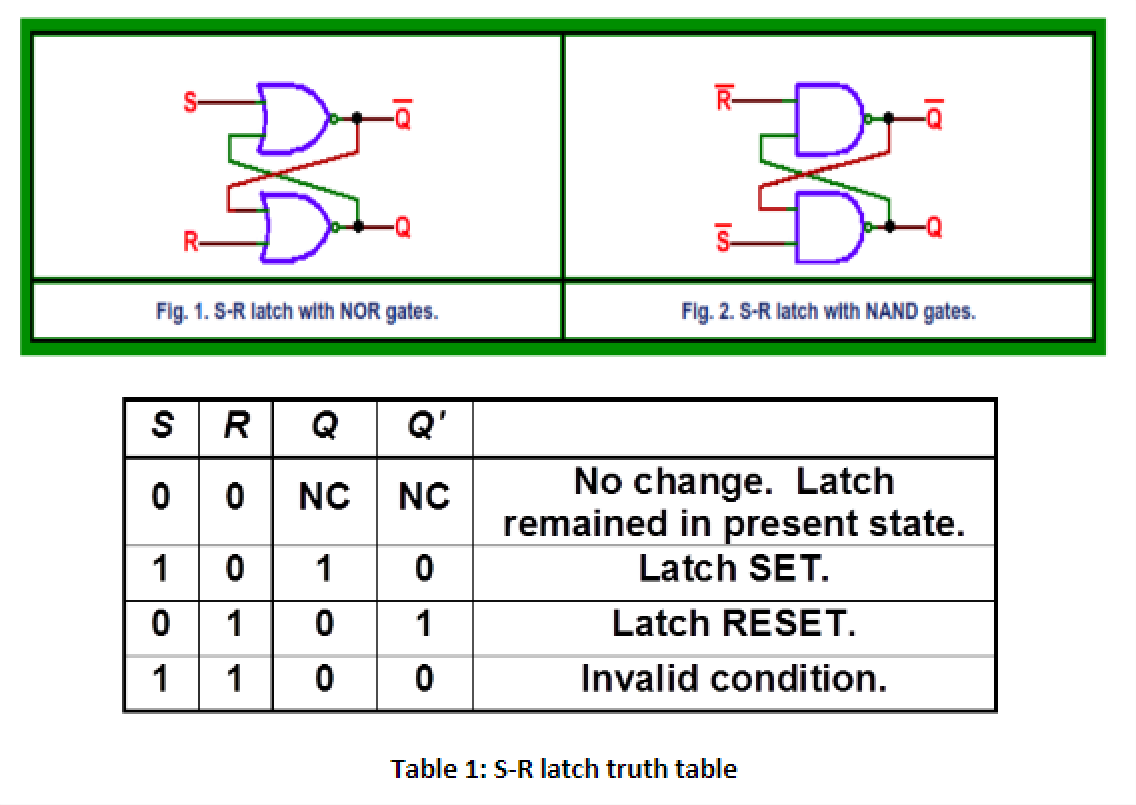
**Assignment-8**

**Name: Dhvani Patel**

**RollNo.: 21BCP116**

**Aim:** Sequential Circuits: Latch and flip flop (SR, JK)

Q 1. An SR latch (Set/Reset) is an asynchronous device: it works independently of control signals and relies only on the state of the S and R inputs. The symbol, the circuit using NOR gates, and the truth table are shown below.



Based on the above figure and truth table, complete following tasks.

1) Generate the Boolean expression for the S-R latch from the truth table given in Table-1.

2) Write a module for NOR gate and develop a structural verilog code for the S-R latch using the NOR gate module.

3) Validate the code via a suitable Testbench code.

**Design Code:**

module NOR(output Y, input A, B);

wire Yd;

or(Yd, A, B);

not(Y, Yd);

endmodule

module srff\_gate(q, qbar, s, r);

input s,r;

output q, qbar;

NOR N1 (q , qbar ,s);

NOR N2 (qbar , q , r);

endmodule

**Testbench Code:**

module tb\_srff\_gate;

reg s,r;

output q , qbar;

srff\_gate srff(q,qbar,s,r);

initial

begin

s=0 ; r=0; #1;

$display("%b %b",q,qbar);

s=0 ; r=1; #1;

$display("%b %b",q,qbar);

s=1 ; r=0; #1;

$display("%b %b",q,qbar);

s=1 ; r=1;

$display("%b %b",q,qbar);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

**Output:**

VCD info: dumpfile dump.vcd opened for output.

x x

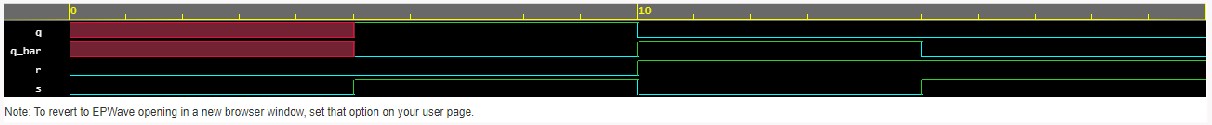
1 0

0 1

0 1

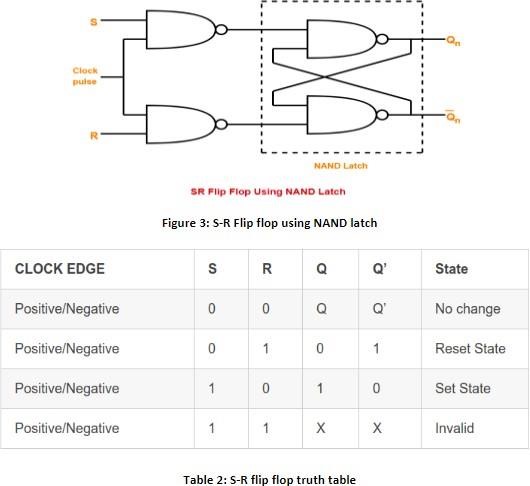
Finding VCD file...

**Waveform:**



Q 2. In S-R latch we do not use a clock. Now if we add an additional clock at input so that the S and R inputs are active only when the clock is high. When the clock goes low, the state of flip-flop is latched and cannot change until the clock goes high again. This clocked addition in S-R latch is also called the SR flip flop. Use the below given figure and truth table for S-R flip flop.

We can add a clock in put in NAND latch in Figure 2, and can convert in S-R flip flop using NAND gates as: (similar results can be achieved via NOR Latch)



1) Generate the Boolean expression for the S-R flipflop using K-map and truth table given in Table 2.

2) Write a verilog module for NAND gate and utlize it to develop a structural verilog module for S-R flip flop as per figure 3.

3) Validate it using suitable Test bench.

**Design Code:**

module sr\_ff(input s,r,clk, output q,q\_bar);

wire nand1\_out, nand2\_out;

nand n1(nand1\_out,s,clk);

nand n2(nand2\_out,r,clk);

nand n3(q,nand1\_out,q\_bar);

nand n4(q\_bar,nand2\_out,q);

endmodule

**Testbench Code:**

module sr\_ff\_tb;

reg s,r,clk;

wire q,q\_bar;

sr\_ff al(s,r,clk,q,q\_bar);

initial

begin

s<=1'b0;r<=1'b0;clk<=1'b0;#5;

$display("s = %b, r = %b, clk = %b, q = %b, q\_bar = %b",s,r,clk,q,q\_bar);

s<=1'b0;r<=1'b1;clk<=1'b1;#5;

$display("s = %b, r = %b, clk = %b, q = %b, q\_bar = %b",s,r,clk,q,q\_bar);

s<=1'b0;r<=1'b0;clk<=1'b1;#5;

$display("s = %b, r = %b, clk = %b, q = %b, q\_bar = %b",s,r,clk,q,q\_bar);

s<=1'b1;r<=1'b0;clk<=1'b1;#5;

$display("s = %b, r = %b, clk = %b, q = %b, q\_bar = %b",s,r,clk,q,q\_bar);

s<=1'b1;r<=1'b1;clk<=1'b1;#5;

$display("s = %b, r = %b, clk = %b, q = %b, q\_bar = %b",s,r,clk,q,q\_bar);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

**Output:**

VCD info: dumpfile dump.vcd opened for output.

s = 0, r = 0, clk = 0, q = x, q\_bar = x

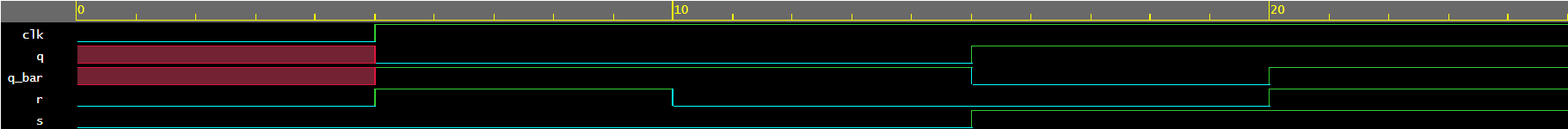
s = 0, r = 1, clk = 1, q = 0, q\_bar = 1

s = 0, r = 0, clk = 1, q = 0, q\_bar = 1

s = 1, r = 0, clk = 1, q = 1, q\_bar = 0

s = 1, r = 1, clk = 1, q = 1, q\_bar = 1

**Waveform:**



Q 3. We can also develop a behavioral modeling based verilog code for the S-R latch. Here we can use the if-else logic to assign values for output based on the input conditions.

For example, the condition S = 1 or H, R = 0 or L then Q = 1 and Q’ = 0 can be expressed under a begin

block using if condition as:

If (S == 1 & R == 0)

begin

Q <= 1;

Q\_bar <= 0;

end

Similar code can be developed for rest of the conditions.

1) Develop a behavioral verilog code using if-else slatemets for S-R latch.

2) Varify it with a suitable testbench code.

**Design Code:**

module sr\_latch(s,r, q, qbar);

input s,r;

output reg q, qbar;

always@(s or r)

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 == 1 & r == 0)

begin

q = 1;

qbar = 0;

end

else if(s == 0 & r == 0)

begin

q <= q;

qbar <= qbar;

end

end

endmodule

**Testbench Code:**

module tb\_sr\_latch;

reg s,r;

wire q,q\_bar;

sr\_latch al(.s(s),.r(r),.q(q),.qbar(qbar));

initial

begin

s<=1'b0;r<=1'b0;

$display("s = %b, r = %b, q = %b, qbar = %b",s,r,q,qbar);

#5;

s<=1'b1;r<=1'b0;

$display("s = %b, r = %b, q = %b, qbar = %b",s,r,q,qbar);

#5;

s<=1'b0;r<=1'b1;

$display("s = %b, r = %b, q = %b, qbar = %b",s,r,q,qbar);

#5;

s<=1'b1;r<=1'b1;

$display("s = %b, r = %b, q = %b, qbar = %b",s,r,q,qbar);

#5;

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

**Output:**

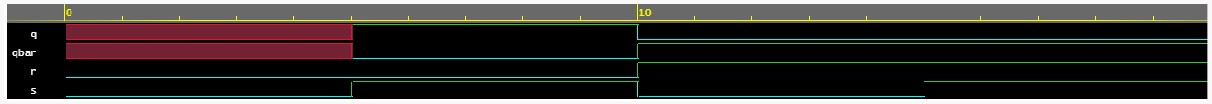
VCD info: dumpfile dump.vcd opened for output.

s = 0, r = 0, q = x, qbar = x

s = 1, r = 0, q = 1, qbar = 0

s = 0, r = 1, q = 0, qbar = 1

**Waveform:**



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

**Design Code:**

module srff\_behave(s, r, clk, q, qbar);

input s,r,clk;

output reg q, qbar;

always@(posedge 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 == 1 & r == 0)

begin

q = 1;

qbar = 0;

end

else if(s == 0 & r == 0)

begin

q <= q;

qbar <= qbar;

end

end

endmodule

**Testbench Code:**

module tb\_srff\_behave;

reg s,r,clk;

wire q,q\_bar;

srff\_behave al(s,r,clk,q,q\_bar);

initial

begin

s<=1'b0;r<=1'b0;clk<=1'b0;#5;

$display("s = %b, r = %b, clk = %b, q = %b, q\_bar = %b",s,r,clk,q,q\_bar);

s<=1'b0;r<=1'b1;clk<=1'b1;#5;

$display("s = %b, r = %b, clk = %b, q = %b, q\_bar = %b",s,r,clk,q,q\_bar);

s<=1'b0;r<=1'b0;clk<=1'b0;#5;

$display("s = %b, r = %b, clk = %b, q = %b, q\_bar = %b",s,r,clk,q,q\_bar);

s<=1'b1;r<=1'b0;clk<=1'b1;#5;

$display("s = %b, r = %b, clk = %b, q = %b, q\_bar = %b",s,r,clk,q,q\_bar);

s<=1'b1;r<=1'b1;clk<=1'b1;#5;

$display("s = %b, r = %b, clk = %b, q = %b, q\_bar = %b",s,r,clk,q,q\_bar);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

**Output:**

s = 0, r = 0, clk = 0, q = x, q\_bar = x

s = 0, r = 1, clk = 1, q = 0, q\_bar = 1

s = 0, r = 0, clk = 0, q = 0, q\_bar = 1

s = 1, r = 0, clk = 1, q = 1, q\_bar = 0

s = 1, r = 1, clk = 1, q = 1, q\_bar = 0

**Waveform:**

