**Assignment 3**

Ameer Ashraf Louly – G3

A screen shot of a computer screen

AI-generated content may be incorrect.Question 1:

Figure TB Code

A screenshot of a computer

AI-generated content may be incorrect.

Figure DO File

Figure Questa Lint

Question 2:

Write a testbench for question 5 part C in assignment 2. Test the parameterized asynchronous FlipFlop using 2 testbenches, testbench 1 that overrides the design with FF\_TYPE = “DFF” and the testbench 2 overrides parameter with FF\_TYPE = “TFF”.

A screenshot of a computer program

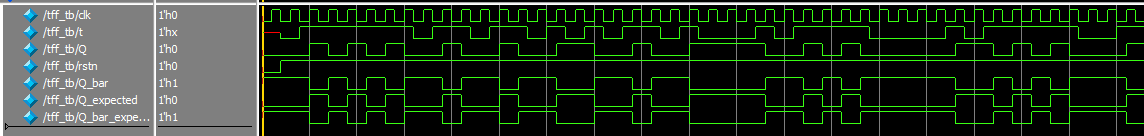
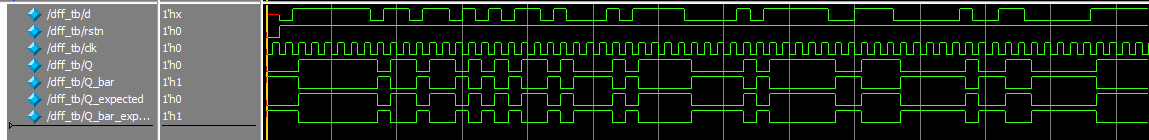
AI-generated content may be incorrect.

Figure DFF TB

A screenshot of a computer program

AI-generated content may be incorrect.

Figure TFF TB

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.Figure DO Files

Question 3:

A screenshot of a computer program

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.Implement BCD up counter (MOD 10 counter), where the counter has 10 states. The counter will divide the clock frequency by 10.

Figure BCD Counter Code

Figure TB Code

A screenshot of a computer program

AI-generated content may be incorrect.

Figure DO File

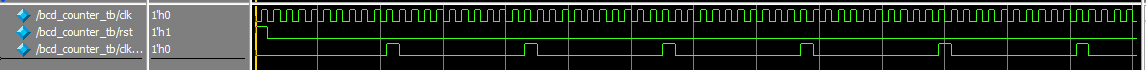
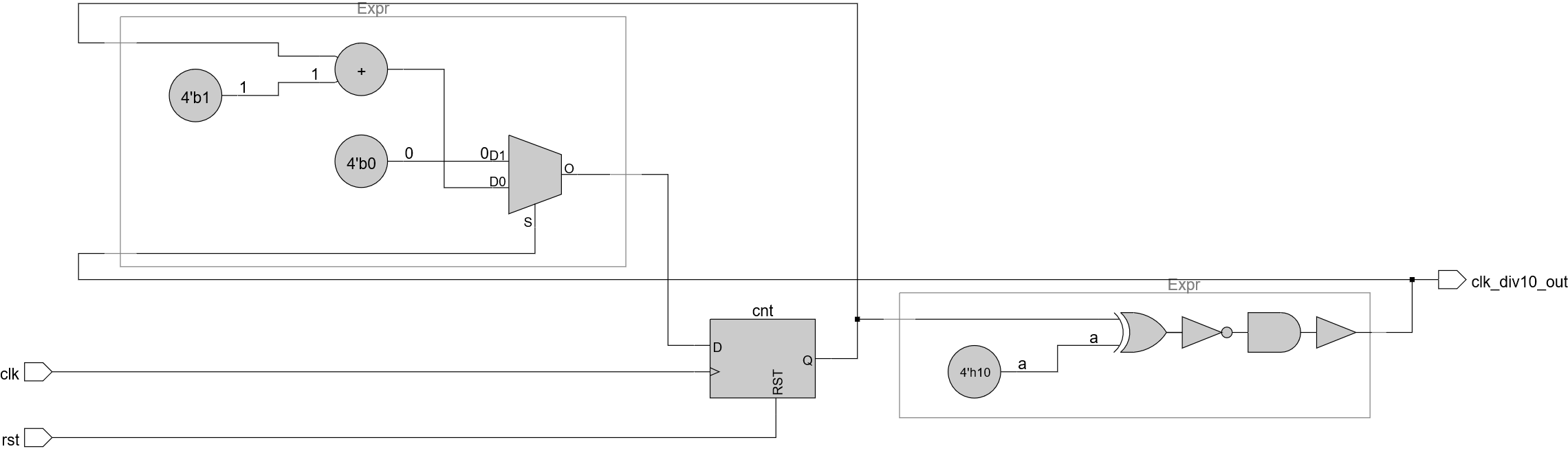


Figure BCD Counter Schematic

Figure Waveform Output

Question 4: Ripple Counter

A computer screen shot of a program code

AI-generated content may be incorrect.

Figure Ripple Counter Code

A screenshot of a computer program

AI-generated content may be incorrect.

Figure Ripple Counter TB

A screenshot of a computer

AI-generated content may be incorrect.

Figure Do File

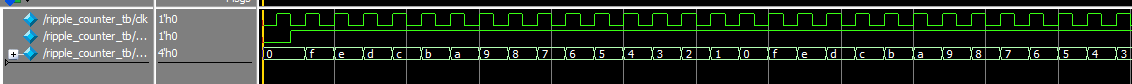


Figure Waveform

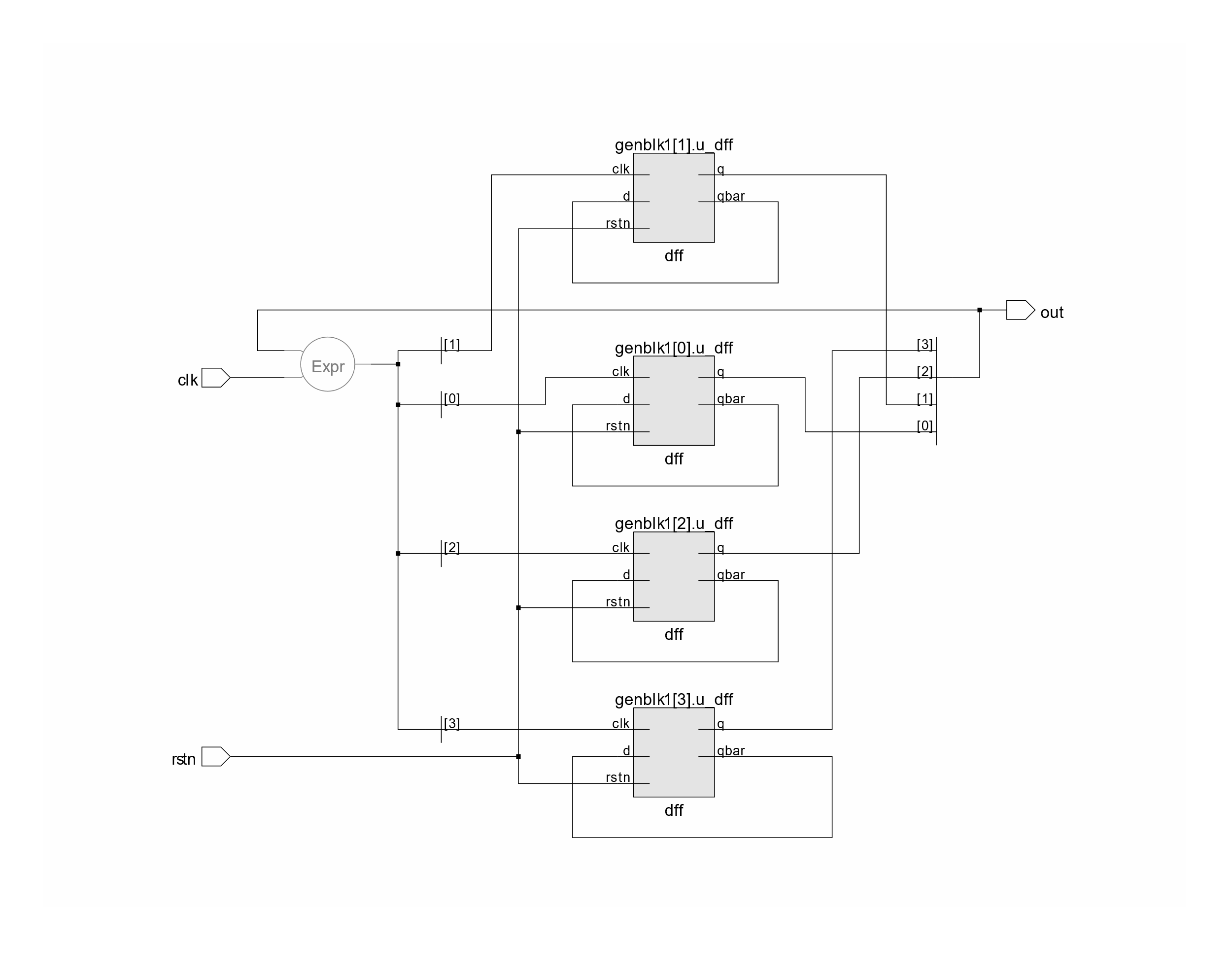


Figure Ripple Counter Schematic

Question 5:



Figure SLE Shift Register Code

module shift\_reg\_tb();

    parameter LOAD\_AVALUE = 2;

    parameter SHIFT\_DIRECTION = "LEFT";

    parameter LOAD\_SVALUE = 4;

    parameter SHIFT\_WIDTH = 8;

    reg sclr, sset, shiftin, load, clk, enable, aclr, aset;

    reg [SHIFT\_WIDTH - 1 : 0] data, q\_expected;

    wire shiftout;

    wire [SHIFT\_WIDTH - 1 : 0] q;

    shift\_reg #(.LOAD\_AVALUE(LOAD\_AVALUE),

                .LOAD\_SVALUE(LOAD\_SVALUE),

                .SHIFT\_DIRECTION(SHIFT\_DIRECTION),

                .SHIFT\_WIDTH(SHIFT\_WIDTH))

                DUT (

                    .sclr(sclr),

                    .sset(sset),

                    .shiftin(shiftin),

                    .load(load),

                    .clk(clk),

                    .enable(enable),

                    .aclr(aclr),

                    .aset(aset),

                    .shiftout(shiftout),

                    .q(q)

                );

    initial begin

        clk = 0;

        forever begin

            #1 clk = ~clk;

        end

    end

    initial begin

        // Check for aclr

        aset = 1;

        aclr = 1;

        repeat(50) begin

            sclr = $random;

            sset = $random;

            shiftin = $random;

            load = $random;

            enable = $random;

            data = $random;

            @(negedge clk);

            if(q != 0) begin

                $display("Error - aclr");

                $exit;

            end

        end

        // check for aset

        aclr = 0;

        aset = 1;

        repeat(50) begin

            sclr = $random;

            sset = $random;

            shiftin = $random;

            load = $random;

            enable = $random;

            data = $random;

            @(negedge clk);

            if(q != LOAD\_AVALUE) begin

                $display("Error - aset");

                $exit;

            end

        end

        // Test for sclr

        aclr = 0;

        aset = 0;

        sclr = 1;

        sset = 1;

        repeat(50) begin

            shiftin = $random;

            load = $random;

            enable = $random;

            data = $random;

            @(negedge clk);

            if(q != 0) begin

                $display("Error - sclr");

                $exit;

            end

        end

        // Test for sset

        aclr = 0;

        aset = 0;

        sclr = 0;

        sset = 1;

        enable = 1;

        repeat(50) begin

            shiftin = $random;

            load = $random;

            data = $random;

            @(negedge clk);

            if(q != LOAD\_SVALUE) begin

                $display("Error - sset");

                $exit;

            end

        end

        // Test for load

        aclr = 0;

        aset = 0;

        sclr = 0;

        sset = 0;

        load = 1;

        enable = 1;

        repeat(50) begin

            shiftin = $random;

            data = $random;

            q\_expected = data;

            @(negedge clk);

            if(q != q\_expected) begin

                $display("Error - load");

                $exit;

            end

        end

        // test for shifting

        // Clearing Input for signifcant amount of time

        aclr = 1;

        repeat(50) @(negedge clk);

        aclr = 0;

        // Setting value

        aset = 1;

        @(negedge clk);

        aset = 0;

        // Deactivating Control Signals

        aclr = 0;

        aset = 0;

        sclr = 0;

        sset = 0;

        load = 0;

        shiftin = 0;

        enable = 1;

        $display("Starting Shift Test");

        repeat(10) @(negedge clk);

        $exit;

    end

endmodule

A screen shot of a computer

AI-generated content may be incorrect.

Figure Shift Functionality Working

A screenshot of a computer program

AI-generated content may be incorrect.

Figure Do file

Question 6:

Implement the following SLE (sequential logic element). This design will act as flipflop or latch based on the LAT signal as demonstrated in the truth table.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure SLE Latch/FF Code

A screenshot of a computer

AI-generated content may be incorrect.

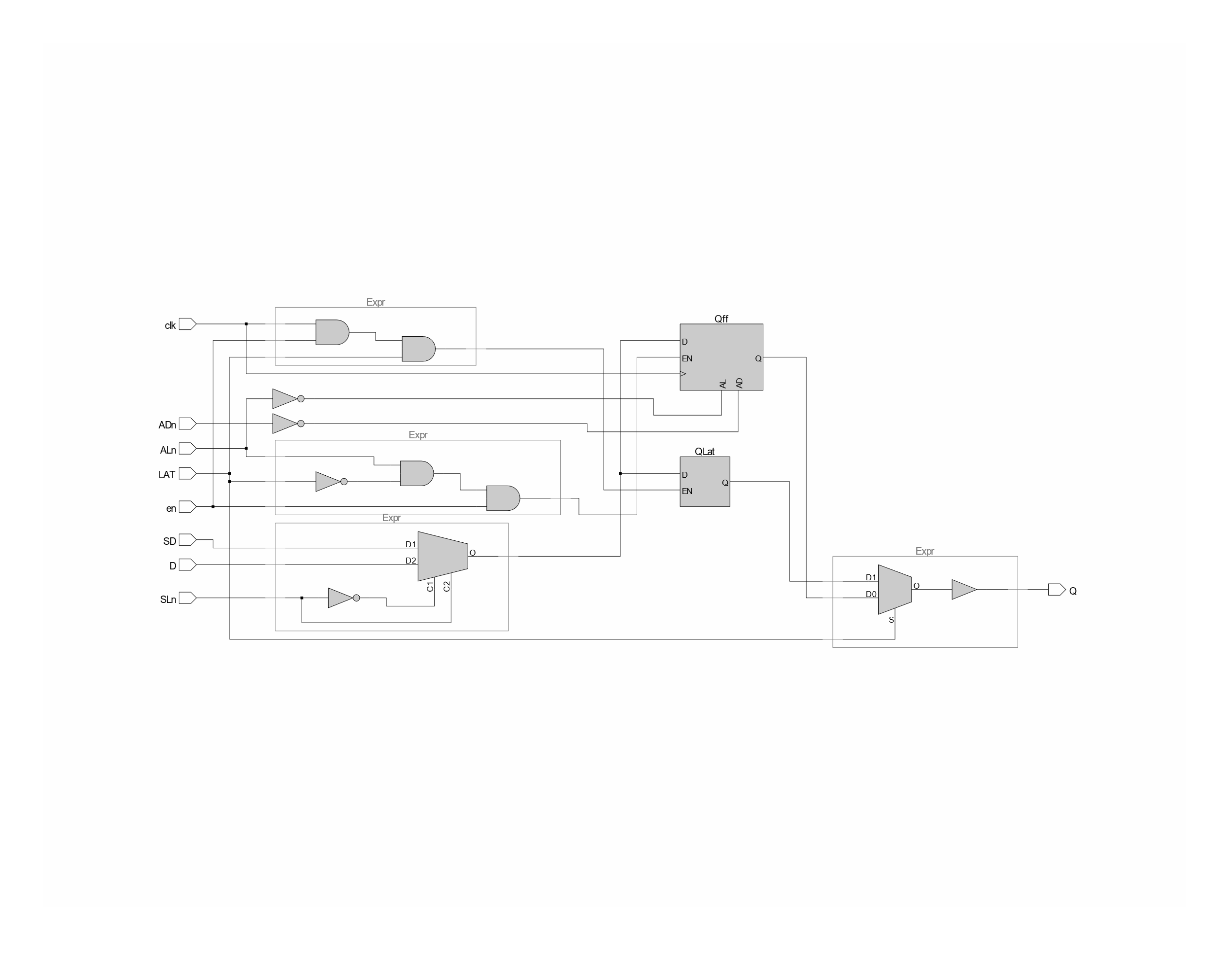
Figure Questa Lint

Figure Schematic

module sle\_tb();

    reg D, clk, en, ALn, ADn, SLn, SD, LAT;

    wire Q;

    sle DUT(D, clk, en, ALn, ADn, SLn, SD, LAT, Q);

    initial begin

        clk = 0;

        forever begin

            #10 clk = ~clk;

        end

    end

    initial begin

        LAT = 0;

        ALn = 0;

        repeat(10) begin

            ADn = $random;

            D = $random;

            @(negedge clk);

            if(Q != ~ADn) begin

                $display("Error Async Load");

                $exit;

            end

        end

        ALn = 1;

        SLn = 0;

        en = 1;

        repeat(10)begin

            SD = $random;

            D = $random;

            @(negedge clk);

            if(Q != SD) begin

                $display("Error Sync Load");

                $exit;

            end

        end

        ALn = 1;

        SLn = 1;

        en = 1;

        repeat(10)begin

            SD = $random;

            D = $random;

            @(negedge clk);

            if(Q != D) begin

                $display("Error Latch Operation");

                $exit;

            end

        end

        LAT = 1;

        ALn = 1;

        SLn = 0;

        en = 1;

        repeat(20)begin

            SD = $random;

            D = $random;

            #1;

        end

        ALn = 1;

        SLn = 1;

        en = 1;

        repeat(20)begin

            SD = $random;

            D = $random;

            #1;

        end

        $display("Design Works");

        $exit;

    end

endmodule

Figure Testbench Code