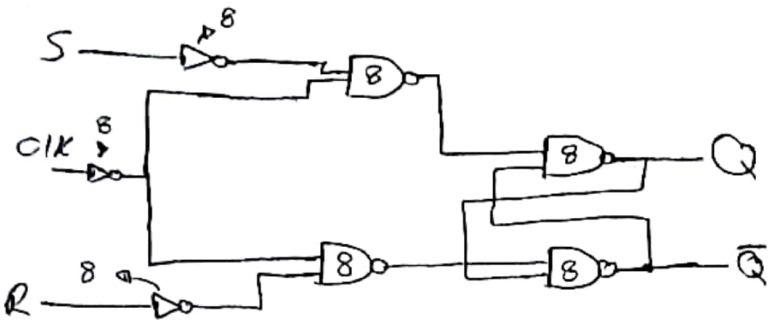
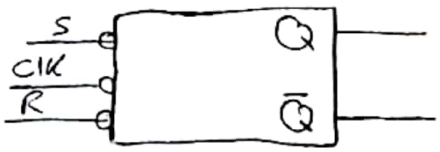
Digital systems
CA4 - Report

Mahdi Bohlouli
8131198365

۱. ①





CLK	S	R	Q	$\bar{Q}$
1	–	–	HOLD	
0	0	0	1	1
0	0	1	1	0
0	1	0	0	1
0	1	1	HOLD	

NAND Delay: 8ns

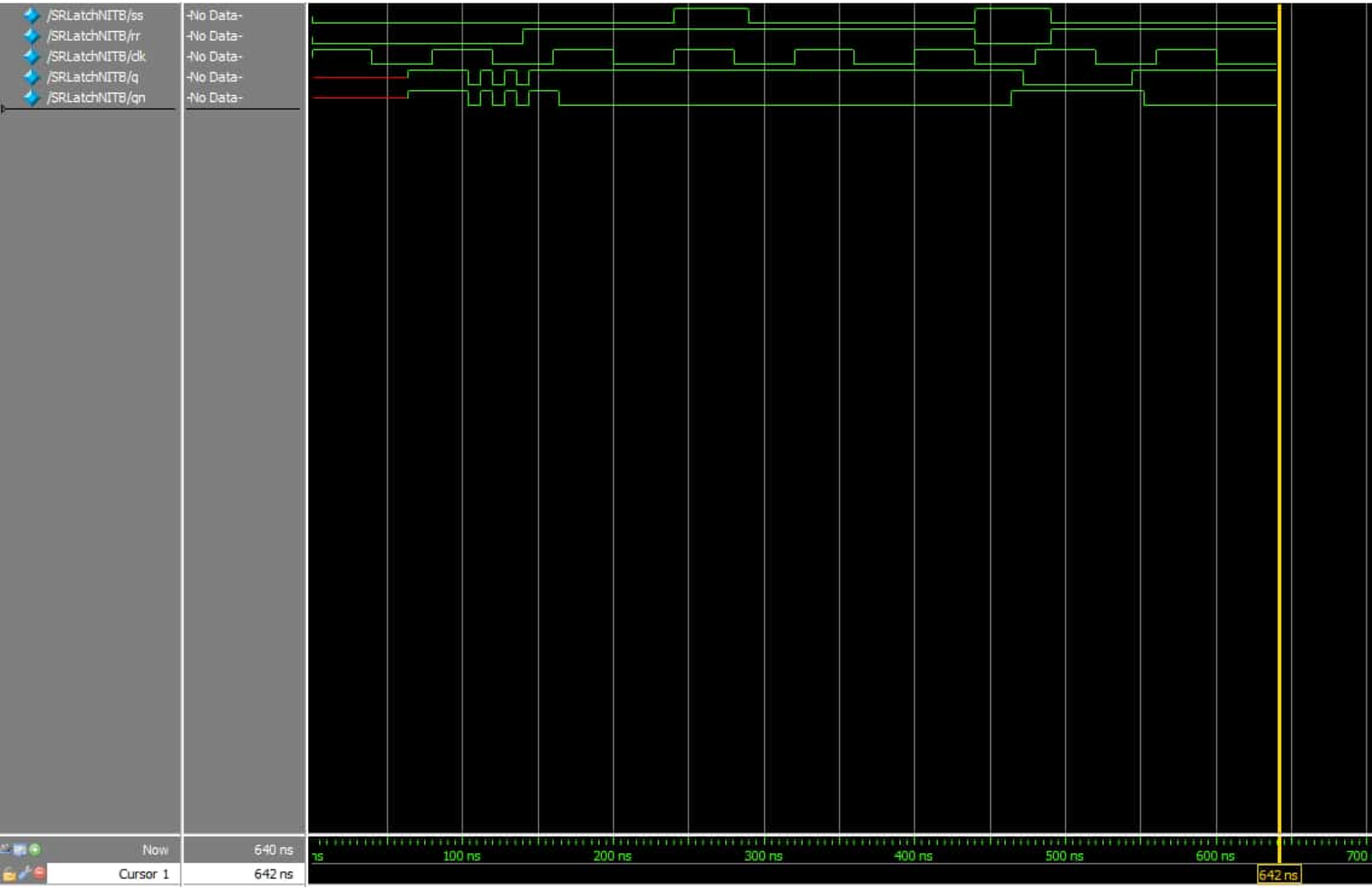
SR-latch = 32ns

SR-Latch NAND Implementation

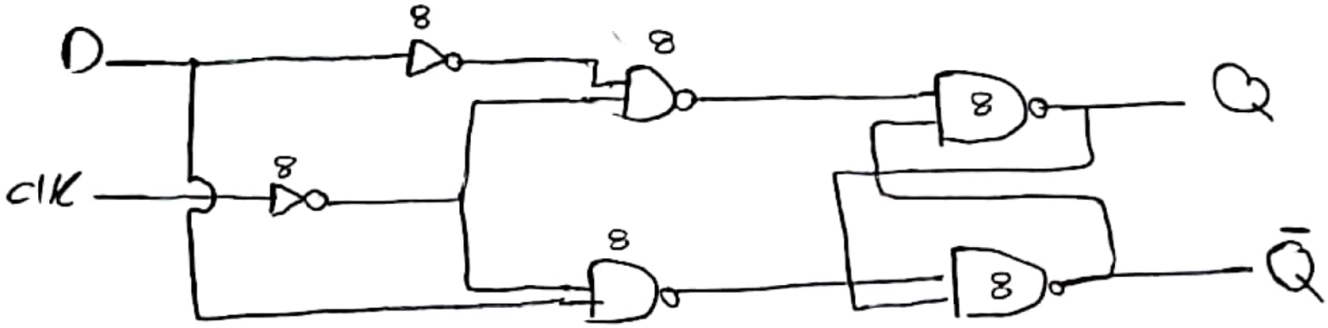
```
`timescale 1ns/1ns


module SRLatchNI (input S, R, clk, output Q, QN);

    wire s_inv, r_inv, clk_inv;
    nand #8 s_inverter(s_inv, S, S), r_inverter(r_inv, R, R), clk_inverter(clk_inv, clk, clk);

    wire s_low_active, r_low_active;
    nand #8 s_low_nand(s_low_active, s_inv, clk_inv), r_low_nand(r_low_active, r_inv, clk_inv);

    nand #8 q_out(Q, QN, s_low_active), qn_out(QN, Q, r_low_active);

endmodule
```

SR-Latch NAND Implementation TB

```
`include "SRLatchNI.v"
`timescale 1ns/1ns

module SRLatchNITB ();
    reg ss=0, rr=0, clk=1;
    wire q, qn;

    SRLatchNI sr_latch(ss, rr, clk, q, qn);
    always #40 clk = ~clk;
    initial begin
        #40
        repeat(10) #50 {ss, rr} = $random;
        #100 $stop;
    end

endmodule
```



سؤال ③ :



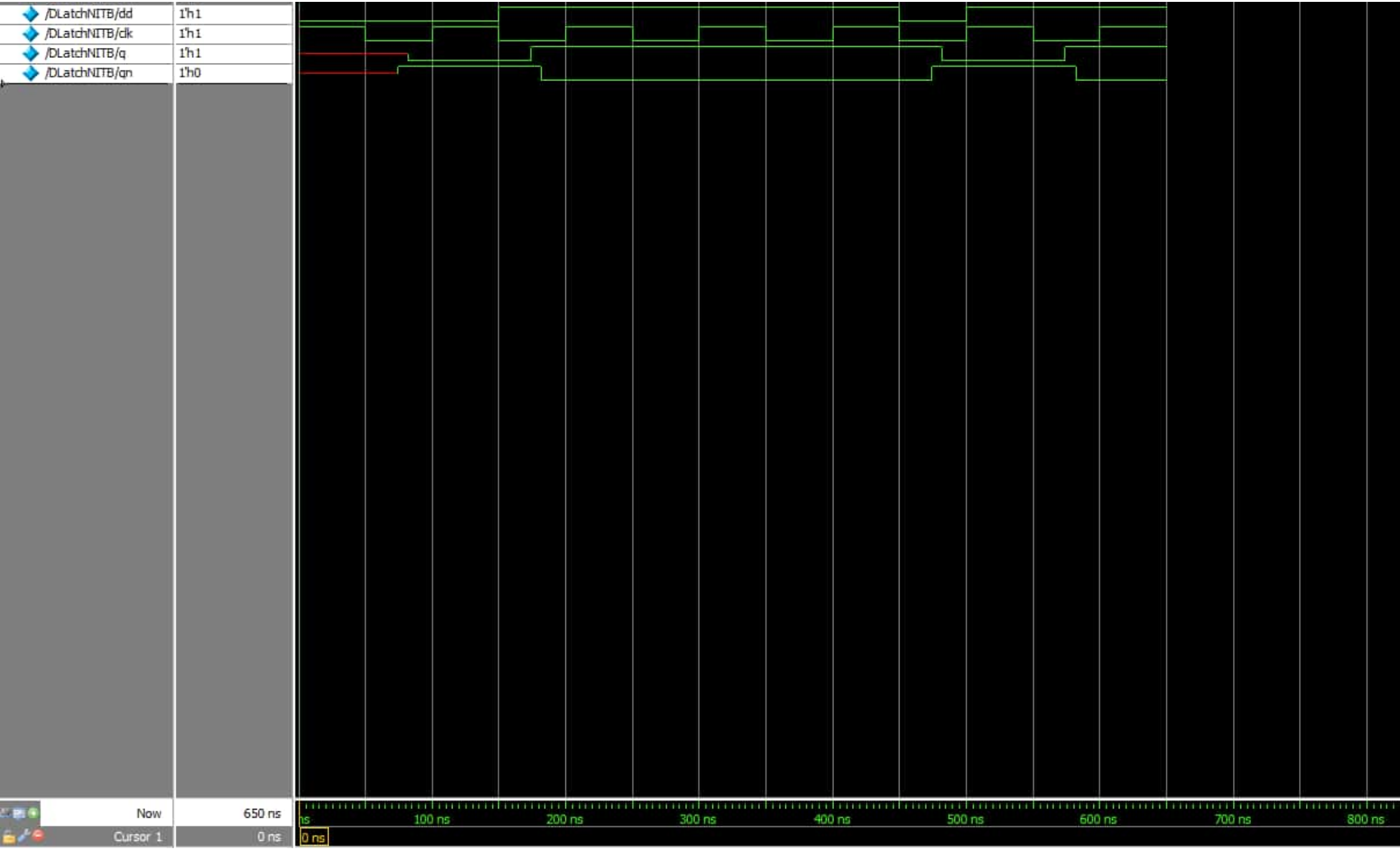
CIK	D	Q	$\bar{Q}$
1	-	Hold	
0	1	0	1
0	0	1	0

D-Latch NAND Implementation

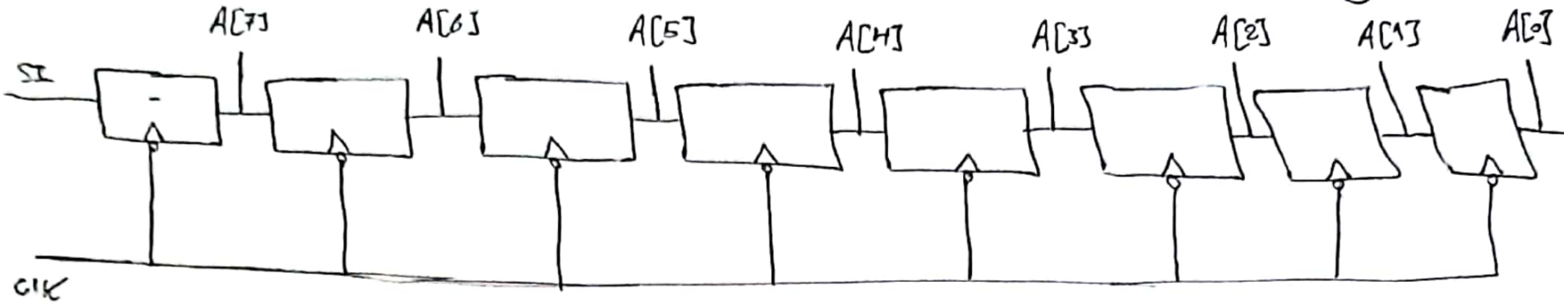
```
`include "SRLatchNI.v"
`timescale 1ns/1ns


module DLatchNI (input D, clk, output Q, QN);
    wire d_inv;
    not (d_inv, D);
    SRLatchNI d_latch(D, d_inv, clk, Q, QN);
endmodule
```

D-Latch NAND Implementation TB

```
`include "DLatchNI.v"
`timescale 1ns/1ns

module DLatchNITB ();
    reg dd=0, clk=1;
    wire q, qn;
    DLatchNI d_latch(dd, clk, q, qn);
    always #50 clk = ~clk;
    initial begin
        #50
        repeat(10) #50 dd = $random;
        #100 $stop;
    end
endmodule
```



سوال (4) :



[redacted]

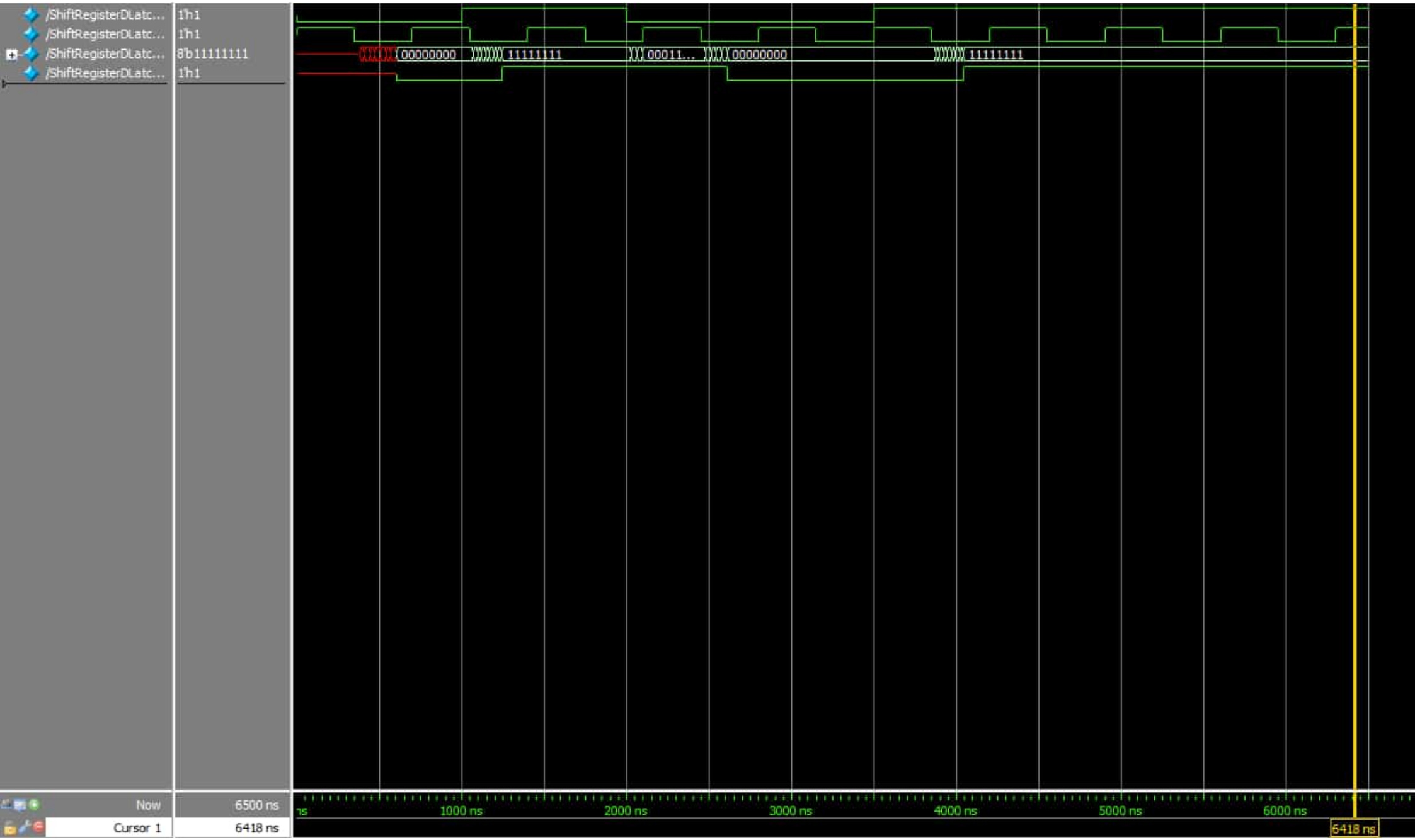
This shift register does not work properly because when the value of CLK is equal to 0 when the value of D is changed, a new value appears on the output at the same time, while we want to change the output only once each time the CLK is changed.

8 Bit Shift Register DLatch

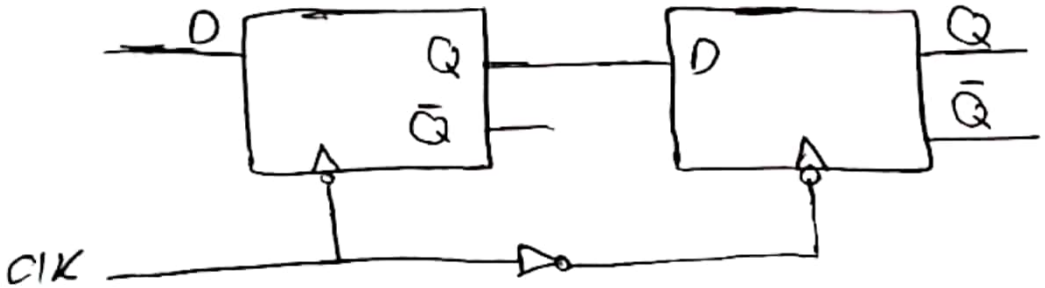
```
`include "DLatchNI.v"
`timescale 1ns/1ns


module ShiftRegisterDLatch (input SI, CLK, output [7:0] Output, output SO);
    wire [7:0] QBar;
    wire [8:0] output_temp;
    assign output_temp[8] = SI;
    genvar i;
    generate
        for (i = 7; i >= 0; i=i-1) begin
            DLatchNI xx(output_temp[i + 1], CLK, output_temp[i], QBar[i]);
        end
    endgenerate
    assign Output = output_temp[7:0];
    assign SO = Output[0];
endmodule
```

8 Bit Shift Register DLatch TB

```
`include "8BitShiftRegisterDLatch.v"
`timescale 1ns/1ns

module ShiftRegisterDLatchTB ();
    reg Si = 0;
    reg Clk = 1;
    wire [7:0] Output;
    wire So;
    ShiftRegisterDLatch shift_register(Si, Clk, Output, So);
    always #350 Clk = ~Clk;
    initial begin
        #500
        #500 Si = 1;
        #1000 Si = 0;
        #1500 Si = 1;
        #3000 $stop;
    end
endmodule
```



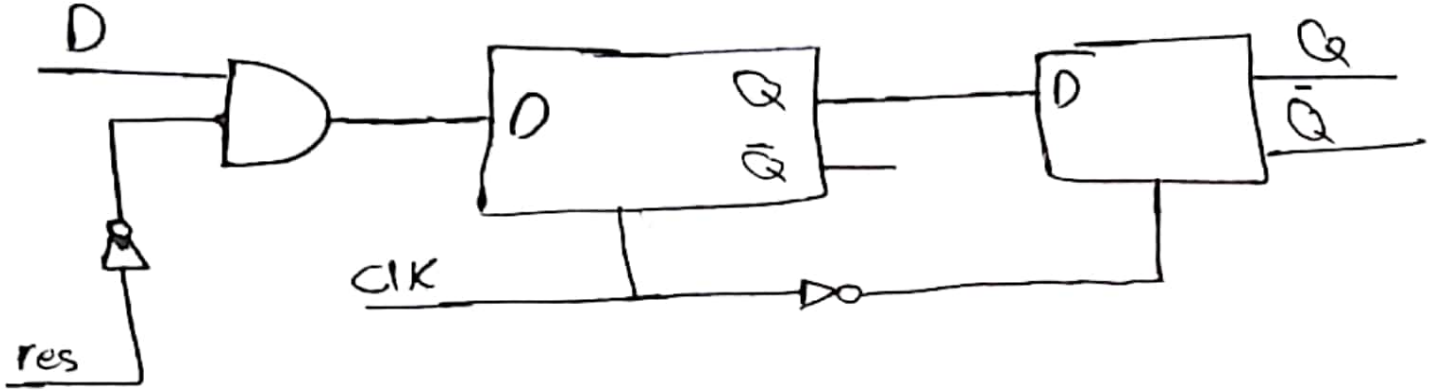
سوال ⑥ :



DFlipFlop

```
`include "DLatchNI.v"
`timescale 1ns/1ns

module DFlipFlop (input D, CLK, output Q, QN);
    wire clk_inv, qm, qmn;
    not(clk_inv, CLK);
    DLatchNI d_latch_1(D, CLK, qm, qmn);
    DLatchNI d_latch_2(qm, clk_inv, Q, QN);
endmodule
```

سؤال (7) :

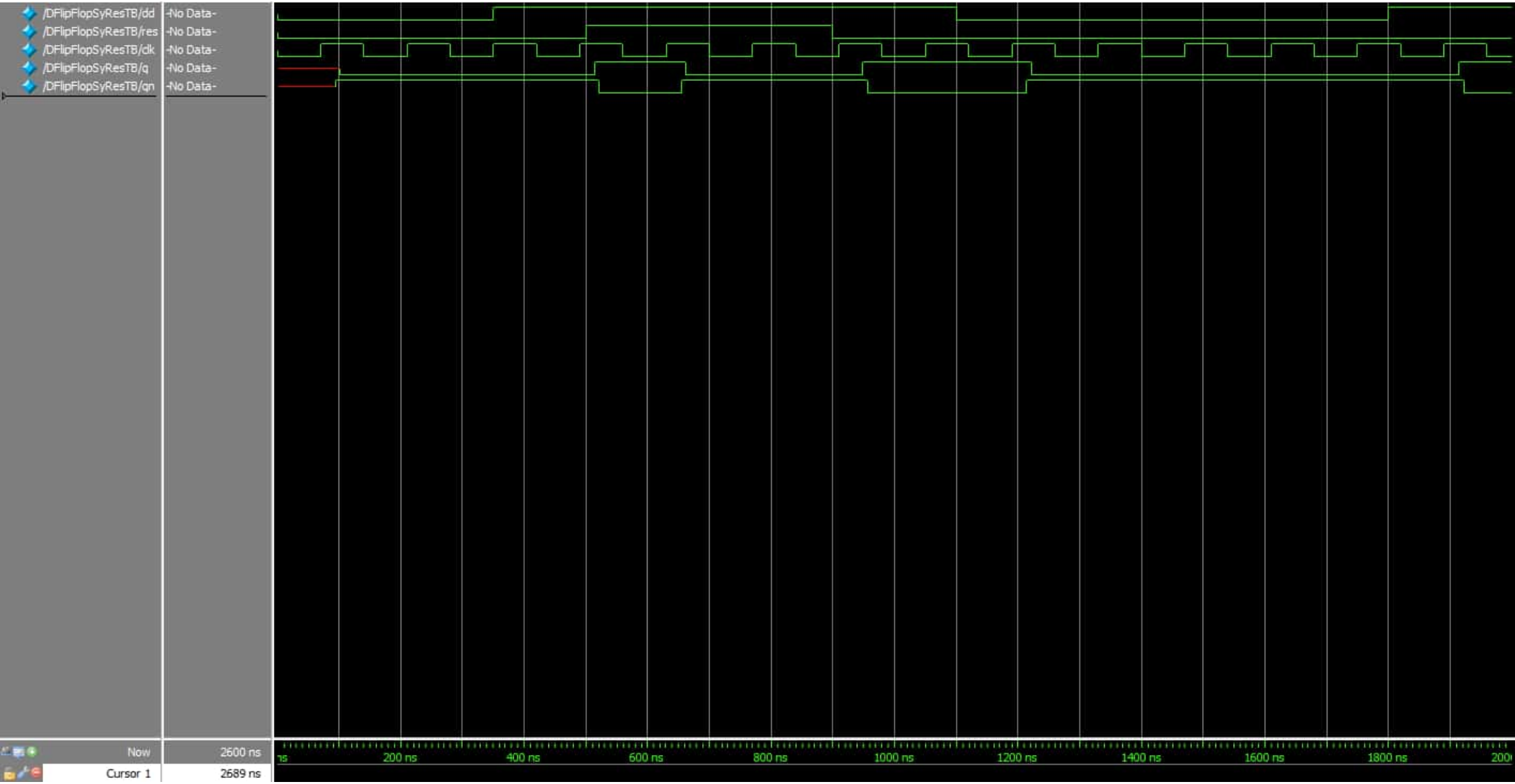


DFlipFlop synchronous Reset

```
`include "DLatchNI.v"
`timescale 1ns/1ns

module DFlipFlopSyRes (input D, res, CLK, output Q, QN);
    wire clk_inv, qm, qmn, d_res_and, res_inv;
    not(clk_inv, CLK);
    not(res_inv, res);
    and(d_res_and, D, res_inv);
    DLatchNI d_latch_1(d_res_and, CLK, qm, qmn);
    DLatchNI d_latch_2(qm, clk_inv, Q, QN);
endmodule
```

DFlipFlop synchronous Reset TB

```
`include "DFlipFlopSyRes.v"
`timescale 1ns/1ns

module DFlipFlopSyResTB ();
    reg dd = 0, res = 0, clk = 0;
    wire q, qn;
    DFlipFlopSyRes DFFSR(dd, res, clk, q, qn);
    always #70 clk = ~clk;
    initial begin
        #300
        #50 dd = 1;
        #150 res = 1;
        #400 res = 0;
        #200 dd = 0;
        #300
        repeat(5) #200 dd = $random;
        #200 $stop;
    end
endmodule
```



السؤال (8) :

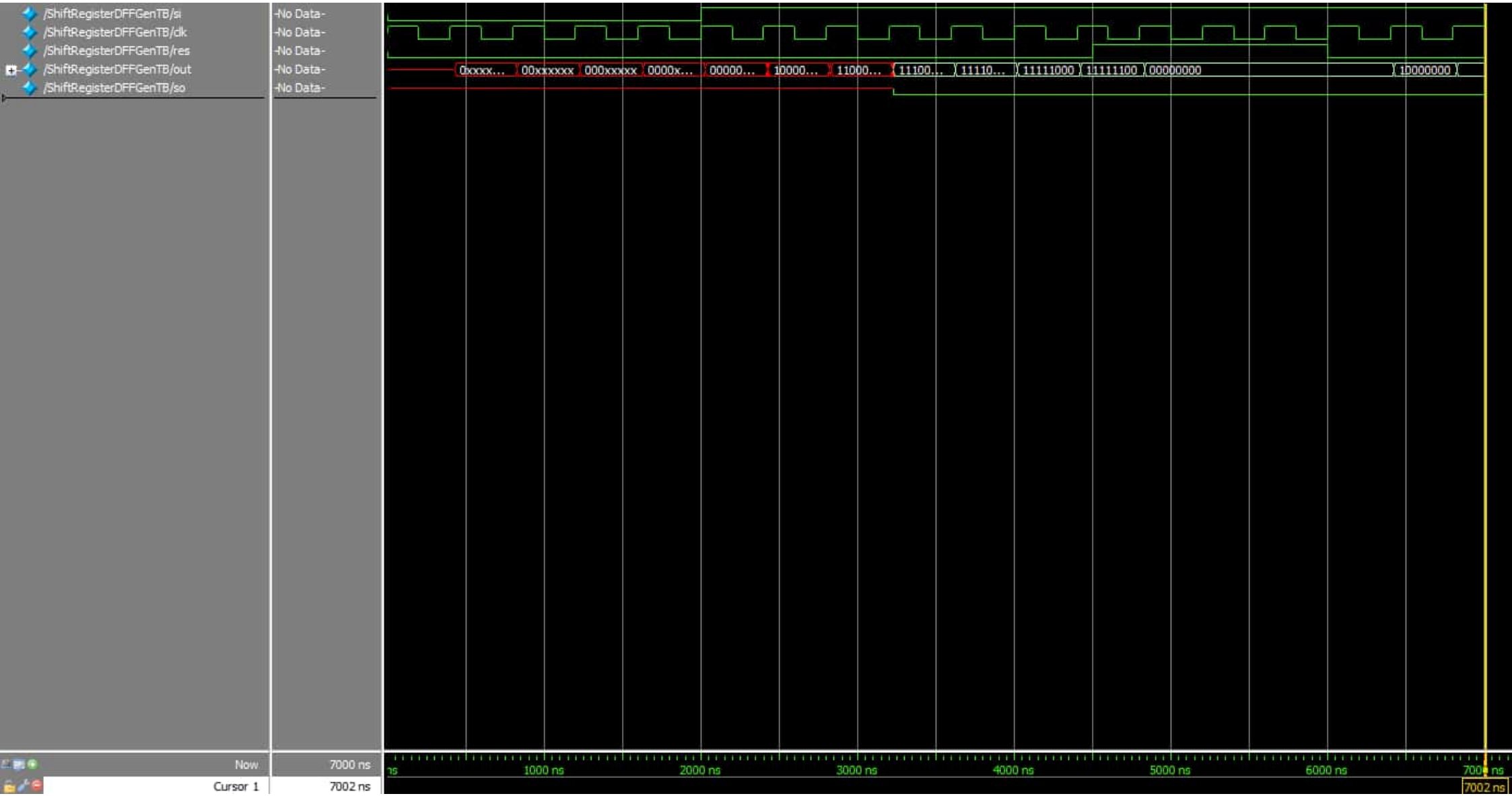
The difference between this section and the shift register we made in question 4 is that we used D-flipflop in this section and D-latch in question 4, and as explained in question 4, whenever the CLK is on. the values the inputs appears immediately on the output values, which is not to our liking, but in this section (due to Active Low) the output changes only in the posedge CLK.

8-Bit Shift Register D-FlipFlop Gen

```
`include "DFlipFlopSyRes.v"
`timescale 1ns/1ns


module ShiftRegisterDFFGen (input SI, res, CLK, output [7:0] Output, output SO);
    wire [7:0] QN;
    wire [8:0] out_temp;
    assign out_temp[8] = SI;
    genvar k;
    generate
        for (k = 7; k >= 0; k=k-1) begin
            DFlipFlopSyRes xx(out_temp[k + 1], res, CLK, out_temp[k], QN[k]);
        end
    endgenerate
    assign Output = out_temp[7:0];
    assign SO = Output[0];
endmodule
```

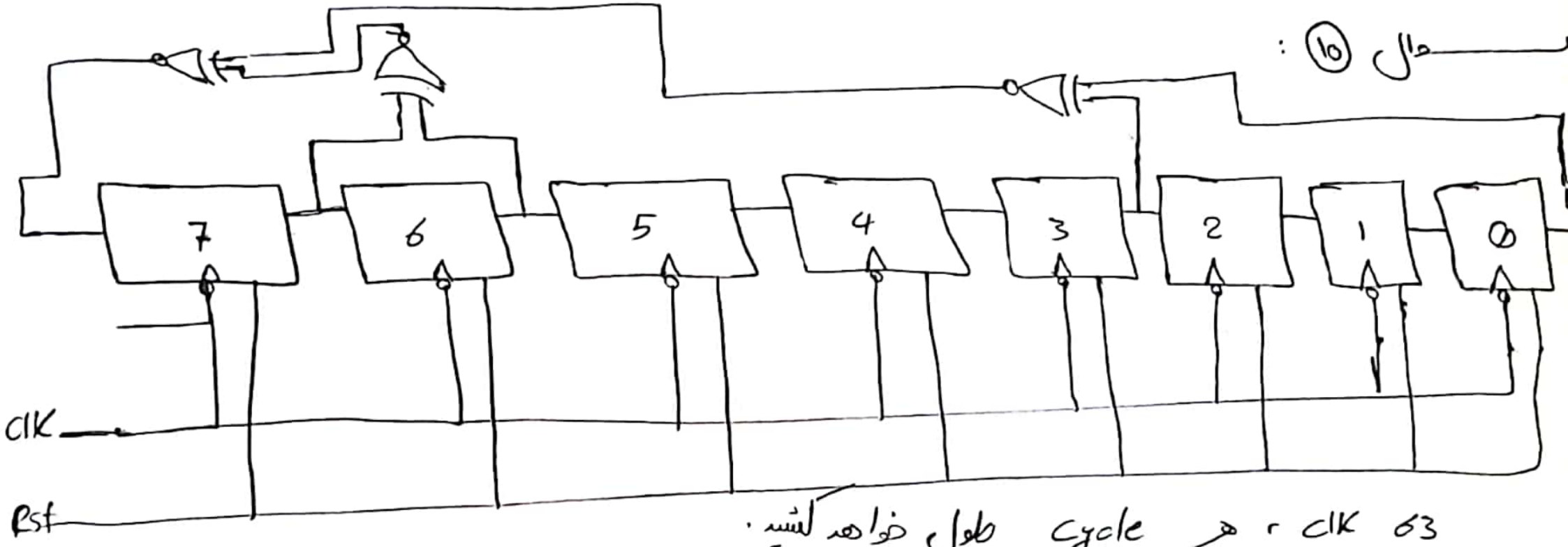
8-Bit Shift Register D-FlipFlop Gen TB

```
`include "8BitShiftRegisterDFFGen.v"
`timescale 1ns/1ns

module ShiftRegisterDFFGenTB ();
    reg si=0, clk=1, res=0;
    wire [7:0] out;
    wire so;
    ShiftRegisterDFFGen SRDFFG(si, res, clk, out, so);
    always #200 clk = ~clk;
    initial begin
        #1000
        repeat(4) #500 si = $random;
        #1000
        repeat(4) #500 res = $random;
        #1000 $stop;
    end
endmodule
```



8 Bit Shift Register D-FlipFlop AS

```
`timescale 1ns/1ns

module ShiftRegisterDFFAS (input SI, res, CLK, output reg [7:0] Output, output SO);
    always @(posedge CLK) begin
        if(res) Output <= 8'b0;
        else Output <= {SI, Output[7:1]};
    end
    assign SO = Output[0];
endmodule
```



حال ⑩ :

هر ۲ CLK 63 طول Cycle خواهد کشید.

LFSR

```
`include "8BitShiftRegisterDFFAS.v"
`timescale 1ns/1ns

module LFSR (input res, CLK, output [7:0] Output, output SO);
    wire coeff_1, coeff_2, si;
    xnor(coeff_1, Output[3], Output[0]);
    xnor(coeff_2, Output[7], Output[6]);
    xnor(si, coeff_1, coeff_2);
    ShiftRegisterDFFAS SRDFFAS(si, res, CLK, Output[7:0], SO);
endmodule
```

LFSR TB

```
`include "LFSR.v"
`timescale 1ns/1ns

module LFSRTB ();
    reg clk = 0, res = 1;
    wire [7:0] out;
    wire so;
    LFSR lfsr(res, clk, out, so);
    always #200 clk = ~clk;
    initial begin
        #2000
        #100 res = 0;
        #30000 $stop;
    end
endmodule
```