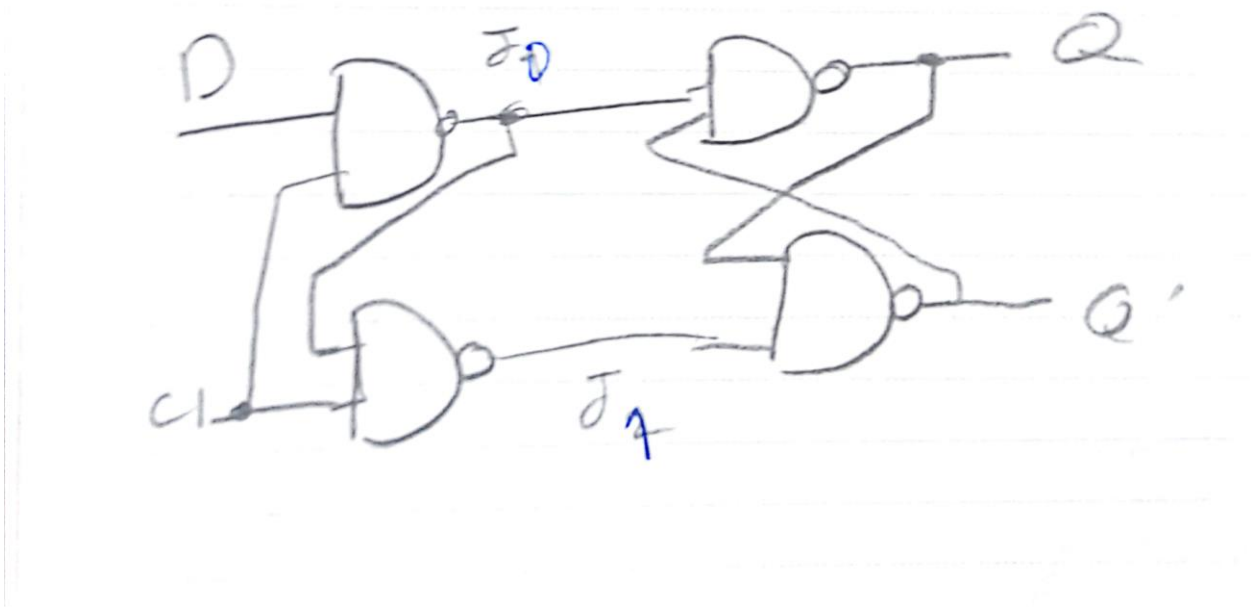
به نام خدا

محمد مشرقی

۸۱۰۱۹۹۴۹۲

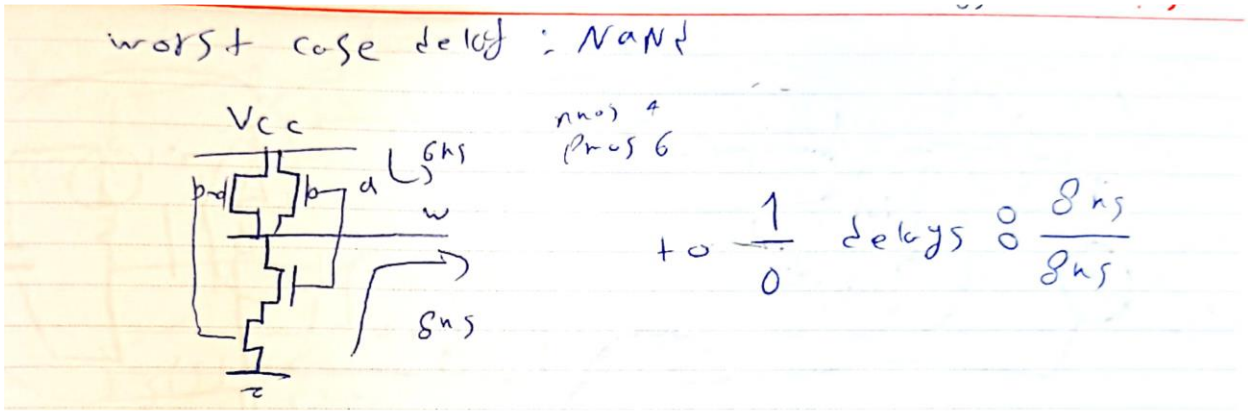
1-

```
module D_latch (input clc,D , output Q,Qb);
    wire j[1:0];
nand #(8,8) t1(j[0],D,clc), t2(j[1],j[0],clc) , t3(Q,Qb,j[0]), t4(Qb,Q,j[1]);
endmodule
```



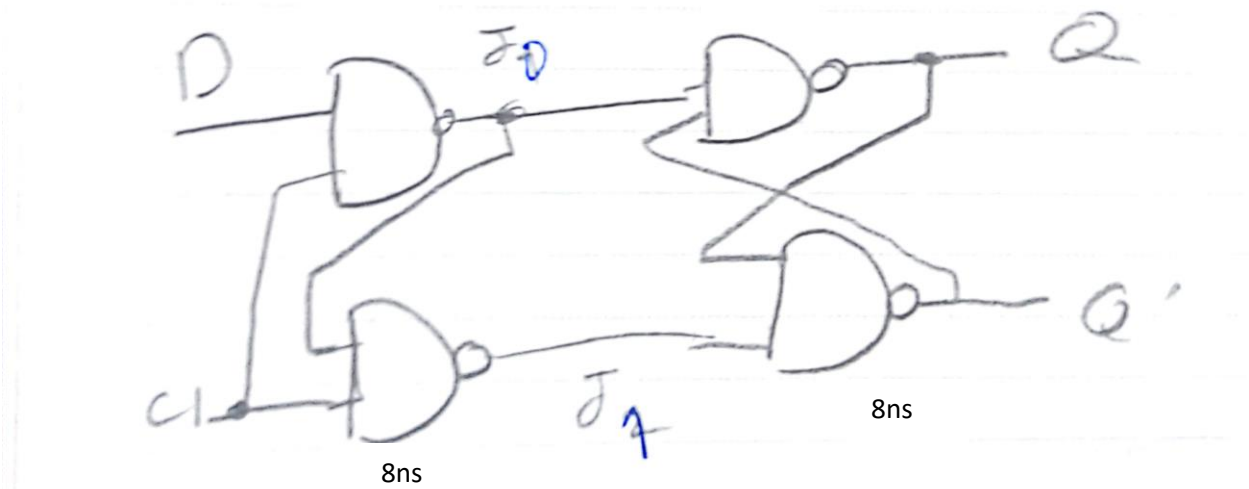
2-

با توجه به دیلی نند داریم:



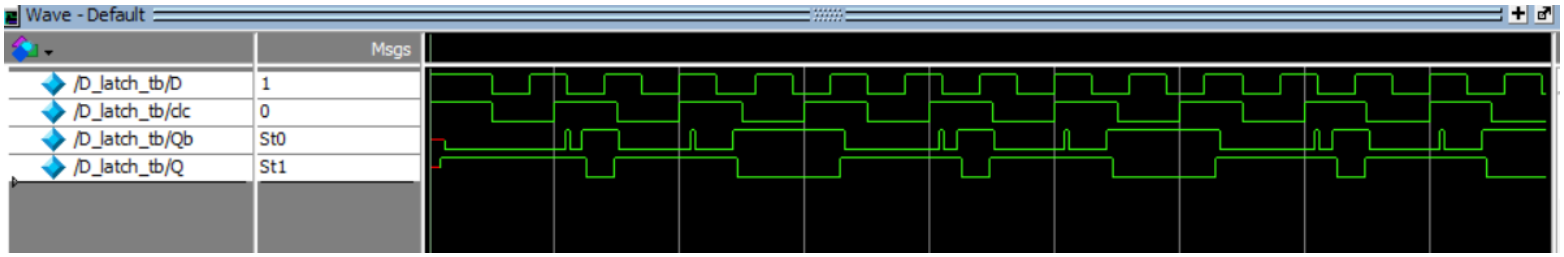
8ns

8ns



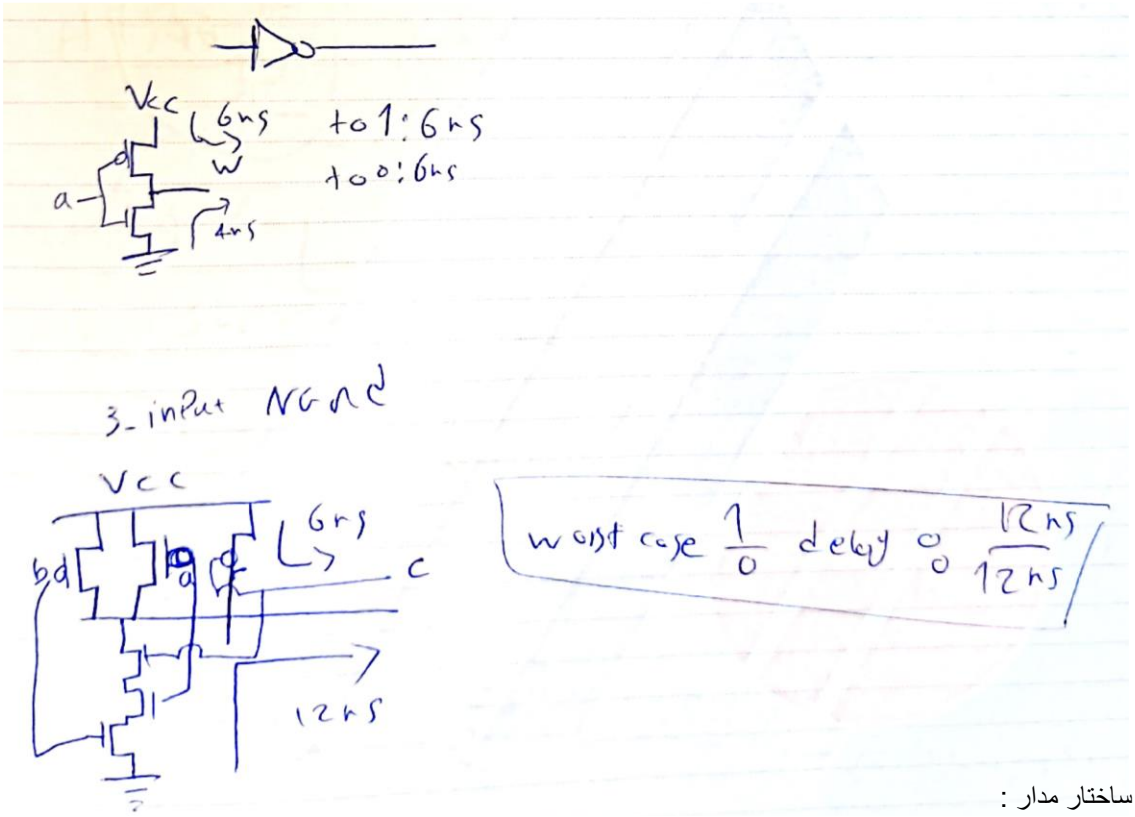
3-

```
`timescale 1ns/1ns
module D_latch();
logic D=1, clc=1 ;
wire Qb,Q;

D_latch cut1(clc , D , Q, Qb);

 always #100 clc=-clc;

 initial begin
    #40
    repeat(40) #60 D=-D;
    #100 $stop;
 end
endmodule
```

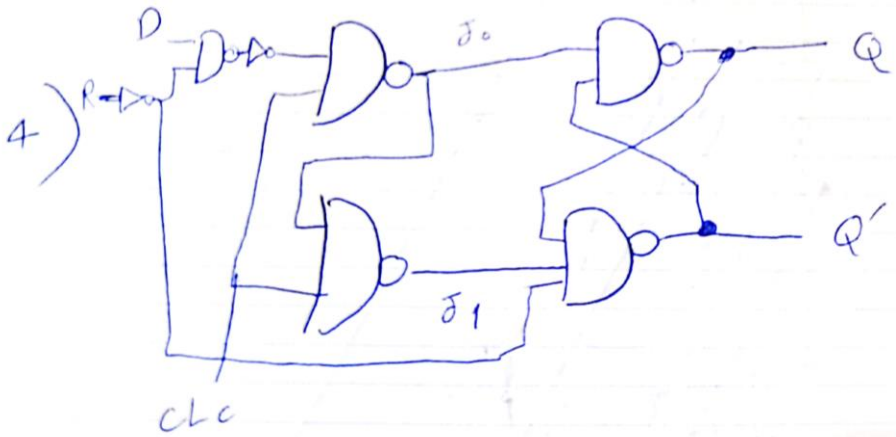


با توجه به شکل هر وقت D و clc یک شوند گلیچ رخ میده

4-

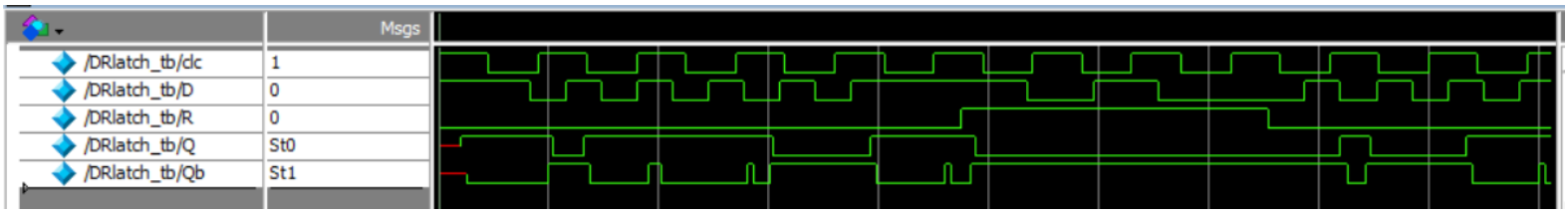


ساختار مدار :



```
`timescale 1ns/1ns
module DRlatch(input clc, D, R, output Q, Qb);
  wire  j[3: 0] ;  wire Rb;
    nand #(8,8) t1 (j[3], D, Rb);
    not # (6,6) n1 (j[2], j[3]);
    not # (6,6) n2 (Rb, R);
    nand # (12, 12) t2 (j [0], j[2], Rb, clc);
    nand #(8,8) t3 (j [1], j [0], clc);
    nand # (8,8) t4 (Q, Qb, j [0]);
    nand #(12,12) t5 (Qb, Q, j [1], Rb);
endmodule
```

```
`timescale 1ns/1ns
module DRlatch_tb ();
logic clc = 1,D = 1,R = 0;
wire Q, Qb;
DRlatch CUT (clc, D, R, Q, Qb);
always # 90 clc = ~clc;
initial begin
    #100
    repeat (10) # 65 D = ~D;
    # 200 R = 1;
    repeat (3)# 120 D = ~D;
    # 200 R = 0;
    repeat (10) # 65 D = ~D;
    # 200 $stop;
end endmodule
```

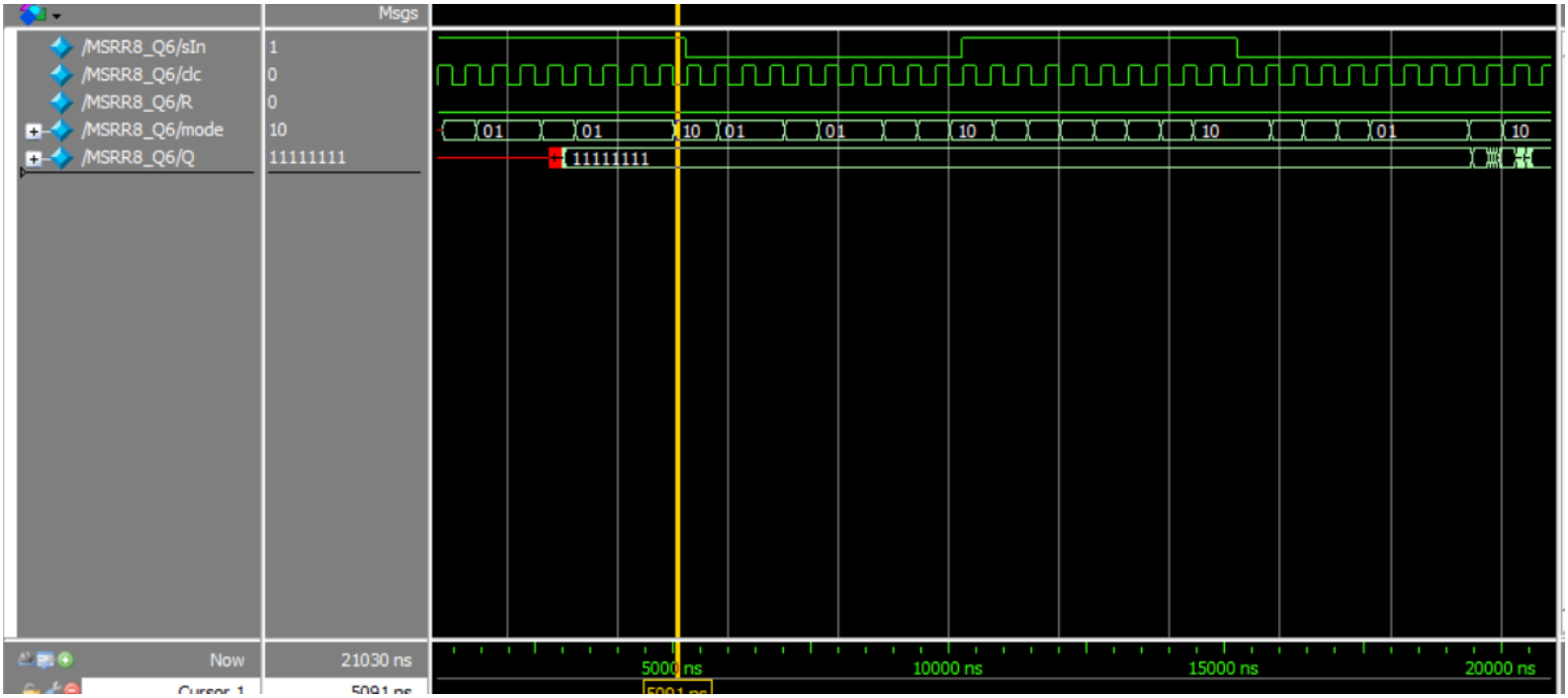


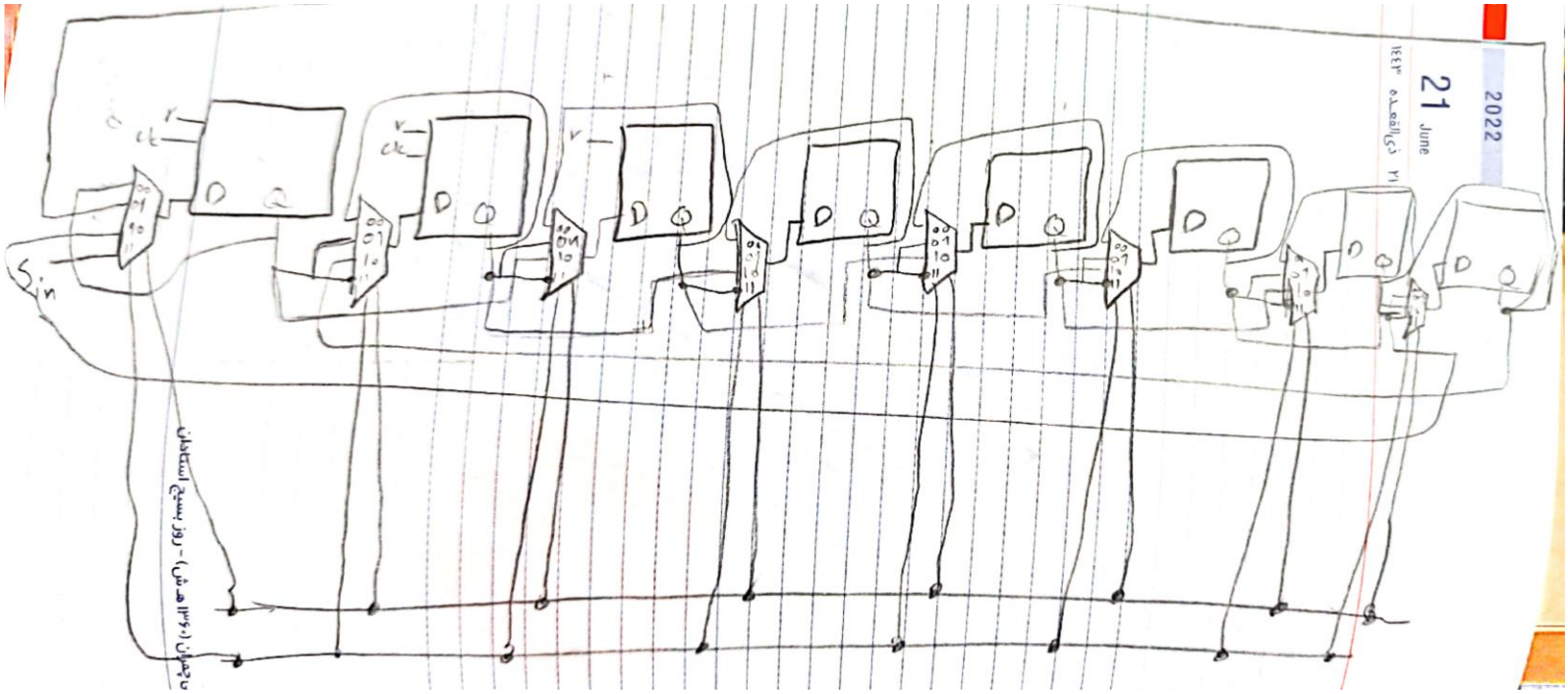
5-

```
`timescale 1ns/1ns
module multiplexer4 (input a0, a1, a2, a3, input [1: 0] s , output w);
 assign w = s == 2'b00 ?  a0:
            s == 2'b01 ?  a1:
            s == 2'b10 ?  a2:
            s == 2'b11 ?  a3: 1'bx;
endmodule

module MSRR8_Q5(input clc, R, sIn, input [1: 0] mode, output [7: 0] Q);
    wire [7: 0] muxin_00, muxin_01, muxin_10, muxin_11, muxop, Qb;
    genvar k;
    generate;
      for (k = 0; k <8; k = k + 1) begin
          multiplexer4 muxx (muxin_00[k], muxin_01[k], muxin_10[k], muxin_11[k], mode, muxop[k]);
          DRlatch DD (clc, muxop[k], R, Q[k], Qb[k]);

          assign muxin_00[k] = Q[k];

          assign muxin_01[k]= k == 7 ?  Q[0]: Q[k + 1];

          assign muxin_10[k] = k == 7 ?  Q[1]:
                               k == 6 ?  Q[0]: Q[k + 2];

          assign muxin_11[k] = k == 7 ?  sIn: Q [k + 1];
      end
    endgenerate
endmodule
```

6-

```
`timescale 1ns/1ns
module MSRR8_Q6();
 logic sIn=1 , clc=0 , R=0;
 logic [1:0]mode = 2'b11;
 wire[7:0]Q;
MSRR8_Q5  CUT(clc, R, sIn, mode, Q);
 always #160 clc = ~clc;

 initial begin
  #30
  #200 sIn=1;
  repeat(8) #200 mode = $random ;
  #200 sIn=0;
  repeat(8) #200 mode = $random ;
    #200 sIn=1;
  repeat(8) #200 mode = $random ;
  #200 sIn=0;
  repeat(8) #200 mode = $random ;
  #1000 $stop;
 end


endmodule
```


Msgs
/MSRR8_Q6/sIn
/MSRR8_Q6/clc
/MSRR8_Q6/R
/MSRR8_Q6/mode
/MSRR8_Q6/Q
1
0
0
10
11111111
01
01
10
01
01
10
10
01
10
11111111
Now
21030 ns
5000 ns
10000 ns
15000 ns
20000 ns

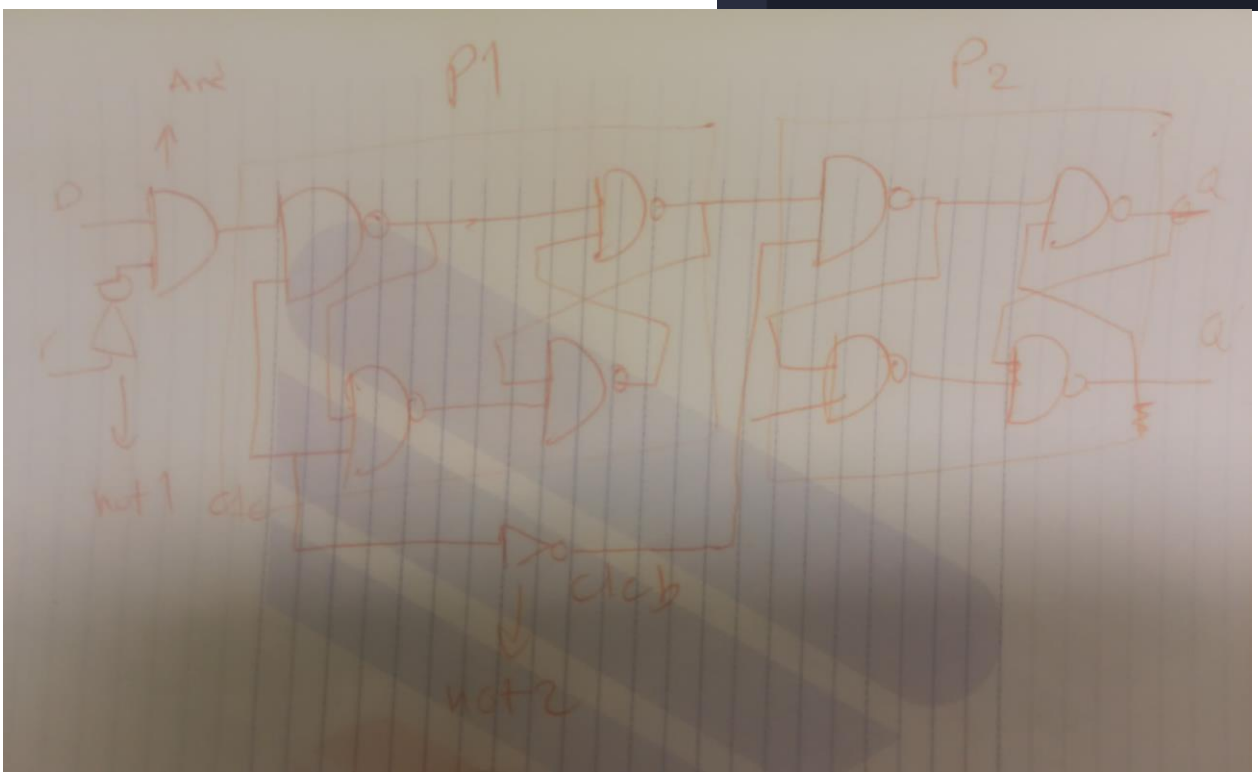

2022
21 June

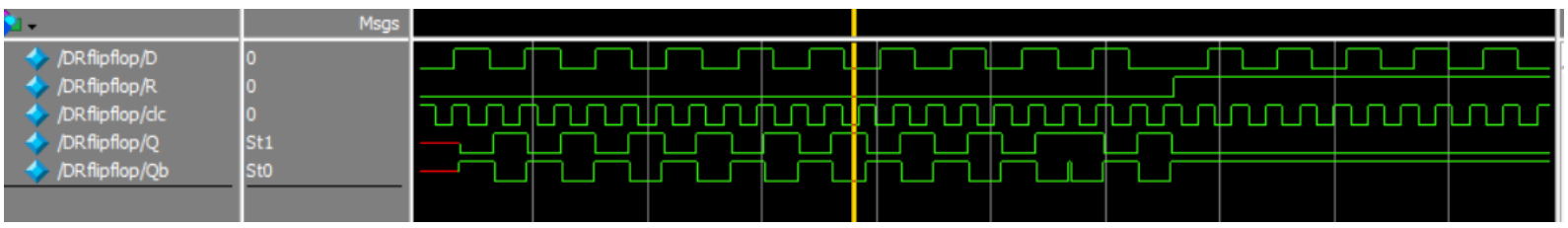
مدار طبق انتظار کار نمی کند چون وقتی فلیپ فلاپ در مدار استفاده نمی کنیم هنگامی که ورودی را تغییر می دهیم چون به خروجی وصل است رو ان نیز تاثیر گذار است و باعث تغییر ان می شود تا و این تا زمانی اتفاق می افتاد که clc یک هست و وقتی هم که صفر شود اتفاقی نمی افتد.

7-

```
`timescale 1ns/1ns
module  Dfilpflop(input clc,D, R, output Q, Qb);
wire Rb, Dj, Q1, Qb1, clcbar;
not #6 n1(Rb, R);
not #6 n2 (clcb, clc);
and #14 A1 (Dj, Rb, D);
D_latch p1(clc, Dj,  Q1, Qb1);
D_latch p2(clcbar, Q1,  Q, Qb);
endmodule
```

```
`timescale 1ns/1ns
module DRflipflop ();
logic D = 0, R= 0 , clc = 1;
wire Q, Qb;
Dfilpflop cut(clc, D, R, Q, Qb);
always # 74 clc = ~clc;
initial begin
repeat (20) # 155 D = ~ D;
# 200 R = 1;
repeat (20) # 150 D = ~D;

# 200 R = 0;
repeat (20) # 155 D = ~ D;
# 300 $stop;
end
endmodule
```

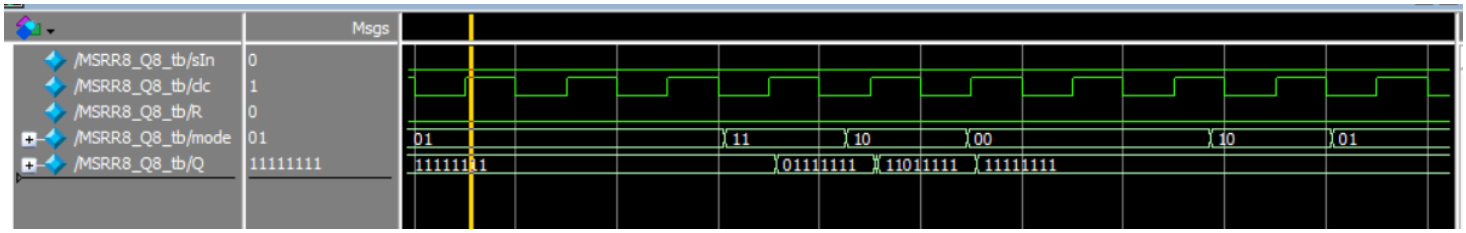


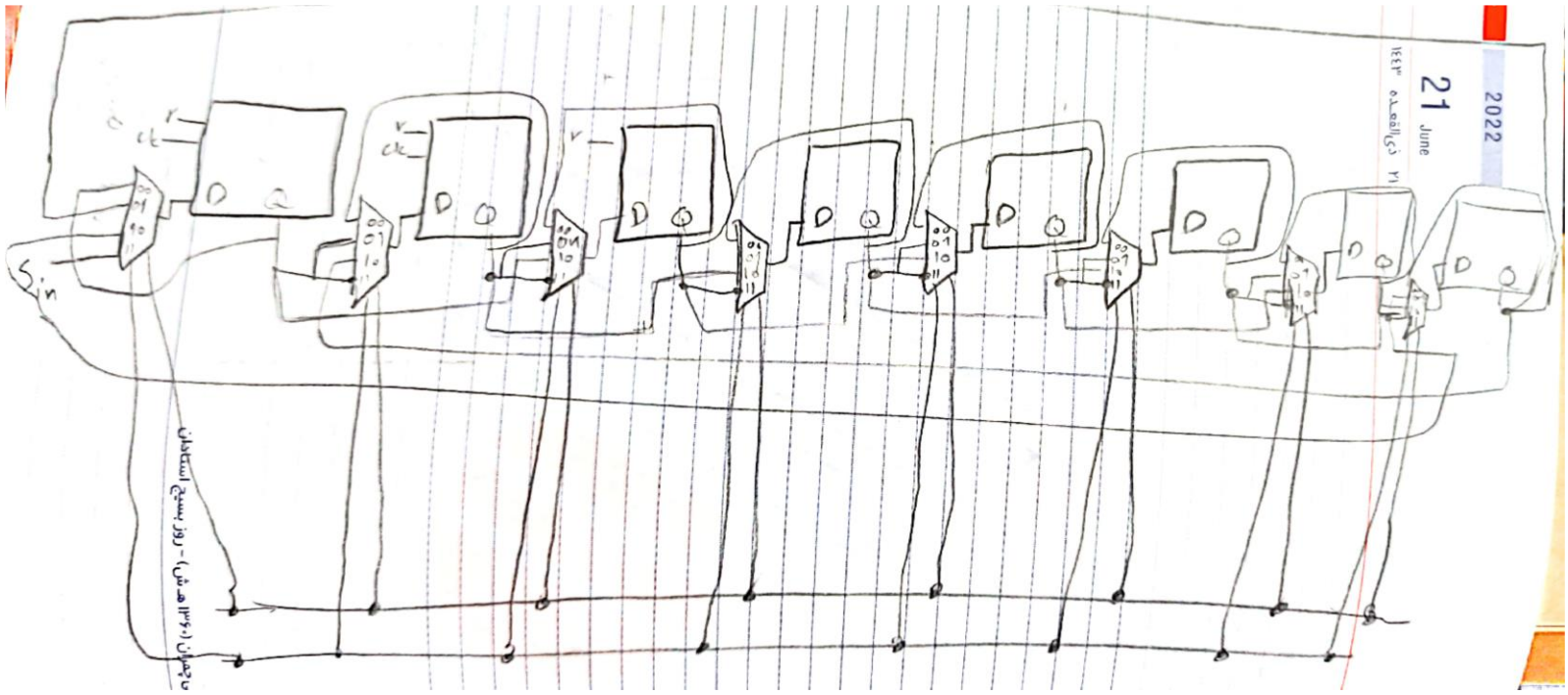


8-

```
`timescale 1ns/1ns
module multiplexer4 (input a0, a1, a2, a3, input [1: 0] s , output w);
 assign w = s == 2'b00 ?  a0:
            s == 2'b01 ?  a1:
            s == 2'b10 ?  a2:
            s == 2'b11 ?  a3: 1'bx;
endmodule

module MSRR8_Q8(input clc, R, sIn, input [1: 0] mode, output [7: 0] Q);
    wire [7: 0] muxin_00, muxin_01, muxin_10, muxin_11, muxop, Qb;
    genvar k;
    generate;
      for (k = 0; k <8; k = k + 1) begin
           multiplexer4 muxx (muxin_00[k], muxin_01[k], muxin_10[k], muxin_11[k], mode, muxop[k]);
           Dfilpflop DD (clc, muxop[k], R, Q[k], Qb[k]);

           assign muxin_00[k] = Q[k];

           assign muxin_01[k]= k == 7 ?  Q[0]: Q[k + 1];

           assign muxin_10[k] = k == 7 ?  Q[1]:
                                k == 6 ?  Q[0]: Q[k + 2];

           assign muxin_11[k] = k == 7 ?  sIn: Q [k + 1];
      end
    endgenerate
endmodule
```

```
`timescale 1ns/1ns
module MSRR8_Q8_tb();
 logic sIn=1 , clc=0 , R=0;
 logic [1:0]mode;
 wire[7:0]Q;
MSRR8_Q5  CUT(clc, R, sIn, mode, Q);
 always #250 clc = ~clc;
 initial begin
  #30
  #200 sIn=1;
  repeat(8) #600 mode = $random ;
  #200 sIn=0;
  repeat(8) #600 mode = $random ;
    #200 sIn=1;
  repeat(8) #600 mode = $random ;
  #200 sIn=0;
  repeat(8) #600 mode = $random ;
  #1000 $stop;
 end
endmodule
```


Msgs
/MSRR8_Q8_tb/sIn
/MSRR8_Q8_tb/dc
/MSRR8_Q8_tb/R
/MSRR8_Q8_tb/mode
/MSRR8_Q8_tb/Q
0
1
0
01
11111111
01
11
10
00
10
01
11111111
01111111
11011111
11111111


2022
21 June

9-

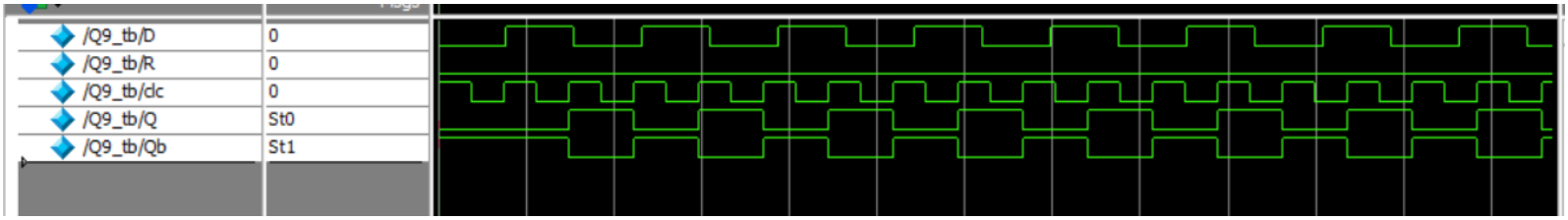
```
`timescale 1ns/1ns
module dff_sync_rst (input R, clk, d,  output reg Q,output  Qb);
    always @(negedge clk)
     begin
          if (R) Q <= 1'b0;
          else Q <= d;
     end
   assign Qb = ~ Q;
endmodule
```

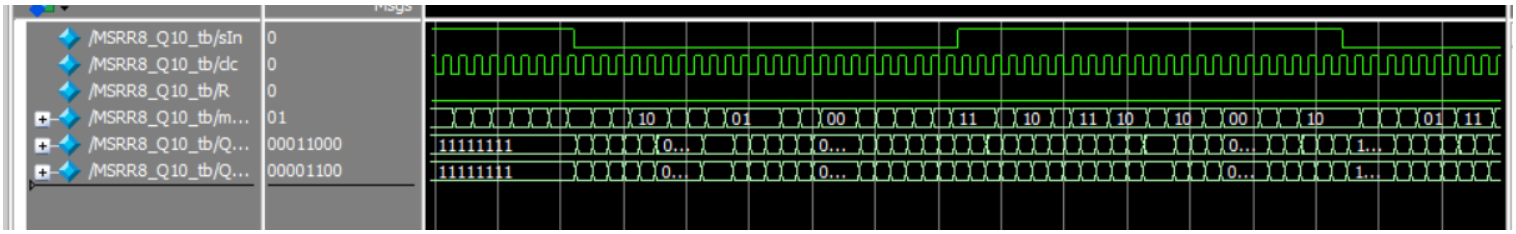
```
`timescale 1ns/1ns
module Q9_tb();
logic D = 0, R= 0 , clc = 1;
wire Q, Qb;
dff_sync_rst cut( R, clc, D , Q, Qb);
always # 74 clc = ~clc;
initial begin
repeat (20) # 155 D = ~ D;
# 200 R = 1;
repeat (20) # 150 D = ~D;

# 200 R = 0;
repeat (20) # 155 D = ~ D;
# 300 $stop;
end
endmodule
```

10-

```
module MSRR8_Q10_tb();
 logic sIn=1 , clc=0 , R=0;
 logic [1:0]mode;
 wire[7:0]Q_gate , Q_assign;
MSRR8_Q8  CUT1(clc, R, sIn, mode, Q_gate);
MSRR8_Q10  CUT2(clc, R, sIn, mode, Q_assign);
 always #250 clc = ~clc;
 initial begin
  #30
  #200 sIn=1;
  repeat(10) #600 mode = 11 ;

  repeat(20) #600 mode = $random ;
  #200 sIn=0;
  repeat(20) #600 mode = $random ;
    #200 sIn=1;
  repeat(20) #600 mode = $random ;
  #200 sIn=0;
  repeat(20) #600 mode = $random ;
  #1000 $stop;
 end
endmodule
```

```
`timescale 1ns/1ns
module multiplexer4 (input a0, a1, a2, a3, input [1: 0] s , output w);
 assign w = s == 2'b00 ?  a0:
            s == 2'b01 ?  a1:
            s == 2'b10 ?  a2:
            s == 2'b11 ?  a3: 1'bx;
endmodule

module MSRR8_Q10(input clc, R, sIn, input [1: 0] mode, output [7: 0] Q);
    wire [7: 0] muxin_00, muxin_01, muxin_10, muxin_11, muxop, Qb;
    genvar k;
    generate;
      for (k = 0; k <8; k = k + 1) begin
           multiplexer4 muxx (muxin_00[k], muxin_01[k], muxin_10[k], muxin_11[k], mode, muxop[k]);
           dff_sync_rst DD (R,clc, muxop[k],  Q[k], Qb[k]);

           assign muxin_00[k] = Q[k];

           assign muxin_01[k]= k == 7 ?  Q[0]: Q[k + 1];

           assign muxin_10[k] = k == 7 ?  Q[1]:
                                k == 6 ?  Q[0]: Q[k + 2];

           assign muxin_11[k] = k == 7 ?  sIn: Q [k + 1];
      end
    endgenerate
endmodule
```





در اینجا چون به دلیل استفاده از هاردور باعث می شود دیلی داشته باشیم اما در always دیلی نداریم و اینکه در ساختار گیت میتوان ایکس داشت برخلاف always

دلیل بعدی هم اینکه در ساختار گیت باید اول کلاک یک شود و به ان فرصت داده شود و سپس کلاک صفر شود و بعد از زمانی خروجی تغییر می کنه اما always با negedge به صورت سریع تغییر می کنه(اگر در test bench زمان ها را کم کنیم احتمال اینکه دو خروجی یکی نباشد هست)