يوسف أحمد محمد ابراهيم

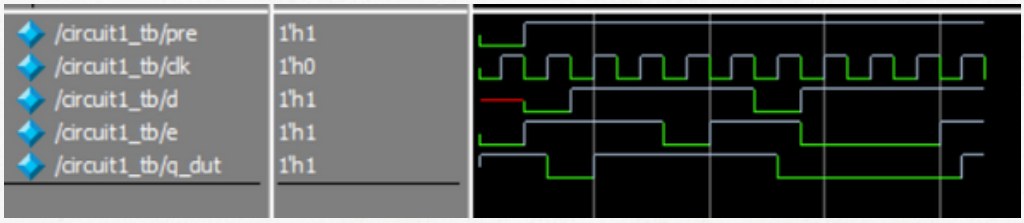
Assignment 3

Circuit_1 :

- **testbench:**

```
module circuit1_tb();

    reg pre, clk, d, e;
    wire q_dut;

    circuit_1 dut(pre, d, e, clk, q_dut);

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

    initial begin
        pre = 0;
        e = 0;
        @(negedge clk);
        pre = 1;

        repeat(10) begin
            d = $random;
            e = $random;
            @(negedge clk);
        end

        $stop;
    end

endmodule
```

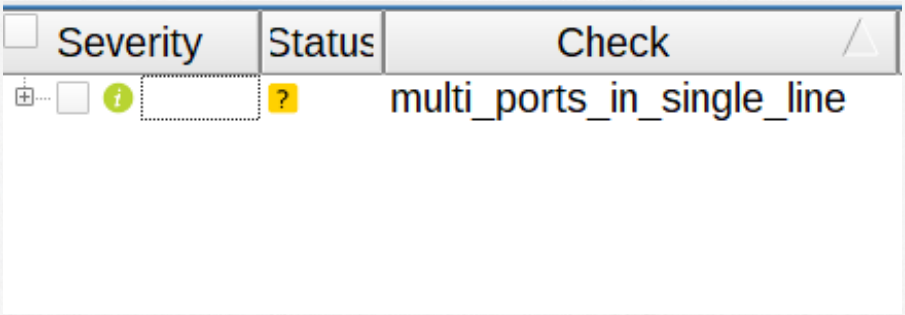
- **output:**



- **do file:**

```
vlib work
vlog 1.v 1_tb.v
vsim -voptargs=+acc work.circuit1_tb
add wave *

run -all

#quit -sim
```

- **lint result:**

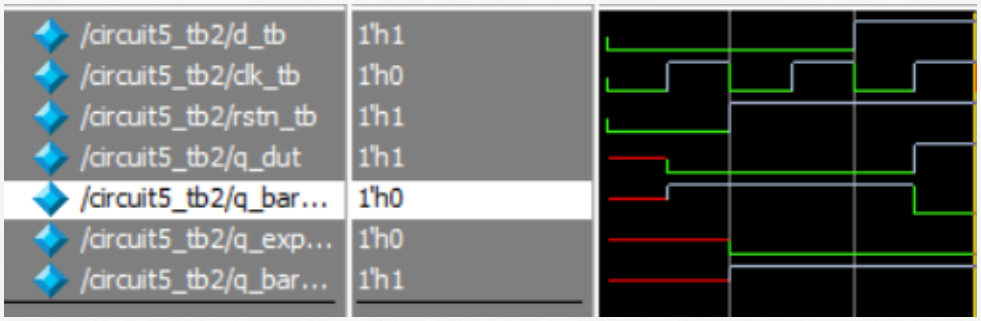


Circuit_2:

- **testbench1:**

```
module circuit5_tb1 ();
    reg d_tb, clk_tb, rstn_tb;
    wire q_dut, q_bar_dut, q_expected, q_bar_expected;

    circuit_5c #("DFF") dut(d_tb, clk_tb, rstn_tb, q_dut, q_bar_dut);
    circuit_5b expected(d_tb, clk_tb, rstn_tb, q_expected, q_bar_expected);

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

    initial begin
        rstn_tb = 0;
        d_tb = 0;
        @(negedge clk_tb);
        rstn_tb = 1;

        repeat(10) begin
            d_tb = $random;
            @(negedge clk_tb);
            if (q_dut != q_expected || q_bar_dut != q_bar_expected) begin
                $display("Error! expected: %d, dut: %d", q_expected, q_dut);
                $stop;
            end
        end
        $stop;
    end
endmodule
```

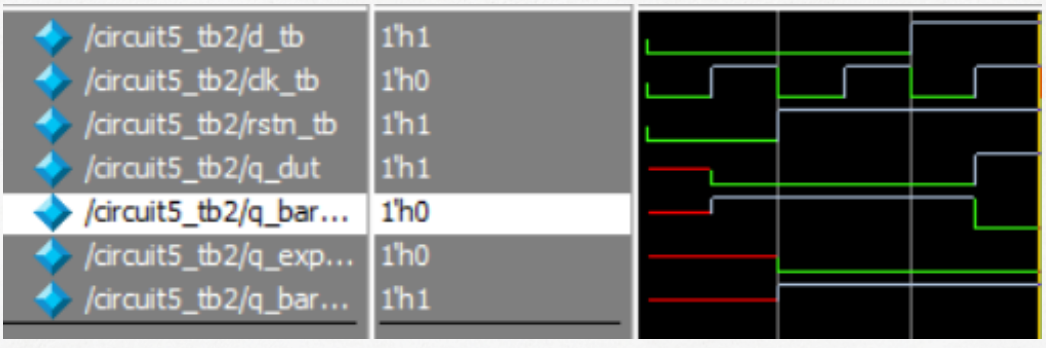
- **output:**



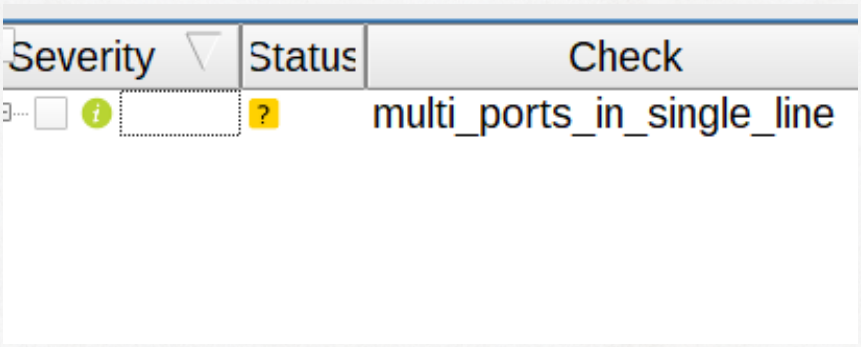
- **testbench2:**

```
33  module circuit5_tb2 ();
34      reg d_tb, clk_tb, rstn_tb;
35      wire q_dut, q_bar_dut, q_expected, q_bar_expected;
36
37      circuit_5c #("TFF") dut(d_tb, clk_tb, rstn_tb, q_dut, q_bar_dut);
38      circuit_5a expected(d_tb, clk_tb, rstn_tb, q_expected, q_bar_expected);
39
40      initial begin
41          clk_tb = 0;
42          forever
43              #1 clk_tb = ~clk_tb;
44      end
45
46      initial begin
47          rstn_tb = 0;
48          d_tb = 0;
49          @(negedge clk_tb);
50          rstn_tb = 1;
51
52          repeat(10) begin
53              d_tb = $random;
54              @(negedge clk_tb);
55              if (q_dut != q_expected || q_bar_dut != q_bar_expected) begin
56                  $display("Error! expected: %d, dut: %d", q_expected, q_dut);
57                  $stop;
58              end
59          end
60          $stop;
61      end
62  endmodule
```

- **output:**



- **lint result**



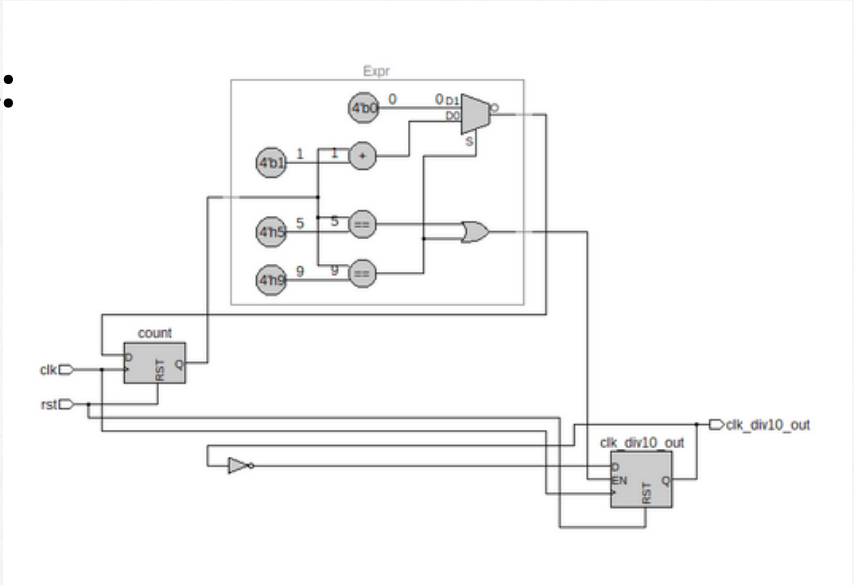
Circuit_3:

```
1   module circuit_3 (input clk, rst, output reg clk_div10_out);
2       reg [3:0]count;
3       always @(posedge clk, posedge rst) begin
4           if (rst) begin
5               count <= 4'b0000;
6               clk_div10_out <= 4'b0000;
7           end
8           else begin
9               if (count == 4'b1001) begin
10                  count <= 4'b0000;
11                  clk_div10_out <= ~clk_div10_out;
12              end
13              else begin
14                  count <= count + 1;
15                  if (count == 4'b0101)
16                      clk_div10_out <= ~clk_div10_out;
17              end
18          end
19      end
20  endmodule
```

- **lint result:**

Severity	Status	Check
	?	assign_width_overflow
	?	async_reset_active_high
	?	multi_ports_in_single_line

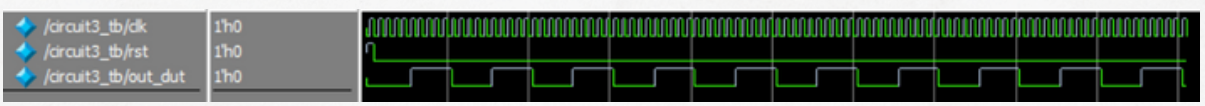
- **schematic:**



- **testbench:**

```
module circuit3_tb();
    reg clk, rst;
    wire out_dut;

    circuit_3 dut(clk, rst, out_dut);

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

    initial begin
        rst = 1;
        @(negedge clk);
        rst = 0;
        repeat(100)
            @(negedge clk);
        $stop;
    end

endmodule
```

- **output:**



Circuit_4 :

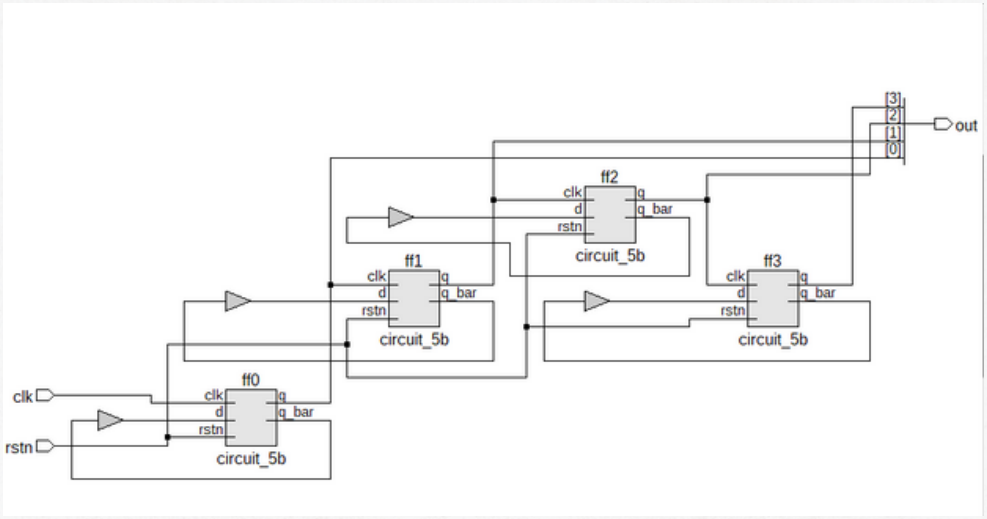
- **module**

```
1   module circuit_4(input clk, rstn, output [3:0] out);
2       wire q0, q1, q2, q3;
3       wire q_bar0, q_bar1, q_bar2, q_bar3;
4       wire d0, d1, d2, d3;
5
6       assign d0 = q_bar0;
7       assign d1 = q_bar1;
8       assign d2 = q_bar2;
9       assign d3 = q_bar3;
10
11      circuit_5b ff0(d0, rstn, clk, q0, q_bar0);
12      circuit_5b ff1(d1, rstn, q0, q1, q_bar1);
13      circuit_5b ff2(d2, rstn, q1, q2, q_bar2);
14      circuit_5b ff3(d3, rstn, q2, q3, q_bar3);
15
16      assign out = {q3, q2, q1, q0};
17
18  endmodule
```

- **lint result:**

Severity	Status	Check
	?	re_entrant_output
	?	re_entrant_output
	?	re_entrant_output
	?	re_entrant_output
	?	multi_ports_in_single_line
	?	multi_ports_in_single_line

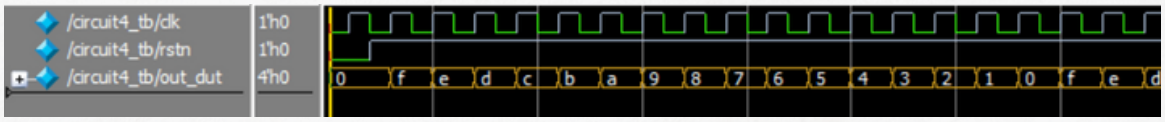
- **schematic:**


[3]
[2]
[1]
[0]
out
ff2
clk
d
rstn
q
q_bar
circuit_5b
ff1
clk
d
rstn
q
q_bar
circuit_5b
ff3
clk
d
rstn
q
q_bar
circuit_5b
ff0
clk
d
rstn
q
q_bar
circuit_5b
clk
rstn

- **testbench:**

```
module circuit4_tb();
    reg clk, rstn;
    wire [3:0]out_dut;

    circuit_4 dut(clk, rstn, out_dut);

    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end
    initial begin
      rstn = 0;
      @(negedge clk);
      rstn = 1;

        repeat(100)
            @(negedge clk);
        $stop;
    end

endmodule
```

- **output:**



Circuit_5:

```
module circuit_5 #(parameter LOAD_AVALUE = 1, SHIFT_DIRECTION = "LEFT", LOAD_SVALUE = 1, SHIFT_WIDTH = 8)
                  (input sclr, sset, shiftin, load, clk, en, aclr, aset,
                  input [SHIFT_WIDTH-1:0]d,
                  output reg shiftout,
                  output reg [SHIFT_WIDTH-1:0]q);

    always @(posedge clk, posedge aclr, posedge aset) begin
        if(aclr)
            q <= 0;
        else if(aset)
            q <= LOAD_AVALUE;
        else
            if(en) begin
                if(sclr)
                    q <= 0;
                else if(sset)
                    q <= LOAD_SVALUE;
                else if(load)
                    q <= d;
                else begin
                    if (SHIFT_DIRECTION == "LEFT")
                        {shiftout, q} <= {q, shiftin};
                    else if(SHIFT_DIRECTION == "RIGHT")
                        {q, shiftout} <= {shiftin, q};
                end
            end
    end

endmodule
```

- **testbench parts:**
  - **stimulus declaration and clock generation:**

```
module circuit5_tb();
    reg sclr, sset, shiftin_tb, load,
        clk, en, aclr, aset;
    reg [7:0]d, q_expected;
    wire shiftout_tb;
    wire [7:0]q_dut;
    circuit_5 #(.LOAD_AVALUE(2), .LOAD_SVALUE(4))
        dut(sclr, sset, shiftin_tb, load, clk,
            en, aclr, aset, d, shiftout_tb, q_dut);
    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end
```

- **check 1 & 2**

```
initial begin
    //check 1
    aclr = 1;
    aset = 1;
    for(i=0; i<5; i = i + 1)begin
        sclr = $random%2;
        sset = $random%2;
        en = $random%2;
        d = $random;
        load = $random%2;
        shiftin_tb = $random%2;
        q_expected = 0;
        @(negedge clk);
        if(q_dut != q_expected) begin
            $display("ERROR 1");
            $stop;
        end
    end

    //check 2
    aclr = 0;
    aset = 1;
    for(i=0; i<5; i = i + 1)begin
        sclr = $random%2;
        sset = $random%2;
        en = $random%2;
        d = $random;
        load = $random%2;
        shiftin_tb = $random%2;
        q_expected = 2;
        @(negedge clk);
        if(q_dut != q_expected) begin
            $display("ERROR 2");
            $stop;
        end
    end
```

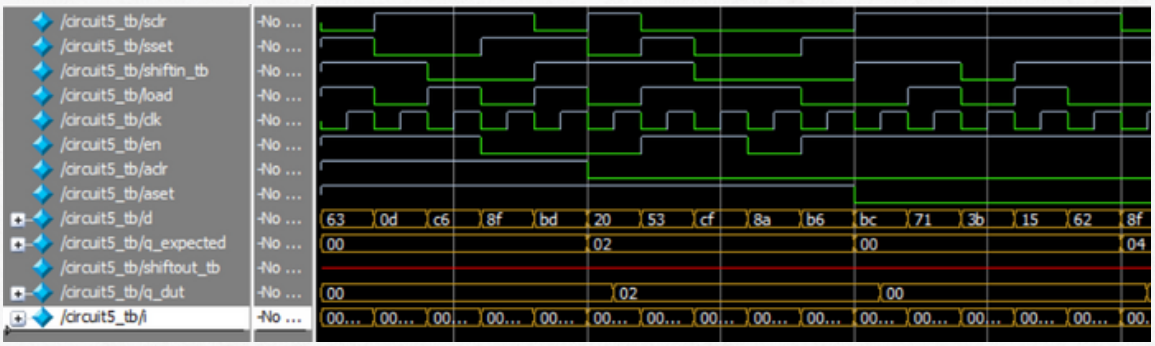
- **check 3 & 4**

```
//check 3
aclr = 0;
aset = 0;
sclr = 1;
sset = 1;
en = 1;
for(i=0; i<5; i = i + 1)begin
    d = $random;
    load = $random%2;
    shiftin_tb = $random%2;
    q_expected = 0;
    @(negedge clk);
    if(q_dut != q_expected) begin
        $display("ERROR 3");
        $stop;
    end
end

//check 4
aclr = 0;
aset = 0;
sclr = 0;
sset = 1;
en = 1;
for(i=0; i<5; i = i + 1)begin
    d = $random;
    load = $random%2;
    shiftin_tb = $random%2;
    q_expected = 4;
    @(negedge clk);
    if(q_dut != q_expected) begin
        $display("ERROR 4");
        $stop;
    end
end
```

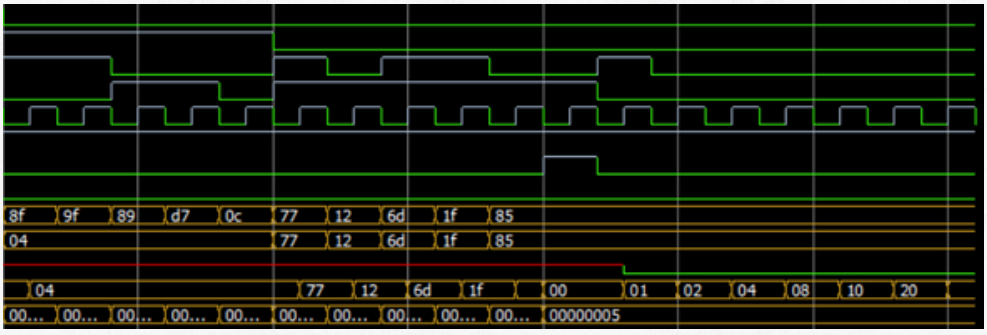
- **check 5 & 6**

```
    //check 5
    aclr = 0;
    aset = 0;
    sclr = 0;
    sset = 0;
    load = 1;
    en = 1;
    for(i=0; i<5; i = i + 1)begin
        d = $random;
        shiftin_tb = $random%2;
        q_expected = d;
        @(negedge clk);
        if(q_dut != q_expected) begin
            $display("ERROR 5");
            $stop;
        end
    end

    //check 6
    aclr = 1;
    @(negedge clk);
    aclr = 0;
    aset = 0;
    sclr = 0;
    sset = 0;
    load = 0;
    en = 1;
    shiftin_tb = 1;
    @(negedge clk);
    shiftin_tb = 0;
    repeat(6)
        @(negedge clk);
    $stop;

end

endmodule
```

- **output:**
  - **check 1, 2 and 3**



  - **check 4, 5 and 6**

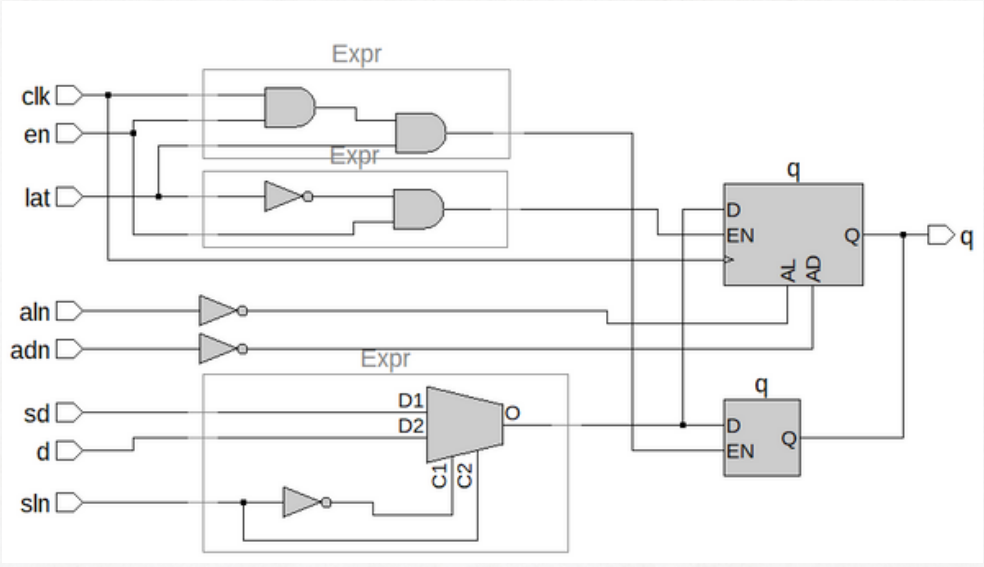


Circuit_6:

- **module:**

```
module circuit_6(input d, clk, en, aln, adn, sln, sd, lat, output reg q);

    always @(*) begin

        if (lat) begin
          if(clk)
            if(en)
                if(~sln)
                    q = sd;
                else
                    q = d;
        end
    end
    always @(posedge clk, negedge aln) begin
        if (~aln)
            q <= ~adn;
        else if(~lat) begin
            if(en) begin
                if(~sln)
                    q <= sd;
                else
                    q <= d;
            end
        end
    end

endmodule
```

- **schematic:**



- **testbench :**

```
module circuit6_tb();
    reg d, clk, en, aln, adn, sln, sd, lat;
    wire q_dut;

    circuit_6 dut(d, clk, en, aln, adn, sln, sd, lat, q_dut);

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

    initial begin
        aln = 0;
        @(negedge clk);
        aln = 1;
        lat = 0;
        repeat(10) begin
            d = $random;
            en = $random;
            adn = $random;
            sln =$random;
            sd = $random;
            @(negedge clk);
        end
        lat = 1;
        repeat(10) begin
            d = $random;
            en = $random;
            adn = $random;
            sln =$random;
            sd = $random;
            @(negedge clk);
        end
        $stop;
    end

endmodule
```

- **output:**