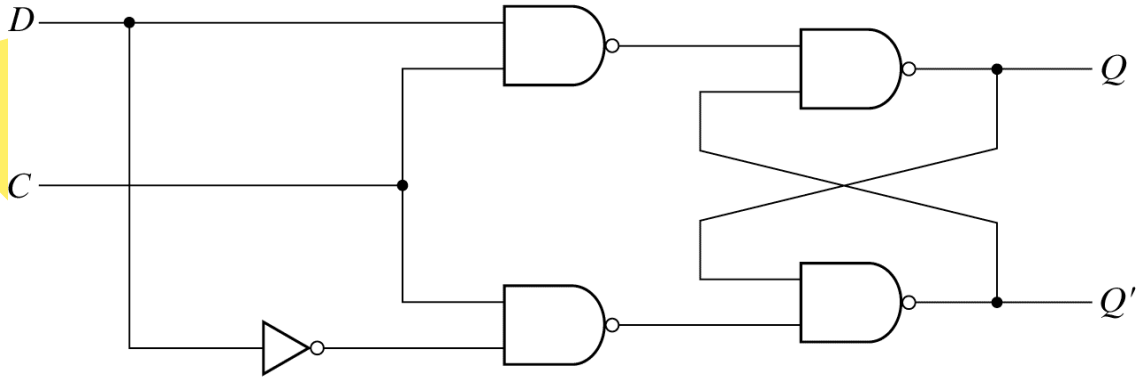
D latch



(a) Logic diagram

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

(b) Function table

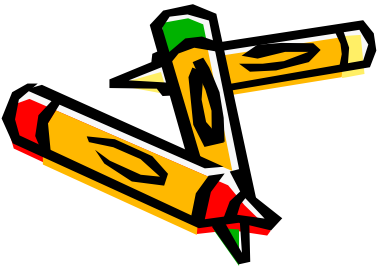
Fig. 5-6 D Latch

```
module D_latch(Q,D,control);
  output Q;
  input D,control;
  reg Q;
  always @(control or D)
  if (control) Q=D;
endmodule
```

D flip-flop

```
module DFF(Q,D,CLK,RST);
module DFF(Q,D,CLK);
   output Q;
   output RST;
   input D,CLK;
   reg Q;
   always @(posedge CLK or negedge RST)
   always @(posedge CLK )
      if (~RST)
         Q=1'b0;
      else
         Q=D;
endmodule
```

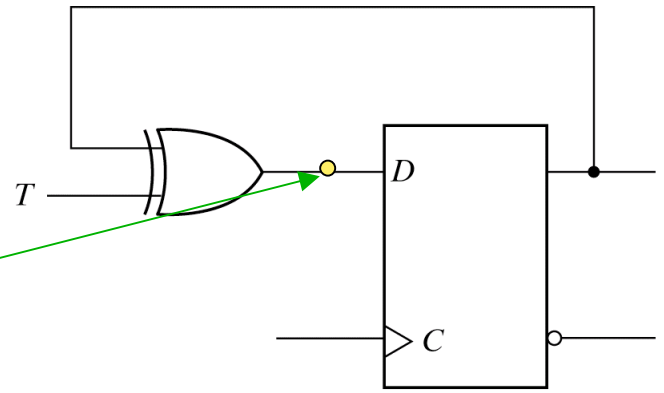
$$Q(t+1) = D$$



T flip-flop from D flip-flop and gate

- module TFF(Q,T,CLK,RST);
- output Q;
- input T,CLk,RST;
- wire DT;
- assign DT=Q^T;
- DFF TF1(Q,DT,CLK,RST);

$$D = T \oplus Q = TQ' + T'Q$$

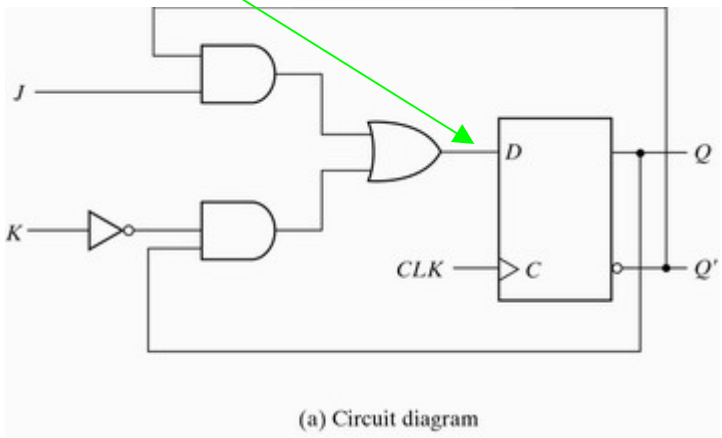


(b) From *D* flip-flop

JK flip-flop from D flip-flop and gate

- module JKFF (Q,J,K,CLK,RST);
- output Q;
- input J,K,CLK,RST;
- wire JK;
- assign JK =(J&~Q)|(~K&Q);
- DFF JK1 (Q,JK,CLK,RST);
- endmodule

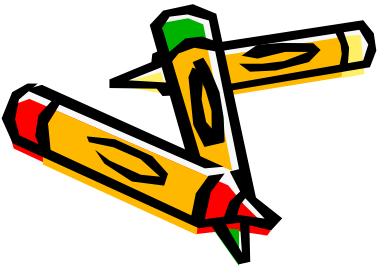
$$Q(t+1) = JQ' + K'Q$$



(a) Circuit diagram

```
module JK_FF (J,K,CLK,Q,Qnot)
   output Q,Qnot;
   input J,K,CLK;
   reg Q;
   assign Qnot =~Q;
   always @(posedge CLK)
      case ({J,K})
         2'b00:Q=Q;
         2'b01:Q=1'b0;
         2'b10:Q=1'b1;
         2'b11:Q=~Q;
      endcase
endmodule
```

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	~Q(t)



Mealy state diagram

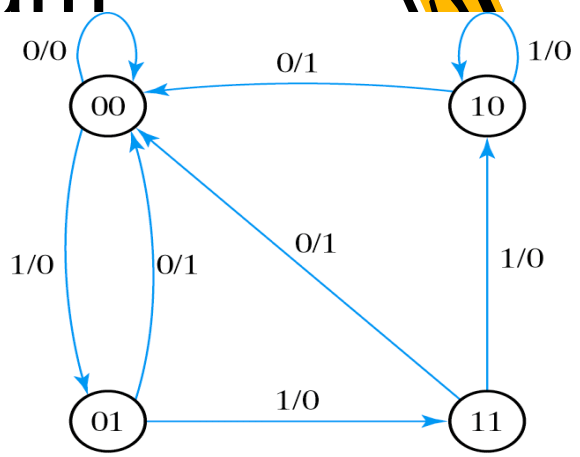


Fig. 5-16 State Diagram of the Circuit of Fig. 5-15

- module mealy_mdl (x,y,CLK,RST
- input x,CLK,RST;
- output y;
- reg y;
- reg [1:0] Prstate,Nxtstate;
- parameter s0=2'b00, s1=2'b01,
- s2=2'b10, s3=2'b11;
- always @ (posedge CLK or negedge RST)
- if (~RST) Prstate =s0;
- else Prstate =Nxtstate;
-

```
always @ (Prstate or x)
    case (Prstate)
        s0: if (x) Nxtstate=S1;
            else Nxtstate=S0;
        s1: if (x) Nxtstate=S3;
            else Nxtstate=S0;
        s2: if (~x) Nxtstate=S0;
            else Nxtstate=S2;
        s3: if (x) Nxtstate=S2;
            else Nxtstate=S0;
    endcase
always @ (Prstate or x)
    case (Prstate)
        s0: y=0;
        s1: if (x)  y=1'b0;      else  y=1'b1;
        s2: if (x)  y=1'b0;      else  y=1'b1;
        s3: if (x)  y=1'b0;      else   y=1'b1;
    endcase
endmodule
```

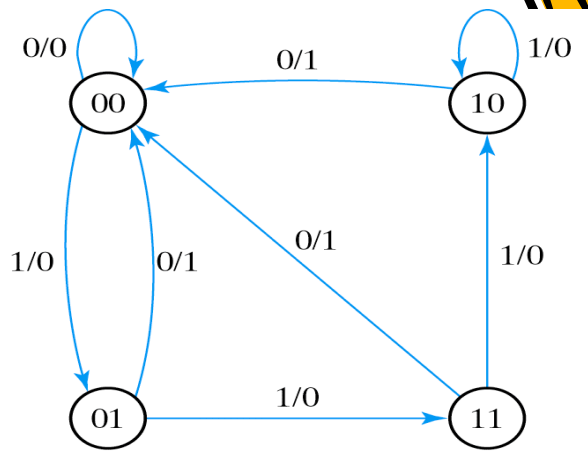


Fig. 5-16 State Diagram of the Circuit of Fig. 5-15

```
module Moore_mdl (x,ab,CLK,RST);
input x,CLK,RST;
output [1:0] ab;
reg [1:0] state ;
parameter s0=2'b00,s1=2'b01,s2=2'b10,s3=2'b11;
   always @ (posedge CLK or negedge RST)
      if (~RST)  state =s0;
      else
      case (state)
         s0: if(~x) state =s1; else state =s0;
         s1: if(x)  state =s2; else state =s3;
         s2: if(~x) state =s3; else state =s2;
         s3: if(~x) state =s0; else state =s3;
      endcase
   assign ab=state
endmodule
```

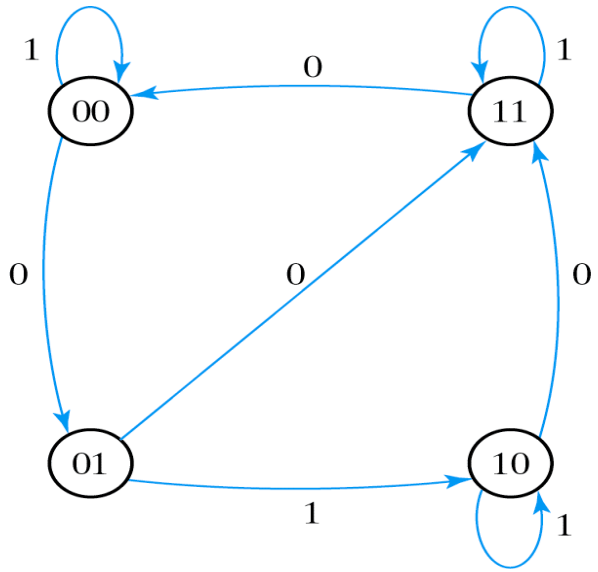
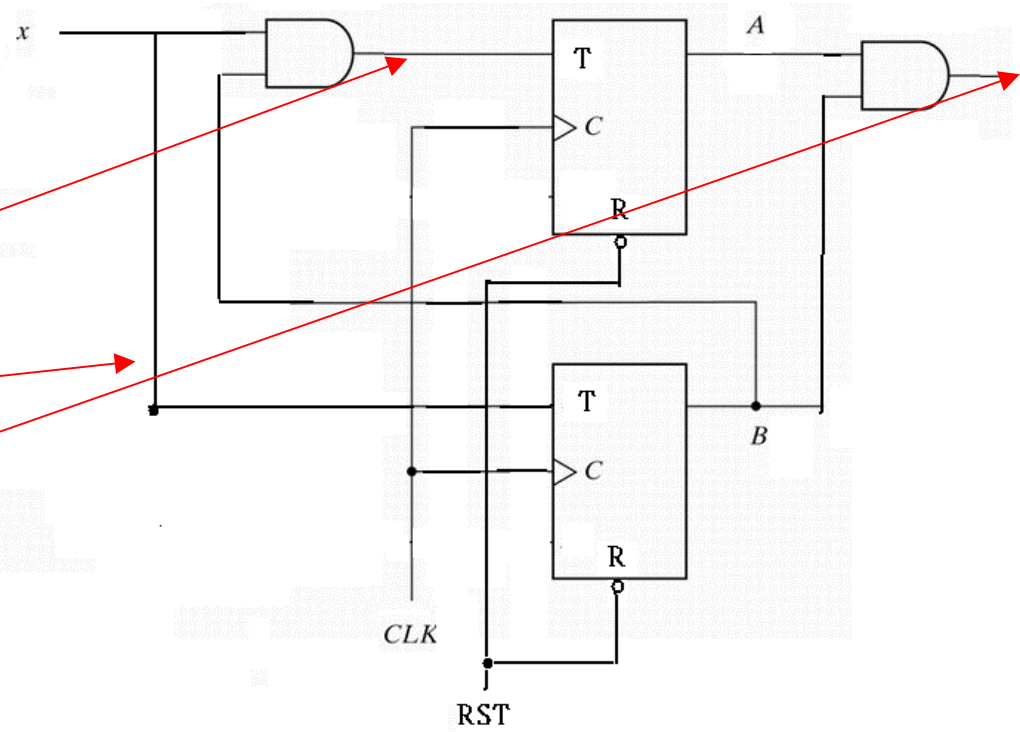


Fig. 5-19 State Diagram of the Circuit of Fig. 5-18

```
module Tcircuit (x,y,A,B,CLK,RST)
   input x,CLK,RST;
   output y,A,B;
   wire TA,TB;
// Flip-flip input equations
   assign TA=x&B;
   assign TB=x;
// Flip-flip output equations
   assign y=A&B;
//Instantiate T flip-flip
   T_FF BF (B,TB,CLK,RST)
   T_FF AF (A,TA,CLK,RST)
endmodule
```




```
module T_FF(Q,T,CLK,RST);
   output Q;
   input T,CLK,RST;
   reg Q;
   always @ (posedge CLK or negedge RST)
      if (~RST) Q=1'b0;
      else  Q=Q^T;
endmodule
```

$$Q(t+1) = T \oplus Q = TQ' + T'Qf$$

```
module testTcircuit;
   reg x,CLK,RST;
   wire y,A,B;
   Tcircuit TC(x,y,A,B,CLK,RST) ;
   initial
     begin
          RST=0;
          CLk=0;
      #5  RST=1;
          repeat(16);
      #5     CLK=~CLK;
     end
   initial
     begin
          x=0;
      #15  x=1;
          repeat(8);
      #5      x=~x;
     end
endmodule
```