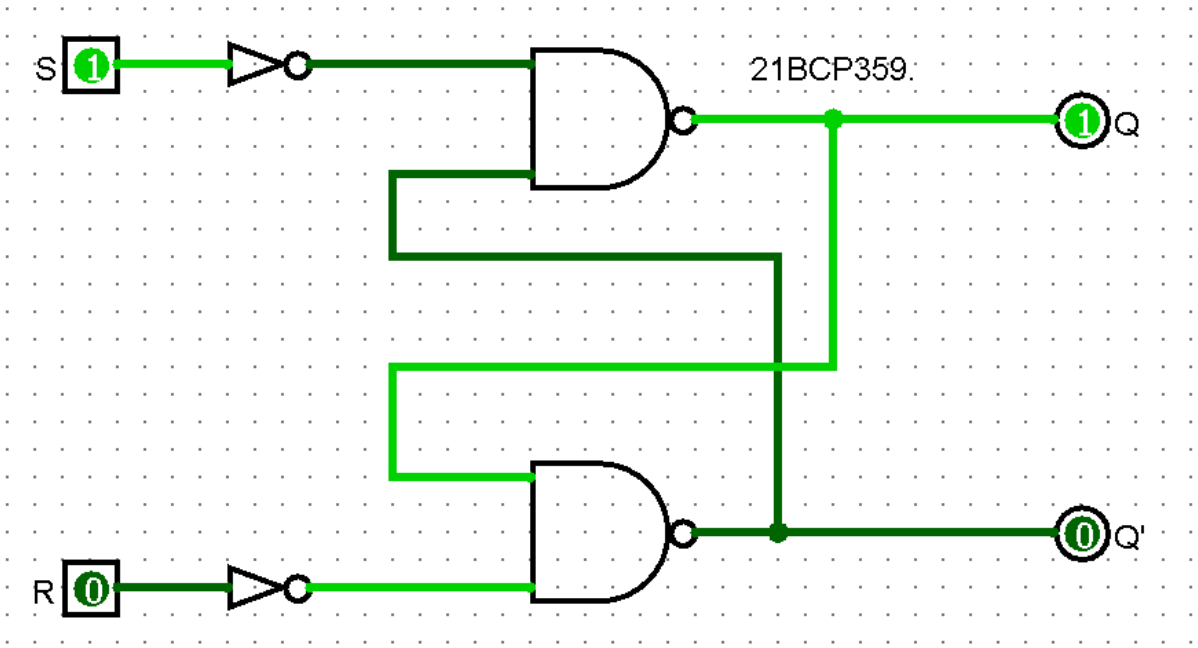
LAB 8: Sequential Circuits: Latch And SR Flip Flop

Question 1: Design an SR flip-flop using Logisim.



Question 2: Write a module for NOR gate and develop a structural Verilog code for the S-R latch using the NOR gate module. Validate the code via a suitable Testbench code.

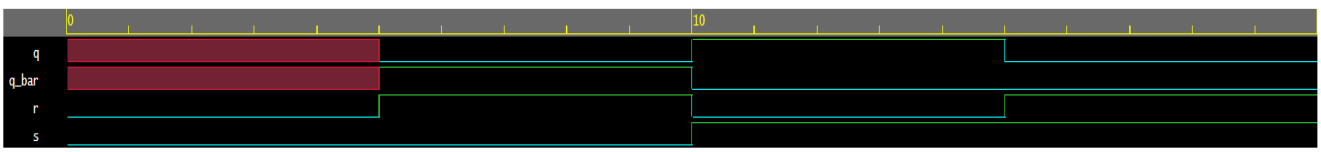
Expression: $Q_{n+1} = E.D + E'.Q_n$

testbench.sv

```
1  module nor_latch_tb;
2    reg s,r; //input
3    wire q,q_bar; //output
4    nor_latch x(.s(s),.r(r),.q(q),.q_bar(q_bar));
5    initial
6      begin
7        s<=1'b0; r<=1'b0; #5;
8        s<=1'b0; r<=1'b1; #5;
9        s<=1'b1; r<=1'b0; #5;
10       s<=1'b1; r<=1'b1; #5;
11       end
12   initial
13     begin
14       $dumpfile("dump.vcd");
15       $dumpvars(1);
16     end
17 endmodule
```

design.sv

```
1  // SR LATCH
2  module nor_latch(input s,r, output q,q_bar);
3    nor(q,r,q_bar);
4    nor(q_bar,q,s);
5  endmodule
```



Question 3: Write a Verilog module for NAND gate and utilize it to develop a structural Verilog module. Validate it using suitable Test bench.

Expression: $Q_n+1 = S + Q_nR'$

testbench.sv

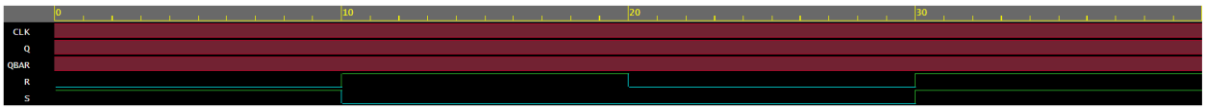
```
module SRflipfloptest;
  reg S, R, CLK;
  wire Q, QBAR;
  SRflipflop f1 (.s(S), .r(R), .clk(CLK), .q(Q), .qbar(QBAR));
  initial
    begin
      S = 1; R = 0; #10;
      S = 0; R = 1; #10;
      S = 0; R = 0; #10;
      S = 1; R = 1; #10;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

design.sv

```
module SRflipflop(q, qbar, s, r, clk);

input s, r, clk;
output q, qbar;

assign q = clk ? (s + ((~r) & q)) : q;
assign qbar = ~q;
endmodule
```



Q 4. Develop a similar behavioral code and test bench for S-R flip flop using if-else condition as per question 3.

testbench.sv

```
// 21BCP294 -Ayush Thakor
// SR -Flip flop behavioral code
module tb_srff_ifelse;
reg S,R, CLK;
wire Q, QBAR;
srff_ifelse dut(S,R,CLK,Q,QBAR);
initial
begin
S= 0; R= 0; CLK=0; #1;
$display ("CLK = %b, S= %b, R = %b , Q = %b , QBAR = %b", CLK,S,R, Q, QBAR);
S= 0; R= 1; CLK=1; #1;
$display ("CLK = %b, S= %b, R = %b , Q = %b , QBAR = %b", CLK,S,R, Q, QBAR);
S= 0; R= 0; CLK=0; #1;
$display ("CLK = %b, S= %b, R = %b , Q = %b , QBAR = %b", CLK,S,R, Q, QBAR);
S= 1; R= 0; CLK=1; #1;
$display ("CLK = %b, S= %b, R = %b , Q = %b , QBAR = %b", CLK,S,R, Q, QBAR);
S= 0; R= 0; CLK=0; #1;
$display ("CLK = %b, S= %b, R = %b , Q = %b , QBAR = %b", CLK,S,R, Q, QBAR);
S= 1; R= 1; CLK=1; #1;
$display ("CLK = %b, S= %b, R = %b , Q = %b , QBAR = %b", CLK,S,R, Q, QBAR);
$finish;
end
initial
begin
$dumpfile("dump.vcd");
$dumpvars (1);
end
```

design.sv

```
// 21BCP294 -Ayush Thakor
module srff_ifelse(S, R, CLK, Q, QBAR);
input S, R, CLK;
output reg Q, QBAR;
always@(CLK)
begin
if(S==1 & R==0)
begin
Q = 1;
QBAR = 0;
end
else if(S==0 & R==1)
begin
Q = 0;
QBAR = 1;
end
else if(S==0 & R==0)
begin
Q = Q;
QBAR = QBAR;
end
else if(S==1 & R==1)
begin
Q = 1;
QBAR = 1;
end
end
endmodule
```


```
CLK = 0, S= 0, R = 0 , Q = x , QBAR = x
CLK = 1, S= 0, R = 1 , Q = 0 , QBAR = 1
CLK = 0, S= 0, R = 0 , Q = 0 , QBAR = 1
CLK = 1, S= 1, R = 0 , Q = 1 , QBAR = 0
CLK = 0, S= 0, R = 0 , Q = 1 , QBAR = 0
CLK = 1, S= 1, R = 1 , Q = 1 , QBAR = 1
```

Question 5: Develop a similar behavioural code and test bench for S-R flip flop using if-else condition as per question 3.

testbench.sv

```
module SRflipfloptest;
  reg S, R, CLK;
  wire Q, QBAR;
  SRflipflop f1 (.s(S), .r(R), .clk(CLK), .q(Q), .qbar(QBAR));
  initial
    begin
      S = 1; R = 0; #10;
      S = 0; R = 1; #10;
      S = 0; R = 0; #10;
      S = 1; R = 1; #10;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

design.sv

```
module SRflipflop(s, r, clk, q , qbar);
  input clk, s, r;
  output reg q, qbar;
  always@(posedge clk)
    begin
      if(s == 1)
        begin
          q <= 1;
          qbar <= 0;
        end
      else if(r == 1)
        begin
          q <= 0;
          qbar <= 1;
        end
      else if(s == 0 & r == 0)
        begin
          q <= q;
          qbar = qbar;
        end
    end
endmodule
```