UNIVERSITY OF TEHRAN

Report for Computer Assignment 4

Latches, flip-flops, and a little beyond

Instructor : Dr. Navabi

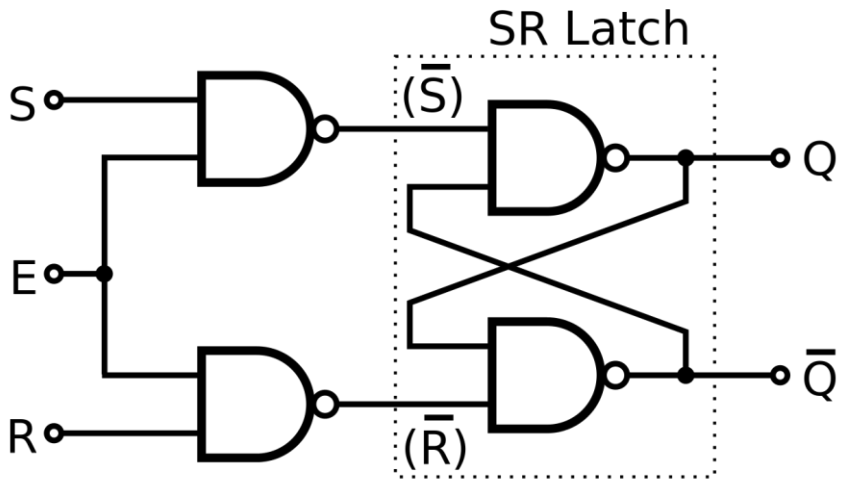
Danial Saeedi

`

Problem 1

Circuit Diagram

The worst-case delay of NAND according to its delays is 4 + 4 = 8ns.



Verilog Code

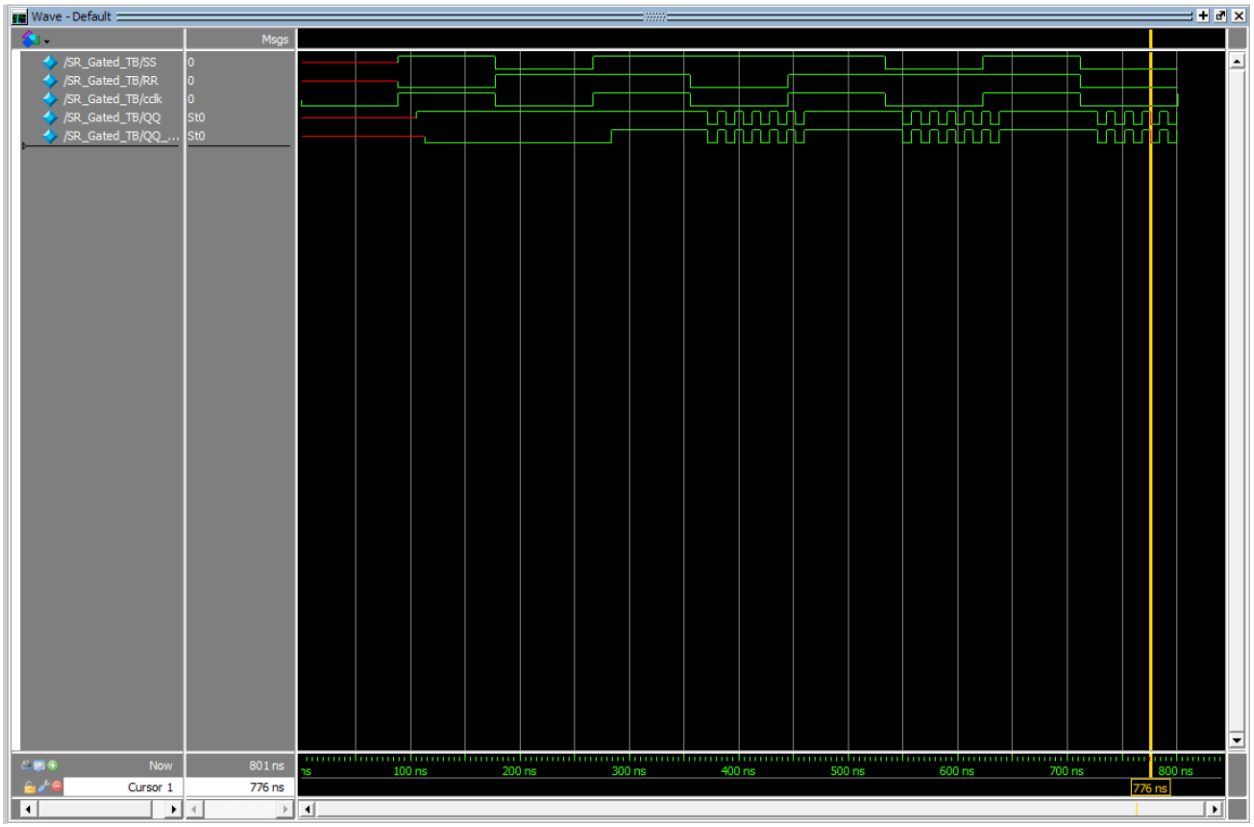
```verilog
`timescale 1ns/1ns
module SR_Latch(input S, R , CLK, output Q, Q_bar);
    wire g,p;
    nand #8 G1(g,S,CLK);
    nand #8 G2(p,R,CLK);

    nand #8 G3(Q,g,Q_bar);
    nand #8 G4(Q_bar,p,Q);
endmodule
```

`

Problem 2

Testbench

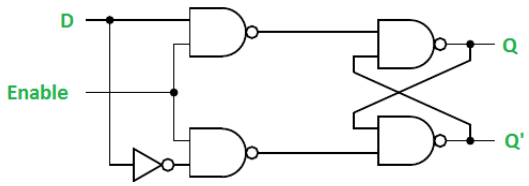
```
1  `timescale 1ns/1ns
2  module SR_Gated_TB();
3      reg SS, RR;
4      reg cclk=1'b0;
5      wire QQ, QQ_prime;
6
7      SR_Latch CUT1 (SS, RR, cclk, QQ, QQ_prime);
8
9      always #80 cclk=~cclk;
10
11     initial begin
12         #89 SS=1;RR=0;
13         #89 SS=0;RR=1;
14         #89 SS=1;
15         #89 SS=1;RR=0;
16         #89 SS=1;RR=1;
17         #89 SS=0;RR=1;
18         #89 SS=1;RR=1;
19         #89 SS=0;RR=0;
20         #89 $stop;
21     end
22 endmodule
```



`

Problem 3

Circuit Diagram



Verilog Code

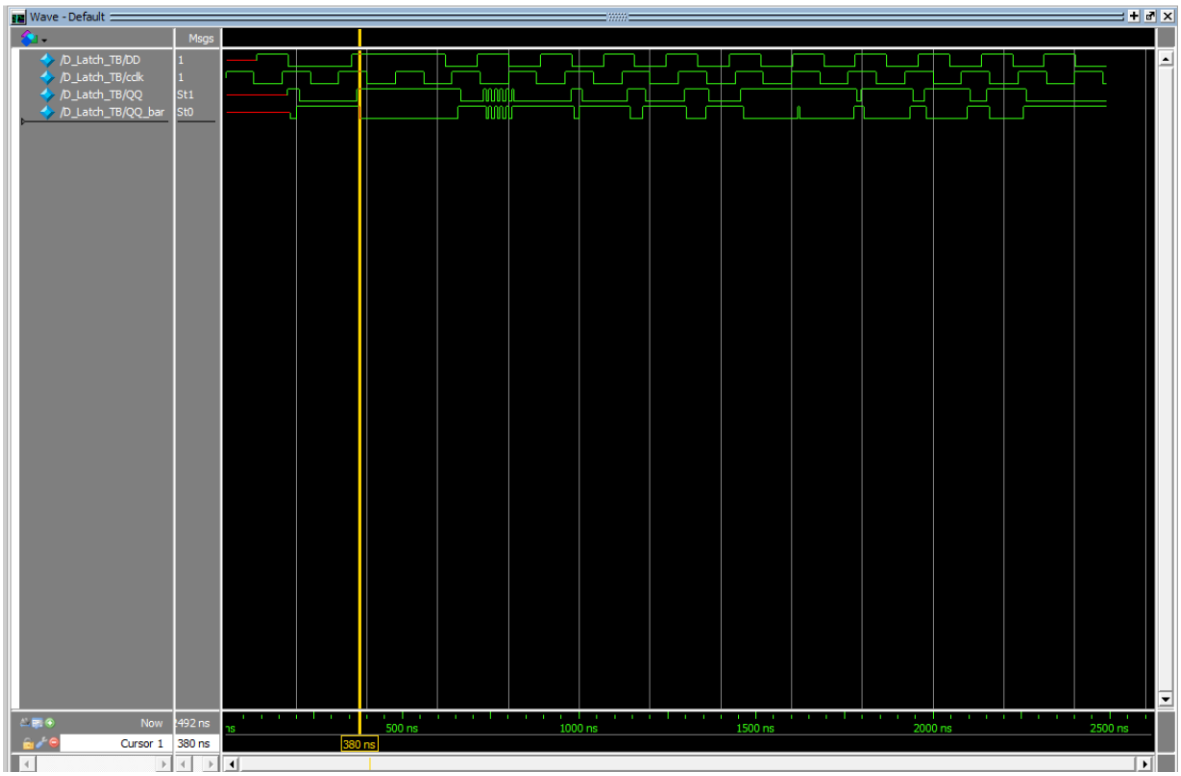
```verilog
`timescale 1ns/1ns
module D_Latch(input D, clk, output Q, Q_BAR);
    wire D_BAR;

    not #6 G1(D_BAR,D);

    SR_Latch CUT1(D, D_BAR, clk, Q, Q_BAR);
endmodule
```

Testbench

```verilog
`timescale 1ns/1ns
module D_Latch_TB();
    reg DD;
    reg cclk=1;
    wire QQ,QQ_bar;

    D_Latch CUT1 (DD, cclk, QQ, QQ_bar);

    always #80 cclk=~cclk;

    initial begin
        #89 DD=1;
        #89 DD=0;
        #89 DD=0;
        #89 DD=1;
        #89 DD=1;
        #89 DD=1;
        #89 DD=0;
        repeat(20) #89 DD = ~DD;
        #89 $stop;
    end
endmodule
```

`

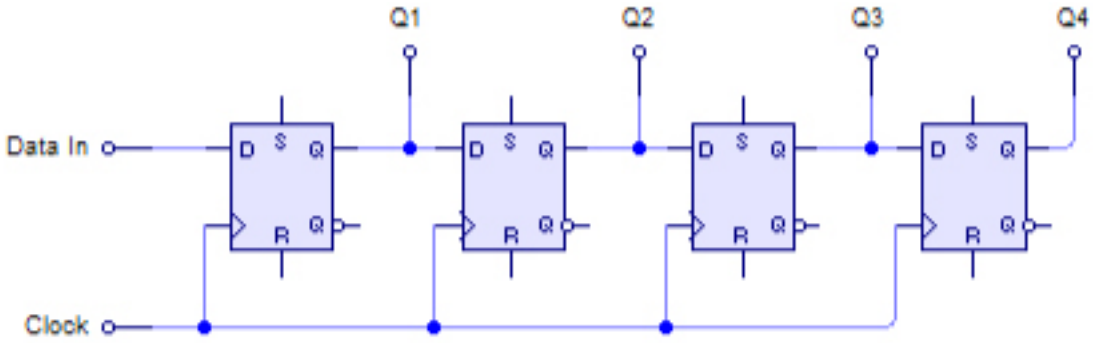
Simulation Result



`

Problem 4

Circuit Diagram



Verilog Code

```verilog
`timescale 1ns/1ns
module shift_register(input sin,clk, output [7:0] PO);
    wire [8:0] Q;
    wire [8:0] Q_BAR;
    assign Q[8] = sin;
    assign PO[0] = Q[0];
    genvar i;
    generate
    for (i=0;i<8;i=i+1) begin: I0
        D_Latch XX(Q[i+1],clk,Q[i],Q_BAR[i]);
        assign PO[i]=Q[i];
    end
    endgenerate
endmoduleA
```

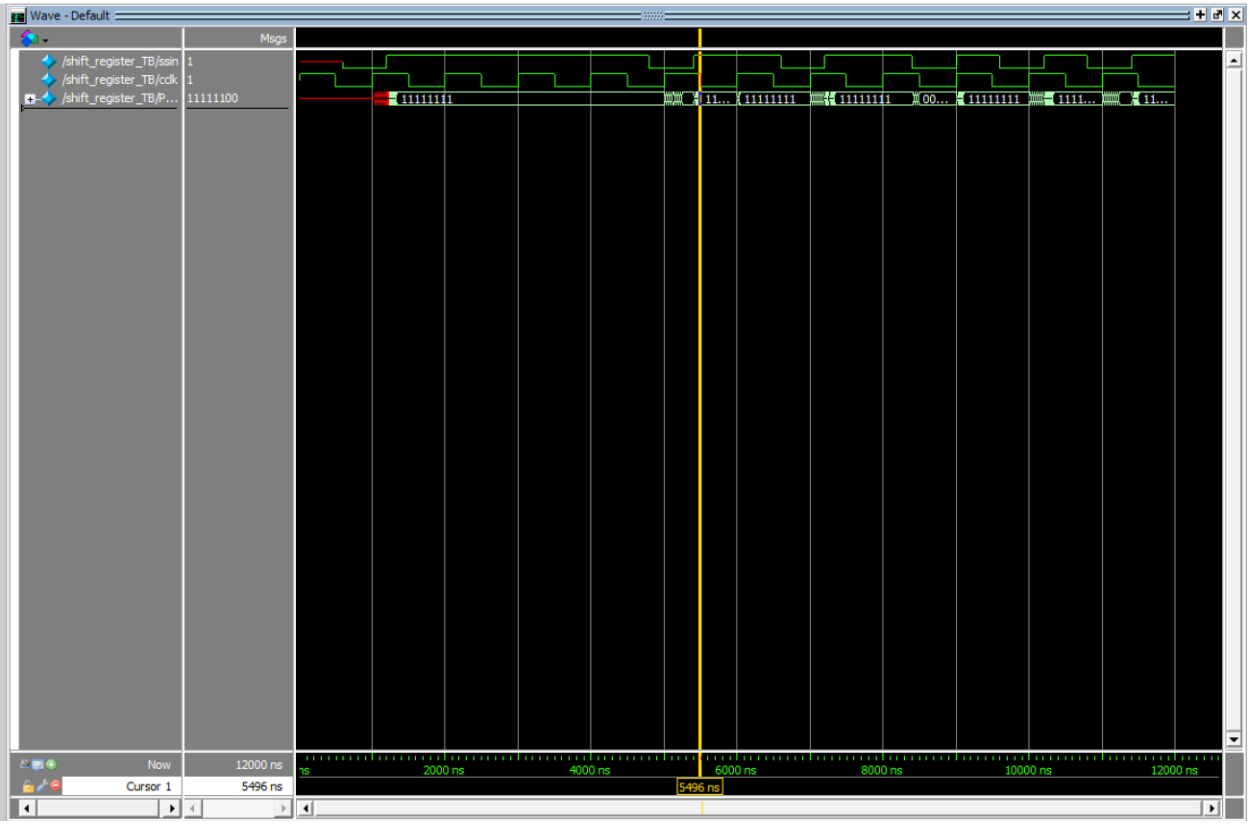
`

Problem 5

Testbench

```
1  `timescale 1ns/1ns
2  module shift_register_TB();
3      reg ssin;
4      reg cclk = 1'b1;
5      //Primary Output
6      wire [7:0] PPO;
7
8      shift_register CUT1(ssin, cclk, PPO);
9
10     always  #600 ssin=$random;
11     always  #500 cclk=~cclk;
12     initial begin
13         #12000 $stop;
14     end
15 endmodule
```

Simulation Result

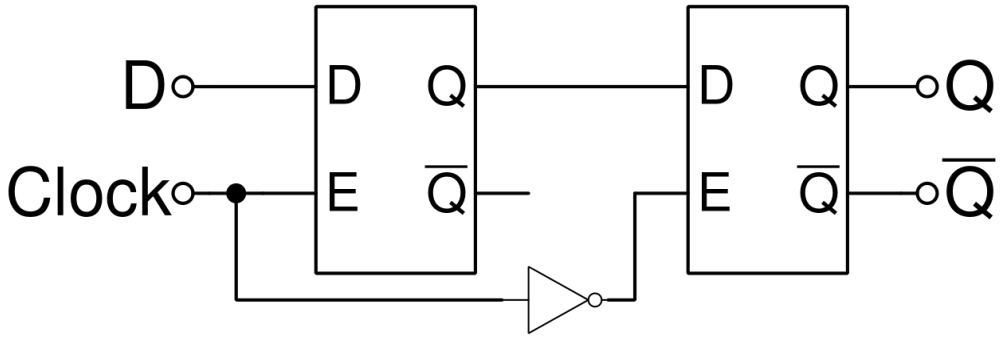


As you can see, this circuit **doesn't** work because it has **transparency** issue.

`

Problem 6

Circuit Diagram



Verilog Code

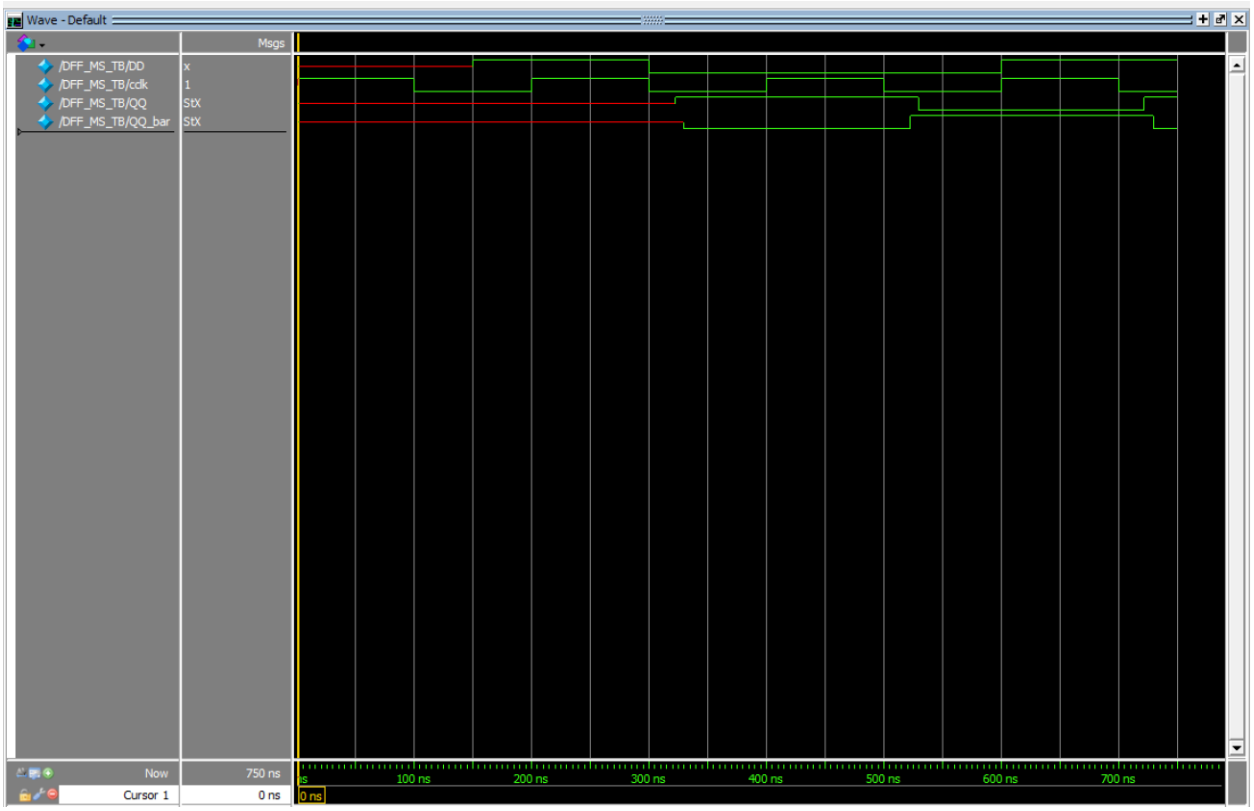
```verilog
`timescale 1ns/1ns
module DFF_MS(input D, clk, output Q,Q_bar);
    wire q1,q1_bar,clk_not;
    D_Latch D1(D,clk,q1,q1_bar);

    not #6 G1(clk_not,clk);
    D_Latch D2(q1,clk_not,Q,Q_bar);
endmodule
```

`

Testbench

```
1  `timescale 1ns/1ns
2  module DFF_MS_TB();
3      reg DD;
4      reg cclk=1'b1;
5      wire QQ,QQ_bar;
6
7      DFF_MS CUT(DD, cclk, QQ, QQ_bar);
8
9      always #100 cclk=~cclk;
10
11     initial begin
12         #150 DD=1;
13         #150 DD=0;
14         #150 DD=0;
15         #150 DD=1;
16         #150 $stop;
17     end
18 endmodule
```

Simulation Result



`

Problem 7

Verilog Code

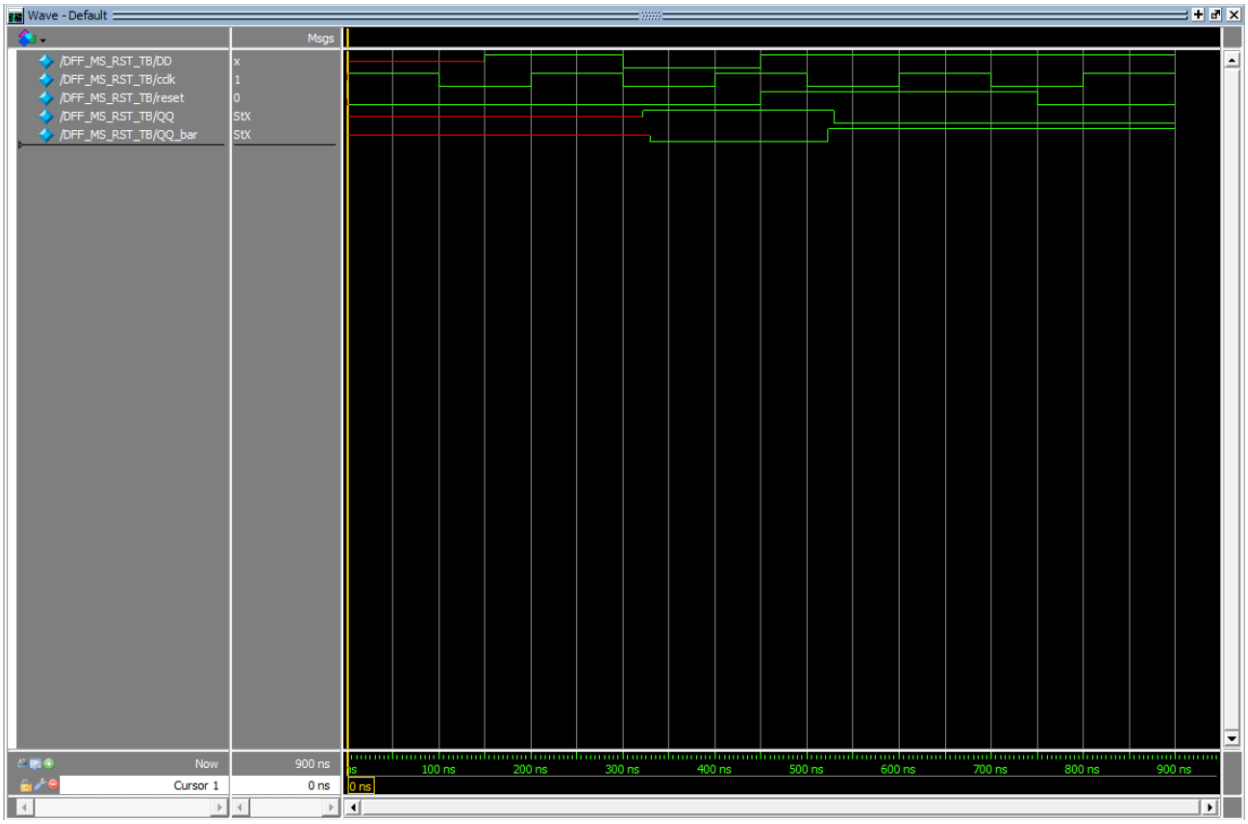
```
`timescale 1ns/1ns
module DFF_MS_RST(input D, clk, rst, output Q,Q_bar);
    wire D2;
    assign D2 = D & ~rst;

    DFF_MS CUT(D2, clk, Q);
endmodule
```

Testbench

```
`timescale 1ns/1ns
module DFF_MS_RST_TB();
    reg DD;
    reg cclk=1'b1;
    reg reset = 1'b0;
    wire QQ,QQ_bar;

    DFF_MS_RST CUT(DD, cclk, reset, QQ, QQ_bar);

    always #100 cclk=~cclk;

    initial begin
        #150 DD=1;
        #150 DD=0;
        #150 DD=1; reset = 1;
        #150 DD=1;
        #150 DD=1; reset = 0;
        #150 $stop;
    end
endmodule
```

`

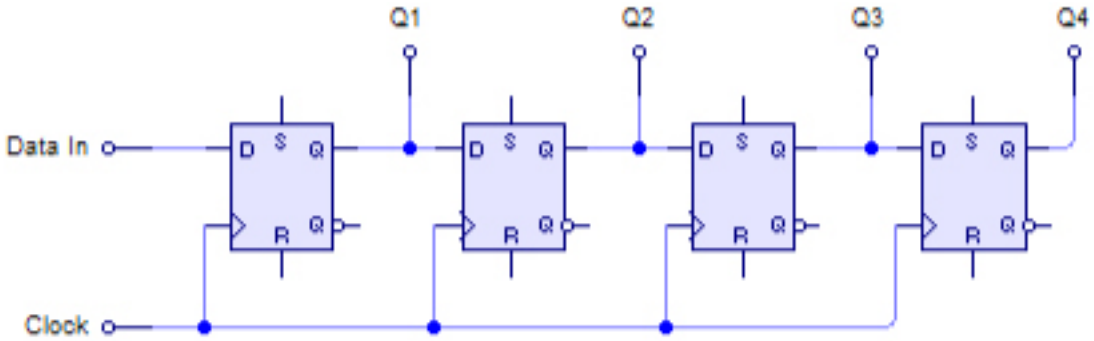
Simulation Result



`

Problem 8

Circuit Diagram



Verilog Code

```verilog
`timescale 1ns/1ns
module shift_register2(input sin,clk,rst, output [7:0] PO);
    wire [8:0] Q;
    wire [8:0] Q_BAR;
    assign Q[8] = sin;
    assign PO[0] = Q[0];
    genvar i;
    generate
    for (i=0;i<8;i=i+1) begin: I0
        DFF_MS_RST XX(Q[i+1],clk, rst ,Q[i],Q_BAR[i]);
        assign PO[i]=Q[i];
    end
    endgenerate
endmodule
```

`

Testbench

```
`timescale 1ns/1ns
module shift_register2_TB();
    reg ssin;
    reg cclk = 1'b1;
    reg reset = 1'b0;
    //Primary Output
    wire [7:0] PPO;

    shift_register2 CUT1(ssin, cclk, reset , PPO);

    always  #600 ssin=$random;
    always  #500 cclk=~cclk;
    initial begin
        #12000 $stop;
    end
endmodule
```

Simulation Result

As you can see, this circuit works because it doesn't has **transparency** issue unlike problem 4.

`

Problem 9

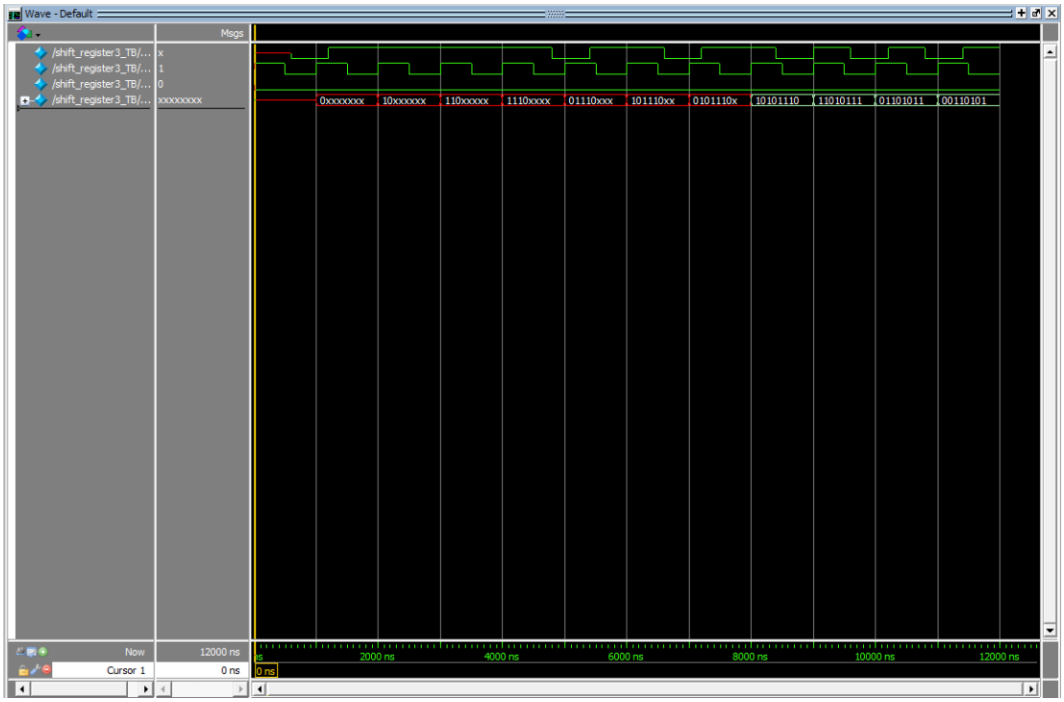
Verilog Code

```
`timescale 1ns/1ns
module shift_register2(input sin, input clk,rs output reg [7:0] PO);
    always @ (posedge clk, posedge rs) begin
        if (rs)
            PO <= 8'b0;
        else
            PO <= {sin, PO[7:1]};
    end
endmodule
```

Testbench

```
`timescale 1ns/1ns
module shift_register3_TB();
    reg ssin;
    reg cclk = 1'b1;
    reg reset = 1'b0;
    //Primary Output
    wire [7:0] PPO;

    shift_register3 CUT1(ssin, cclk, reset , PPO);

    always  #600 ssin=$random;
    always  #500 cclk=~cclk;
    initial begin
        #12000 $stop;
    end
endmodule
```

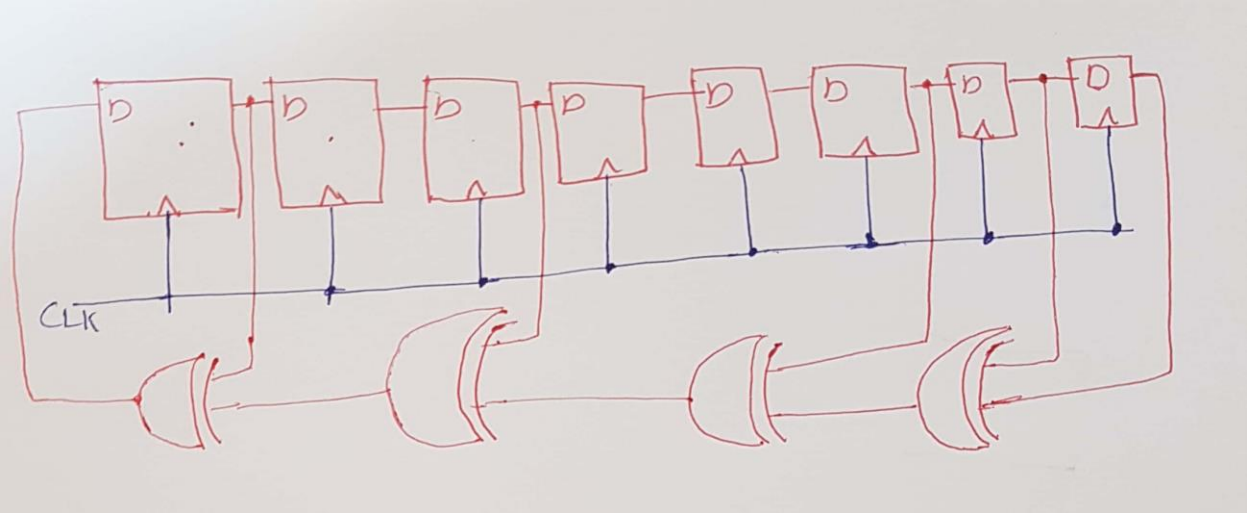
Simulation Result



`

Problem 10

Circuit Diagram



Verilog Code

```verilog
// LFSR with x^8+x^7+x^6+x^3+1 polynomial
module LFSR#(parameter N = 8)( input clk, reset, output [N:0] q);

    reg [N:0] r_reg;
    wire [N:0] r_next;
    wire feedback_value;

    always @(posedge clk, posedge reset)
    begin
        if (reset)
            begin
            r_reg <= 1;
            end
        else if (clk == 1'b1)
            r_reg <= r_next;
    end
    assign feedback_value = r_reg[8] ^ r_reg[7] ^ r_reg[6] ^ r_reg[3] ^ r_reg[0];

    assign r_next = {feedback_value, r_reg[N:1]};
    assign q = r_reg;
endmodule
```

`

Testbench

```
`timescale 1ns/1ns
module LFSR_TB();
    reg cclk = 1'b0;
    reg rrs = 1'b0;
    wire [8:0] q;
    LFSR CUT2(cclk,rrs,q);
    always #100 cclk = ~cclk;
    initial begin
        #50 rrs = 1;
        #50 rrs = 0;
        #2000 $stop;
    end
endmodule
```

Simulation Result