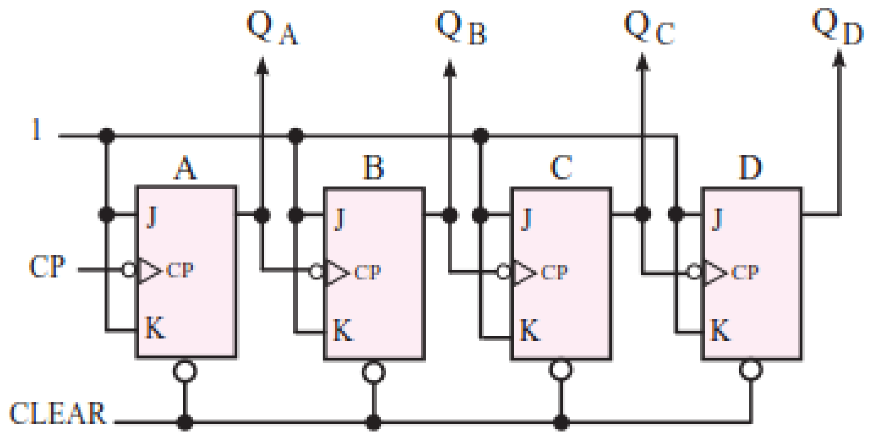
آزمایشگاه سیستم های دیجیتال2

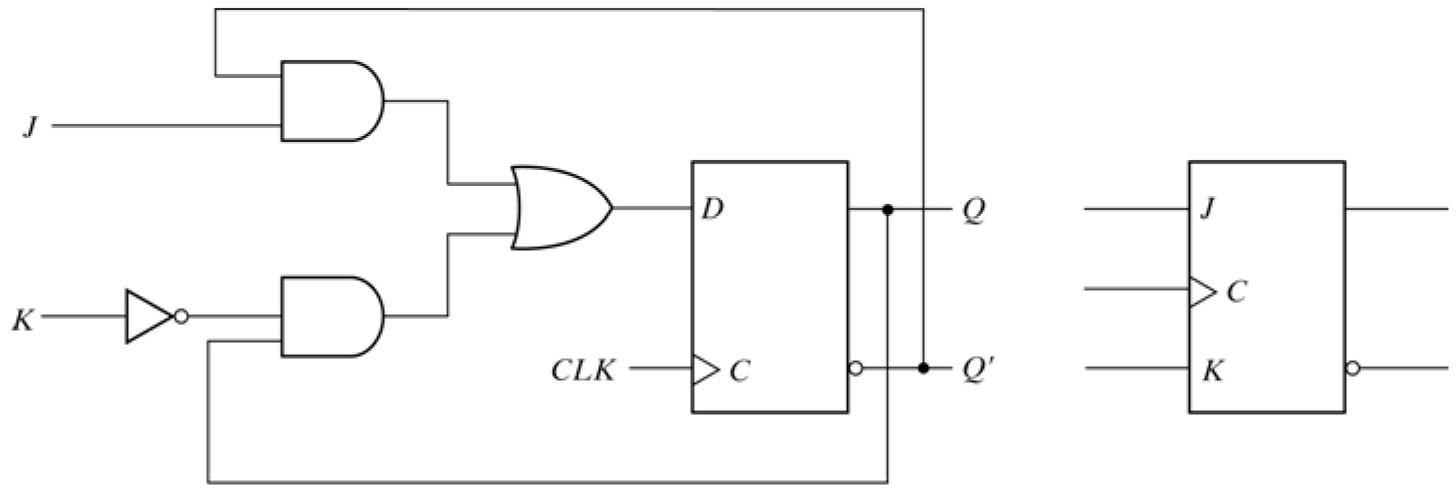
آزمایش 3

شمارنده ها

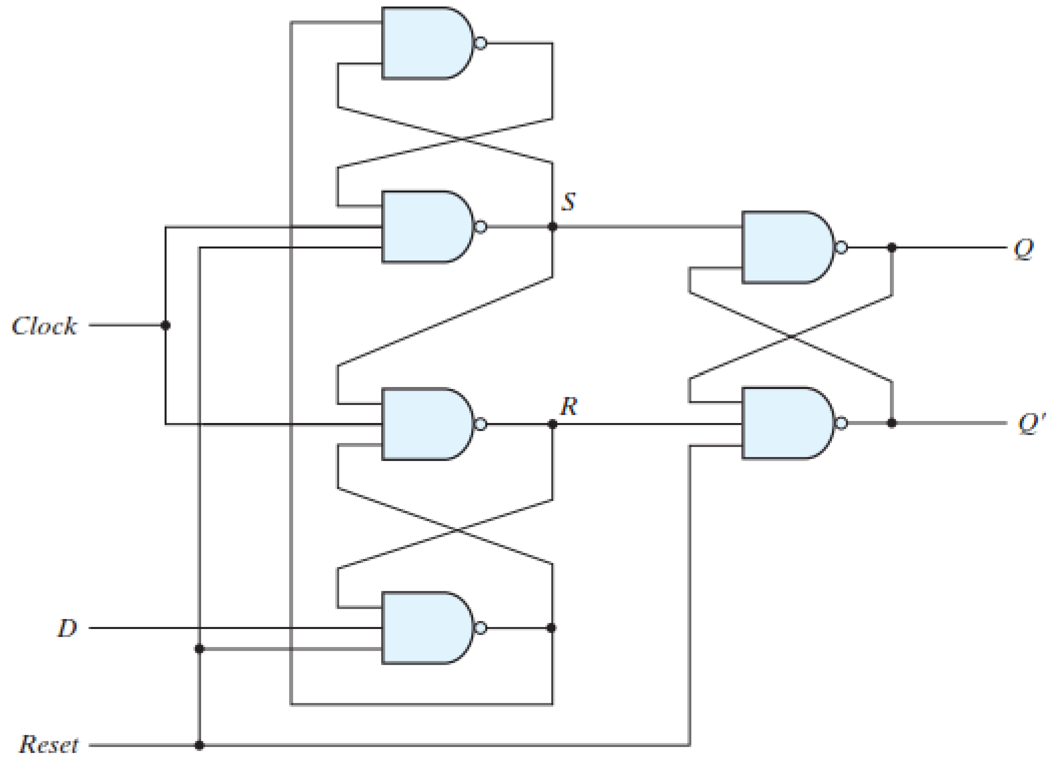
شمارنده ها (Counters)



مدار داخلی هر بلوک



فیلیپ فلاپ D دارای پایه Reset

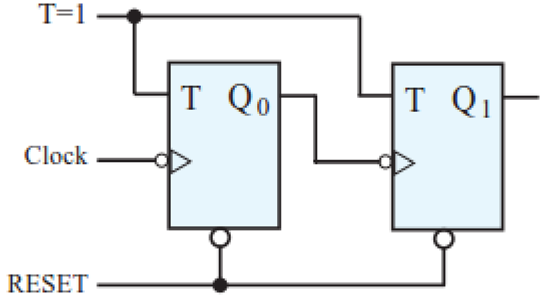


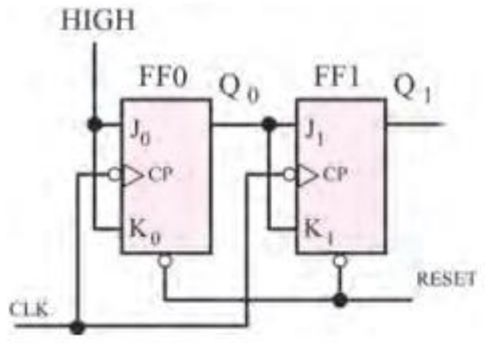
ادامه

انواع شمارنده ها

- شمارنده های آسنکرون (Asynchronous)

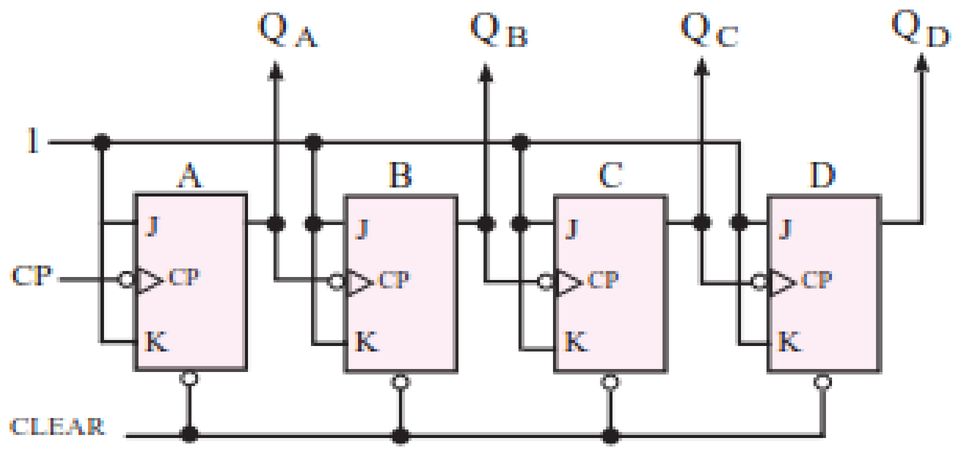
- شمارنده های سنکرون (synchronous)

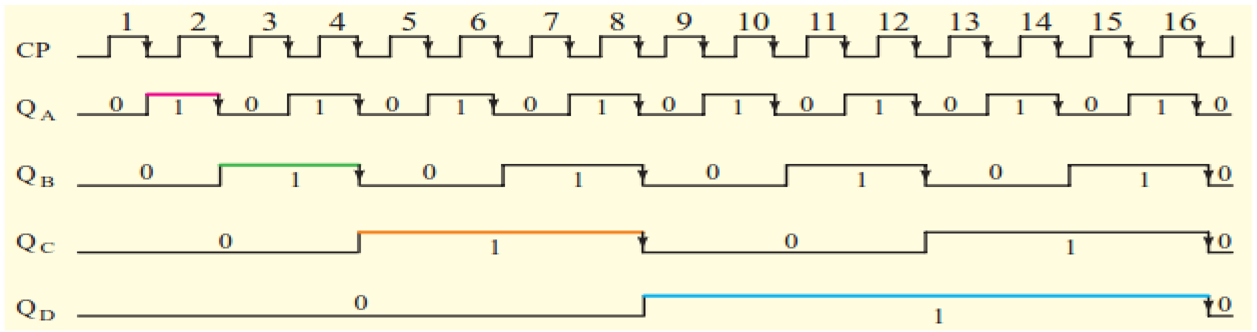


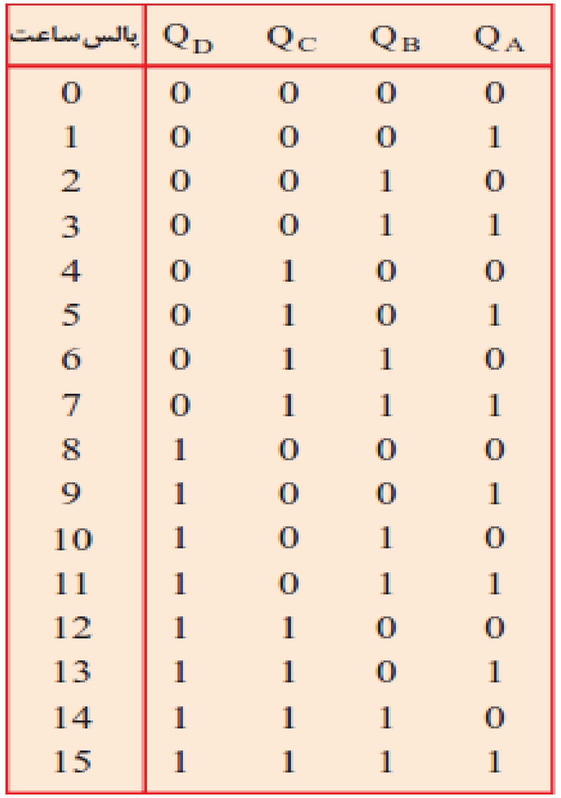


شمارنده های آسنکرون

- شمارنده های آسنکرون صعودی (Up Counter) :



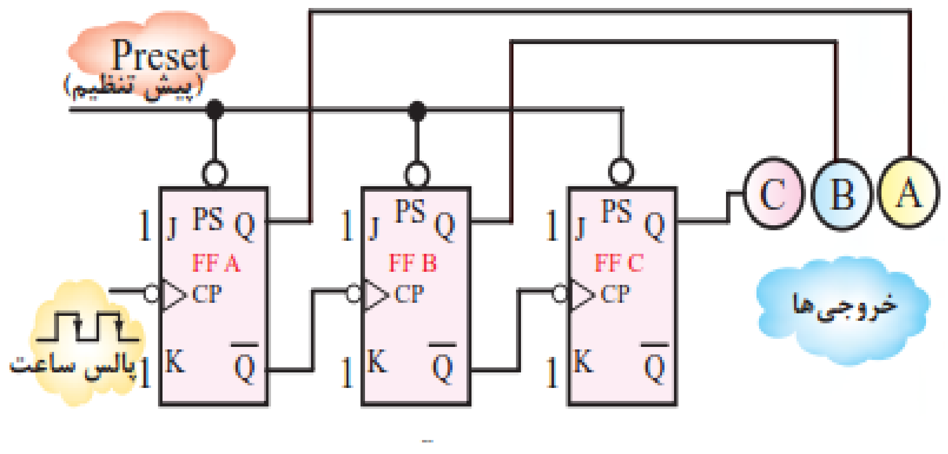


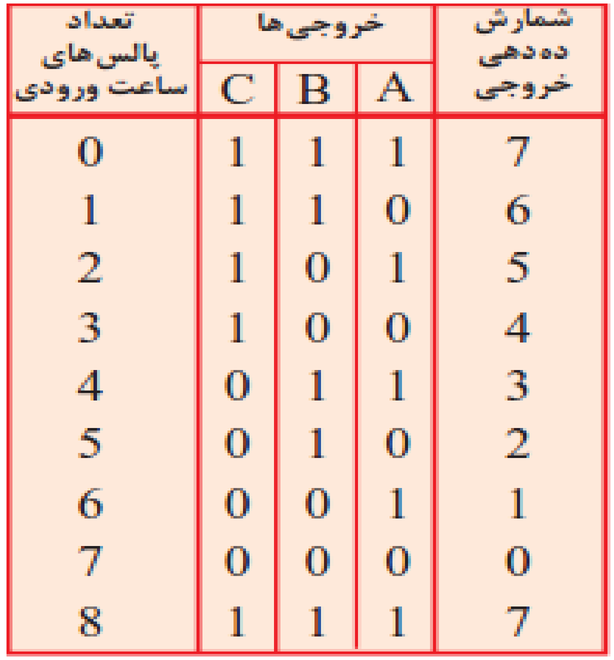


پالس ساعت	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

ادامه

- شمارنده های آسنکرون نزولی (Down Counter) :

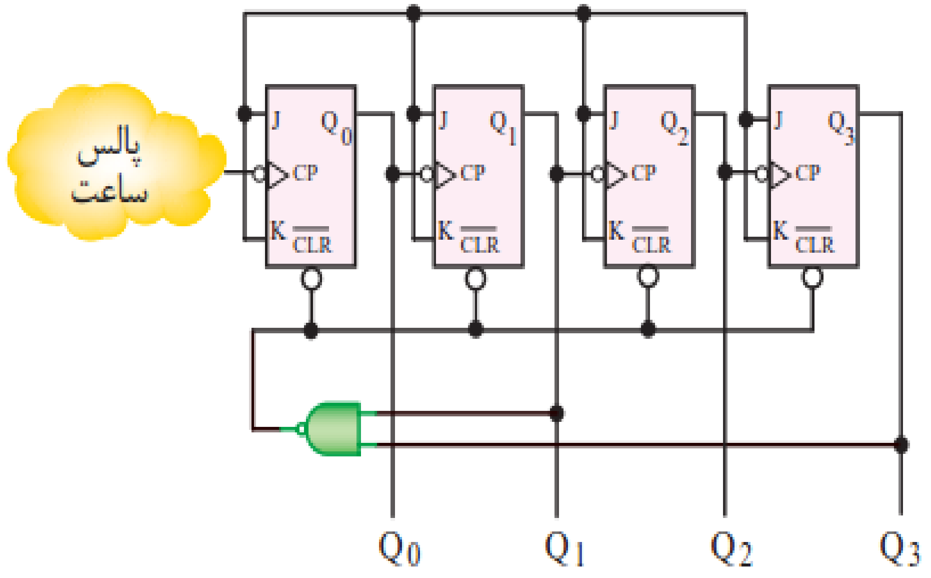


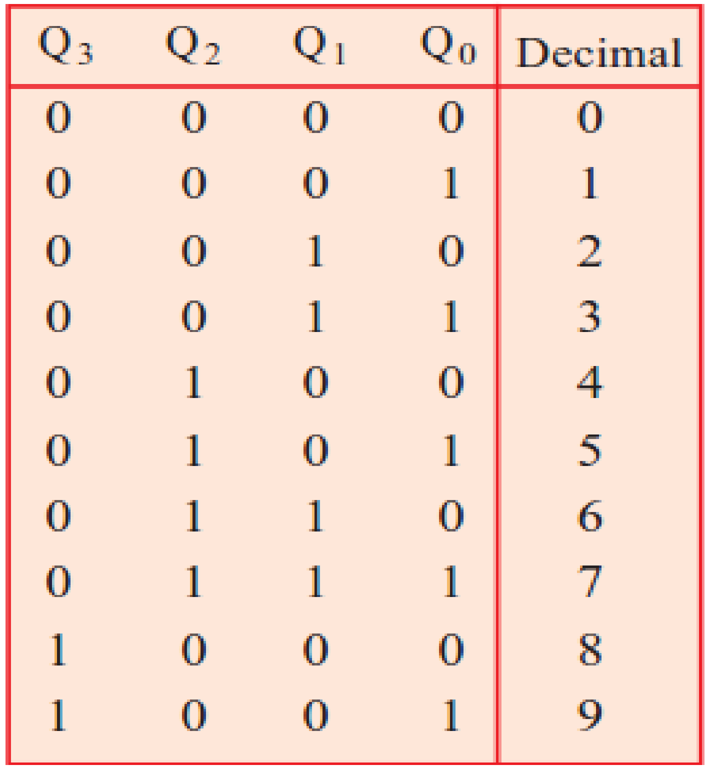


تعداد پالس‌های ساعت ورودی	خروجی‌ها			شمارش ده‌دهی خروجی
	C	B	A	
0	1	1	1	7
1	1	1	0	6
2	1	0	1	5
3	1	0	0	4
4	0	1	1	3
5	0	1	0	2
6	0	0	1	1
7	0	0	0	0
8	1	1	1	7

ادامه

• شمارنده های آسنکرون ده دهی BCD (Binary Coded Decimal) :

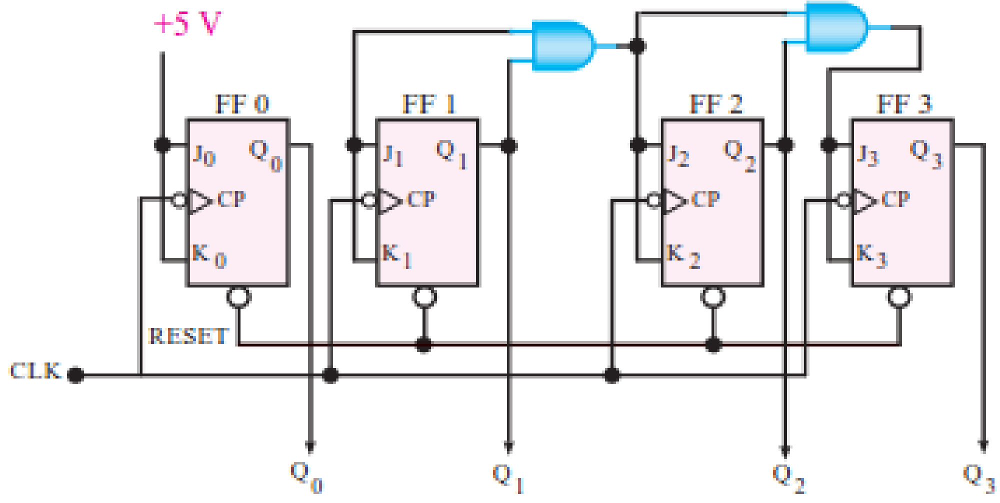


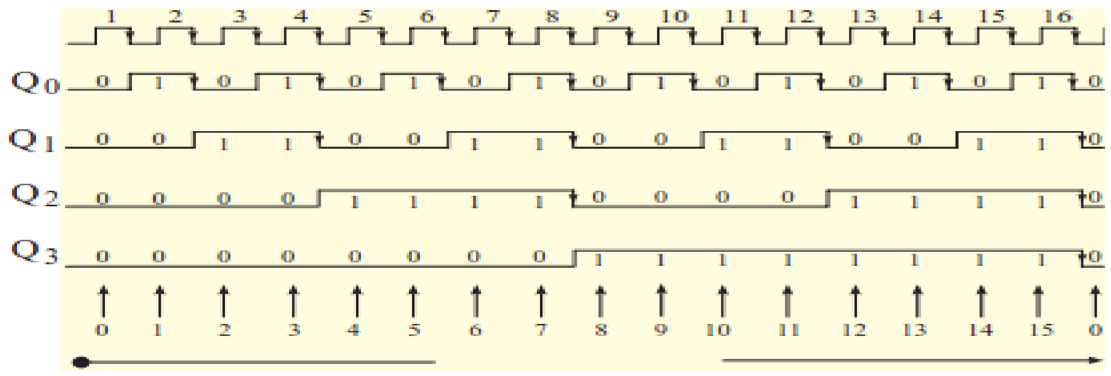


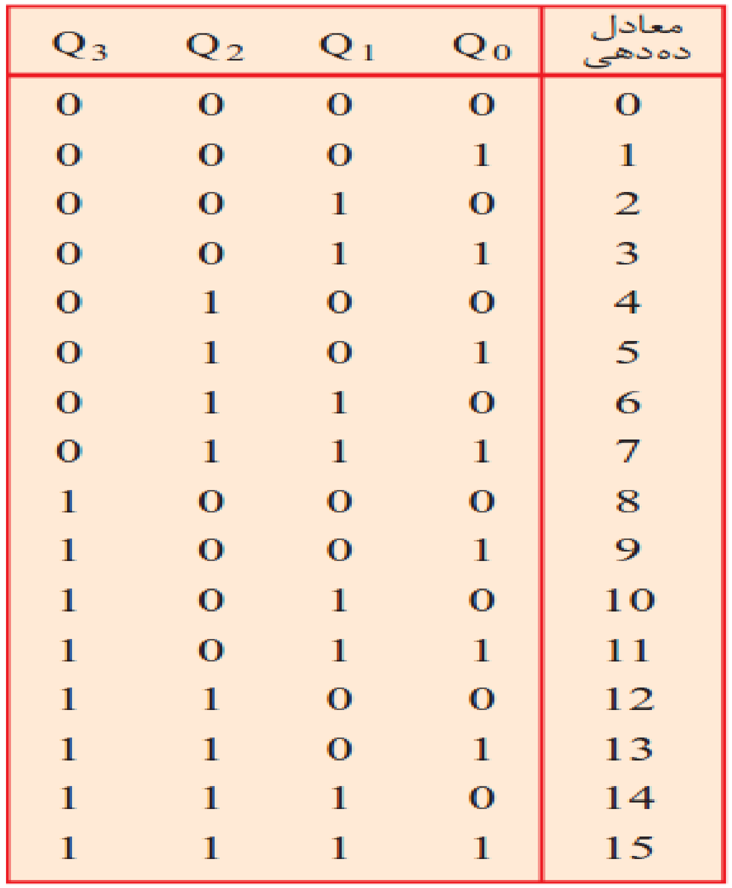
Q_3	Q_2	Q_1	Q_0	Decimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

شمارنده های سنکرون

- شمارنده های سنکرون صعودی :



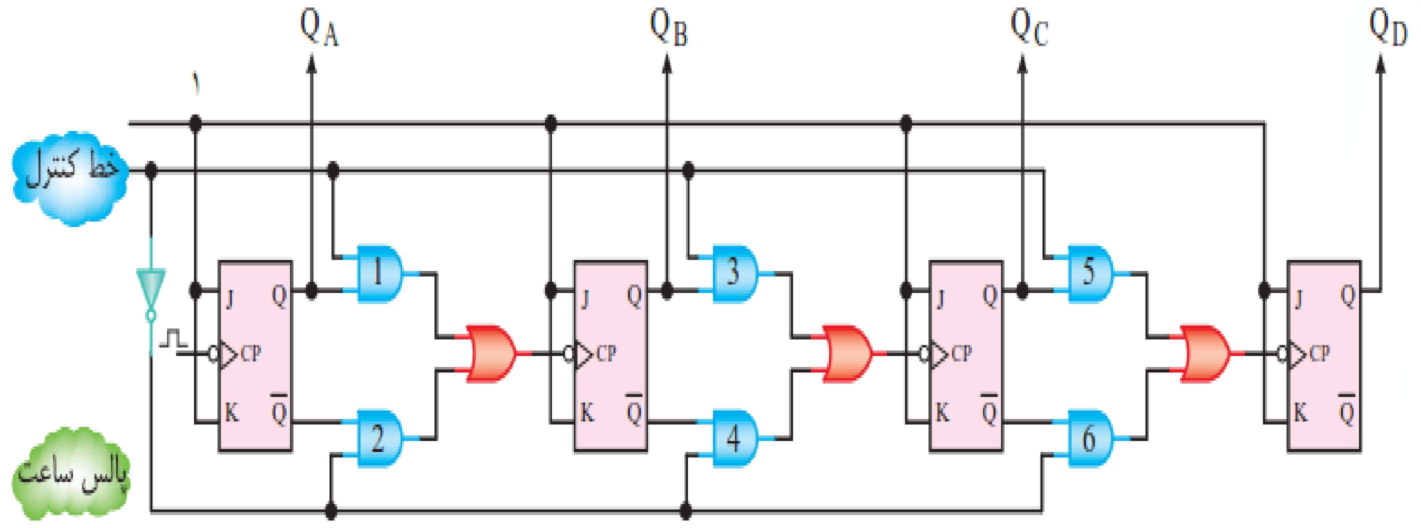




Q3	Q2	Q1	Q0	معادل دهدهی
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

ادامه

- شمارنده های صعودی ونزولی (Up/Down Counter) :



خط کنترل=۱ $\Rightarrow$ $\begin{cases} \text{AND}_{۱}, \text{AND}_{۳}, \text{AND}_{۵} & \text{فعال} \\ \text{AND}_{۲}, \text{AND}_{۴}, \text{AND}_{۶} & \text{غیر فعال} \end{cases}$

شمارندهٔ صعودی

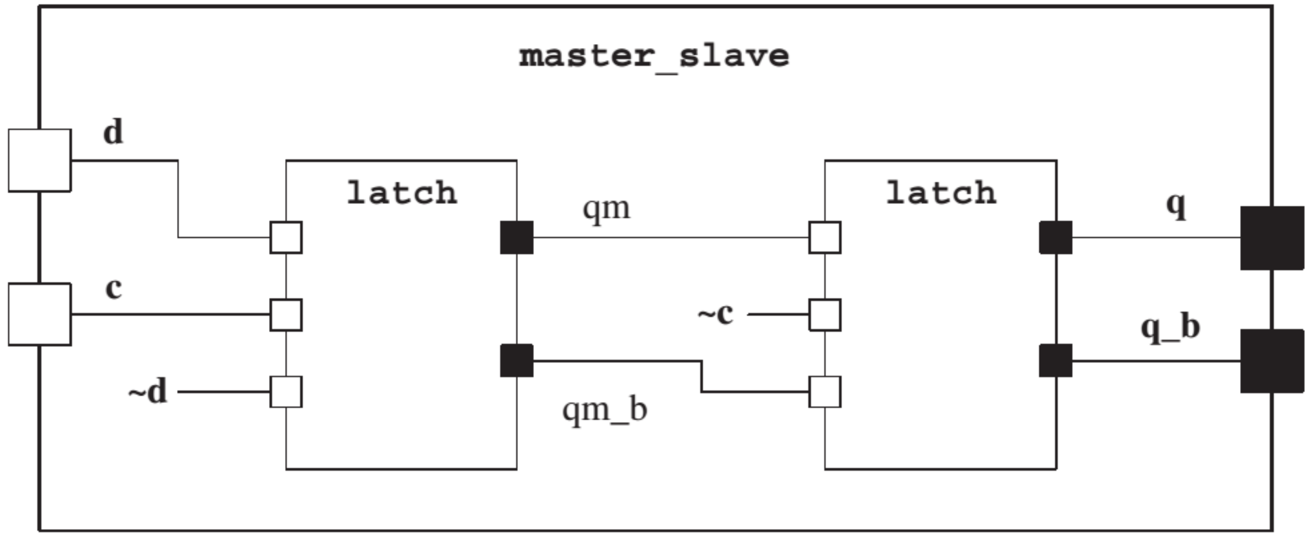
خط کنترل=۰ $\Rightarrow$ $\begin{cases} \text{AND}_{۱}, \text{AND}_{۳}, \text{AND}_{۵} & \text{غیرفعال} \\ \text{AND}_{۲}, \text{AND}_{۴}, \text{AND}_{۶} & \text{فعال} \end{cases}$

شمارندهٔ نزولی

Sequential Circuit Description

Basic Memory Components

Master-Slave D Flip-Flop



Sequential Circuit Description

Basic Memory Components

All NAND Clocked SR-Latch

```
`timescale 1ns/100ps

module latch_p #(parameter tplh=3, tphl=5) (input s, r, c,
                          output q, q_b );
   wire _s, _r;
   nand #(tplh,tphl)
        g1 ( _s, s, c ),
        g2 ( _r, r, c ),
        g3 ( q, _s, q_b ),
        g4 ( q_b, _r, q );
endmodule
```

Sequential Circuit Description

Basic Memory Components

Master-Slave D Flip-Flop

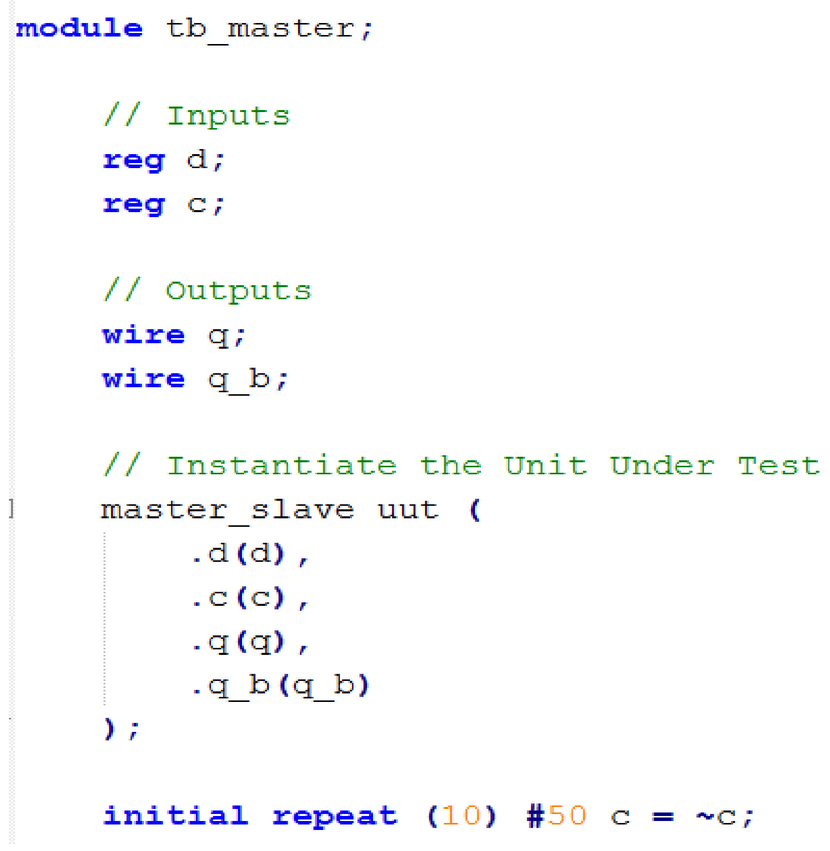
```
`timescale 1ns/100ps

module master_slave (input d, c, output q, q_b );
   wire qm, qm_b;
   defparam master.tplh=4, master.tphl=4, slave.tplh=4,
            slave.tphl=4;
   latch_p
      master ( d, ~d, c, qm, qm_b ),
      slave  ( qm, qm_b, ~c, q, q_b );
endmodule
```

Sequential Circuit Description

Basic Memory Components

Master-Slave D Flip-Flop



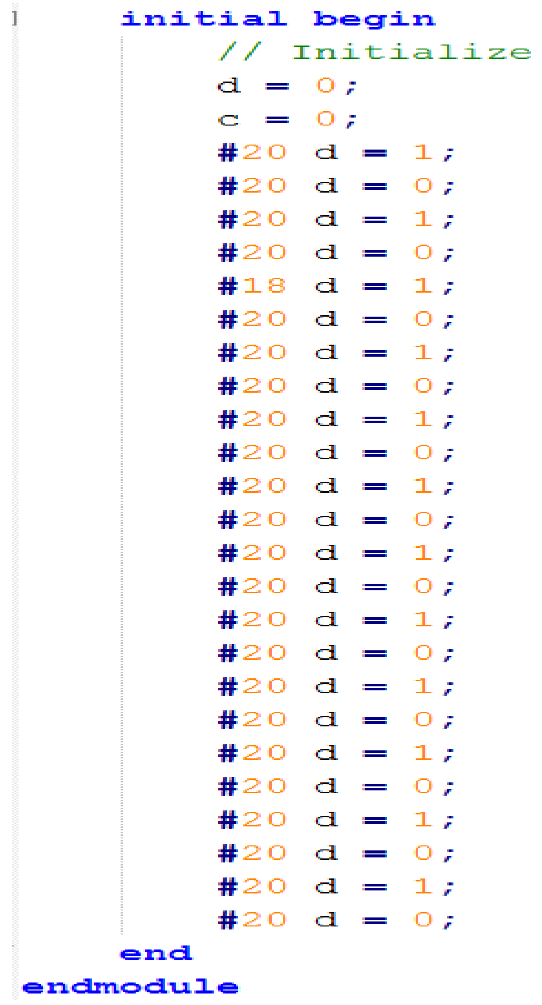
```
module tb_master;

    // Inputs
    reg d;
    reg c;

    // Outputs
    wire q;
    wire q_b;

    // Instantiate the Unit Under Test
    master_slave uut (
        .d(d),
        .c(c),
        .q(q),
        .q_b(q_b)
    );

    initial repeat (10) #50 c = ~c;
```

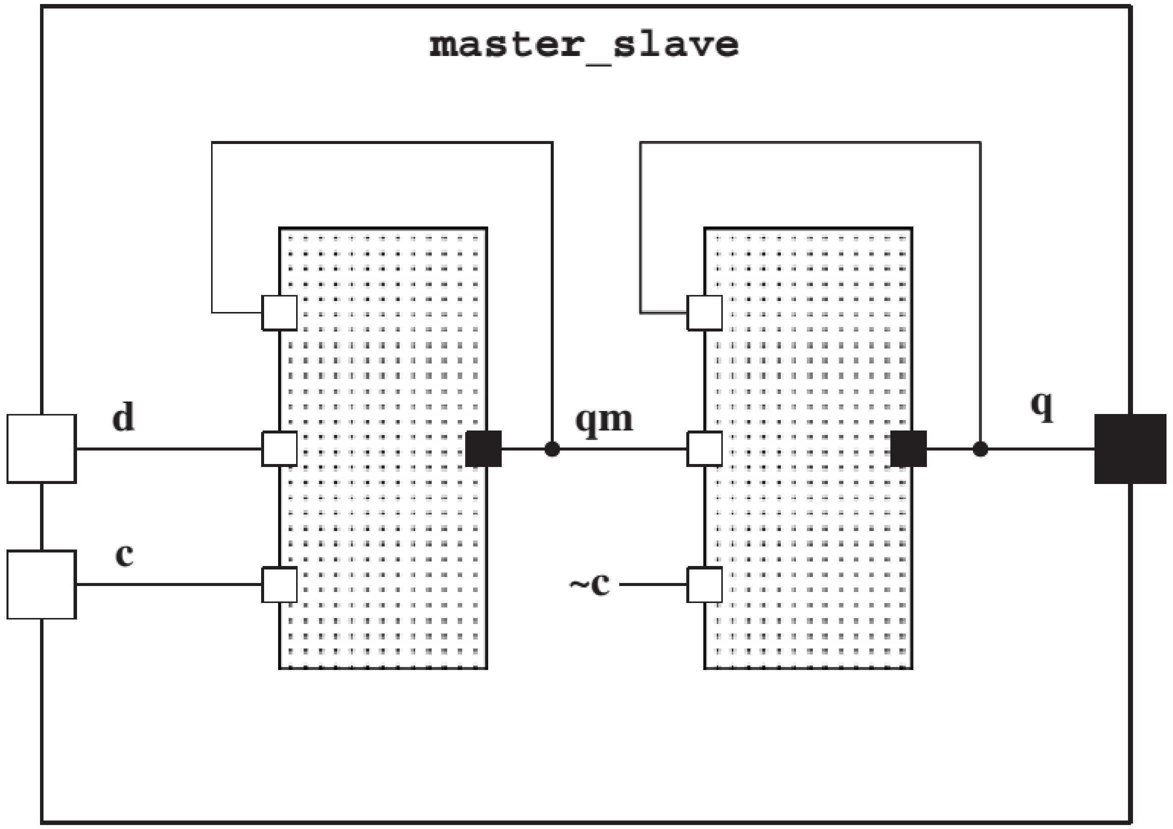


```
    initial begin
        // Initialize
        d = 0;
        c = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #18 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
    end
endmodule
```

Sequential Circuit Description

Basic Memory Components

Master-Slave Using Two Feedback Blocks



Sequential Circuit Description

Basic Memory Components

Master-Slave Using Two Feedback Blocks

```
`timescale 1ns/100ps

module master_slave_p #(parameter delay=3) (input d,c, output q);
   wire qm;
   assign #(delay) qm = c ? d : qm;
   assign #(delay) q = ~c ? qm : q;
endmodule
```

Sequential Circuit Description

Basic Memory Components

Master-Slave Using Two Feedback Blocks

```verilog
module tb_master;

    // Inputs
    reg d;
    reg c;

    // Outputs
    wire q;

    // Instantiate the Unit Under Test
    maste_slave_e uut (
        .d(d),
        .c(c),
        .q(q)
    );
```

```verilog
    initial repeat (10) #50 c = ~c;
    initial begin
        d = 0;
        c = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #18 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        #20 d = 0;
    end
endmodule
```

Sequential Circuit Description

Behavioral memory elements

A D-Type Latch Verilog Code

```
 `timescale 1ns/100ps

module latch (input d, c, output reg q, q_b );
    always @( c or d )
       if ( c ) begin
          #4 q = d;
          #3 q_b = ~d;
       end
endmodule
```

Sequential Circuit Description

Behavioral memory elements

Latch Using Nonblocking Assignments

```
`timescale 1ns/100ps

module latch (input d, c, output reg q, q_b );
   always @( c or d )
      if ( c ) begin
         q <= #4 d;
         q_b <= #3 ~d;
      end
endmodule
```

Sequential Circuit Description

Behavioral memory elements

Positive Edge Trigger Flip-Flop

```
`timescale 1ns/100ps

module d_ff (input d, clk, output reg q, q_b );
   always @( posedge clk ) begin
       q <= #4 d;
       q_b <= #3 ~d;
   end
endmodule
```

Sequential Circuit Description

Behavioral memory elements

D Type Flip-Flop with Synchronous Control

```
module d_ff_sr_Synch (input d, s, r, clk, output reg q, q_b );

   always @(posedge clk) begin
      if( s ) begin
         q <= #4 1'b1;
         q_b <= #3 1'b0;
      end else if( r ) begin
         q <= #4 1'b0;
         q_b <= #3 1'b1;
      end else begin
         q <= #4 d;
         q_b <= #3 ~d;
      end
   end
endmodule
```

Sequential Circuit Description

Memory vectors and arrays

8-bit Transparent D-Latch

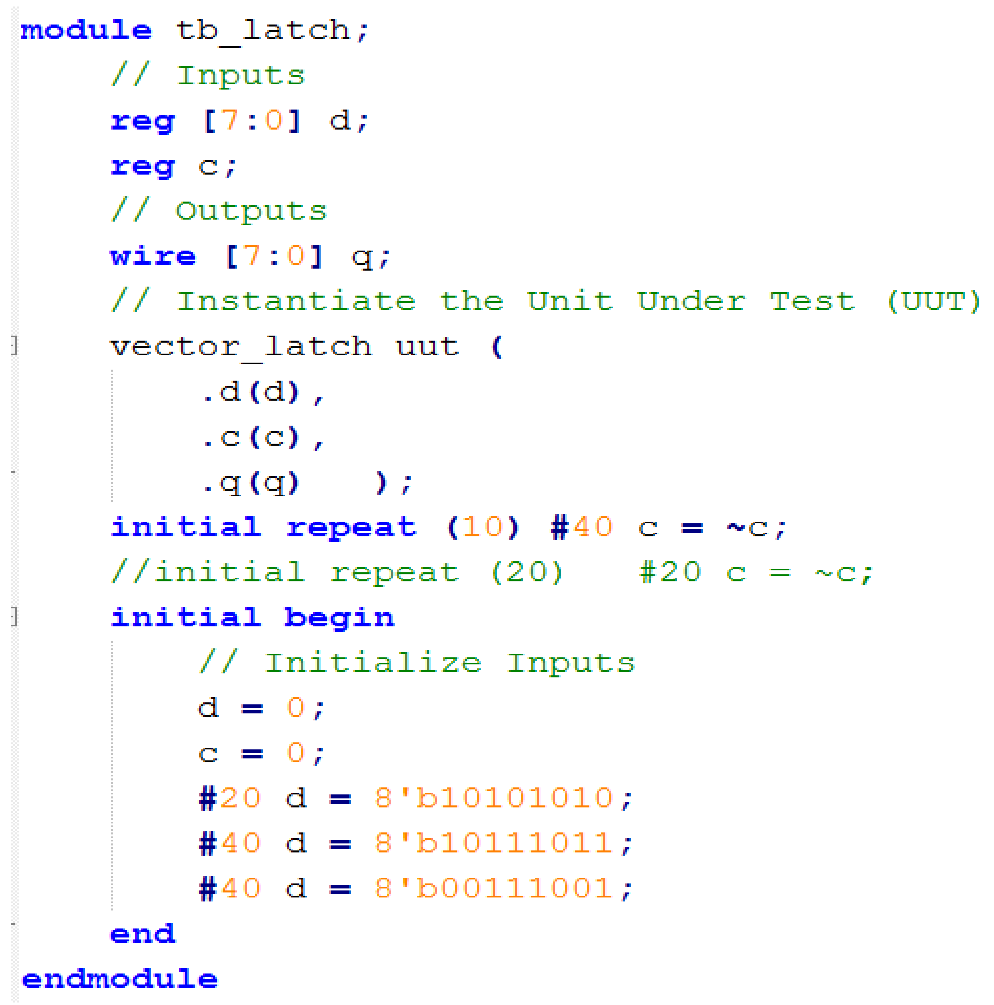
```
`timescale 1ns/100ps

module vector_latch (input [7:0] d, input c, output reg [7:0] q);
    always @( c or d )
        if( c )
            #4 q = d;
endmodule
```

Sequential Circuit Description

Memory vectors and arrays

8-bit Transparent D-Latch, Test



```
module tb_latch;
    // Inputs
    reg [7:0] d;
    reg c;
    // Outputs
    wire [7:0] q;
    // Instantiate the Unit Under Test (UUT)
    vector_latch uut (
        .d(d),
        .c(c),
        .q(q)   );
    initial repeat (10) #40 c = ~c;
    //initial repeat (20)   #20 c = ~c;
    initial begin
        // Initialize Inputs
        d = 0;
        c = 0;
        #20 d = 8'b10101010;
        #40 d = 8'b10111011;
        #40 d = 8'b00111001;
    end
endmodule
```

Sequential Circuit Description

Memory vectors and arrays

An 8-bit Register with Tri-State Output

```
module vector_ff (input [7:0] d, input clk, rst, oe,
                  output [7:0] q);
   reg [7:0] internal_q;

   always @( posedge clk )
      if( rst )
         #4 internal_q <= 8'b0000_0000;
      else
         #4 internal_q <= d;

   assign q = oe ? internal_q : 8'bZ;
endmodule
```

Sequential Circuit Description

Memory vectors and arrays

A Sizable Register

```
`timescale 1ns/100ps
module sizable_reg #(size) ( input [size-1:0] d, input clk, rst,
                             output reg [size-1:0] q );
  always @( posedge clk, negedge rst )
    begin
      if( ~rst )
        #4 q <= 0;
      else
        #4 q <= d;
    end

endmodule
```

Sequential Circuit Description

Separate register&combinational blocks

Up-Down Counter

```
module counter (input [3:0] d_in, input clk, rst, ld, u_d,
                output reg [3:0] q);
   always @( posedge clk ) begin
      if( rst )
         q = 4’b0000;
      else if( ld )
         q = d_in;
      else if( u_d )
         q = q + 1;
      else
         q = q - 1;
   end
endmodule
```