



به نام خدا  
دانشگاه تهران  
پردیس دانشکده‌های فنی  
دانشکده مهندسی برق و کامپیوتر



## سیستم‌های دیجیتال 1

--- ECE 894 ---

نیمسال دوم (99-00)

استاد: پروفسور نوابی

## COMPUTER ASSIGNMENT 4

LATCHES, FLIP-FLOPS, AND A LITTLE BEYOND

محمد مهدی عبدالحسینی

810 198 434



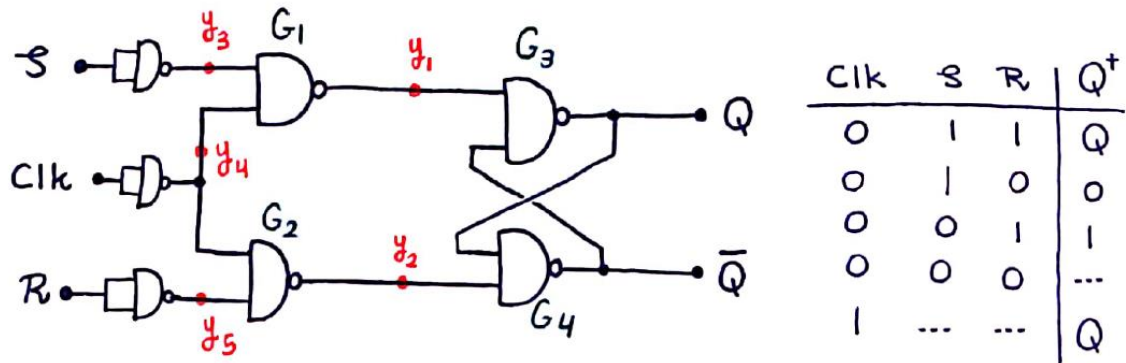
D I G I T A L   S Y S T E M S

## فهرست مطالب

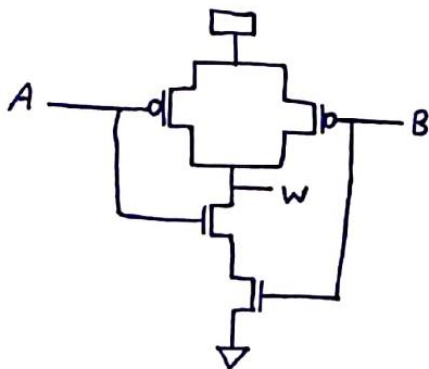
1 .....	بخش اول: SR-Latch
2 .....	بخش دوم: شیه سازی بخش اول
3 .....	بخش سوم: D-Latch
4 .....	بخش چهارم: 8-Bit Shift Register
5 .....	بخش پنجم: شیه سازی بخش چهارم
6 .....	بخش ششم: Master-Slave D-type Flip-Flop
7 .....	بخش هفتم: Synchronous Reset Flip-Flop
8 .....	بخش هشتم: 8-Bit Shift Register FF-Type
9 .....	بخش نهم: 8-Bit Shift Register
10 .....	بخش دهم: LFSR

## بخش اول : SR-Latch

My Clocked SR-Latch With Active Low S, R, And Clock Inputs : (All-NAND)



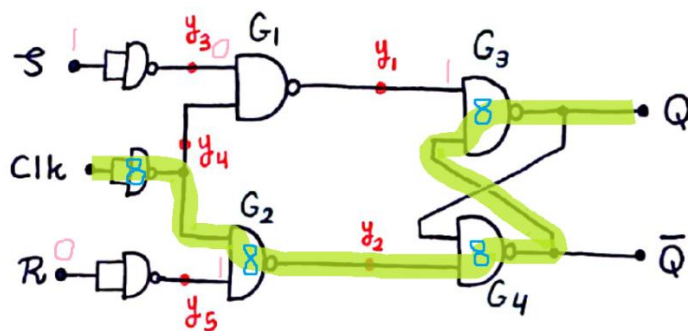
Switch level delays : nmos #4 ; pmos #6



$$\left\{ \begin{array}{l} t_{01} : 4 + 4 = 8 \\ t_{10} : 4 + 4 = 8 \end{array} \right\} \rightarrow \text{nand \#8}$$

$$\rightarrow \text{Worst Case Delay} : 8 + 8 + 8 = 24$$

Actually The Worst Case Delay Is #32



And Here's My Code :

```

1 module Part1_mySRLatchClk (input S, R, Clk, output logic Q);
2     wire [1:5] y;
3     wire QBar;
4     nand #8 G1(y[1],y[3],y[4]);
5     nand #8 G2(y[2],y[4],y[5]);
6     nand #8 G3(Q,y[1],QBar);
7     nand #8 G4(QBar,y[2],Q);
8     nand #8 N1(y[3],S,S);
9     nand #8 N2(y[4],Clk,Clk);
10    nand #8 N3(y[5],R,R);
11 endmodule

```

## بخش دوم: شبیه سازی بخش اول

The Simulation of Part One :

```

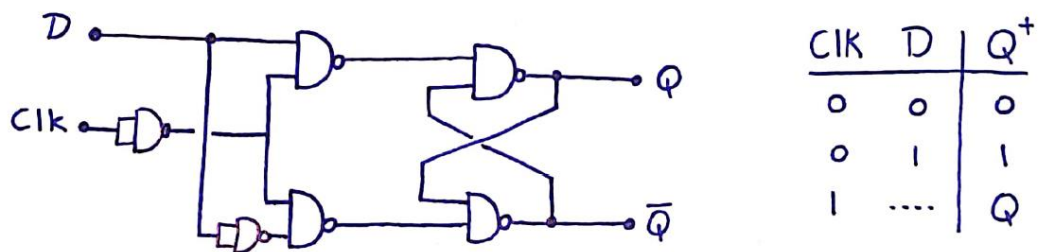
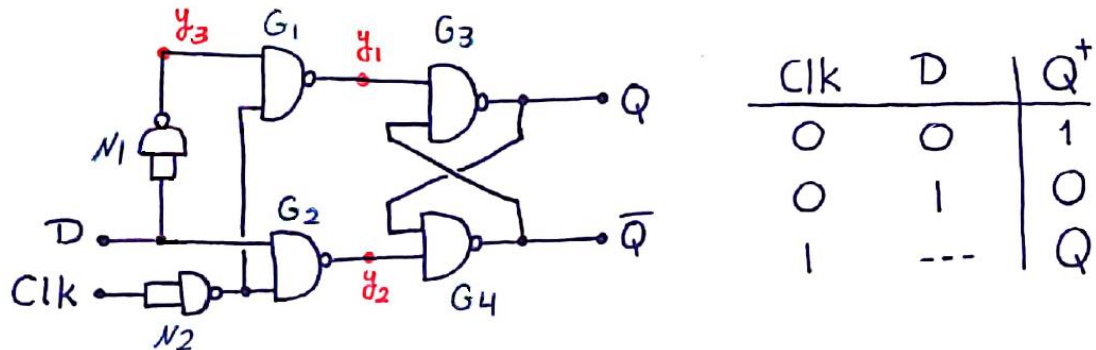
1 module Part2_mySRLatchClk_TB ();
2     logic SS = 1;
3     logic RR = 0;
4     logic CLK = 0;
5     wire QQ;
6     Part1_mySRLatchClk G1 (SS,RR,CLK,QQ);
7     always #50 CLK = ~CLK;
8     initial begin
9         #100 SS = 1;
10        #100 RR = 1;
11        #100 SS = 0;
12        #100 SS = 1;
13        #100 RR = 0;
14        #100 SS = 0;
15        #100 SS = 1;
16        #200 $stop;
17    end
18 endmodule

```

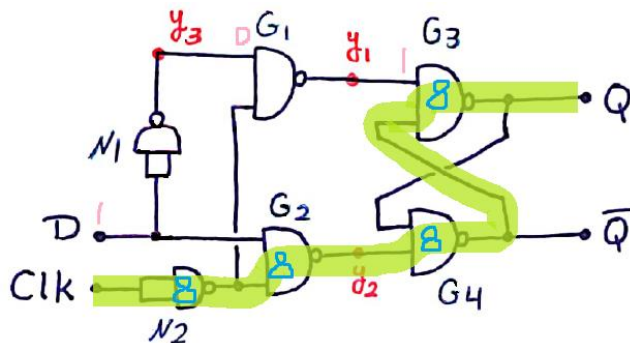


## بخش سوم : D-Latch

Converting The SR Latch Of Part 1 To A Clocked D-Latch :



The Worst Case Delay Is #32



My Code For Clocked D-Latch :

```

1 module Part3_myDLatchClk (input D, Clk, output logic Q);
2     wire [1:4] y;
3     wire QBar;
4     nand #8 G1(y[1], D, y[4]);
5     nand #8 G2(y[2], y[4], y[3]);
6     nand #8 G3(Q, y[1], QBar);
7     nand #8 G4(QBar, y[2], Q);
8     nand #8 N1(y[3], D, D);
9     nand #8 N2(y[4], Clk, Clk);
10 endmodule

```

The TestBench To Verify Its Operation :

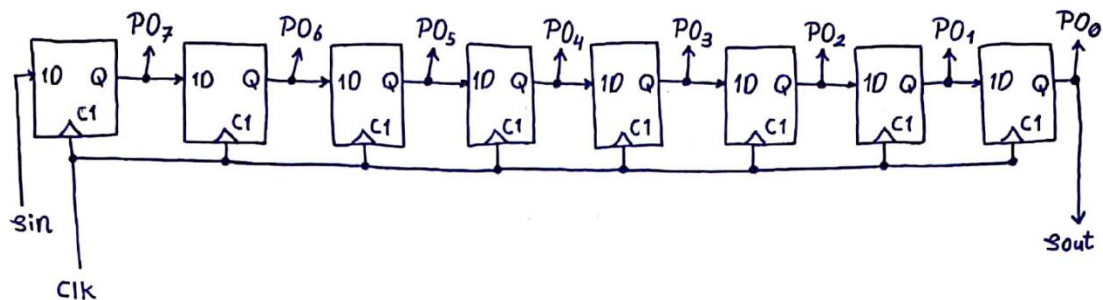
```

14 module Part3_myDLatchClk_TB ();
15     logic DD = 1;
16     logic CLK = 0;
17     wire QQ;
18     Part3_myDLatchClk G1 (DD,CLK,QQ);
19     always #50 CLK = ~CLK;
20     initial begin
21         #100 DD = 0;
22         #100 DD = 1;
23         #100 DD = 0;
24         #100 DD = 1;
25         #200 $stop;
26     end
27 endmodule

```

### بخش چهارم : 8-Bit Shift Register

8-Bit Shift Register Using D-Latch Of Part 3 :



My Code For D-Type 8-Bit Shift Register :

```

1 module Part4_my8BitShiftRegister (input Pin, input Clk, output logic [7:0] Pout);
2     wire [8:0] y;
3     assign y[8] = Pin;
4     genvar k;
5     generate
6         for (k = 7; k >= 0; k = k - 1) begin: latch
7             Part3_myDLatchClk DL (y[k+1], Clk, y[k]);
8         end
9         for (k = 7; k >= 0; k = k - 1) begin: outputs
10             assign Pout[k] = y[k];
11         end
12     endgenerate
13 endmodule

```


## بخش پنجم: شبیه سازی بخش چهارم

The Simulation of Part Four :

```

1  module Part5_my8BitShiftRegister_TB ();
2      logic Pin;
3      logic [7:0] A = 8'b00100111;
4      logic CLK = 0;
5      wire [7:0] Pout;
6      Part4_my8BitShiftRegister G1 (Pin,CLK,Pout);
7      always #50 CLK = ~CLK;
8      initial begin
9          #100 Pin = A[0];
10         #100 Pin = A[1];
11         #100 Pin = A[2];
12         #100 Pin = A[3];
13         #100 Pin = A[4];
14         #100 Pin = A[5];
15         #100 Pin = A[6];
16         #100 Pin = A[7];
17         #200 $stop;
18     end
19 endmodule

```



همانگونه که انتظار داشتیم مدار به درستی کار نکرد. در واقع بدلیل کم بودن بازه هر سیکل نسبت به دلیلی قطعه‌مان ، در هر سیکل هر چقدر که دلیلی اجازه دهد دیوایس کار میکند و خروجی میدهد. که باعث بوجود آمدن خطا در عملکرد سیستم میشود.

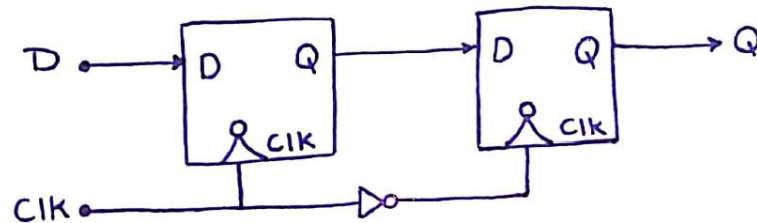
بطور مثال اگر در یک سیکل بعنوان ورودی عدد یک را بدهیم انتظار داریم همین مقدار را در خروجی ببینیم اما در عمل به محض شروع سیکل ، دیوایس تمایل دارد تا پایان سیکل ، همین مقدار را مدام به خروجی تحویل دهد. برای همین مثلا ما عدد

0010011 را به سیستم میدهیم اما عدد 11111111 در خروجی ظاهر میشود :



## بخش ششم : Master-Slave D-type Flip-Flop

Master-Slave D-Type Flip-Flop Using D-Latch Of Part 3 :



My Code Master-Slave D-Type Flip-Flop :

```

1 module Part6_myMasterSlaveDFlipFlop (input D, input Clk, output logic Q);
2     wire y;
3     Part3_myDLatchClk DL1 (D, Clk, y);
4     Part3_myDLatchClk DL2 (y, ~Clk, Q);
5 endmodule
6
7 module Part6_TB ();
8     logic DD = 1;
9     logic CLK = 0;
10    wire QQ;
11    Part6_myMasterSlaveDFlipFlop G1 (DD, CLK, QQ);
12    always #50 CLK = ~CLK;
13    initial begin
14        #100 DD = 0;
15        #100 DD = 1;
16        #100 DD = 0;
17        #100 DD = 1;
18        #200 $stop;
19    end
20 endmodule

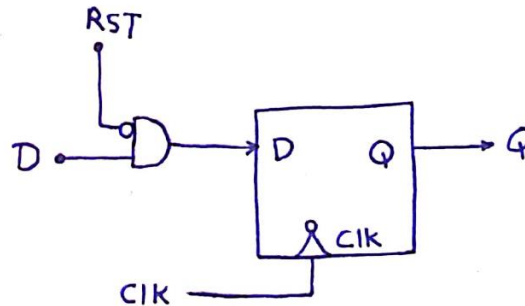
```





## بخش هفتم : Synchronous Reset Flip-Flop

Synchronous Reset (*RST*) Using The Flip-Flop Of Part 6 :

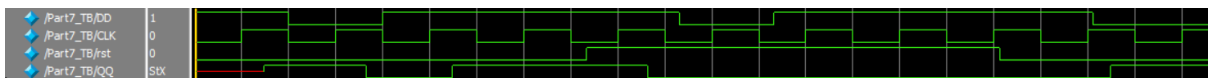


My Code For This Part :

```

1 module Part7_myMasterSlaveDFlipFlop (input D, Clk, RST, output logic Q);
2     wire y;
3     assign y = D & ~RST;
4     Part6_myMasterSlaveDFlipFlop G1 (y, Clk, Q);
5 endmodule
6
7 module Part7_TB ();
8     logic DD = 1;
9     logic CLK = 0;
10    logic rst = 0;
11    wire QQ;
12    Part7_myMasterSlaveDFlipFlop G1 (DD, CLK, rst, QQ);
13    always #50 CLK = ~CLK;
14    initial begin
15        #100 DD = 0;
16        #100 DD = 1;
17        #216 rst = 1;
18        #100 DD = 0;
19        #100 DD = 1;
20        #240 rst = 0;
21        #100 DD = 0;
22        #200 rst = 1;
23        #200 $stop;
24    end
25 endmodule

```



بخش هشتم : 8-Bit Shift Register FF-Type

## 8-Bit Shift Register Using The Flip-Flop Of Part 7 :

```

1  module Part8_my8BitShiftRegister (input Pin,Clk,RST, output logic [7:0] Pout);
2      wire [8:0] y;
3      assign y[8] = Pin;
4      genvar k;
5      generate
6          for (k = 7; k >= 0; k = k - 1) begin: latch
7              Part7_myMasterSlaveDFlipFlop DL (y[k+1], Clk, RST, y[k]);
8          end
9          for (k = 7; k >= 0; k = k - 1) begin: outputs
10             assign Pout[k] = y[k];
11         end
12     endgenerate
13 endmodule

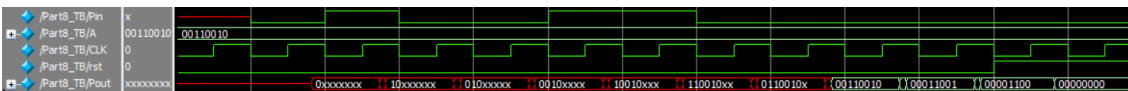
```

## TestBench :

```

17 module Part8_TB ();
18     logic Pin;
19     logic [7:0] A = 8'b00110010;
20     logic CLK = 0;
21     logic rst = 0;
22     wire [7:0] Pout;
23     Part8_my8BitShiftRegister G1 (Pin,CLK,rst,Pout);
24     always #50 CLK = ~CLK;
25     initial begin
26         #100 Pin = A[0];
27         #100 Pin = A[1];
28         #100 Pin = A[2];
29         #100 Pin = A[3];
30         #100 Pin = A[4];
31         #100 Pin = A[5];
32         #100 Pin = A[6];
33         #100 Pin = A[7];
34         #100 rst = 1;
35         #200 $stop;
36     end
37 endmodule

```



## بخش نهم : 8-Bit Shift Register

### 8-Bit Shift-Register Using An Always Statement :

```

1 module Part9_my8BitShiftRegister (input Pin, Clk, RST, output logic [7:0] Pout);
2   always @(posedge Clk,posedge RST)begin
3     if (RST)
4       Pout <= 8'd0;
5     else
6       Pout<={Pin,Pout[7:1]};
7   end
8 endmodule

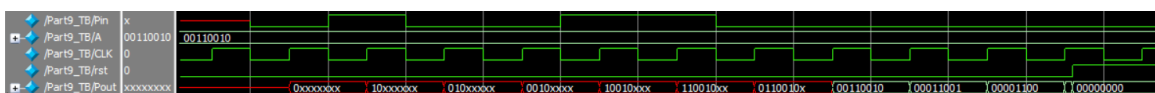
```

### TestBench :

```

11 module Part9_TB ();
12     logic Pin;
13     logic [7:0] A = 8'b00110010;
14     logic CLK = 0;
15     logic rst = 0;
16     wire [7:0] Pout;
17     Part9_my8BitShiftRegister G1 (Pin,CLK,rst,Pout);
18     always #50 CLK = ~CLK;
19     initial begin
20         #100 Pin = A[0];
21         #100 Pin = A[1];
22         #100 Pin = A[2];
23         #100 Pin = A[3];
24         #100 Pin = A[4];
25         #100 Pin = A[5];
26         #100 Pin = A[6];
27         #100 Pin = A[7];
28         #360 rst = 1;
29         #200 $stop;
30     end
31 endmodule

```

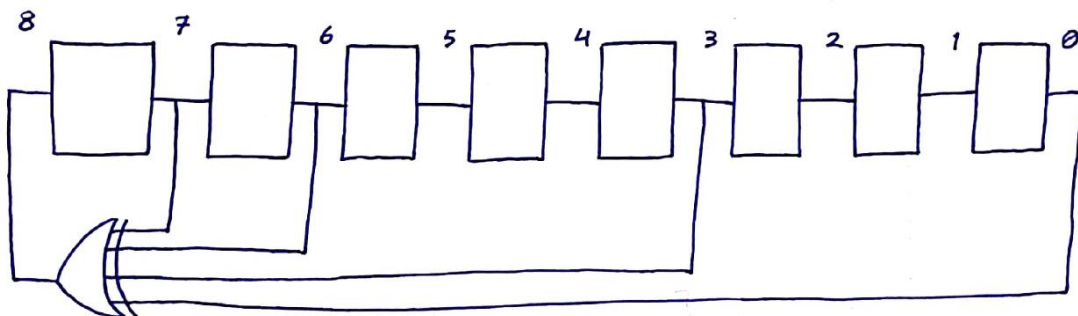
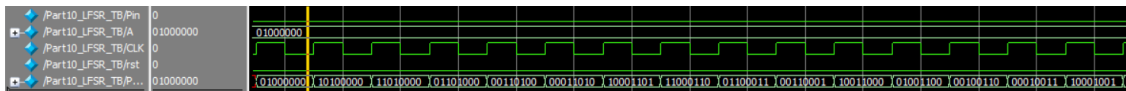


## بخش دهم : LFSR

```

1 module Part10_LFSR (input Pin, Clk, RST, output logic [7:0] Pout);
2     wire y;
3     int i = 0;
4     always @(*)begin
5         if (Pout[0] == 0 || Pout[0] == 1)
6             i = 1;
7     end
8     assign y = i? (Pout[7]^Pout[6]^Pout[3]^Pout[0]) : Pin;
9     Part9_my8BitShiftRegister SR (y, Clk, RST, Pout);
10 endmodule
11
12
13 module Part10_LFSR_TB ();
14     logic Pin;
15     logic [7:0] A = 8'b01000000;
16     logic CLK = 0;
17     logic rst = 0;
18     wire [7:0] Pout;
19     Part10_LFSR G1 (Pin, CLK, rst, Pout);
20     always #50 CLK = ~CLK;
21     initial begin
22         #100 Pin = A[0];
23         #100 Pin = A[1];
24         #100 Pin = A[2];
25         #100 Pin = A[3];
26         #100 Pin = A[4];
27         #100 Pin = A[5];
28         #100 Pin = A[6];
29         #100 Pin = A[7];
30         #7150 $stop;
31     end
32 endmodule

```



$$x^8 + x^7 + x^6 + x^3 + 1$$