

به نام خدا



UNIVERSITY OF TEHRAN  
Electrical and Computer Engineering Department

# Computer Assignment 4

Digital Systems I - ECE 894

Deadline : 12 Khordad 1400

**Erfan Panahi**

**810198369**

Spring 1400

# List of Problems:

## Computer Assignment 4

Problem 1.	Page 2
Problem 2.	Page 4
Problem 3.	Page 6
Problem 4.	Page 8
Problem 5.	Page 9
Problem 6.	Page 10
Problem 7.	Page 12
Problem 8.	Page 14
Problem 9.	Page 16
Problem 10.	Page 18

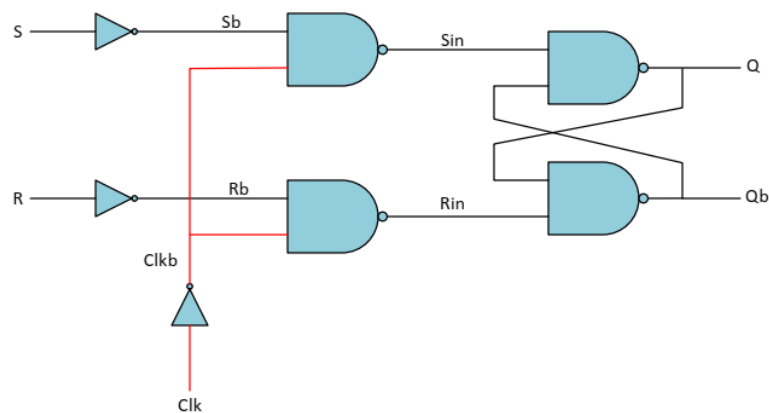
## Problem 1. File: Problem1.v

### A. Truth table:

Clk	S	R	Q+
0	1	1	Q
0	1	0	0
0	0	1	1
0	0	0	-
1	-	-	Q

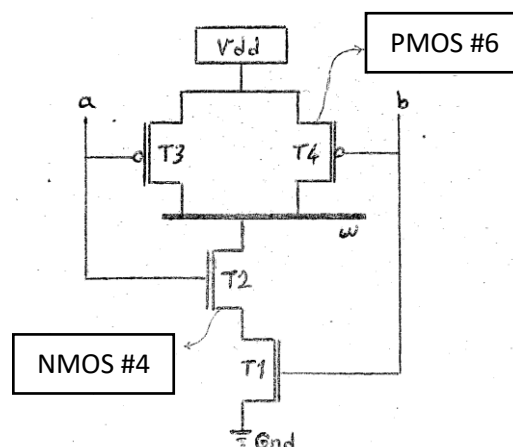
### B. Circuit diagram:

In following figure we show the circuit of a clocked SR-latch using NAND gates:



We have used NAND gates in this structure. So, we show the transistor level structure of this gates to calculate the worst case of a NAND.

#### NAND gate structure:



### c. Worst-cases:

#### NAND gate :

Worst case **To1 : 8NS** ( $ab = 11 \rightarrow 10$ )

Worst case **To0 : 8NS** ( $ab = 10 \rightarrow 11$ )

Using 'nand' statement:

```
nand #8 (w,a,b) ;
```

#### NOT gate :

Worst case **To1 : 6NS**

Worst case **To0 : 6NS**

Using 'not' statement:

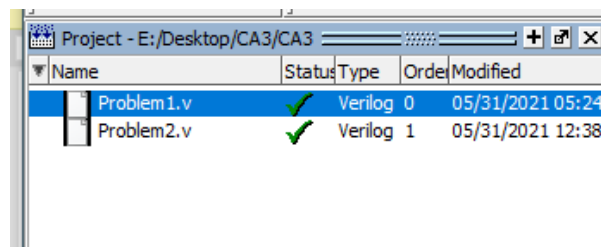
```
not #6 (ab,a) ;
```

### D. Verilog Description:

#### File: Problem1.v

```
`timescale 1ns/1ns
module CSRLatch(input S,R,Clk,output Q) ;
    wire Sb,Rb,Clkb,Qb,Rin,Sin;
    not #6 G1(Sb,S) ;
    not #6 G2(Rb,R) ;
    not #6 G3(Clk,Clkb) ;
    nand #8 G4(Rin,Rb,Clkb) ;
    nand #8 G5(Sin,Sb,Clkb) ;
    nand #8 G6(Qb,Rin,Q) ;
    nand #8 G7(Q,Sin,Qb) ;
endmodule
```

### E. Image of project and compiling:



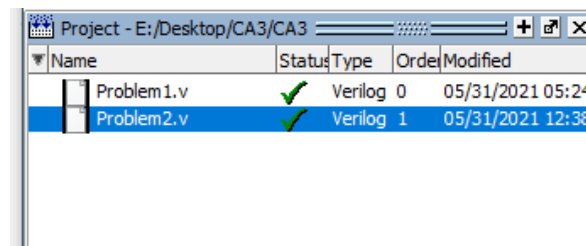
## Problem 2. File: Problem2.v

### A. Verilog Description:

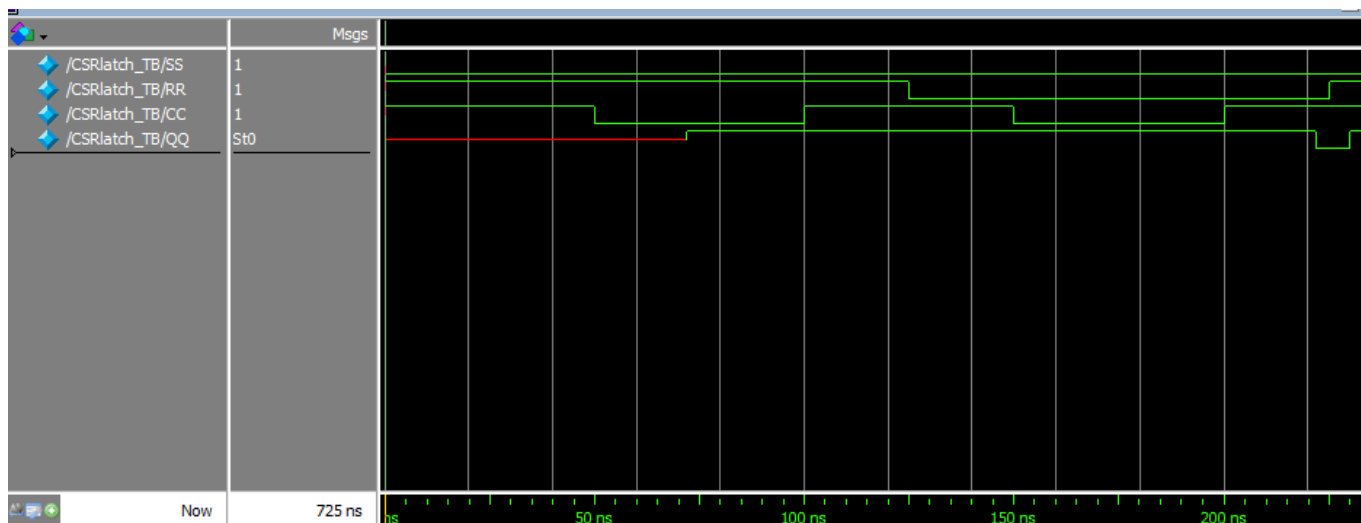
File: File: Problem2.v

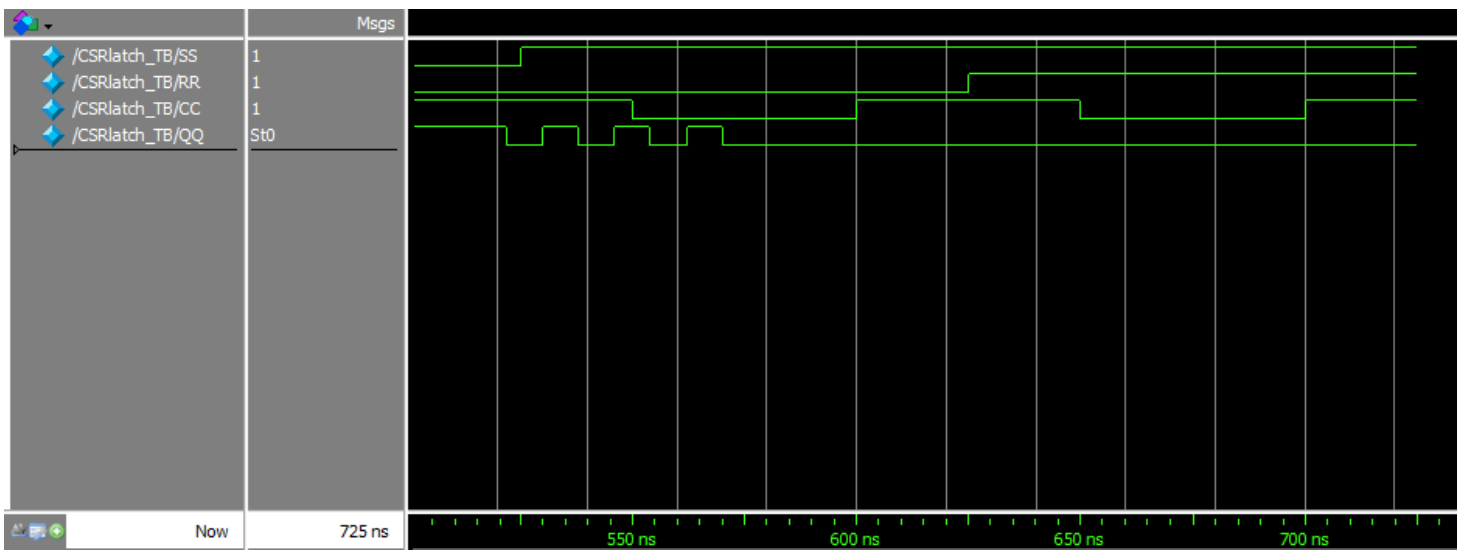
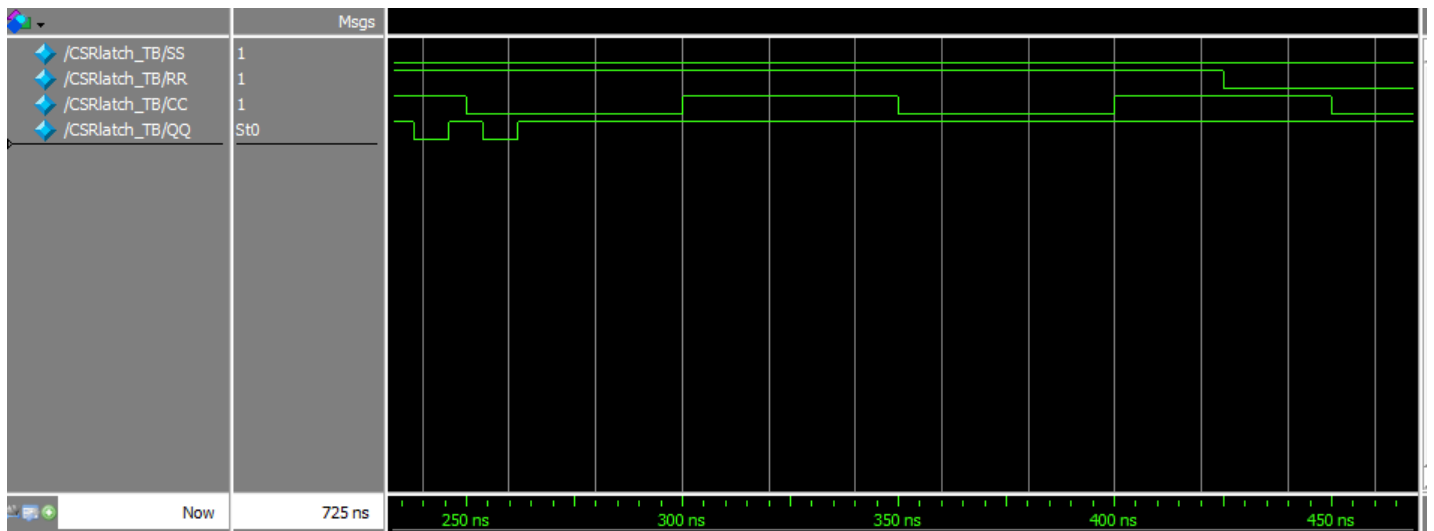
```
`timescale 1ns/1ns
module CSRLatch_TB();
    reg SS=0,RR=1,CC=1;
    wire QQ;
    CSRLatch UUT(SS,RR,CC,QQ);
    always #50 CC<=~CC;
    initial begin
        #125 RR=0;
        #100 RR=1;
        #100 SS=0;
        #100 RR=0;
        #100 SS=1;
        #100 RR=1;
        #100 $stop;
    end
endmodule
```

### B. Image of project and compiling:



### c. Waveform:





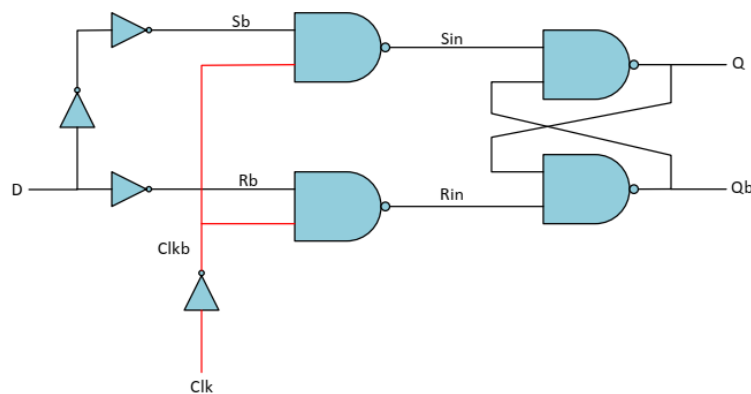
## Problem 3. File: Problem3.v

### A. Truth table:

Clk	D	Q+
0	1	1
0	0	0
1	-	Q

### B. Circuit diagram:

In following figure we show the circuit of a clocked SR-latch using NAND gates:



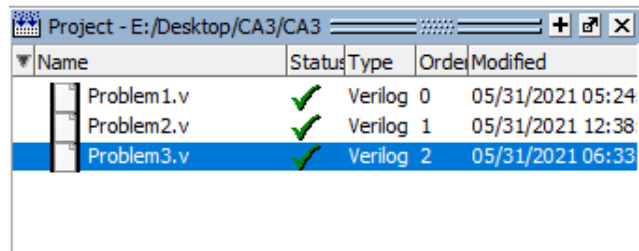
### c. Verilog Description:

File: Problem3.v

```
`timescale 1ns/1ns
module CDlatch(input D,Clk,output Q);
    wire Dbar;
    not #6 G1(Dbar,D);
    CSRLatch SRL(Dbar,D,Clk,Q);
endmodule

module CDlatch_TB();
    reg DD=0,CC=1;
    wire QQ;
    CDlatch UUT(DD,CC,QQ);
    always #50 CC<=~CC;
    initial begin
        #100 DD=1;
        #100 DD=0;
        #200 $stop;
    end
endmodule
```

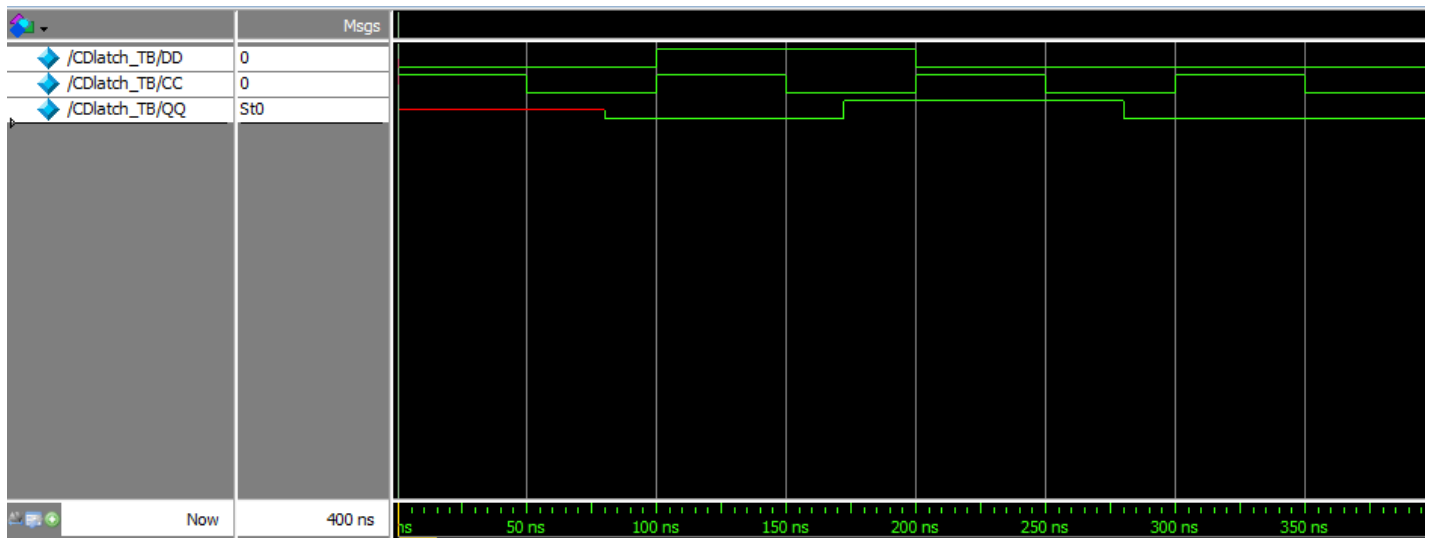
#### D. Image of project and compiling:



The screenshot shows a window titled "Project - E:/Desktop/CA3/CA3". It contains a table with the following data:

Name	Status	Type	Order	Modified
Problem1.v	✓	Verilog	0	05/31/2021 05:24
Problem2.v	✓	Verilog	1	05/31/2021 12:38
Problem3.v	✓	Verilog	2	05/31/2021 06:33

#### E. Waveform:

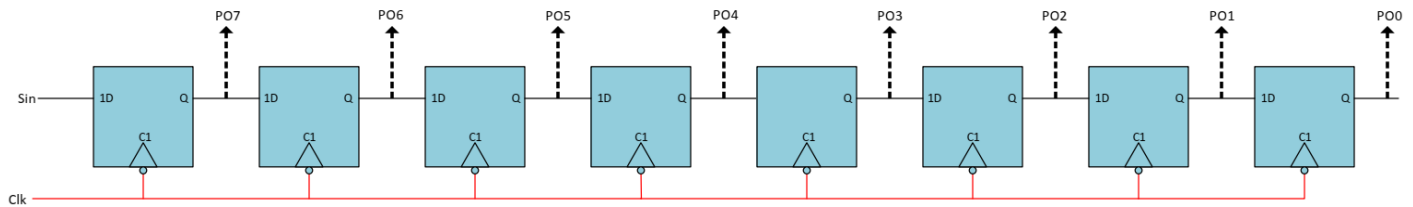




## Problem 4. Files: Problem4.v

### A. Circuit diagram:

In following figure we show the design of an 8-bit shift register: (Using clocked D-latch built in Problem 3.)



### B. Verilog Description:

File: Problem4.v

```
`timescale 1ns/1ns
module DlatchShftReg8bit(input Clk,Si,output [7:0]Po);
    wire [7:0]A;
    assign A = Po;
    CDlatch D1(Si,Clk,Po[7]);
    CDlatch D2(A[7],Clk,Po[6]);
    CDlatch D3(A[6],Clk,Po[5]);
    CDlatch D4(A[5],Clk,Po[4]);
    CDlatch D5(A[4],Clk,Po[3]);
    CDlatch D6(A[3],Clk,Po[2]);
    CDlatch D7(A[2],Clk,Po[1]);
    CDlatch D8(A[1],Clk,Po[0]);
endmodule
```

### c. Image of project and compiling:

Project - E:/Desktop/CA3/CA3				
Name	Status	Type	Order	Modified
Problem5.v	✓	Verilog	4	05/31/2021 08:11
Problem4.v	✓	Verilog	3	05/31/2021 07:41
Problem3.v	✓	Verilog	2	05/31/2021 06:47
Problem1.v	✓	Verilog	0	05/31/2021 05:24
Problem2.v	✓	Verilog	1	05/31/2021 07:47

## Problem 5. Files: Problem5.v

### A. System Verilog Description:

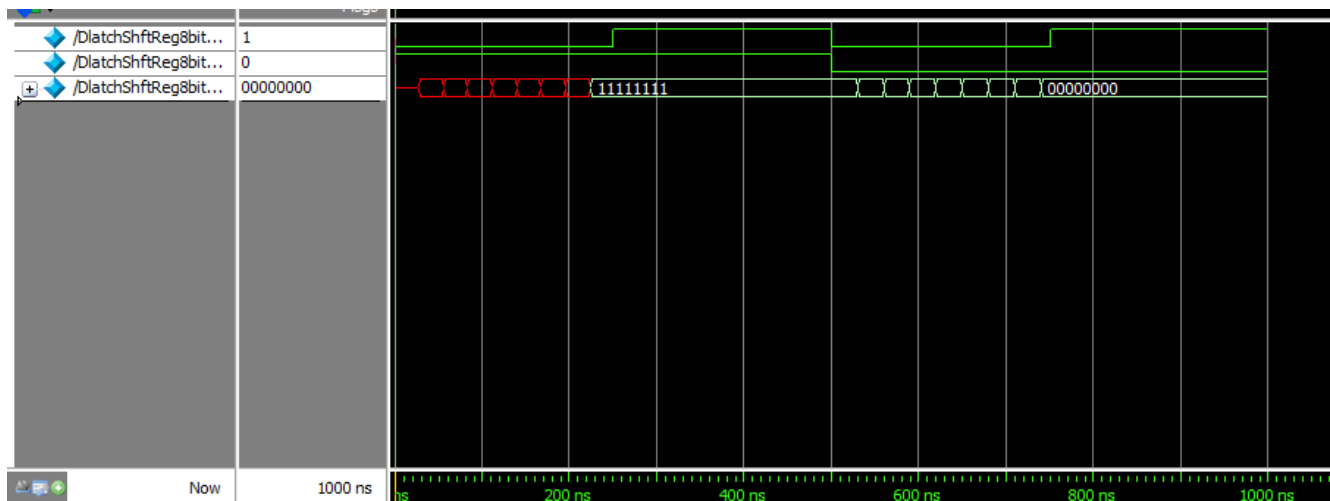
File: Problem5.v

```
`timescale 1ns/1ns
module DlatchShftReg8bit_TB();
    reg CC = 0, Si = 1;
    wire [7:0] PP;
    DlatchShftReg8bit UUT(CC, Si, PP);
    always #250 CC<=~CC;
    initial begin
        #500 Si=0;
        #500 $stop;
    end
endmodule
```

### B. Image of project and compiling:

Project - E:/Desktop/CA3/CA3				
Name	Status	Type	Order	Modified
Problem5.v	✓	Verilog	4	05/31/2021 08:11
Problem4.v	✓	Verilog	3	05/31/2021 07:41
Problem3.v	✓	Verilog	2	05/31/2021 06:47
Problem1.v	✓	Verilog	0	05/31/2021 05:24
Problem2.v	✓	Verilog	1	05/31/2021 07:47

### C. Waveform:



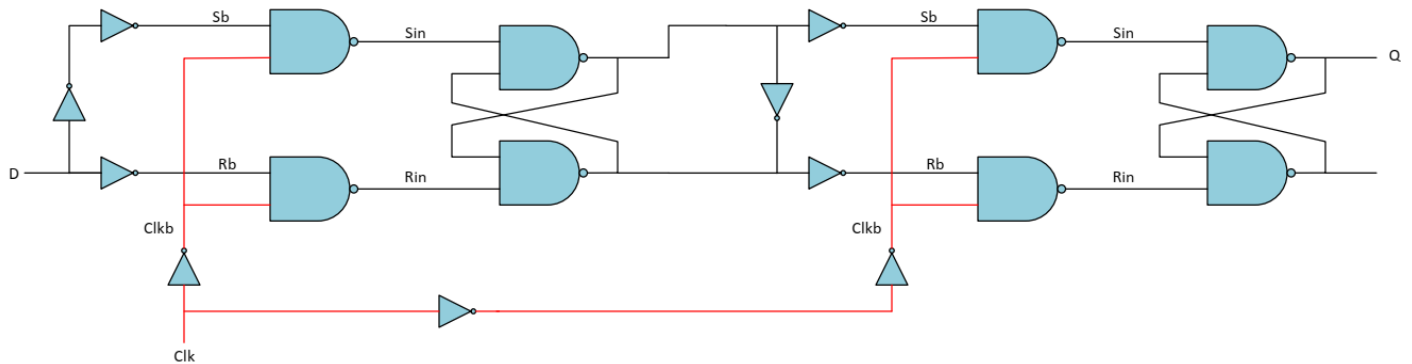
### D. Why-not this circuit works as expected?

As we see in the wave form if the clock duration was more than the delay of a D-latch, every clock would shift 2 and more bits to right, that is bad for a shift-register.

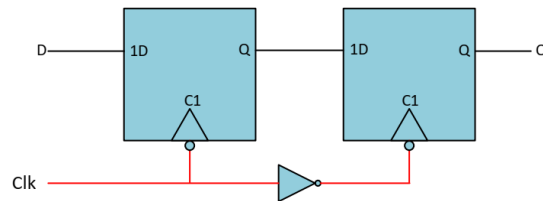
## Problem 6. File: Problem6.v

### A. Circuit diagram:

In following figures we show the diagram of a master-slave D-type flip-flop:



master-slave D-type flip-flop using two D-latches:



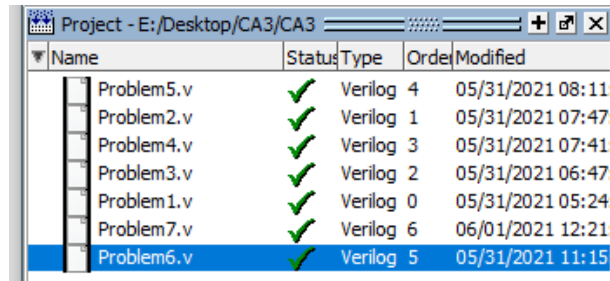
### B. Verilog Description:

File: Problem6.v

```
`timescale 1ns/1ns
module msDFF(input D,Clk,output Q);
    wire DQ,bClk;
    not #6 G1(bClk,Clk);
    CDlatch D1(D,Clk,DQ);
    CDlatch D2(DQ,bClk,Q);
endmodule

module msDFF_TB();
    reg DD=0,CC=1;
    wire QQ;
    msDFF UUT(DD,CC,QQ);
    always #100 CC<=~CC;
    initial begin
        #200 DD=1;
        #200 DD=0;
        #300 $stop;
    end
endmodule
```

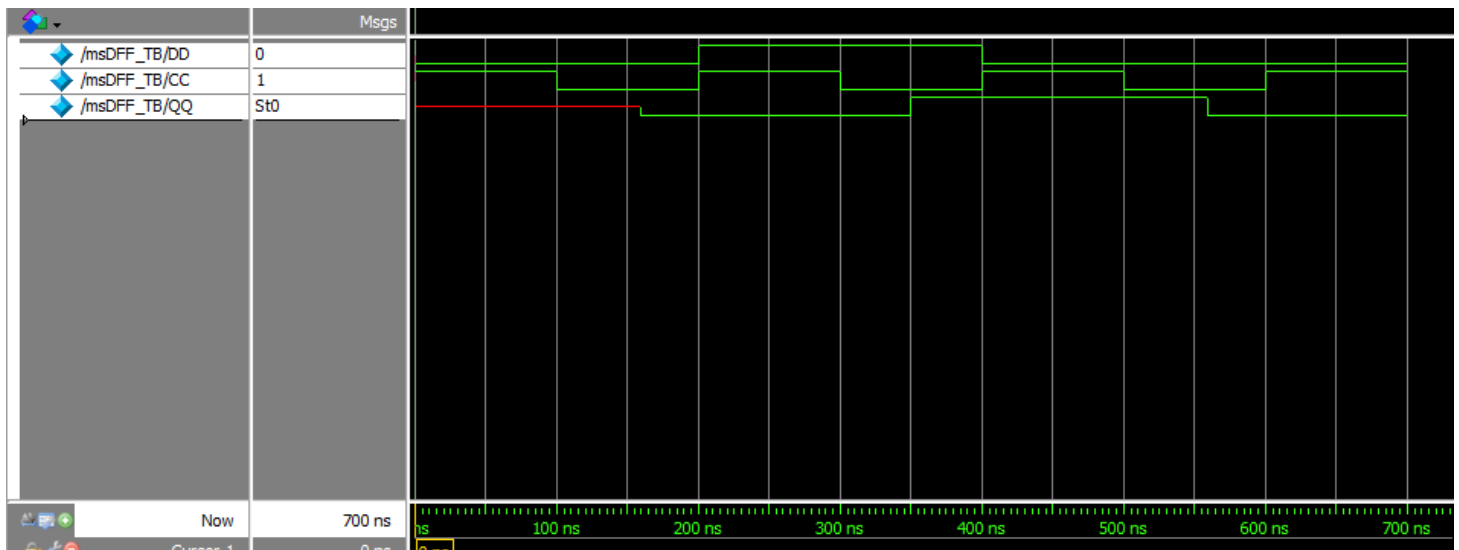
## c. Image of project and compiling:



Project - E:/Desktop/CA3/CA3

Name	Status	Type	Order	Modified
Problem5.v	✓	Verilog	4	05/31/2021 08:11
Problem2.v	✓	Verilog	1	05/31/2021 07:47
Problem4.v	✓	Verilog	3	05/31/2021 07:41
Problem3.v	✓	Verilog	2	05/31/2021 06:47
Problem1.v	✓	Verilog	0	05/31/2021 05:24
Problem7.v	✓	Verilog	6	06/01/2021 12:21
Problem6.v	✓	Verilog	5	05/31/2021 11:15

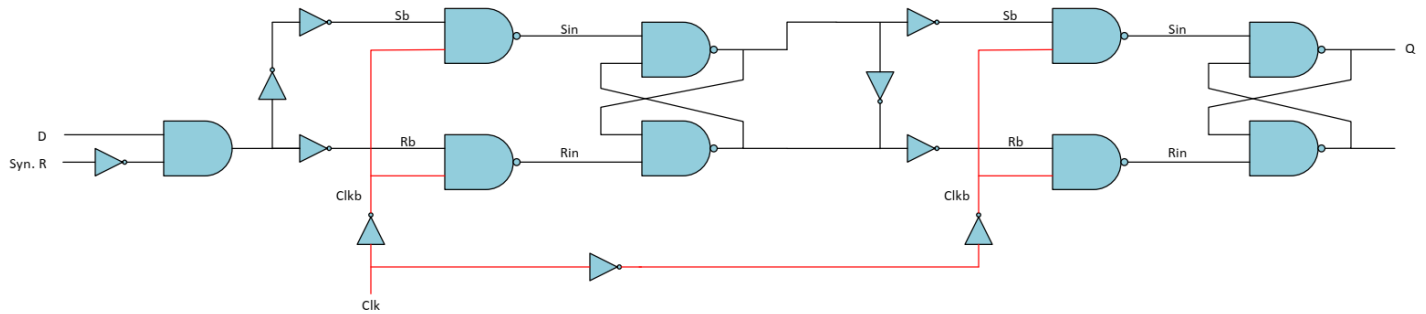
## d. Waveform:



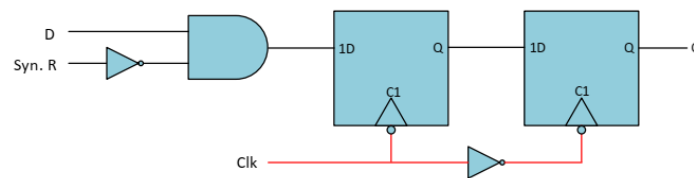
## Problem 7. File: Problem7.v

### A. Circuit diagram:

In following figure we show the design of a synchronous reset (rs) master-slave D-type flip-flop using two D-latches:



Synchronous reset (rs) master-slave D-type flip-flop using two D-latches:



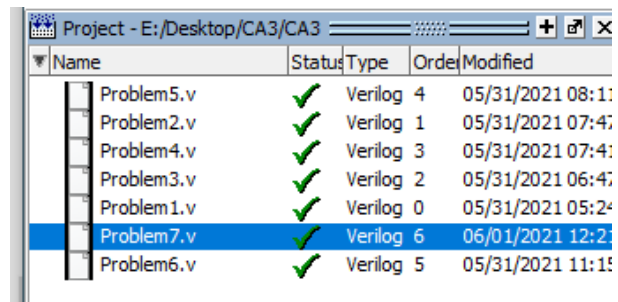
### B. Verilog Description:

File: Problem7.v

```
`timescale 1ns/1ns
module aRmsDFF(input D,Clk,R,output Q);
    wire Rb,Din,DQ,bClk;
    not #6 G1(Rb,R);
    and #14 G2(Din,D,Rb);
    not #6 G3(bClk,Clk);
    CDlatch D1(Din,Clk,DQ);
    CDlatch D2(DQ,bClk,Q);
endmodule

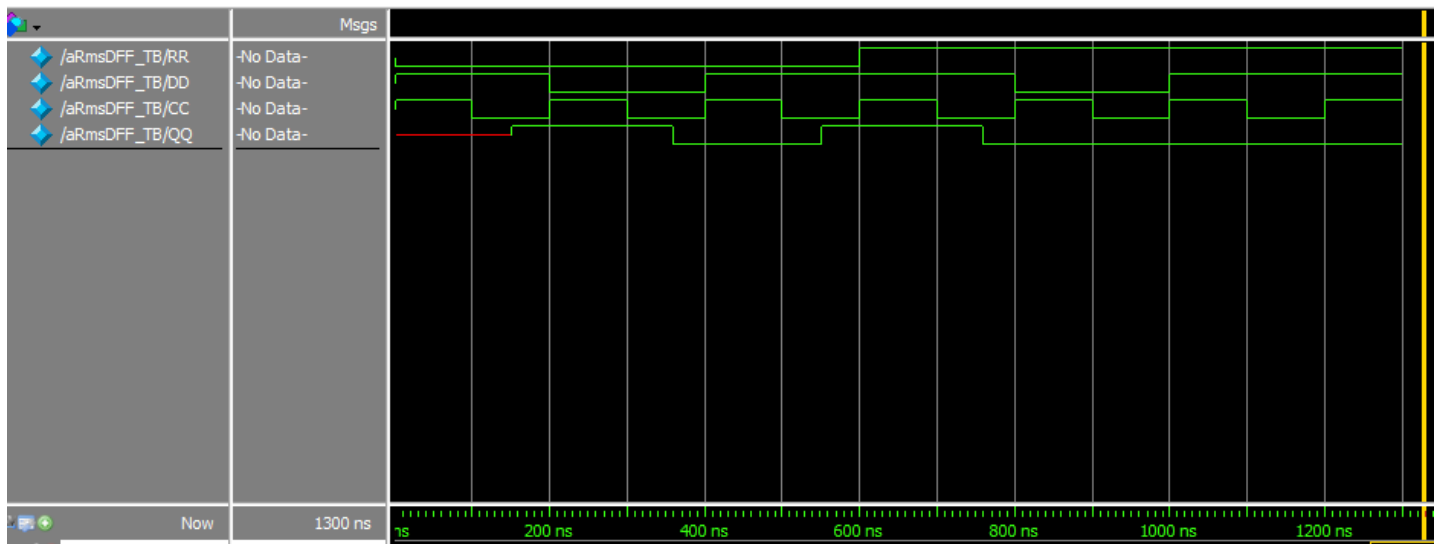
module aRmsDFF_TB();
    reg RR=0,DD=1,CC=1;
    wire QQ;
    aRmsDFF UUT(DD,CC,RR,QQ);
    always #100 CC<=~CC;
    initial begin
        #200 DD=0;
        #200 DD=1;
        #200 RR=1;
        #200 DD=0;
        #200 DD=1;
        #300 $stop;
    end
endmodule
```

### c. Image of project and compiling:



Name	Status	Type	Order	Modified
Problem5.v	✓	Verilog	4	05/31/2021 08:11
Problem2.v	✓	Verilog	1	05/31/2021 07:47
Problem4.v	✓	Verilog	3	05/31/2021 07:47
Problem3.v	✓	Verilog	2	05/31/2021 06:47
Problem1.v	✓	Verilog	0	05/31/2021 05:24
Problem7.v	✓	Verilog	6	06/01/2021 12:21
Problem6.v	✓	Verilog	5	05/31/2021 11:15

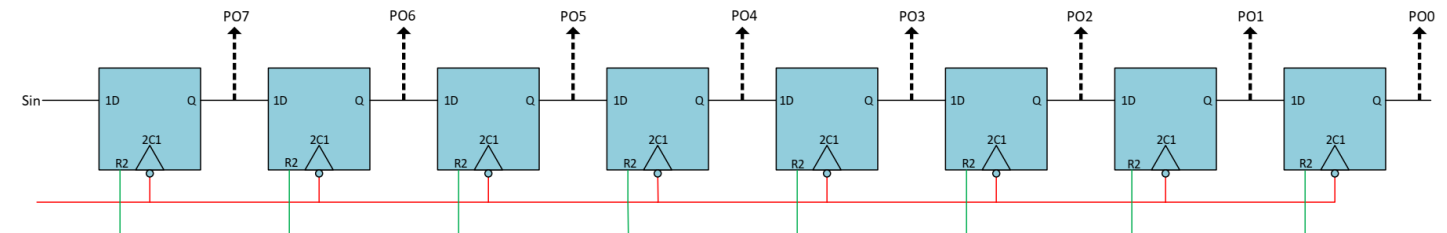
### d. Waveform:



## Problem 8. File: Problem8.v

### A. Circuit diagram:

In following figure we show the design of an 8-bit shift register: (Using clocked D-latch built in Problem 3.)



### B. Verilog Description:

File: Problem8.v

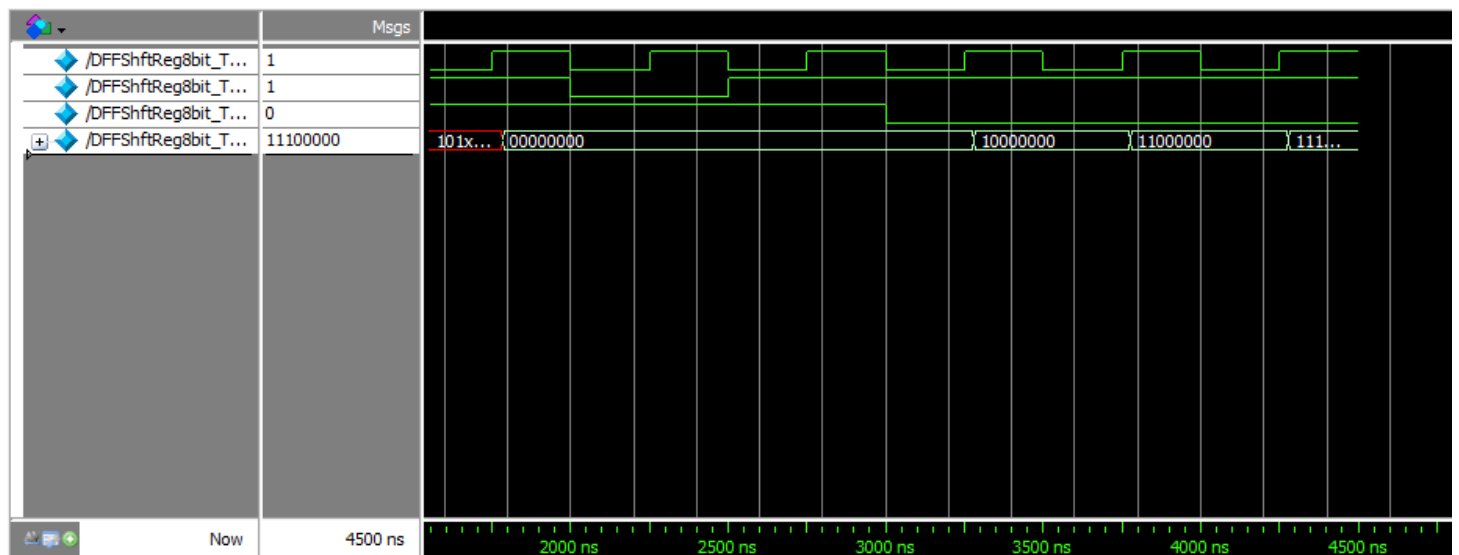
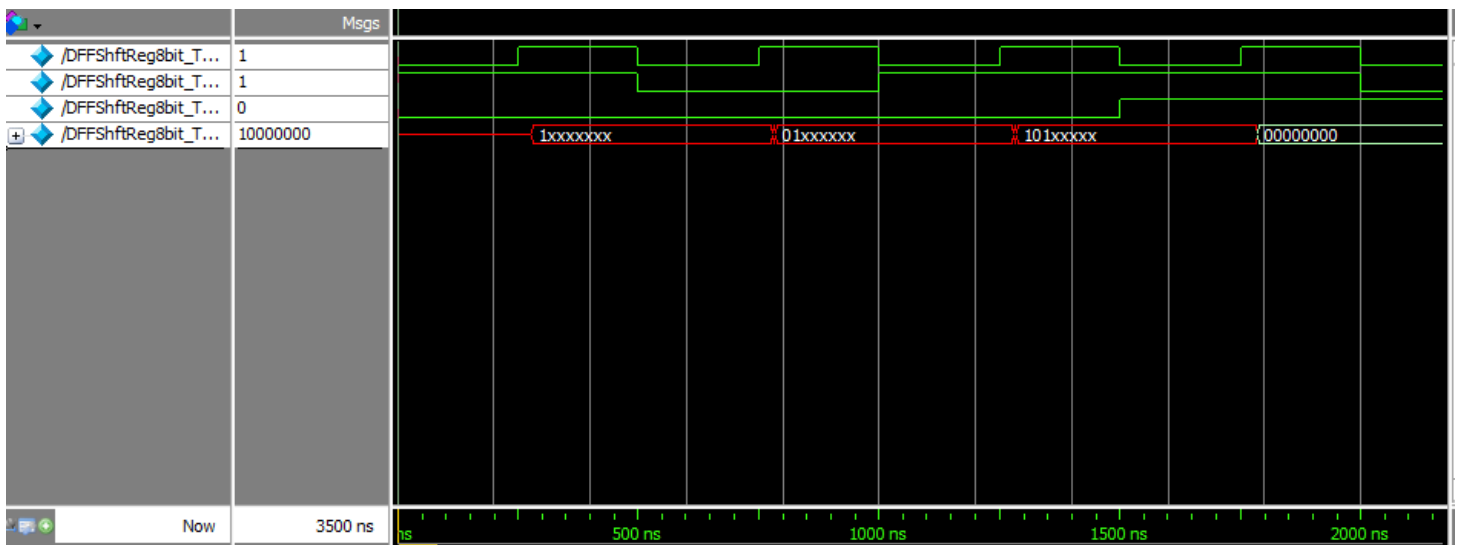
```
`timescale 1ns/1ns
module DFFShftReg8bit(input Clk,Si,R,output [7:0]Po);
    wire [8:0]A;
    assign A = {Si,Po};
    genvar i;
    generate
        for (i=8;i>0;i=i-1)begin:aRDFFs
            aRmsDFF DFF(A[i],Clk,R,Po[i-1]);
        end
    endgenerate
endmodule

`timescale 1ns/1ns
module DFFShftReg8bit_TB();
    reg CC = 0,Si = 1,RR = 0;
    wire [7:0]PP;
    DFFShftReg8bit UUT(CC,Si,RR,PP);
    always #250 CC<=~CC;
    initial begin
        #500 Si=0;
        #500 Si=1;
        #500 RR=1;
        #500 Si=0;
        #500 Si=1;
        #500 RR=0;
        #500 Si=1;
        #1000 $stop;
    end
endmodule
```

### c. Image of project and compiling:

Project - E:/Desktop/CA3/CA3				
Name	Status	Type	Order	Modified
Problem5.v	✓	Verilog	4	06/01/2021 12:42
Problem7.v	✓	Verilog	6	06/01/2021 01:21
Problem6.v	✓	Verilog	5	06/01/2021 01:20
Problem2.v	✓	Verilog	1	05/31/2021 07:47
Problem4.v	✓	Verilog	3	05/31/2021 07:41
Problem3.v	✓	Verilog	2	05/31/2021 06:47
Problem1.v	✓	Verilog	0	05/31/2021 05:24
Problem8.v	✓	Verilog	7	06/01/2021 01:32

#### D. Waveform:





## Problem 9. File: Problem9.v

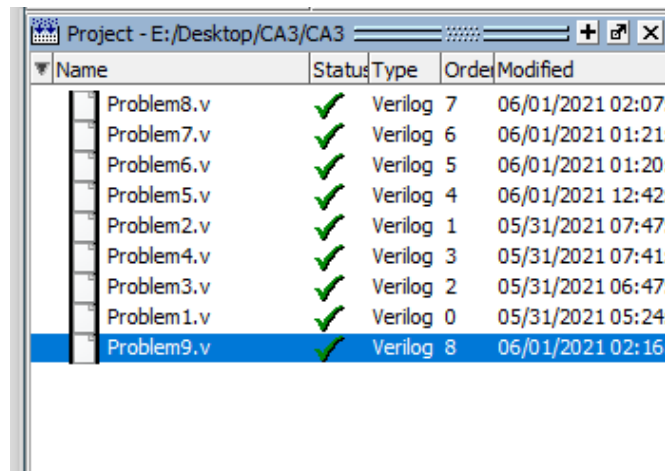
### A. Verilog Description:

File: Problem9.v

```
`timescale 1ns/1ns
module P9DFFShftReg8bit(input Clk,Si,R,output reg[7:0]Po);
    always @ (posedge Clk) begin
        if (R) Po <= 8'd0;
        else Po <= {Si,Po[7:1]};
    end
endmodule

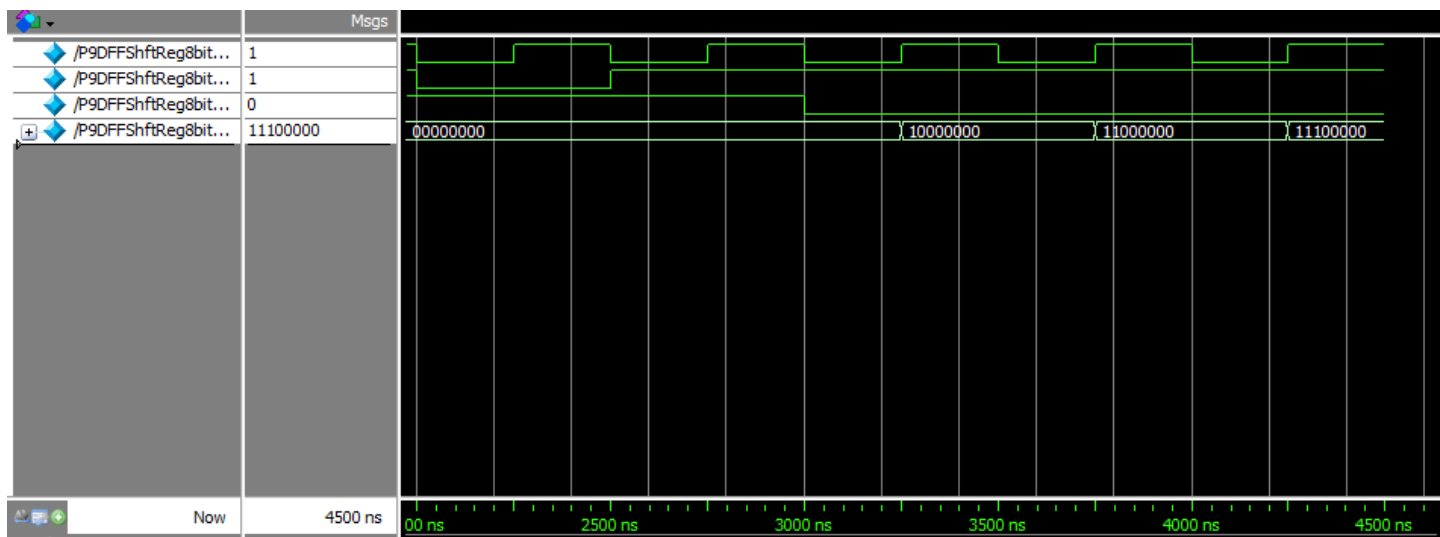
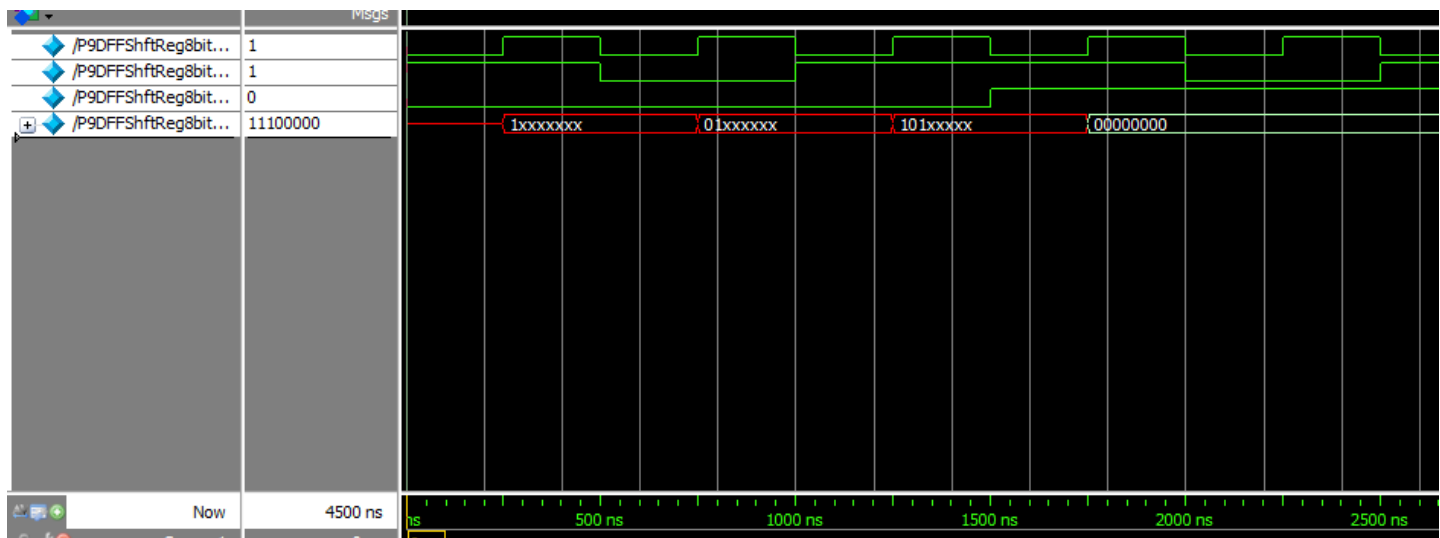
`timescale 1ns/1ns
module P9DFFShftReg8bit_TB();
    reg CC = 0,Si = 1,RR = 0;
    wire [7:0]PP;
    P9DFFShftReg8bit UUT(CC,Si,RR,PP);
    always #250 CC<=~CC;
    initial begin
        #500 Si=0;
        #500 Si=1;
        #500 RR=1;
        #500 Si=0;
        #500 Si=1;
        #500 RR=0;
        #500 Si=1;
        #1000 $stop;
    end
endmodule
```

### B. Image of project and compiling:



Name	Status	Type	Order	Modified
Problem8.v	✓	Verilog	7	06/01/2021 02:07
Problem7.v	✓	Verilog	6	06/01/2021 01:21
Problem6.v	✓	Verilog	5	06/01/2021 01:20
Problem5.v	✓	Verilog	4	06/01/2021 12:42
Problem2.v	✓	Verilog	1	05/31/2021 07:47
Problem4.v	✓	Verilog	3	05/31/2021 07:41
Problem3.v	✓	Verilog	2	05/31/2021 06:47
Problem1.v	✓	Verilog	0	05/31/2021 05:24
Problem9.v	✓	Verilog	8	06/01/2021 02:16

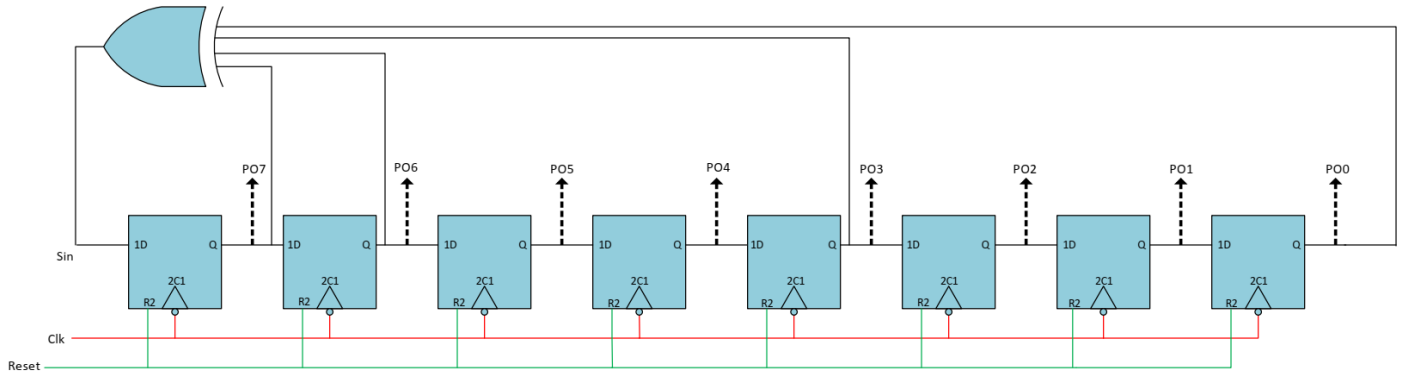
### c. Waveform:



## Problem 10. File: Problem10.v

### A. Circuit diagram:

In following figure we show the diagram of a LFSR: (We can add 8 2-to-1 MUX to have load input)



### B. Verilog Description:

File: Problem10.v

```

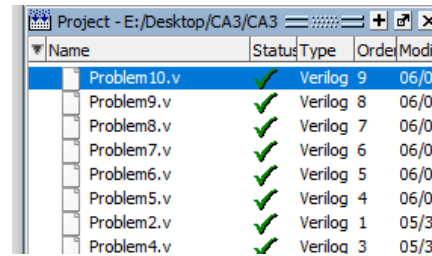
module myLFSR(input [7:0]load,input load_active,Clk,R,output[7:0]Po,output So);
    wire FB;
    assign So = Po[0];
    assign FB = Po[0]^Po[3]^Po[6]^Po[7];
    mySHFREG SHF(load,load_active,Clk,R,FB,Po);
endmodule

module P9withLoad(input [7:0]load,input load_active,Clk,R,Si,output reg[7:0]Po);
    always @ (negedge Clk,posedge R) begin
        if (R) Po <= 8'd0;
        else if (load_active) Po <= load;
        else Po <= {Si,Po[7:1]};
    end
endmodule

module myLFSR_TB();
    reg CC = 1,RR = 0,lload_active=0;
    reg [7:0] lload = 8'd1;
    wire [7:0]PP;
    wire SSo;
    myLFSR uut(lload,lload_active,CC,RR,PP,SSo);
    always #5 CC<=~CC;
    initial begin
        #10 lload_active=1;
        #10 lload_active=0;
        #10000 RR = 1;
        #500 $stop;
    end
endmodule

```

### c. Image of project and compiling:

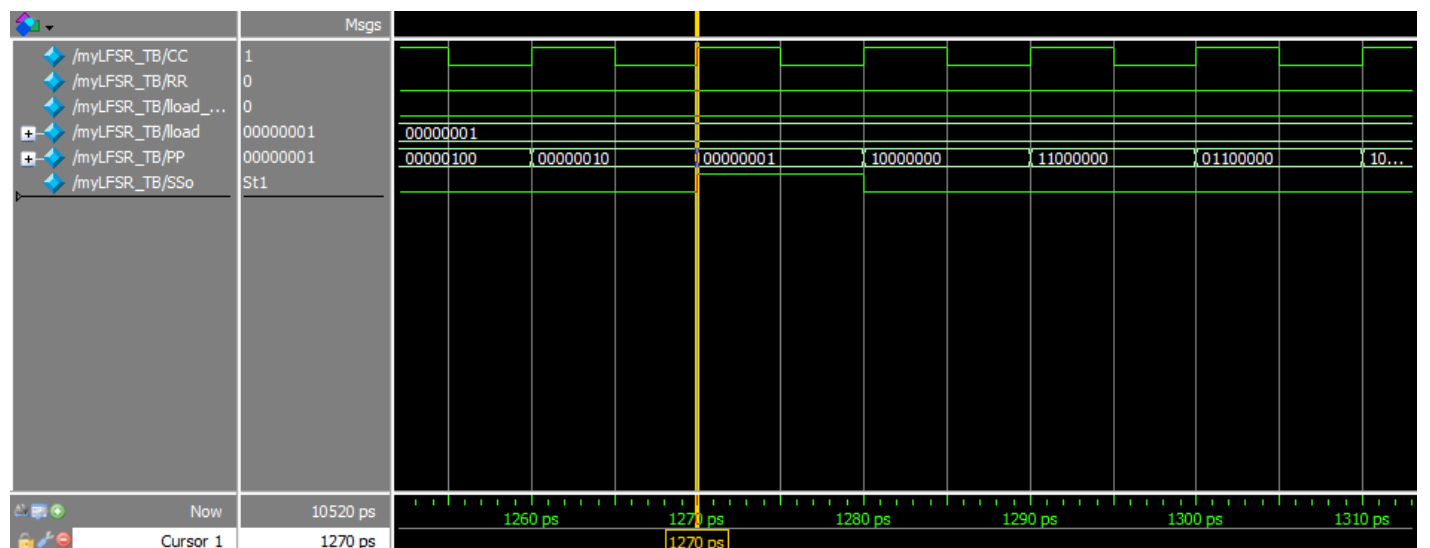
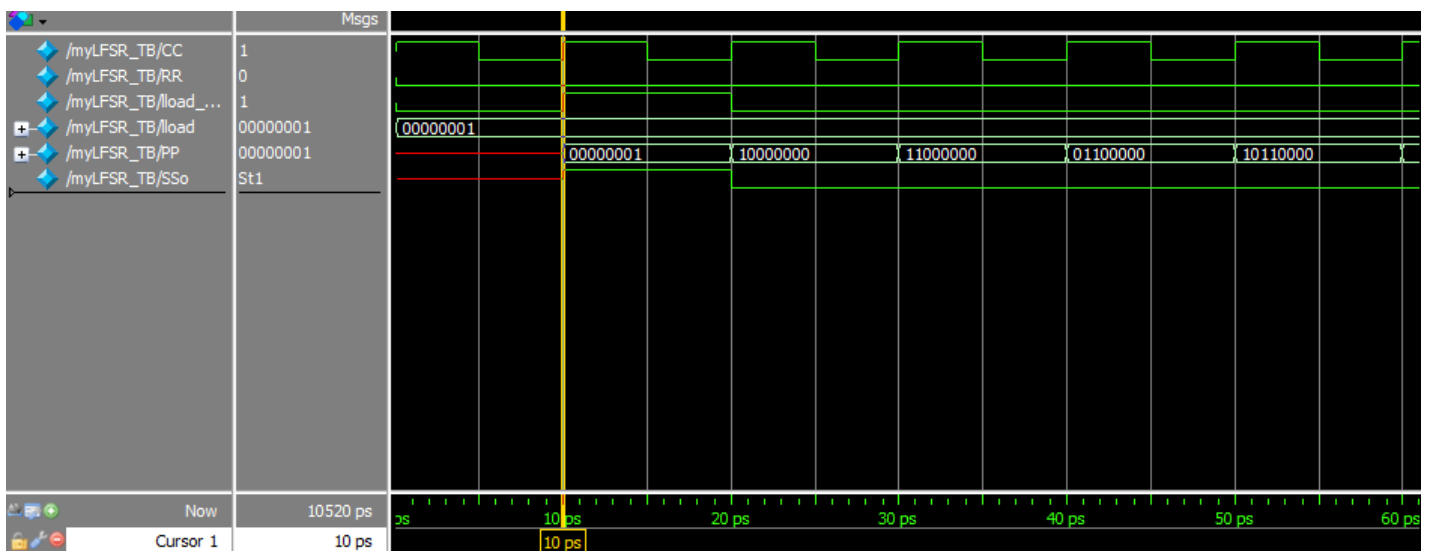


Name	Status	Type	Order	Modi
Problem10.v	✓	Verilog	9	06/0
Problem9.v	✓	Verilog	8	06/0
Problem8.v	✓	Verilog	7	06/0
Problem7.v	✓	Verilog	6	06/0
Problem6.v	✓	Verilog	5	06/0
Problem5.v	✓	Verilog	4	06/0
Problem2.v	✓	Verilog	1	05/3
Problem4.v	✓	Verilog	3	05/3

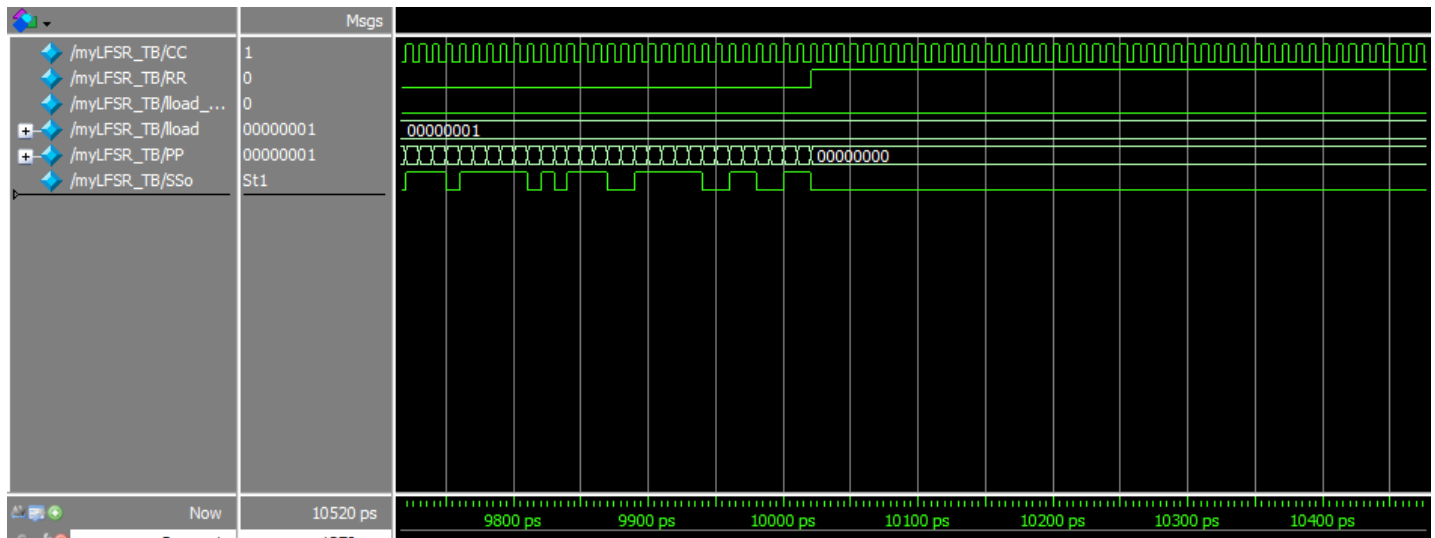
### d. Waveform:

As we see in following waveforms, after every 630NS the input return to its first value (00000001).

$$\text{Clock period} = 5 + 5 = 10\text{NS} \rightarrow \text{Period of LFSR} = \frac{630}{10} = 63$$



When we initial the first value with **00000001** we won't have any change. Because the output of XOR doesn't change.



If the input of XOR was Po[6] ,Po[5] ,Po[2] and Po[0]:

As we see in following waveforms, after every 2550NS the input return to its first value (**00000001**).

$$\text{Clock priod} = 5 + 5 = 10\text{NS} \rightarrow \text{Period of LFSR} = \frac{2550}{10} = \mathbf{255}$$

