Nbit_parameterized_Full_Half_adder_tb2/clk_tb        1'h0
/Nbit_parameterized_Full_Half_adder_tb2/sum_tb        4'h8
/Nbit_parameterized_Full_Half_adder_tb2/cout_tb       1'h1
/Nbit_parameterized_Full_Half_adder_tb2/sum_exp       4'h8
/Nbit_parameterized_Full_Half_adder_tb2/cout_exp      1'h1
```

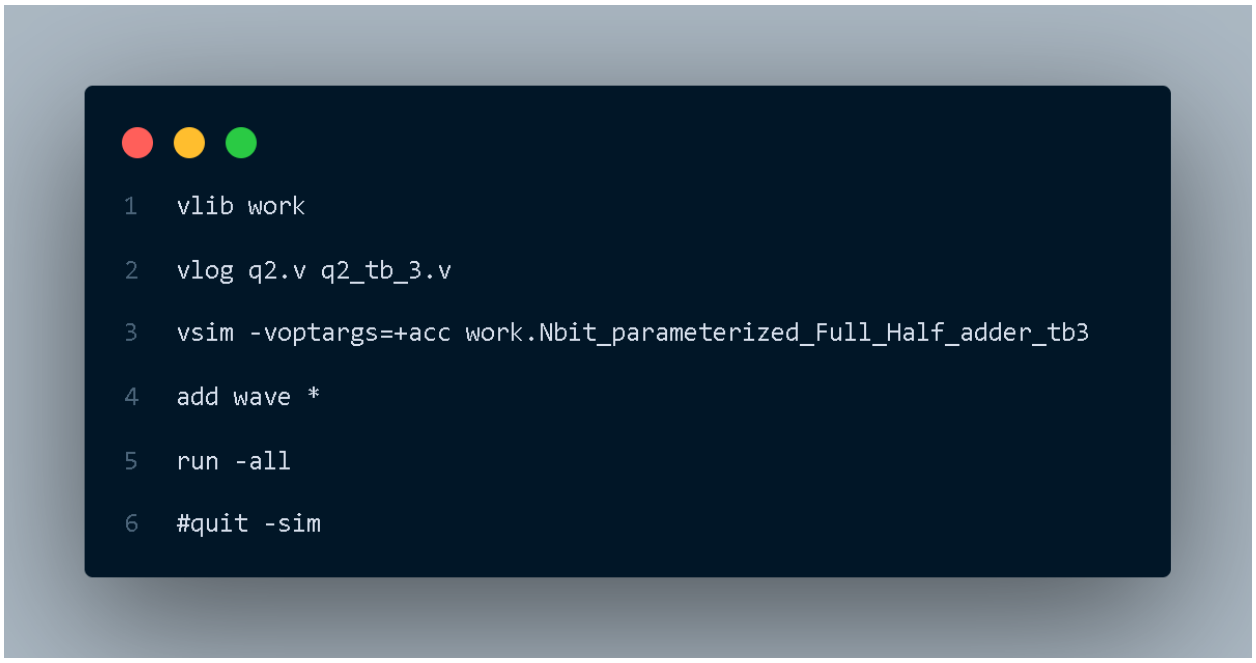
Testbench 3:



```
module Nbit_parameterized_Full_Half_adder_tb3 ();
    parameter WIDTH_tb = 4;
    parameter PIPELINE_ENABLE_tb = 0;
    parameter USE_FULL_ADDER_tb = 1;

    reg [WIDTH_tb - 1:0] a_tb, b_tb;
    reg clk_tb, rst_tb, cin_tb;

    wire [WIDTH_tb - 1:0] sum_tb;
    wire cout_tb;

    reg [WIDTH_tb - 1:0] sum_exp;
    reg cout_exp;

    Nbit_parameterized_Full_Half_adder
    #(.WIDTH(WIDTH_tb), .PIPELINE_ENABLE(PIPELINE_ENABLE_tb), .USE_FULL_ADDER(USE_FULL_ADDER_tb))
    DUT (sum_tb, cout_tb, a_tb, b_tb, clk_tb, rst_tb, cin_tb);

    initial begin
        clk_tb=0;
        forever begin
           #1 clk_tb=~clk_tb;
        end
    end

    initial begin
        rst_tb = 1;
        repeat(10)begin
            a_tb= $random; b_tb= $random; cin_tb=$random;
            {cout_exp, sum_exp} =0;
            #1;
            if(sum_exp != sum_tb || cout_exp != cout_tb)begin
                 $display("reset error: a=%d b=%d cin=%d output: sum=%d cout=%d  Expected: sum=%d cout=%d",
                           a_tb, b_tb, cin_tb, sum_tb, cout_tb, sum_exp, cout_exp);
            end

        end

        rst_tb = 0;
        repeat(50)begin
            a_tb= $random; b_tb= $random; cin_tb =$random;
            {cout_exp, sum_exp} = a_tb + b_tb + cin_tb;
            #1;
            if(sum_exp != sum_tb || cout_exp != cout_tb)begin
                 $display("function error: a=%d b=%d cin=%d output: sum=%d cout=%d  Expected: sum=%d cout=%d",
                           a_tb, b_tb, cin_tb, sum_tb, cout_tb, sum_exp, cout_exp);
            end

        end
        $display("test complete");
        $finish;
    end


endmodule
```

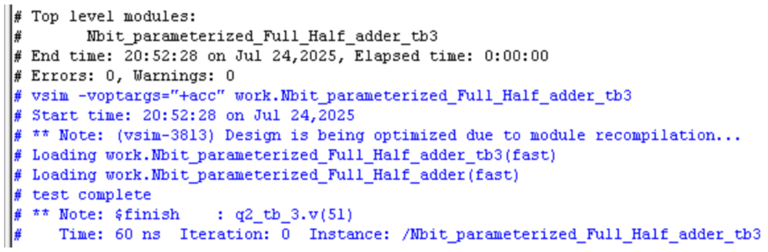
Simulation Tool for testbench 3:

a) Do file

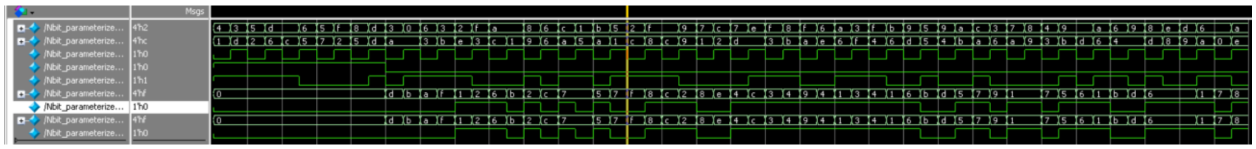


```
vlib work
vlog q2.v q2_tb_3.v
vsim -voptargs=+acc work.Nbit_parameterized_Full_Half_adder_tb3
add wave *
run -all
#quit -sim
```

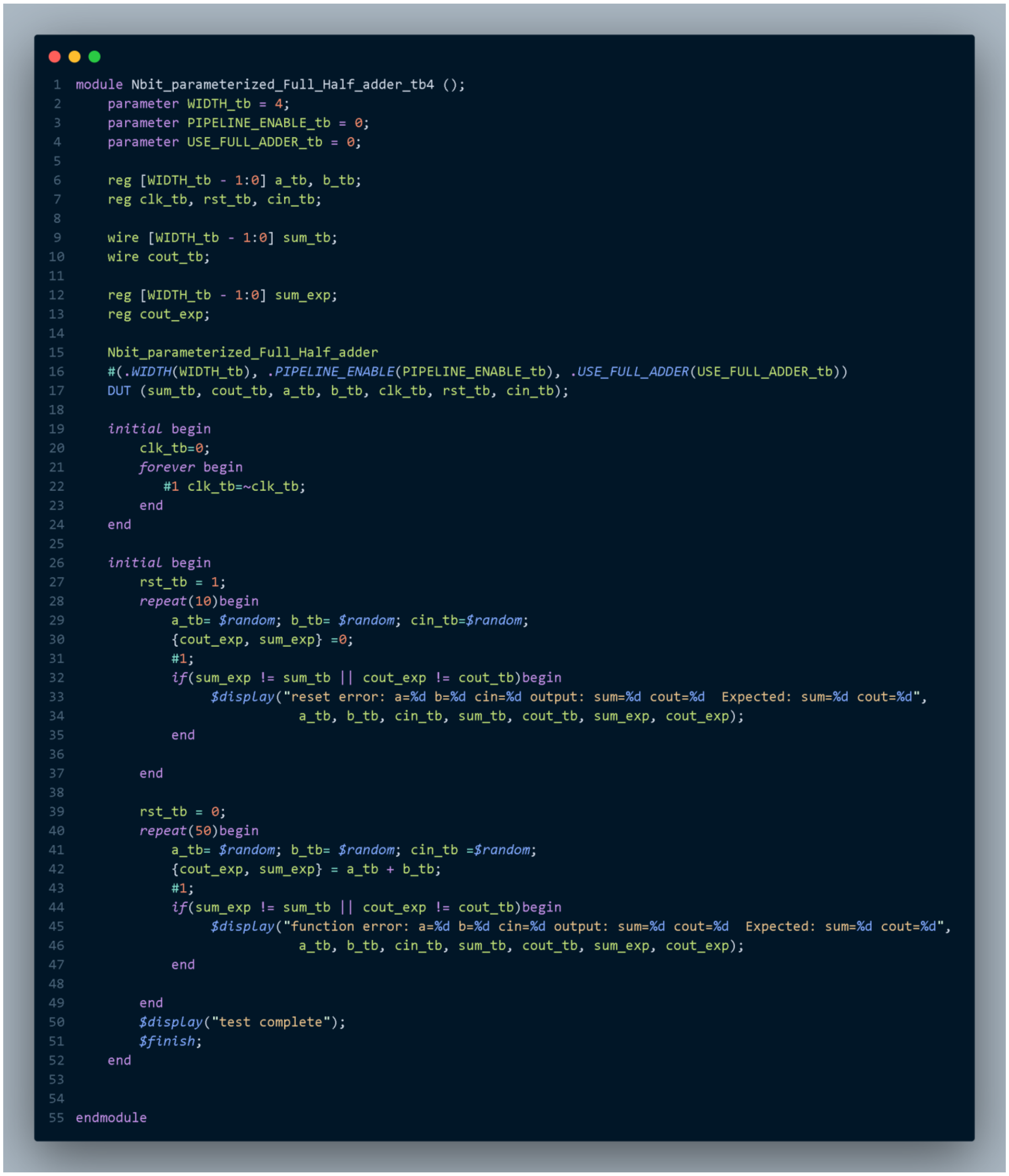
b) Questasim



```
# Top level modules:
#       Nbit_parameterized_Full_Half_adder_tb3
# End time: 20:52:28 on Jul 24,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.Nbit_parameterized_Full_Half_adder_tb3
# Start time: 20:52:28 on Jul 24,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading work.Nbit_parameterized_Full_Half_adder_tb3(fast)
# Loading work.Nbit_parameterized_Full_Half_adder(fast)
# test complete
# ** Note: $finish    : q2_tb_3.v(51)
#    Time: 60 ns  Iteration: 0  Instance: /Nbit_parameterized_Full_Half_adder_tb3
```


Msgs
/Nbit_parameterize... 4'h2
/Nbit_parameterize... 4'hc
/Nbit_parameterize... 1'h0
/Nbit_parameterize... 1'h0
/Nbit_parameterize... 1'h1
/Nbit_parameterize... 4'hf
/Nbit_parameterize... 1'h0
/Nbit_parameterize... 4'hf
/Nbit_parameterize... 1'h0

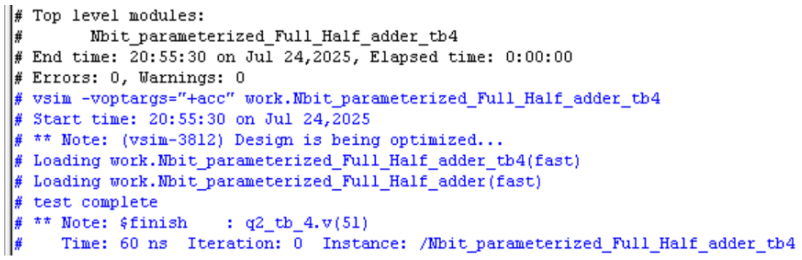
Testbench 4:



```verilog
module Nbit_parameterized_Full_Half_adder_tb4 ();
    parameter WIDTH_tb = 4;
    parameter PIPELINE_ENABLE_tb = 0;
    parameter USE_FULL_ADDER_tb = 0;

    reg [WIDTH_tb - 1:0] a_tb, b_tb;
    reg clk_tb, rst_tb, cin_tb;

    wire [WIDTH_tb - 1:0] sum_tb;
    wire cout_tb;

    reg [WIDTH_tb - 1:0] sum_exp;
    reg cout_exp;

    Nbit_parameterized_Full_Half_adder
    #(.WIDTH(WIDTH_tb), .PIPELINE_ENABLE(PIPELINE_ENABLE_tb), .USE_FULL_ADDER(USE_FULL_ADDER_tb))
    DUT (sum_tb, cout_tb, a_tb, b_tb, clk_tb, rst_tb, cin_tb);

    initial begin
        clk_tb=0;
        forever begin
           #1 clk_tb=~clk_tb;
        end
    end

    initial begin
        rst_tb = 1;
        repeat(10)begin
            a_tb= $random; b_tb= $random; cin_tb=$random;
            {cout_exp, sum_exp} =0;
            #1;
            if(sum_exp != sum_tb || cout_exp != cout_tb)begin
                 $display("reset error: a=%d b=%d cin=%d output: sum=%d cout=%d  Expected: sum=%d cout=%d",
                                a_tb, b_tb, cin_tb, sum_tb, cout_tb, sum_exp, cout_exp);
            end

        end

        rst_tb = 0;
        repeat(50)begin
            a_tb= $random; b_tb= $random; cin_tb =$random;
            {cout_exp, sum_exp} = a_tb + b_tb;
            #1;
            if(sum_exp != sum_tb || cout_exp != cout_tb)begin
                 $display("function error: a=%d b=%d cin=%d output: sum=%d cout=%d  Expected: sum=%d cout=%d",
                                a_tb, b_tb, cin_tb, sum_tb, cout_tb, sum_exp, cout_exp);
            end

        end
        $display("test complete");
        $finish;
    end


endmodule
```

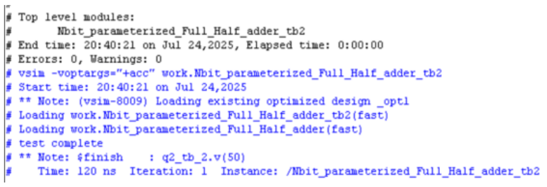
Simulation Tool for testbench 4:

a) Do file

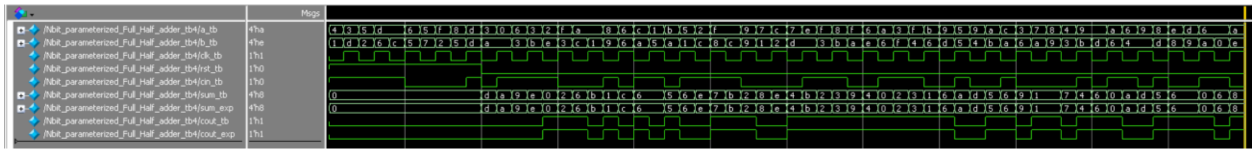


```
# Top level modules:
#       Nbit_parameterized_Full_Half_adder_tb4
# End time: 20:55:30 on Jul 24,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.Nbit_parameterized_Full_Half_adder_tb4
# Start time: 20:55:30 on Jul 24,2025
# ** Note: (vsim-3812) Design is being optimized...
# Loading work.Nbit_parameterized_Full_Half_adder_tb4(fast)
# Loading work.Nbit_parameterized_Full_Half_adder(fast)
# test complete
# ** Note: $finish    : q2_tb_4.v(51)
#    Time: 60 ns  Iteration: 0  Instance: /Nbit_parameterized_Full_Half_adder_tb4
```

b) Questasim



```
# Top level modules:
#       Nbit_parameterized_Full_Half_adder_tb2
# End time: 20:40:21 on Jul 24,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.Nbit_parameterized_Full_Half_adder_tb2
# Start time: 20:40:21 on Jul 24,2025
# ** Note: (vsim-8009) Loading existing optimized design _opt1
# Loading work.Nbit_parameterized_Full_Half_adder_tb2(fast)
# Loading work.Nbit_parameterized_Full_Half_adder(fast)
# test complete
# ** Note: $finish    : q2_tb_2.v(50)
#    Time: 120 ns  Iteration: 1  Instance: /Nbit_parameterized_Full_Half_adder_tb2
```

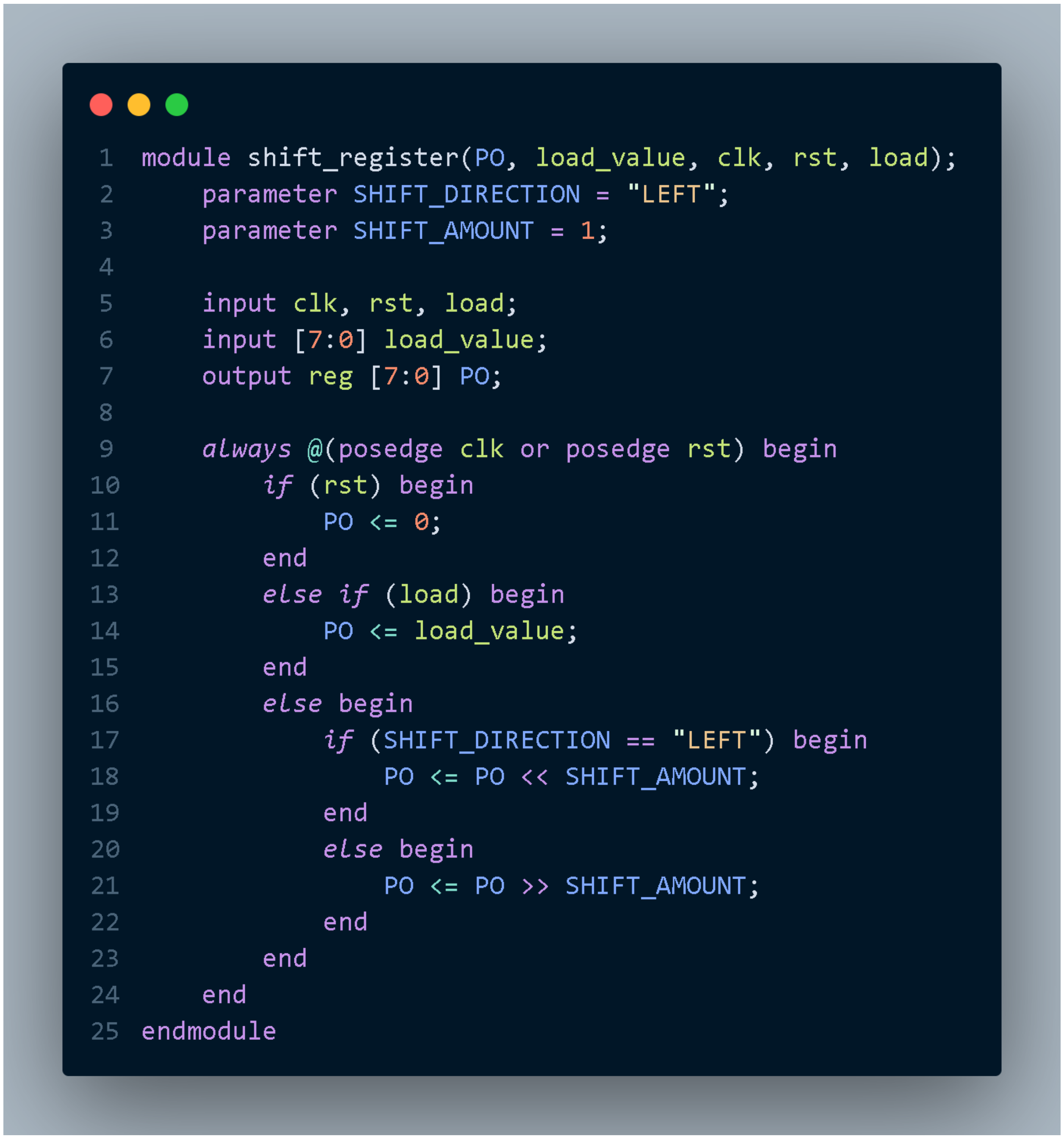


```
                                                       Msgs
/Nbit_parameterized_Full_Half_adder_tb4/a_tb          4'ha
/Nbit_parameterized_Full_Half_adder_tb4/b_tb          4'he
/Nbit_parameterized_Full_Half_adder_tb4/clk_tb        1'h1
/Nbit_parameterized_Full_Half_adder_tb4/rst_tb        1'h0
/Nbit_parameterized_Full_Half_adder_tb4/cin_tb        1'h0
/Nbit_parameterized_Full_Half_adder_tb4/sum_tb        4'h8
/Nbit_parameterized_Full_Half_adder_tb4/sum_exp       4'h8
/Nbit_parameterized_Full_Half_adder_tb4/cout_tb       1'h1
/Nbit_parameterized_Full_Half_adder_tb4/cout_exp      1'h1
```

Q3)

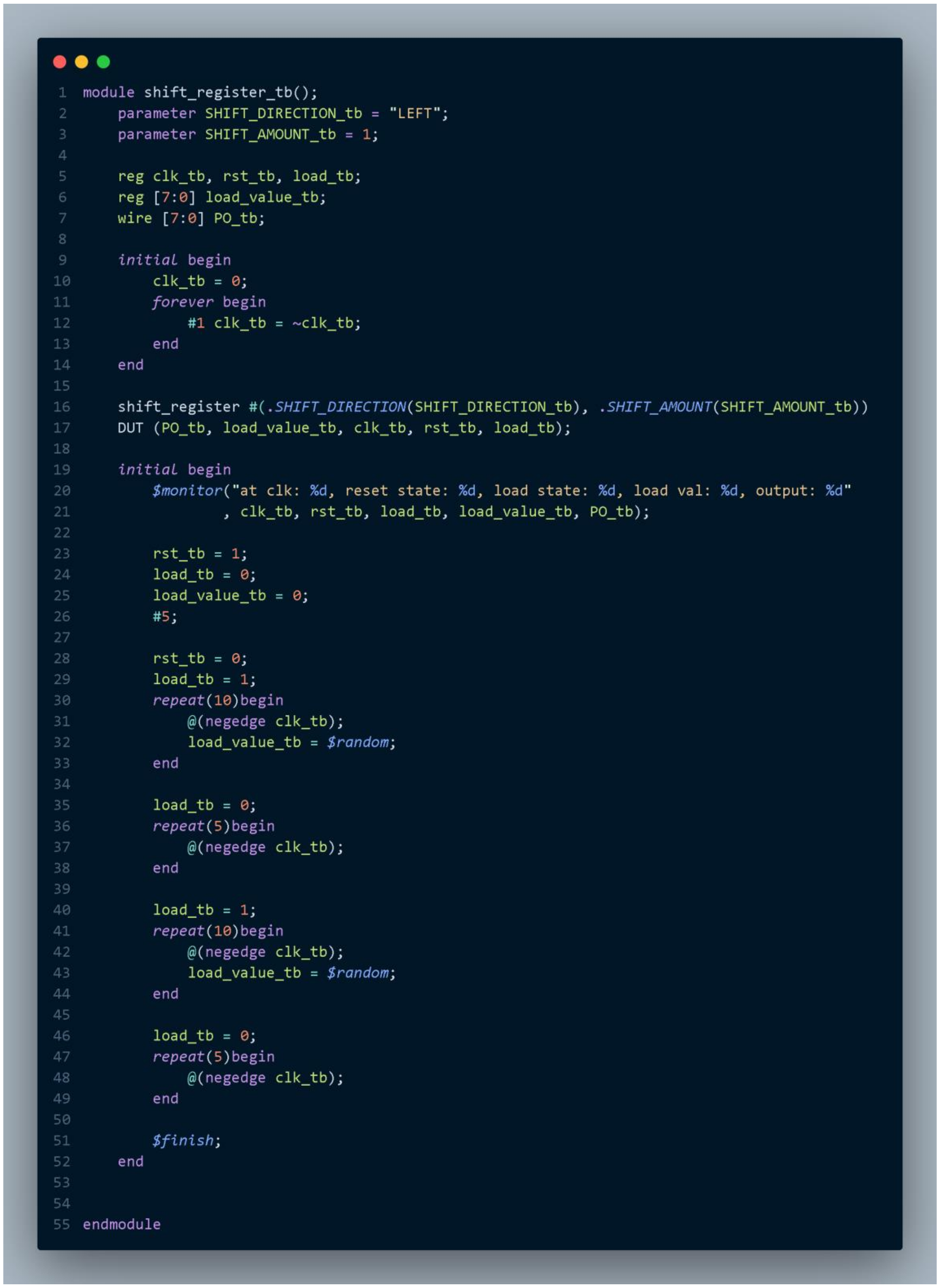
Verilog code:

a) **RTL design:**



```verilog
module shift_register(PO, load_value, clk, rst, load);
    parameter SHIFT_DIRECTION = "LEFT";
    parameter SHIFT_AMOUNT = 1;

    input clk, rst, load;
    input [7:0] load_value;
    output reg [7:0] PO;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            PO <= 0;
        end
        else if (load) begin
            PO <= load_value;
        end
        else begin
            if (SHIFT_DIRECTION == "LEFT") begin
                PO <= PO << SHIFT_AMOUNT;
            end
            else begin
                PO <= PO >> SHIFT_AMOUNT;
            end
        end
    end
endmodule
```

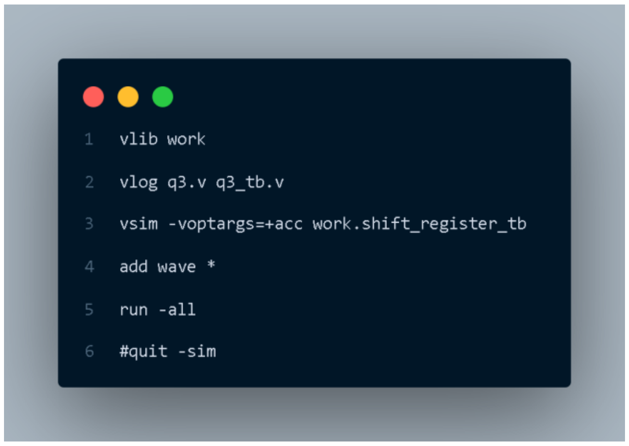
b) Testbench 1:



```
module shift_register_tb();
    parameter SHIFT_DIRECTION_tb = "LEFT";
    parameter SHIFT_AMOUNT_tb = 1;

    reg clk_tb, rst_tb, load_tb;
    reg [7:0] load_value_tb;
    wire [7:0] PO_tb;

    initial begin
        clk_tb = 0;
        forever begin
            #1 clk_tb = ~clk_tb;
        end
    end

    shift_register #(.SHIFT_DIRECTION(SHIFT_DIRECTION_tb), .SHIFT_AMOUNT(SHIFT_AMOUNT_tb))
    DUT (PO_tb, load_value_tb, clk_tb, rst_tb, load_tb);

    initial begin
        $monitor("at clk: %d, reset state: %d, load state: %d, load val: %d, output: %d"
                , clk_tb, rst_tb, load_tb, load_value_tb, PO_tb);

        rst_tb = 1;
        load_tb = 0;
        load_value_tb = 0;
        #5;

        rst_tb = 0;
        load_tb = 1;
        repeat(10)begin
            @(negedge clk_tb);
            load_value_tb = $random;
        end

        load_tb = 0;
        repeat(5)begin
            @(negedge clk_tb);
        end

        load_tb = 1;
        repeat(10)begin
            @(negedge clk_tb);
            load_value_tb = $random;
        end

        load_tb = 0;
        repeat(5)begin
            @(negedge clk_tb);
        end

        $finish;
    end


endmodule
```

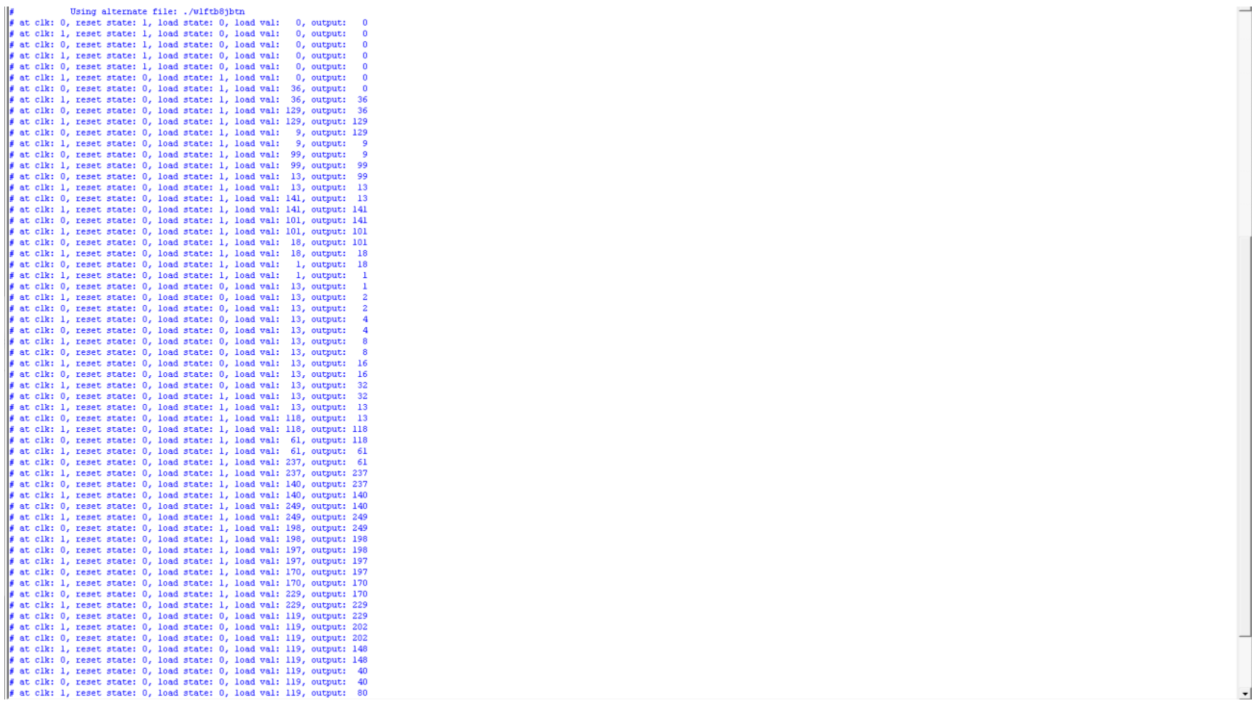
Simulation Tool

c) Do file 1

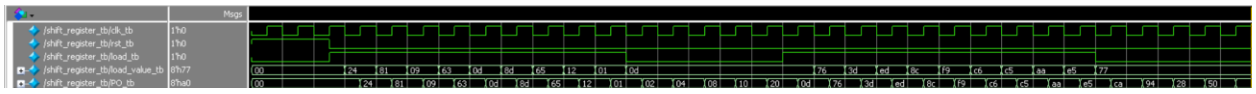


```
vlib work
vlog q3.v q3_tb.v
vsim -voptargs=+acc work.shift_register_tb
add wave *
run -all
#quit -sim
```

d) Questasim 1

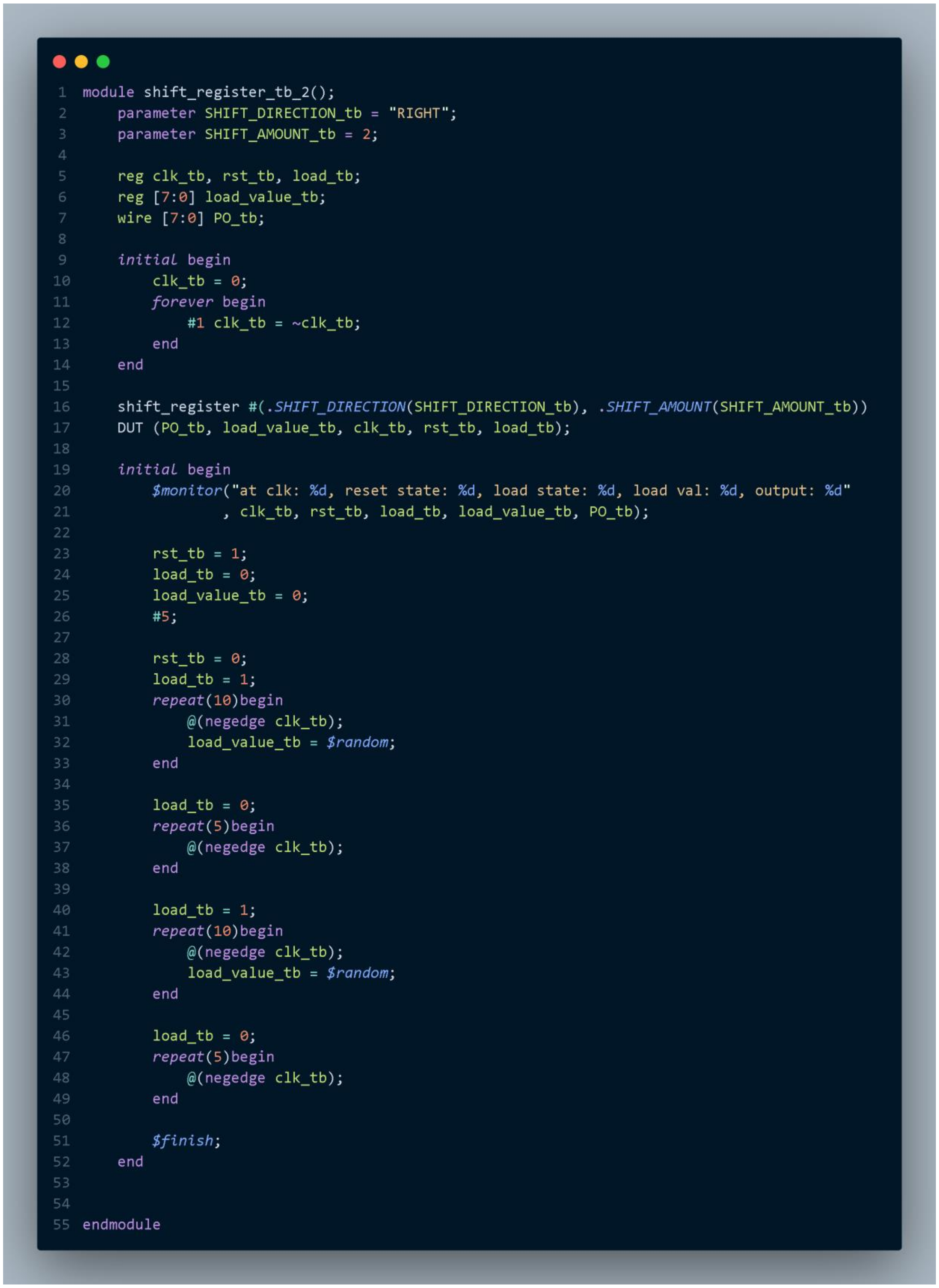


```
#          Using alternate file: ./wlftb8jbtn
# at clk: 0, reset state: 1, load state: 0, load val:   0, output:   0
# at clk: 1, reset state: 1, load state: 0, load val:   0, output:   0
# at clk: 0, reset state: 1, load state: 0, load val:   0, output:   0
# at clk: 1, reset state: 1, load state: 0, load val:   0, output:   0
# at clk: 0, reset state: 1, load state: 0, load val:   0, output:   0
# at clk: 1, reset state: 0, load state: 1, load val:   0, output:   0
# at clk: 0, reset state: 0, load state: 1, load val:  36, output:   0
# at clk: 1, reset state: 0, load state: 1, load val:  36, output:  36
# at clk: 0, reset state: 0, load state: 1, load val: 129, output:  36
# at clk: 1, reset state: 0, load state: 1, load val: 129, output: 129
# at clk: 0, reset state: 0, load state: 1, load val:   9, output: 129
# at clk: 1, reset state: 0, load state: 1, load val:   9, output:   9
# at clk: 0, reset state: 0, load state: 1, load val:  99, output:   9
# at clk: 1, reset state: 0, load state: 1, load val:  99, output:  99
# at clk: 0, reset state: 0, load state: 1, load val:  13, output:  99
# at clk: 1, reset state: 0, load state: 1, load val:  13, output:  13
# at clk: 0, reset state: 0, load state: 1, load val: 141, output:  13
# at clk: 1, reset state: 0, load state: 1, load val: 141, output: 141
# at clk: 0, reset state: 0, load state: 1, load val: 101, output: 141
# at clk: 1, reset state: 0, load state: 1, load val: 101, output: 101
# at clk: 0, reset state: 0, load state: 1, load val:  18, output: 101
# at clk: 1, reset state: 0, load state: 1, load val:  18, output:  18
# at clk: 0, reset state: 0, load state: 1, load val:   1, output:  18
# at clk: 1, reset state: 0, load state: 1, load val:   1, output:   1
# at clk: 0, reset state: 0, load state: 0, load val:  13, output:   1
# at clk: 1, reset state: 0, load state: 0, load val:  13, output:   2
# at clk: 0, reset state: 0, load state: 0, load val:  13, output:   2
# at clk: 1, reset state: 0, load state: 0, load val:  13, output:   4
# at clk: 0, reset state: 0, load state: 0, load val:  13, output:   4
# at clk: 1, reset state: 0, load state: 0, load val:  13, output:   8
# at clk: 0, reset state: 0, load state: 0, load val:  13, output:   8
# at clk: 1, reset state: 0, load state: 0, load val:  13, output:  16
# at clk: 0, reset state: 0, load state: 0, load val:  13, output:  16
# at clk: 1, reset state: 0, load state: 0, load val:  13, output:  32
# at clk: 0, reset state: 0, load state: 1, load val:  13, output:  32
# at clk: 1, reset state: 0, load state: 1, load val:  13, output:  13
# at clk: 0, reset state: 0, load state: 1, load val: 118, output:  13
# at clk: 1, reset state: 0, load state: 1, load val: 118, output: 118
# at clk: 0, reset state: 0, load state: 1, load val:  61, output: 118
# at clk: 1, reset state: 0, load state: 1, load val:  61, output:  61
# at clk: 0, reset state: 0, load state: 1, load val: 237, output:  61
# at clk: 1, reset state: 0, load state: 1, load val: 237, output: 237
# at clk: 0, reset state: 0, load state: 1, load val: 140, output: 237
# at clk: 1, reset state: 0, load state: 1, load val: 140, output: 140
# at clk: 0, reset state: 0, load state: 1, load val: 249, output: 140
# at clk: 1, reset state: 0, load state: 1, load val: 249, output: 249
# at clk: 0, reset state: 0, load state: 1, load val: 198, output: 249
# at clk: 1, reset state: 0, load state: 1, load val: 198, output: 198
# at clk: 0, reset state: 0, load state: 1, load val: 197, output: 198
# at clk: 1, reset state: 0, load state: 1, load val: 197, output: 197
# at clk: 0, reset state: 0, load state: 1, load val: 170, output: 197
# at clk: 1, reset state: 0, load state: 1, load val: 170, output: 170
# at clk: 0, reset state: 0, load state: 1, load val: 229, output: 170
# at clk: 1, reset state: 0, load state: 1, load val: 229, output: 229
# at clk: 0, reset state: 0, load state: 0, load val: 119, output: 229
# at clk: 1, reset state: 0, load state: 0, load val: 119, output: 202
# at clk: 0, reset state: 0, load state: 0, load val: 119, output: 202
# at clk: 1, reset state: 0, load state: 0, load val: 119, output: 148
# at clk: 0, reset state: 0, load state: 0, load val: 119, output: 148
# at clk: 1, reset state: 0, load state: 0, load val: 119, output:  40
# at clk: 0, reset state: 0, load state: 0, load val: 119, output:  40
# at clk: 1, reset state: 0, load state: 0, load val: 119, output:  80
```



```
                                      Msgs
/shift_register_tb/clk_tb             1'h0
/shift_register_tb/rst_tb             1'h0
/shift_register_tb/load_tb            1'h0
/shift_register_tb/load_value_tb      8'h77
/shift_register_tb/PO_tb              8'ha0
```

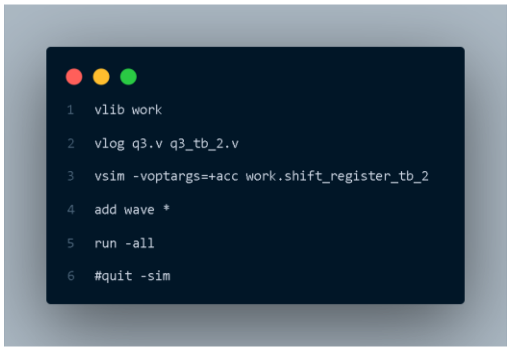
e) Testbench 2:



```verilog
module shift_register_tb_2();
    parameter SHIFT_DIRECTION_tb = "RIGHT";
    parameter SHIFT_AMOUNT_tb = 2;

    reg clk_tb, rst_tb, load_tb;
    reg [7:0] load_value_tb;
    wire [7:0] PO_tb;

    initial begin
        clk_tb = 0;
        forever begin
            #1 clk_tb = ~clk_tb;
        end
    end

    shift_register #(.SHIFT_DIRECTION(SHIFT_DIRECTION_tb), .SHIFT_AMOUNT(SHIFT_AMOUNT_tb))
    DUT (PO_tb, load_value_tb, clk_tb, rst_tb, load_tb);

    initial begin
        $monitor("at clk: %d, reset state: %d, load state: %d, load val: %d, output: %d"
                , clk_tb, rst_tb, load_tb, load_value_tb, PO_tb);

        rst_tb = 1;
        load_tb = 0;
        load_value_tb = 0;
        #5;

        rst_tb = 0;
        load_tb = 1;
        repeat(10)begin
            @(negedge clk_tb);
            load_value_tb = $random;
        end

        load_tb = 0;
        repeat(5)begin
            @(negedge clk_tb);
        end

        load_tb = 1;
        repeat(10)begin
            @(negedge clk_tb);
            load_value_tb = $random;
        end

        load_tb = 0;
        repeat(5)begin
            @(negedge clk_tb);
        end

        $finish;
    end


endmodule
```

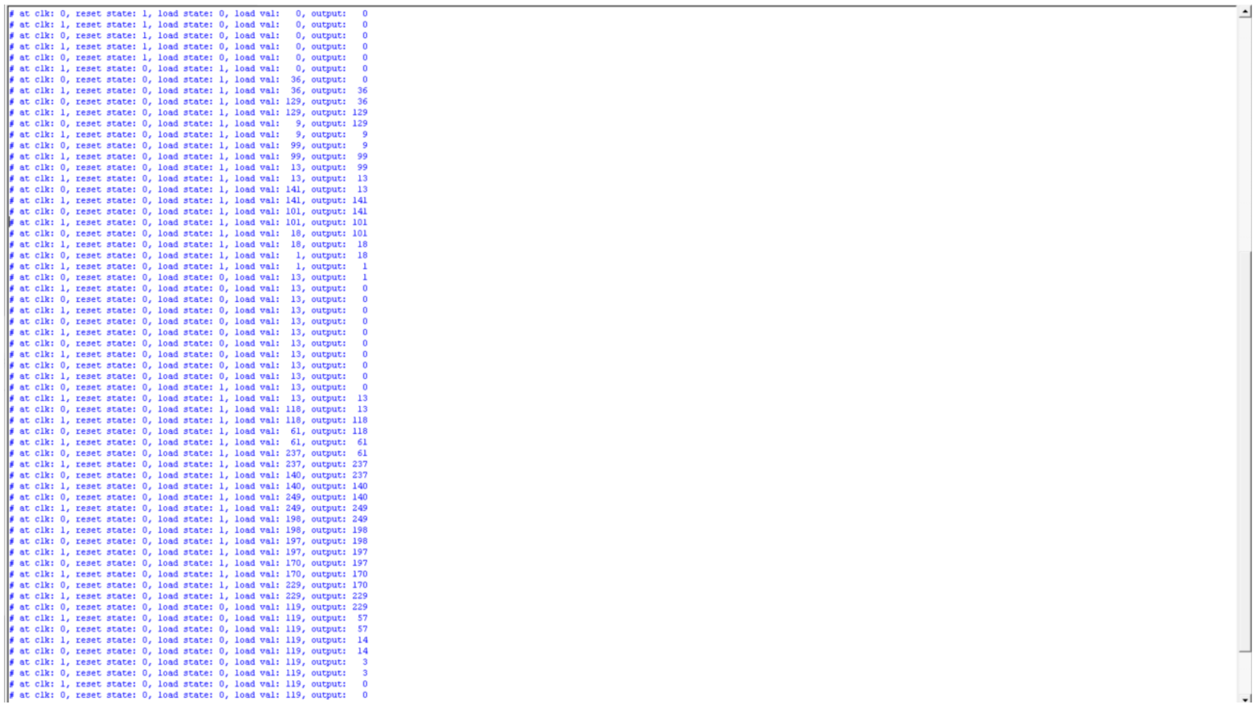
Simulation Tool

f) Do file 2

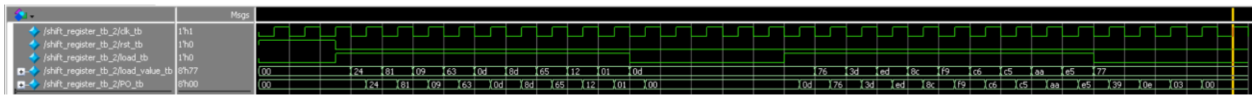


```
vlib work
vlog q3.v q3_tb_2.v
vsim -voptargs=+acc work.shift_register_tb_2
add wave *
run -all
#quit -sim
```

g) Questasim 2



```
# at clk: 0, reset state: 1, load state: 0, load val:   0, output:   0
# at clk: 1, reset state: 1, load state: 0, load val:   0, output:   0
# at clk: 0, reset state: 1, load state: 0, load val:   0, output:   0
# at clk: 1, reset state: 1, load state: 0, load val:   0, output:   0
# at clk: 0, reset state: 1, load state: 0, load val:   0, output:   0
# at clk: 1, reset state: 0, load state: 1, load val:   0, output:   0
# at clk: 0, reset state: 0, load state: 1, load val:  36, output:   0
# at clk: 1, reset state: 0, load state: 1, load val:  36, output:  36
# at clk: 0, reset state: 0, load state: 1, load val: 129, output:  36
# at clk: 1, reset state: 0, load state: 1, load val: 129, output: 129
# at clk: 0, reset state: 0, load state: 1, load val:   9, output: 129
# at clk: 1, reset state: 0, load state: 1, load val:   9, output:   9
# at clk: 0, reset state: 0, load state: 1, load val:  99, output:   9
# at clk: 1, reset state: 0, load state: 1, load val:  99, output:  99
# at clk: 0, reset state: 0, load state: 1, load val:  13, output:  99
# at clk: 1, reset state: 0, load state: 1, load val:  13, output:  13
# at clk: 0, reset state: 0, load state: 1, load val: 141, output:  13
# at clk: 1, reset state: 0, load state: 1, load val: 141, output: 141
# at clk: 0, reset state: 0, load state: 1, load val: 101, output: 141
# at clk: 1, reset state: 0, load state: 1, load val: 101, output: 101
# at clk: 0, reset state: 0, load state: 1, load val:  18, output: 101
# at clk: 1, reset state: 0, load state: 1, load val:  18, output:  18
# at clk: 0, reset state: 0, load state: 1, load val:   1, output:  18
# at clk: 1, reset state: 0, load state: 1, load val:   1, output:   1
# at clk: 0, reset state: 0, load state: 0, load val:  13, output:   1
# at clk: 1, reset state: 0, load state: 0, load val:  13, output:   0
# at clk: 0, reset state: 0, load state: 0, load val:  13, output:   0
# at clk: 1, reset state: 0, load state: 0, load val:  13, output:   0
# at clk: 0, reset state: 0, load state: 0, load val:  13, output:   0
# at clk: 1, reset state: 0, load state: 0, load val:  13, output:   0
# at clk: 0, reset state: 0, load state: 0, load val:  13, output:   0
# at clk: 1, reset state: 0, load state: 0, load val:  13, output:   0
# at clk: 0, reset state: 0, load state: 0, load val:  13, output:   0
# at clk: 1, reset state: 0, load state: 0, load val:  13, output:   0
# at clk: 0, reset state: 0, load state: 1, load val:  13, output:   0
# at clk: 1, reset state: 0, load state: 1, load val:  13, output:  13
# at clk: 0, reset state: 0, load state: 1, load val: 118, output:  13
# at clk: 1, reset state: 0, load state: 1, load val: 118, output: 118
# at clk: 0, reset state: 0, load state: 1, load val:  61, output: 118
# at clk: 1, reset state: 0, load state: 1, load val:  61, output:  61
# at clk: 0, reset state: 0, load state: 1, load val: 237, output:  61
# at clk: 1, reset state: 0, load state: 1, load val: 237, output: 237
# at clk: 0, reset state: 0, load state: 1, load val: 140, output: 237
# at clk: 1, reset state: 0, load state: 1, load val: 140, output: 140
# at clk: 0, reset state: 0, load state: 1, load val: 249, output: 140
# at clk: 1, reset state: 0, load state: 1, load val: 249, output: 249
# at clk: 0, reset state: 0, load state: 1, load val: 198, output: 249
# at clk: 1, reset state: 0, load state: 1, load val: 198, output: 198
# at clk: 0, reset state: 0, load state: 1, load val: 197, output: 198
# at clk: 1, reset state: 0, load state: 1, load val: 197, output: 197
# at clk: 0, reset state: 0, load state: 1, load val: 170, output: 197
# at clk: 1, reset state: 0, load state: 1, load val: 170, output: 170
# at clk: 0, reset state: 0, load state: 1, load val: 229, output: 170
# at clk: 1, reset state: 0, load state: 1, load val: 229, output: 229
# at clk: 0, reset state: 0, load state: 0, load val: 119, output: 229
# at clk: 1, reset state: 0, load state: 0, load val: 119, output:  57
# at clk: 0, reset state: 0, load state: 0, load val: 119, output:  57
# at clk: 1, reset state: 0, load state: 0, load val: 119, output:  14
# at clk: 0, reset state: 0, load state: 0, load val: 119, output:  14
# at clk: 1, reset state: 0, load state: 0, load val: 119, output:   3
# at clk: 0, reset state: 0, load state: 0, load val: 119, output:   3
# at clk: 1, reset state: 0, load state: 0, load val: 119, output:   0
# at clk: 0, reset state: 0, load state: 0, load val: 119, output:   0
```



```
Msgs
/shift_register_tb_2/clk_tb          1'h1
/shift_register_tb_2/rst_tb          1'h0
/shift_register_tb_2/load_tb         1'h0
/shift_register_tb_2/load_value_tb   8'h77   00 24 81 09 63 0d 8d 65 12 01 0d 76 3d ed 8c f9 c6 c5 aa e5 77
/shift_register_tb_2/PO_tb           8'h00   00 24 81 09 63 0d 8d 65 12 01 00 0d 76 3d ed 8c f9 c6 c5 aa e5 39 0e 03 00
```