Date:12-10-2022

Assignment-8

Aim: To implement sequential circuits(particularly SR latch and SR flipflop).

List of Questions:

Q1. 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.

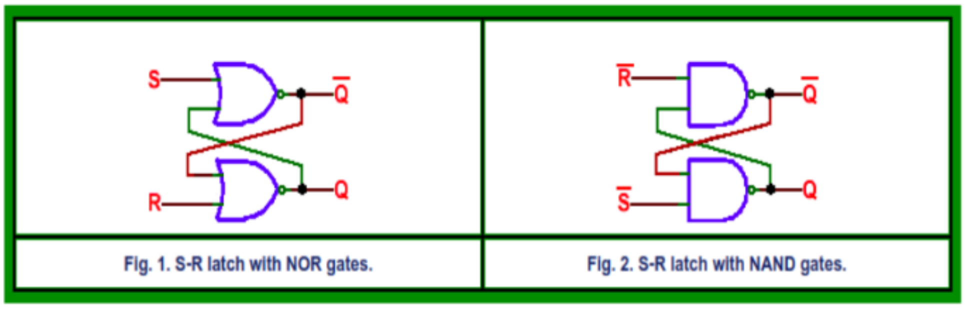
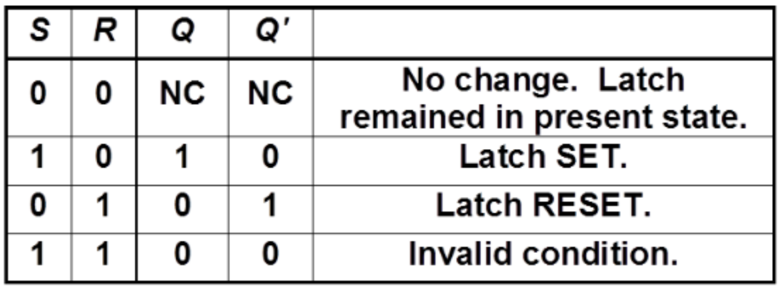


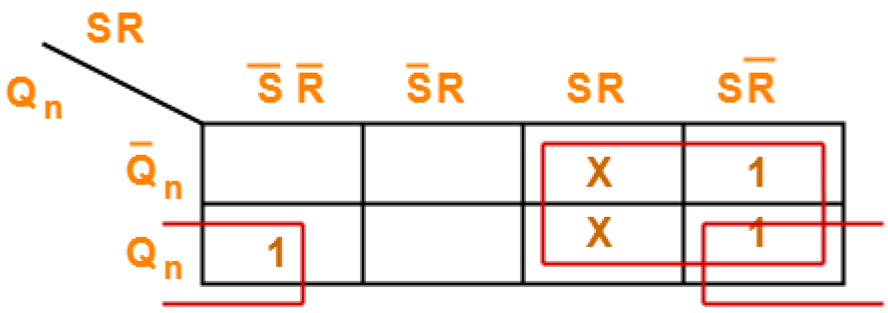
Fig. 1. S-R latch with NOR gates.

Fig. 2. S-R latch with NAND gates.



S	*R*	*Q*	*Q'*	
0	0	NC	NC	No change. Latch remained in present state.
1	0	1	0	Latch SET.
0	1	0	1	Latch RESET.
1	1	0	0	Invalid condition.

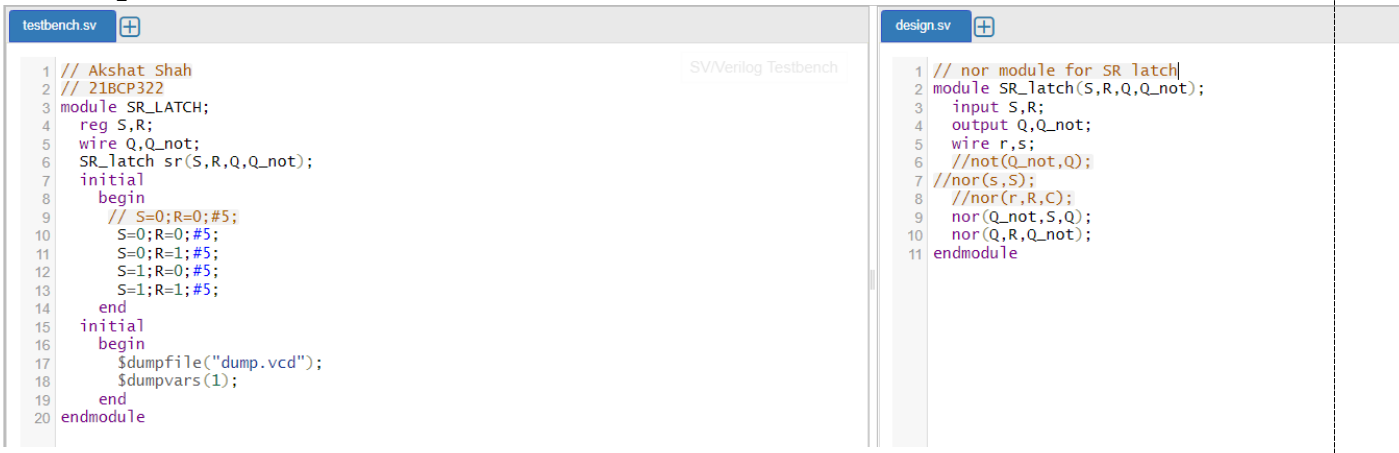
1. Generate the Boolean expression for the S-R latch from the truth table given in Table.



Finally We get

$$Q_{n+1} = S + Q_nR'$$

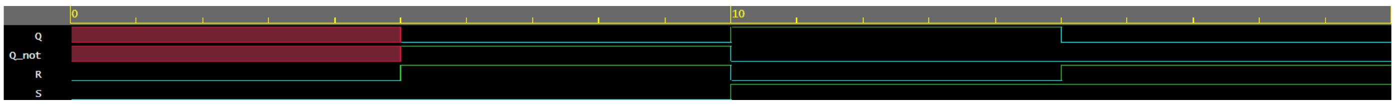
2. Write a module for NOR gate and develop a structural verilog code for the S-R latch using the NOR gate module.
   Verilog code:



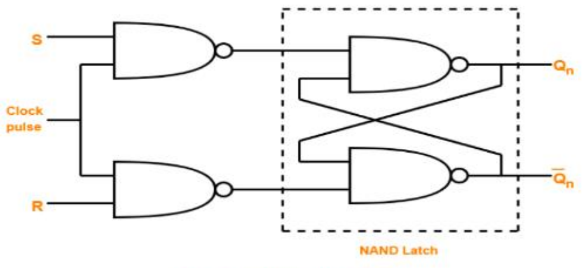
```
// Akshat Shah
// 21BCP322
module SR_LATCH;
  reg S,R;
  wire Q,Q_not;
  SR_latch sr(S,R,Q,Q_not);
  initial
    begin
     // S=0;R=0;#5;
      S=0;R=0;#5;
      S=0;R=1;#5;
      S=1;R=0;#5;
      S=1;R=1;#5;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

```
// nor module for SR latch
module SR_latch(S,R,Q,Q_not);
  input S,R;
  output Q,Q_not;
  wire r,s;
  //not(Q_not,Q);
//nor(s,S);
  //nor(r,R,C);
  nor(Q_not,S,Q);
  nor(Q,R,Q_not);
endmodule
```

Waveform:

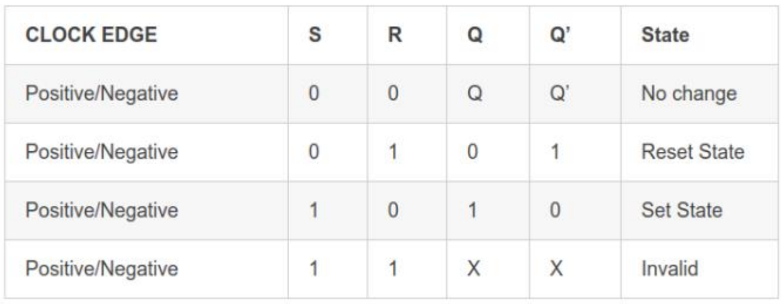


Q2. In S-R latch we do not use a clock. Now if we add an additional clock at input so that the S and R inputs are active only when the clock is high. When the clock goes low, the state of flip-flop is latched and cannot change until the clock goes high again. This clocked addition in S-R latch is also called the S-R 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 , and can convert in S-R flip flop using NAND gates as: (similar results can be achieved via NOR Latch).



SR Flip Flop Using NAND Latch

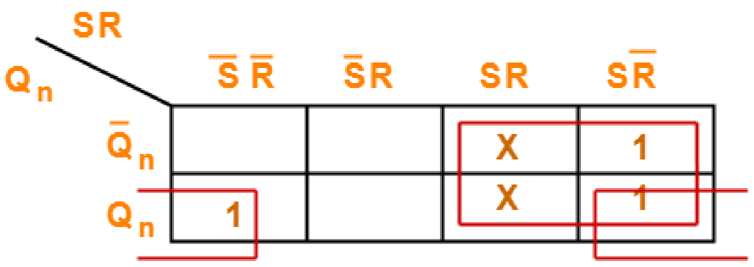
Figure 3: S-R Flip flop using NAND latch



CLOCK EDGE	S	R	Q	Q'	State
Positive/Negative	0	0	Q	Q'	No change
Positive/Negative	0	1	0	1	Reset State
Positive/Negative	1	0	1	0	Set State
Positive/Negative	1	1	X	X	Invalid

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

From the truth table, K-map made is as under:



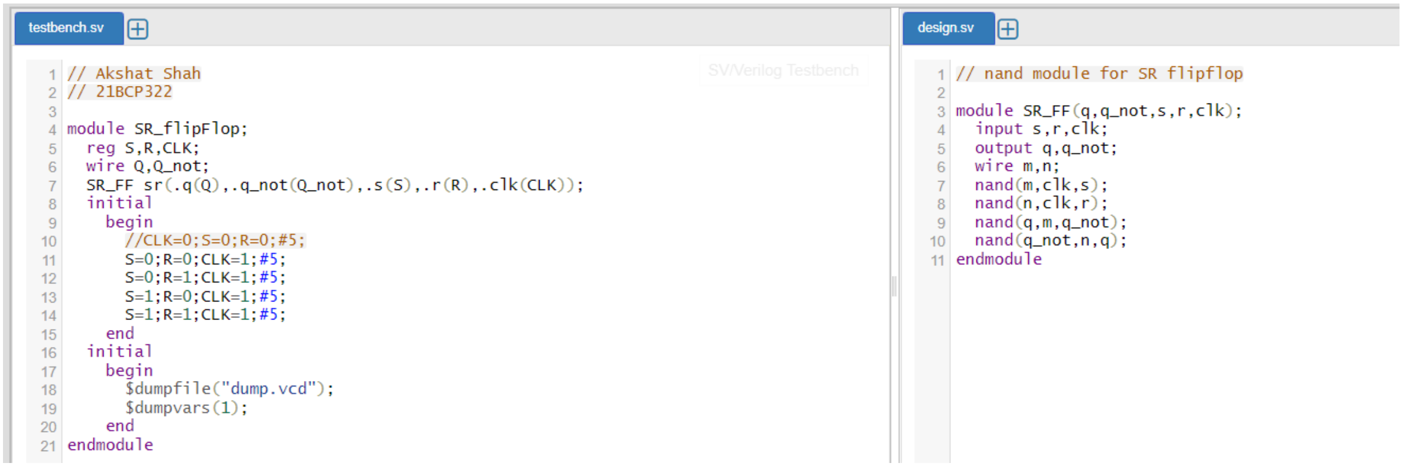
We get

$$Q_{n+1} = (SR + SR')(Q_n + Q'_n) + Q_n(S'R' + SR')$$

$$\mathbf{Q_{n+1} = S + Q_nR'}$$

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.

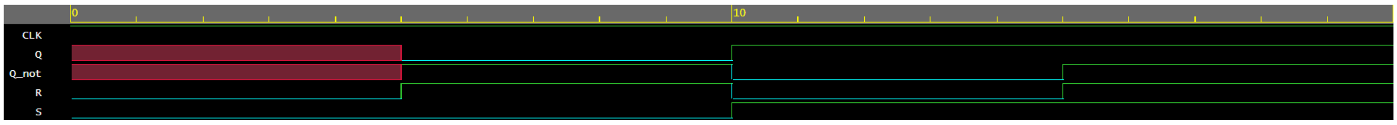
Verilog Code:



```verilog
// Akshat Shah
// 21BCP322

module SR_flipFlop;
  reg S,R,CLK;
  wire Q,Q_not;
  SR_FF sr(.q(Q),.q_not(Q_not),.s(S),.r(R),.clk(CLK));
  initial
    begin
      //CLK=0;S=0;R=0;#5;
      S=0;R=0;CLK=1;#5;
      S=0;R=1;CLK=1;#5;
      S=1;R=0;CLK=1;#5;
      S=1;R=1;CLK=1;#5;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

```verilog
// nand module for SR flipflop

module SR_FF(q,q_not,s,r,clk);
  input s,r,clk;
  output q,q_not;
  wire m,n;
  nand(m,clk,s);
  nand(n,clk,r);
  nand(q,m,q_not);
  nand(q_not,n,q);
endmodule
```

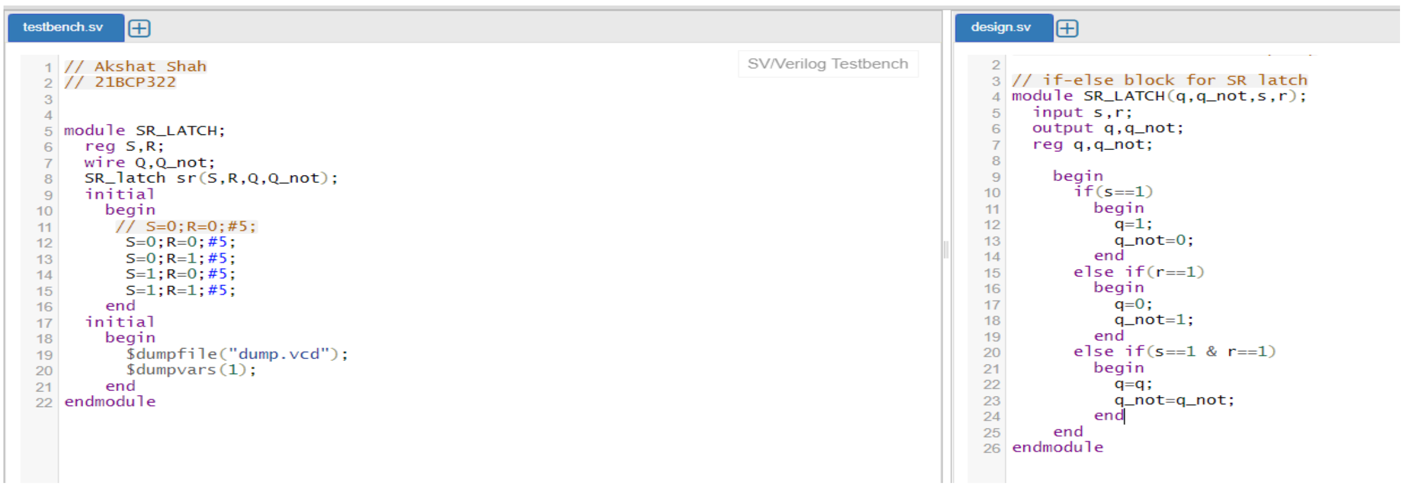
Waveform:



Q3. 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.

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

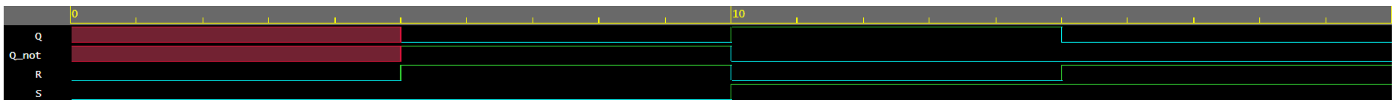
Verilog code:



```
// Akshat Shah
// 21BCP322


module SR_LATCH;
  reg S,R;
  wire Q,Q_not;
  SR_latch sr(S,R,Q,Q_not);
  initial
    begin
     // S=0;R=0;#5;
      S=0;R=0;#5;
      S=0;R=1;#5;
      S=1;R=0;#5;
      S=1;R=1;#5;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

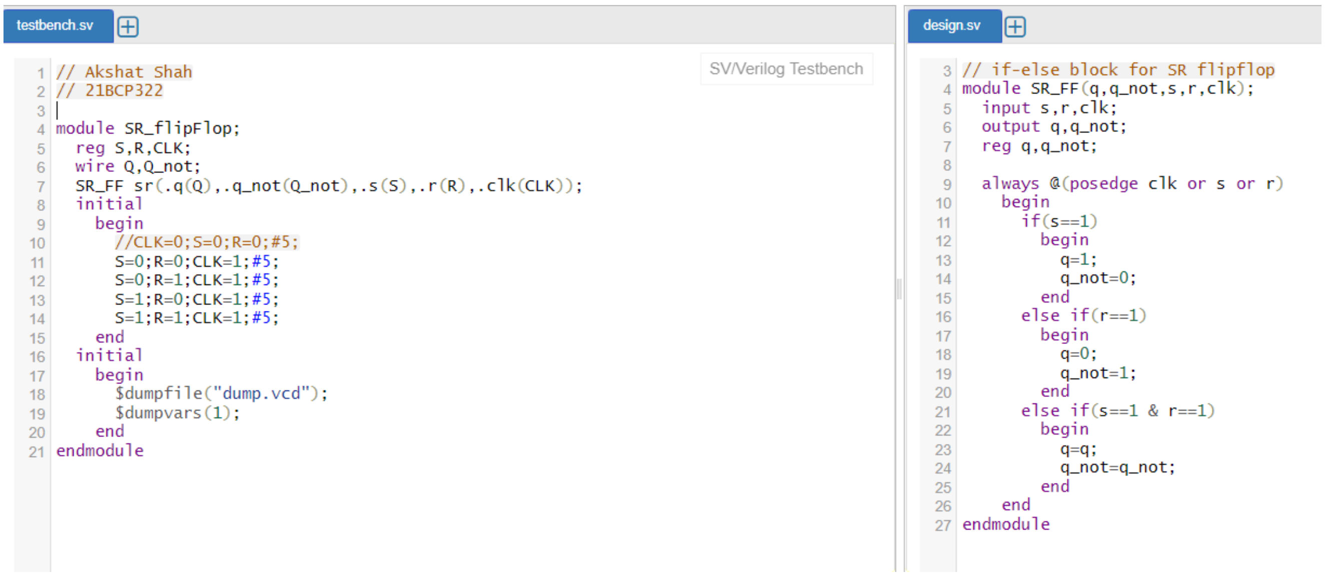
```

// if-else block for SR latch
module SR_LATCH(q,q_not,s,r);
  input s,r;
  output q,q_not;
  reg q,q_not;

    begin
      if(s==1)
        begin
          q=1;
          q_not=0;
        end
      else if(r==1)
        begin
          q=0;
          q_not=1;
        end
      else if(s==1 & r==1)
        begin
          q=q;
          q_not=q_not;
        end
    end
endmodule
```

Waveform:



Q4. Develop a similar behavioral code and test bench for S-R flip flop using if-else condition as per question 3.

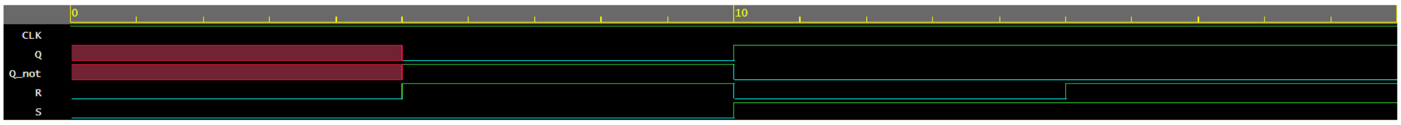
Verilog code:



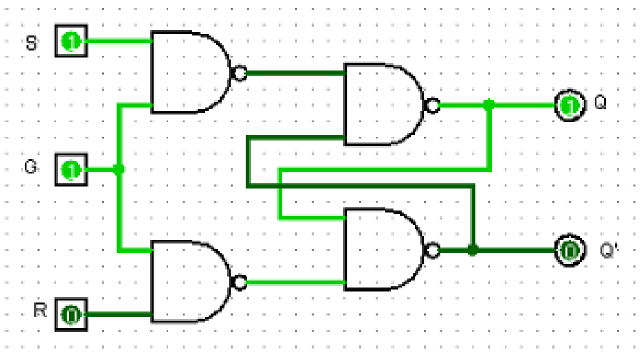
```
// Akshat Shah
// 21BCP322

module SR_flipFlop;
  reg S,R,CLK;
  wire Q,Q_not;
  SR_FF sr(.q(Q),.q_not(Q_not),.s(S),.r(R),.clk(CLK));
  initial
    begin
      //CLK=0;S=0;R=0;#5;
      S=0;R=0;CLK=1;#5;
      S=0;R=1;CLK=1;#5;
      S=1;R=0;CLK=1;#5;
      S=1;R=1;CLK=1;#5;
    end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

```
// if-else block for SR flipflop
module SR_FF(q,q_not,s,r,clk);
  input s,r,clk;
  output q,q_not;
  reg q,q_not;

  always @(posedge clk or s or r)
    begin
      if(s==1)
        begin
          q=1;
          q_not=0;
        end
      else if(r==1)
        begin
          q=0;
          q_not=1;
        end
      else if(s==1 & r==1)
        begin
          q=q;
          q_not=q_not;
        end
    end
endmodule
```

Waveform:



S-R flipflop using Logisim:


S
1
G
1
R
0
1
Q
0
Q'