Chapter 5 Synchronous Sequential Logic
5-1 Sequential Circuits

Every digital system is likely to have combinational circuits, most systems encountered in practice also include storage elements, which require that the system be described in term of sequential logic.

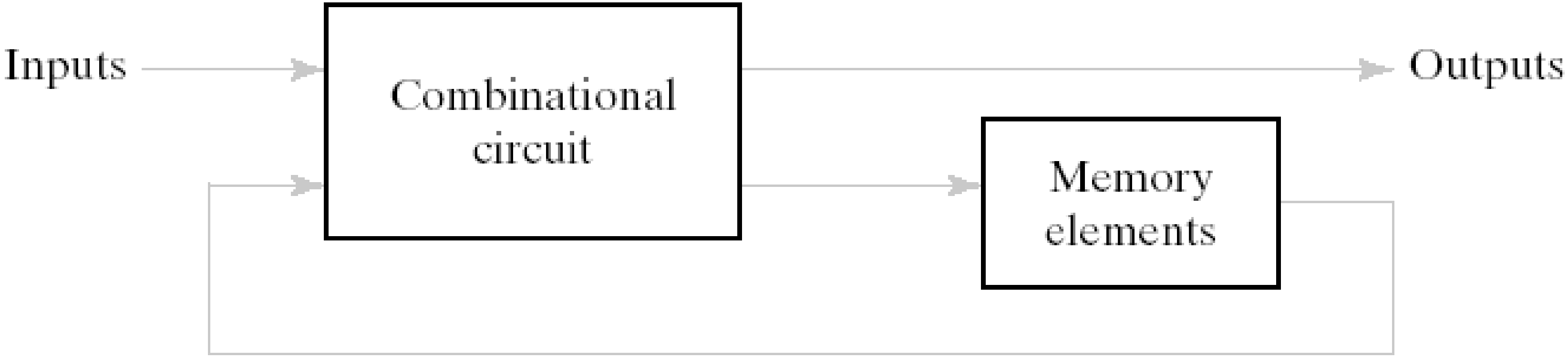
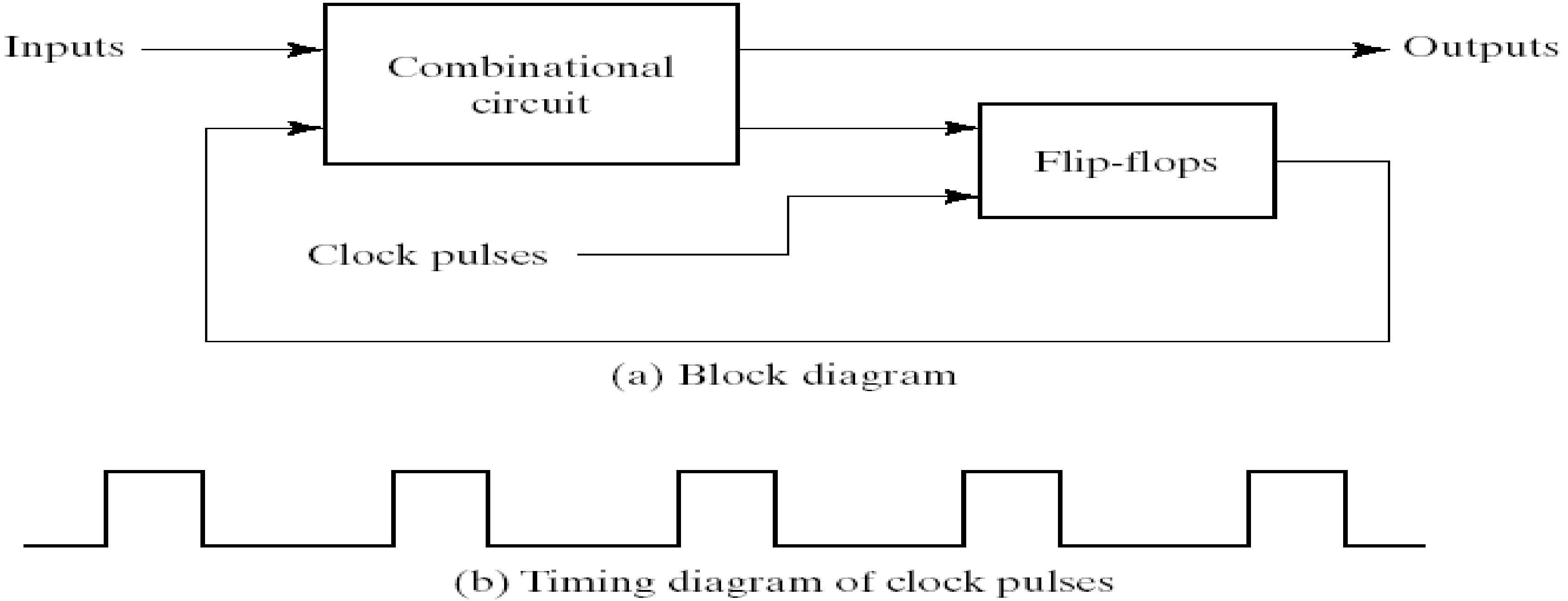


Fig. 5-1 Block Diagram of Sequential Circuit

Synchronous Clocked Sequential Circuit

A sequential circuit may use many flip-flops to store as many bits as necessary. The outputs can come either from the combinational circuit or from the flip-flops or both.



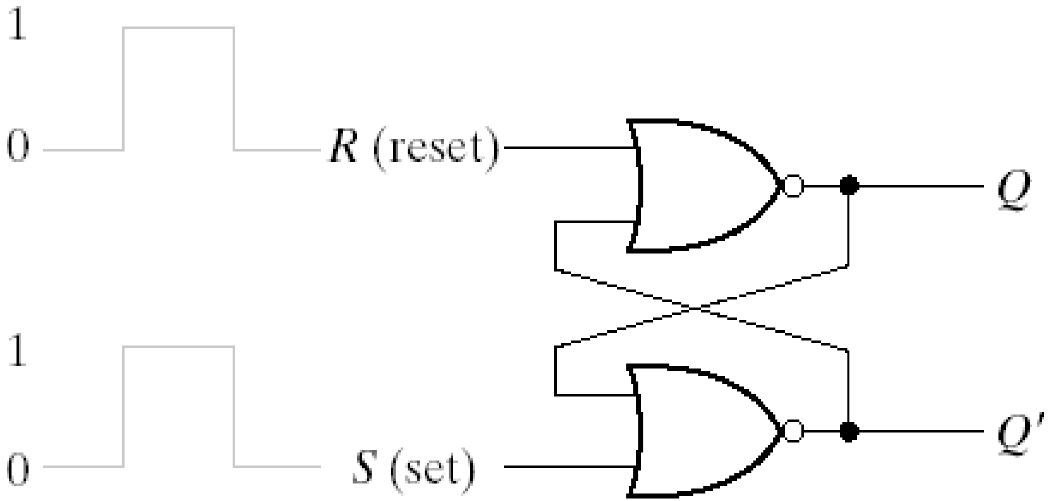
(a) Block diagram

(b) Timing diagram of clock pulses

Fig. 5-2 Synchronous Clocked Sequential Circuit

5-2 Latches SR Latch

The SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates. It has two inputs labeled S for set and R for reset.



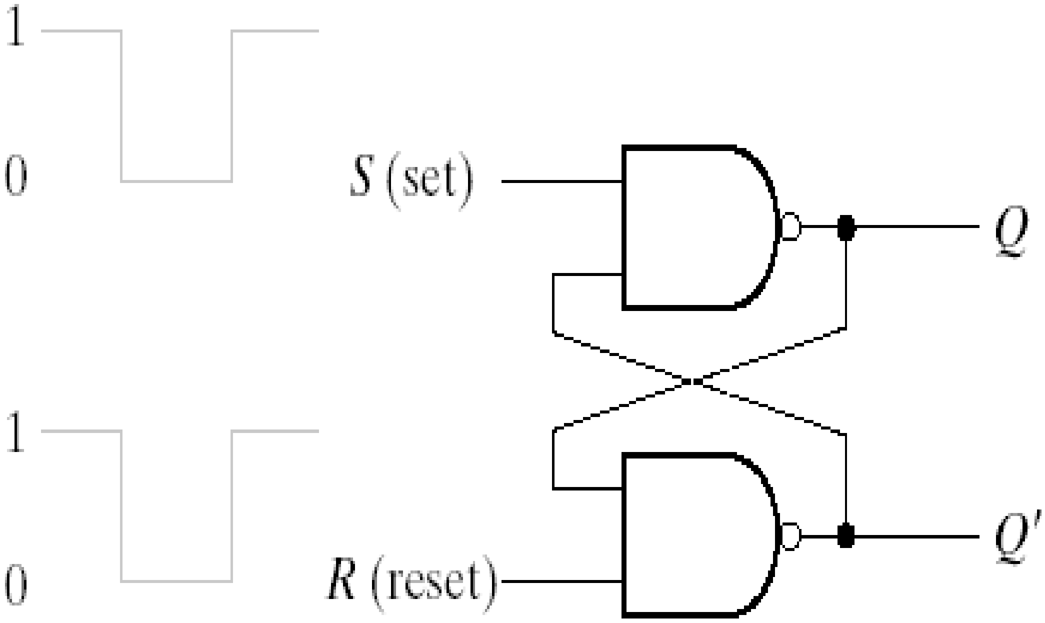
(a) Logic diagram

S	R	Q	Q′	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	

(b) Function table

Fig. 5-3 *SR* Latch with NOR Gates

SR Latch with NAND Gates



(a) Logic diagram

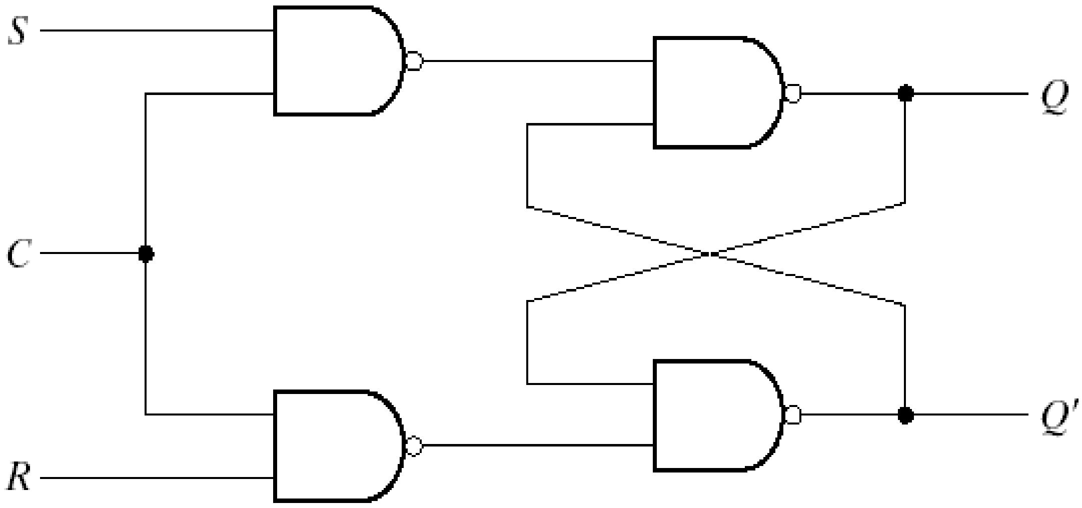
S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	

(b) Function table

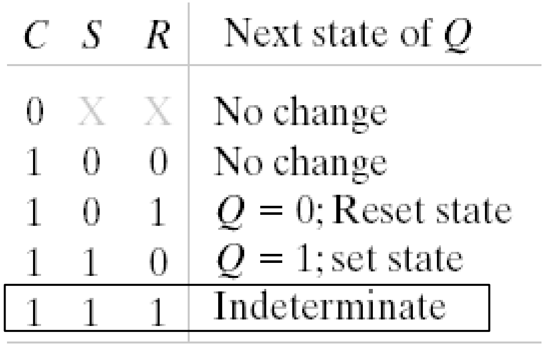
Fig. 5-4 *SR* Latch with NAND Gates

SR Latch with Control Input

The operation of the basic SR latch can be modified by providing an additional control input that determines when the state of the latch can be changed. In Fig. 5-5, it consists of the basic SR latch and two additional NAND gates.



(a) Logic diagram



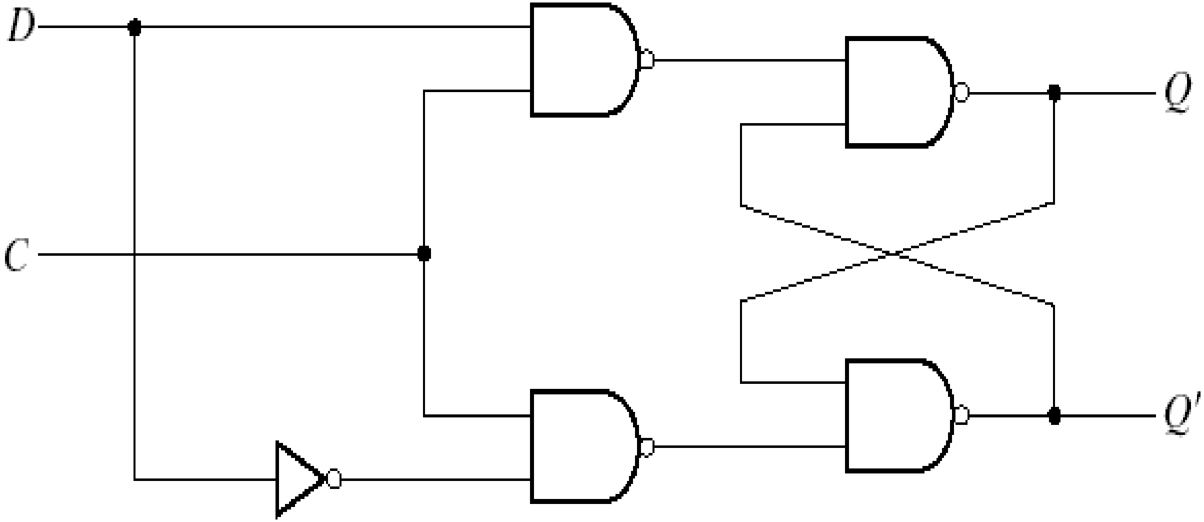
C	*S*	*R*	Next state of *Q*
0	X	X	No change
1	0	0	No change
1	0	1	*Q* = 0; Reset state
1	1	0	*Q* = 1; set state
1	1	1	Indeterminate

(b) Function table

Fig. 5-5 SR Latch with Control Input

D Latch

One way to eliminate the undesirable condition of the indeterminate state in SR latch is to ensure that inputs S and R are never equal to 1 at the same time in Fig 5-5. This is done in the 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

Graphic Symbols for latches

A latch is designated by a rectangular block with inputs on the left and outputs on the right. One output designates the normal output, and the other designates the complement output.

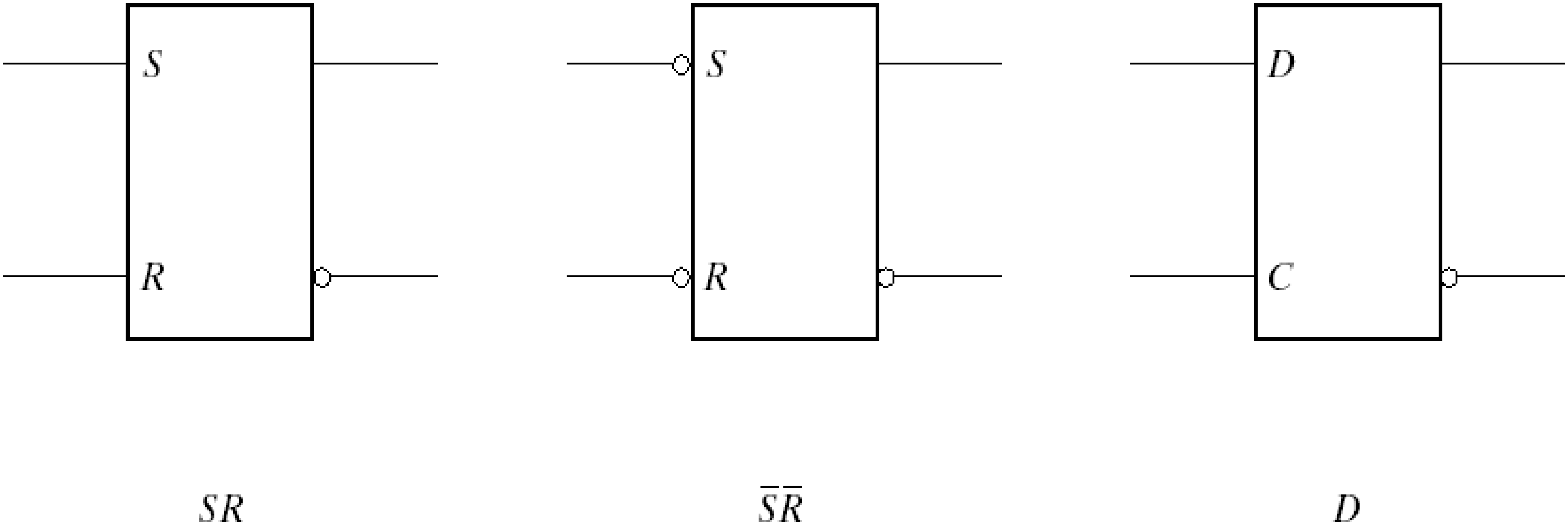


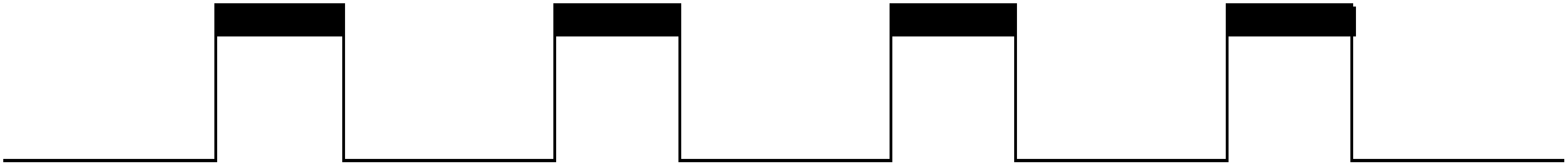
Fig. 5-7 Graphic Symbols for Latches

5-3 Flip-Flops

The state of a latch or flip-flop is switched by a change in the control input. This momentary change is called a trigger and the transition it cause is said to trigger the flip-flop. The D latch with pulses in its control input is essentially a flip-flop that is triggered every time the pulse goes to the logic 1 level. As long as the pulse input remains in the level, any changes in the data input will change the output and the state of the latch.

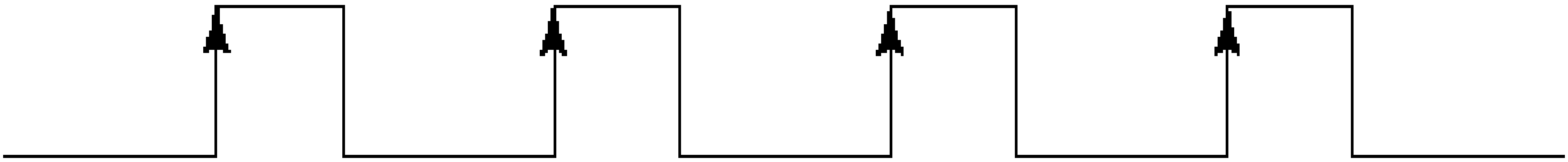
Clock Response in Latch

In Fig (a) a positive level response in the control input allows changes, in the output when the D input changes while the clock pulse stays at logic 1.

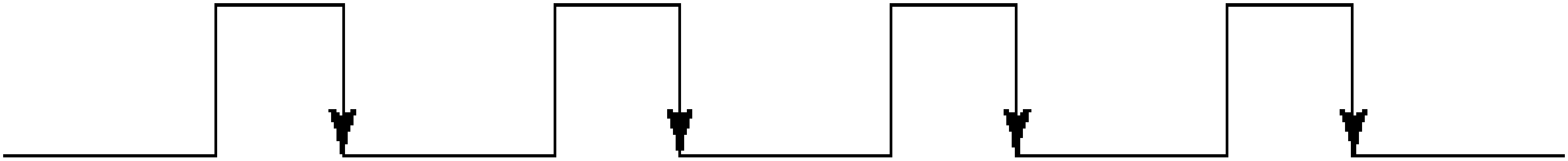


(a) Response to positive level

Clock Response in Flip-Flop



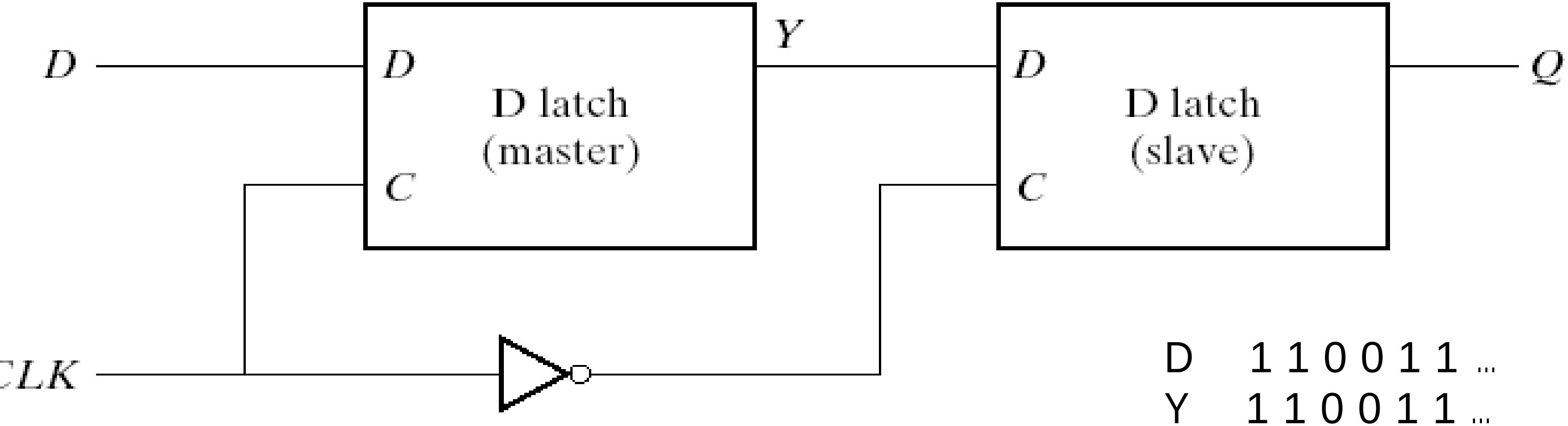
(b) Positive-edge response



(c) Negative-edge response

Edge-Triggered D Flip-Flop

The first latch is called the master and the second the slave. The circuit samples the D input and changes its output Q only at the negative-edge of the controlling clock.



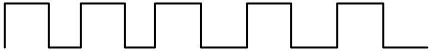
D 1 1 0 0 1 1 …

Y 1 1 0 0 1 1 …

Q ? 1 1 0 0 1 … .

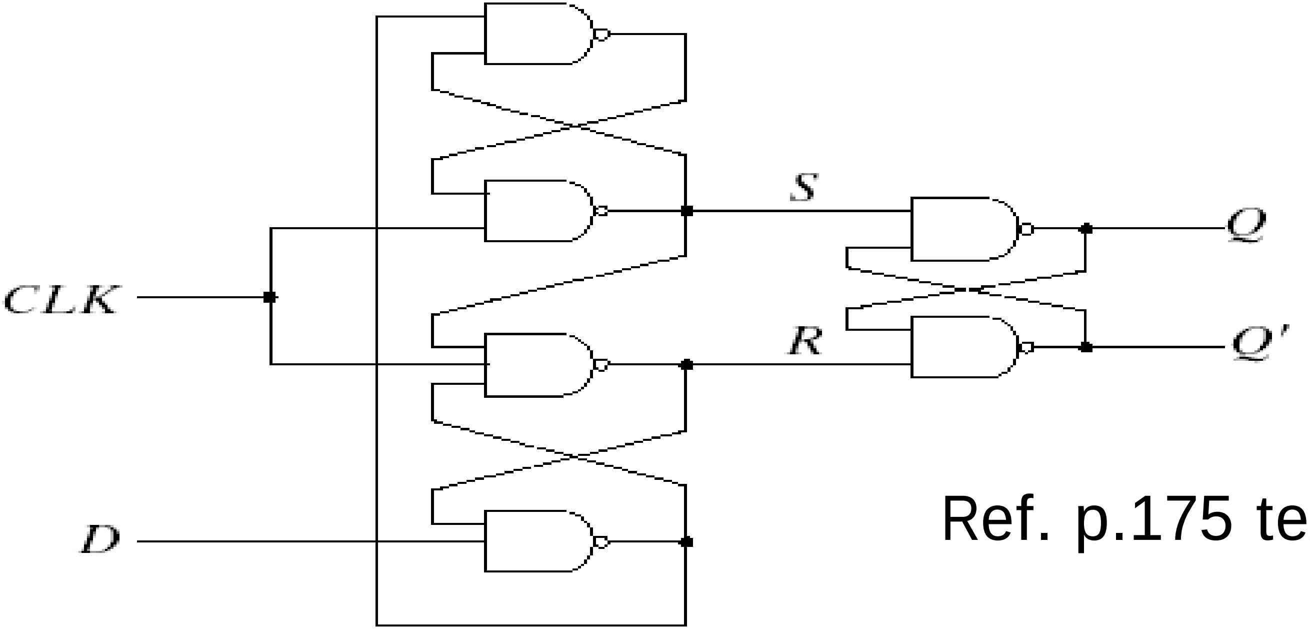
Fig. 5-9 Master-Slave *D* Flip-Flop

CLK



D-Type Positive-Edge-Triggered Flip-Flop

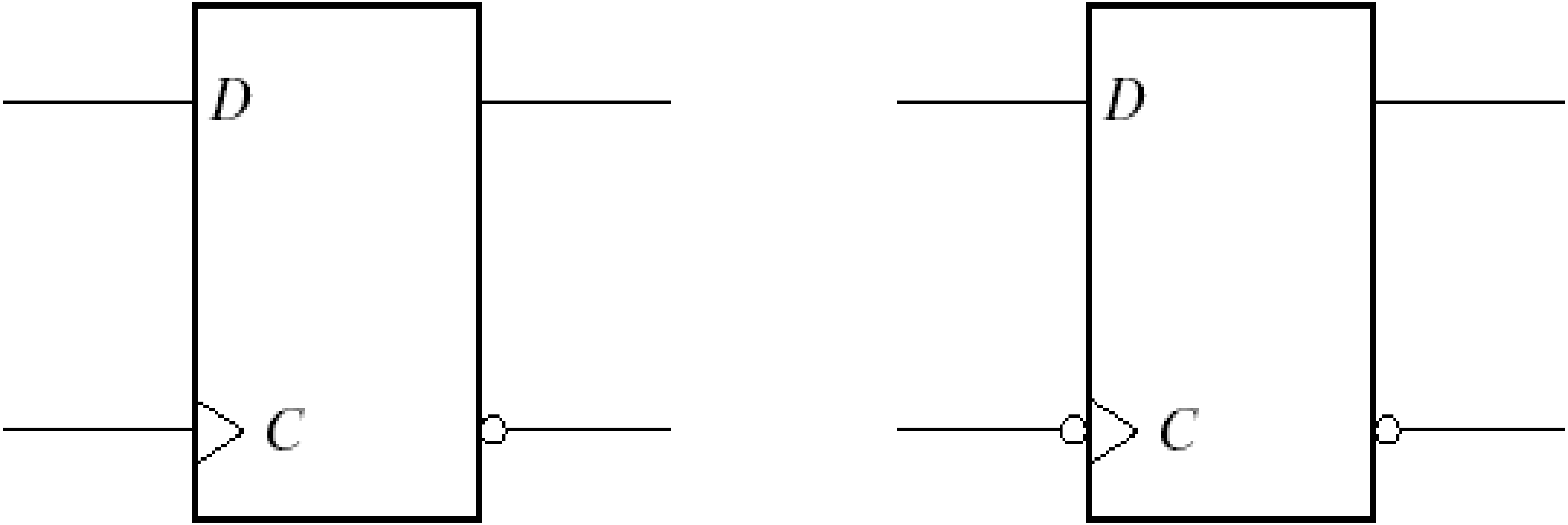
Another more efficient construction of an edge-triggered D flip-flop uses three SR latches. Two latches respond to the external D(data) and CLK(clock) inputs. The third latch provides the outputs for the flip-flop.



Ref. p.175 texts

Fig. 5-10 *D*-Type Positive-Edge-Triggered Flip-Flop

Graphic Symbol for Edge-Triggered D Flip-Flop



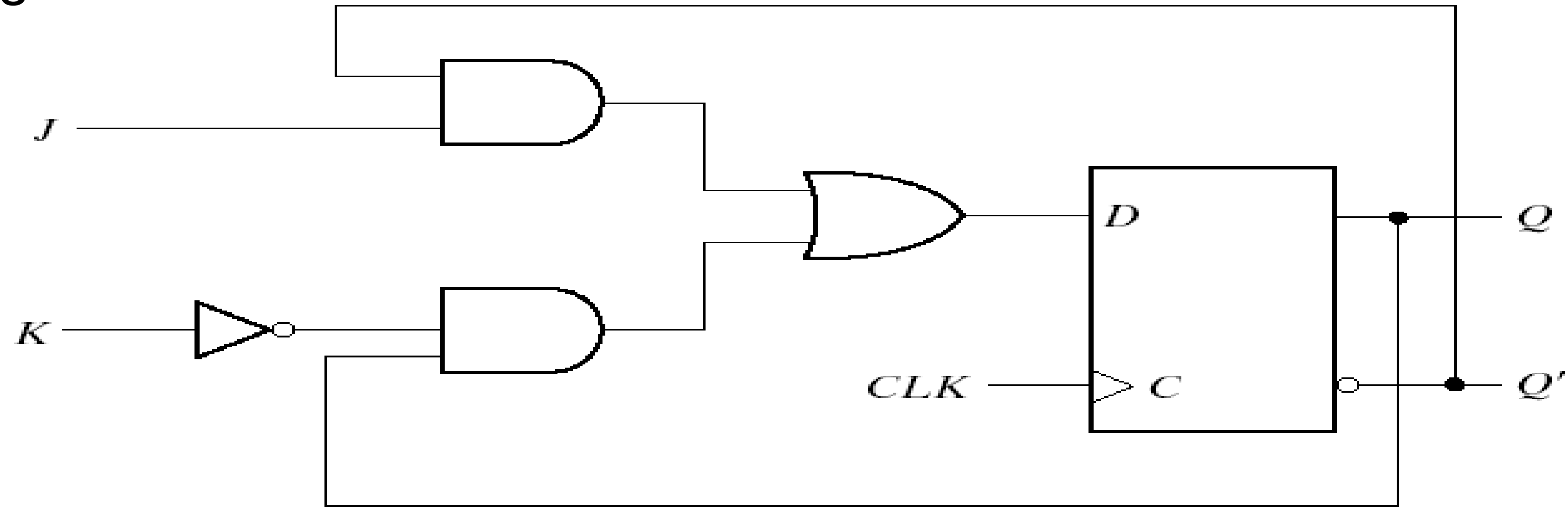
(a) Positive-edge

(a) Negative-edge

Fig. 5-11 Graphic Symbol for Edge-Triggered *D* Flip-Flop

Other Flip-Flops JK Flip-Flop

There are three operations that can be performed with a flip-flop: set it to 1, reset it to 0, or complement its output. The JK flip-flop performs all three operations. The circuit diagram of a JK flip-flop constructed with a D flip-flop and gates.

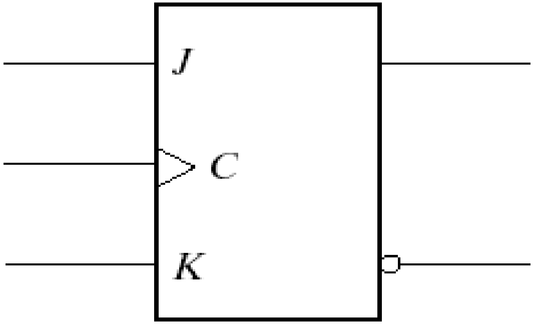


(a) Circuit diagram

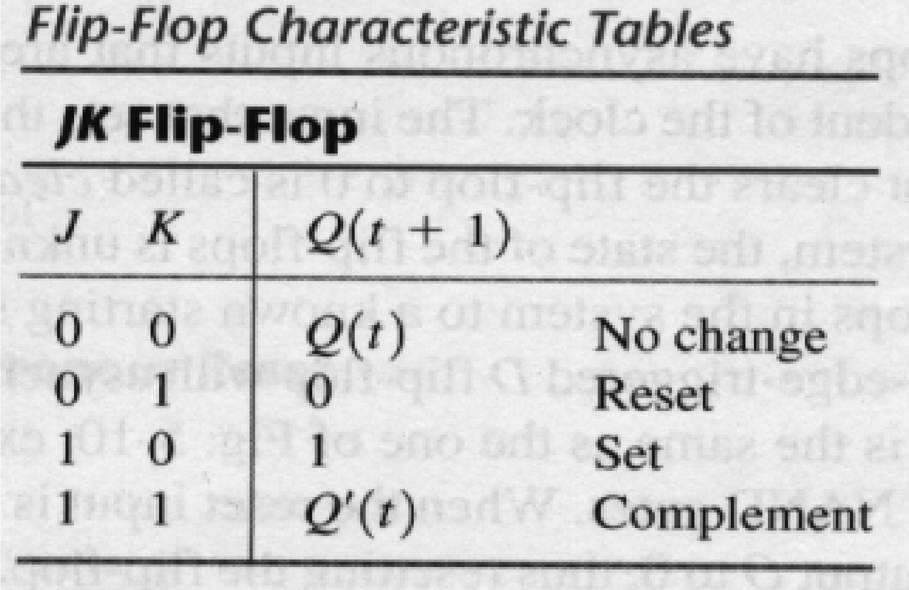
JK Flip-Flop

The J input sets the flip-flop to 1, the K input resets it to 0, and when both inputs are enabled, the output is complemented. This can be verified by investigating the circuit applied to the D input:

$$\boxed{D = J Q` + K` Q}$$



(b) Graphic symbol



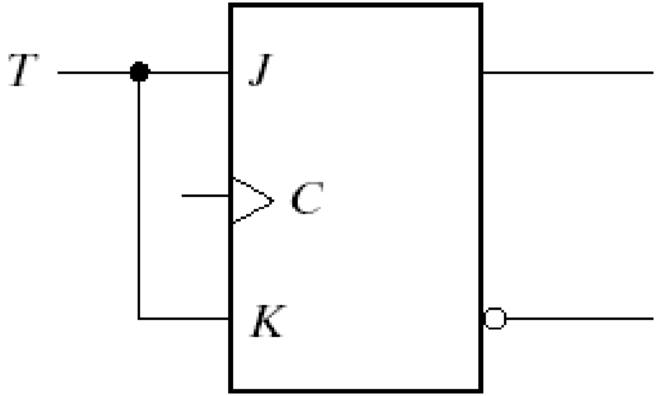
Flip-Flop Characteristic Tables

JK Flip-Flop

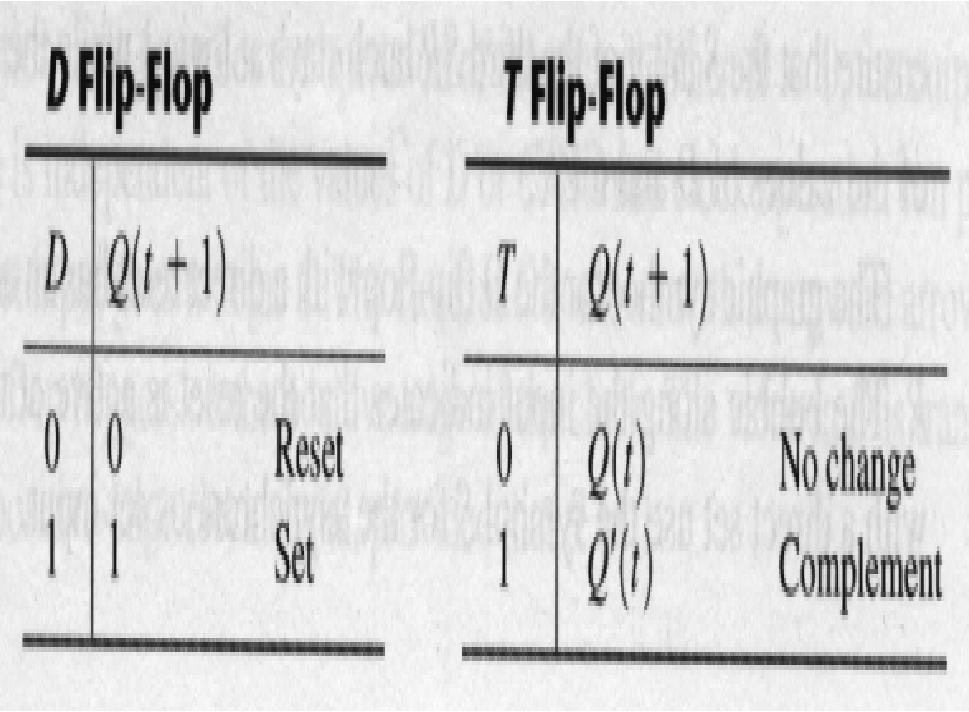
J	K	Q(t + 1)	
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q′(t)	Complement

T Flip-Flop

The T(toggle) flip-flop is a complementing flip-flop and can be obtained from a JK flip-flop when inputs J and K are tied together.



(a) From *JK* flip-flop



D Flip-Flop

D	$Q(t+1)$	
0	0	Reset
1	1	Set

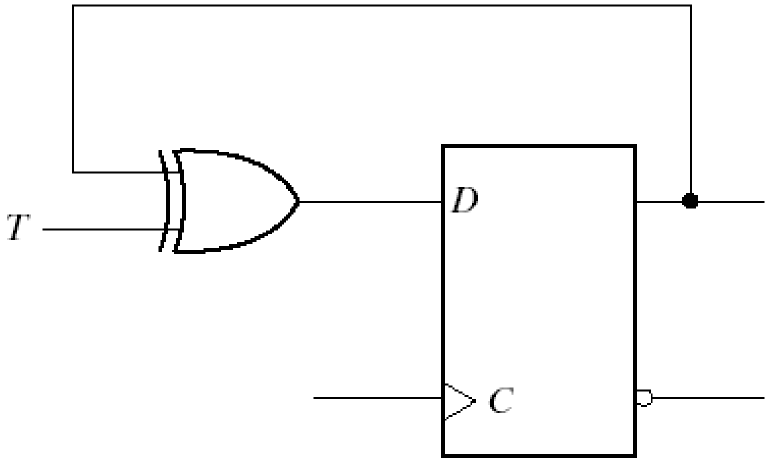
T Flip-Flop

T	$Q(t+1)$	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

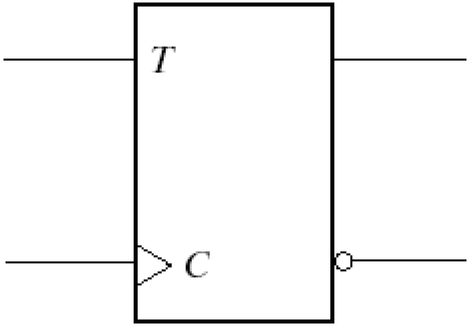
T Flip-Flop

The T flip-flop can be constructed with a D flip-flop and an exclusive-OR gates as shown in Fig. (b). The expression for the D input is

$$D = T \oplus Q = TQ` + T`Q$$



(b) From *D* flip-flop



(c) Graphic symbol

Characteristic Equations

D flip-flop Characteristic Equations

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

JK flip-flop Characteristic Equations

$$Q(t + 1) = JQ` + K`Q$$

T flip-flop Characteristic Equations

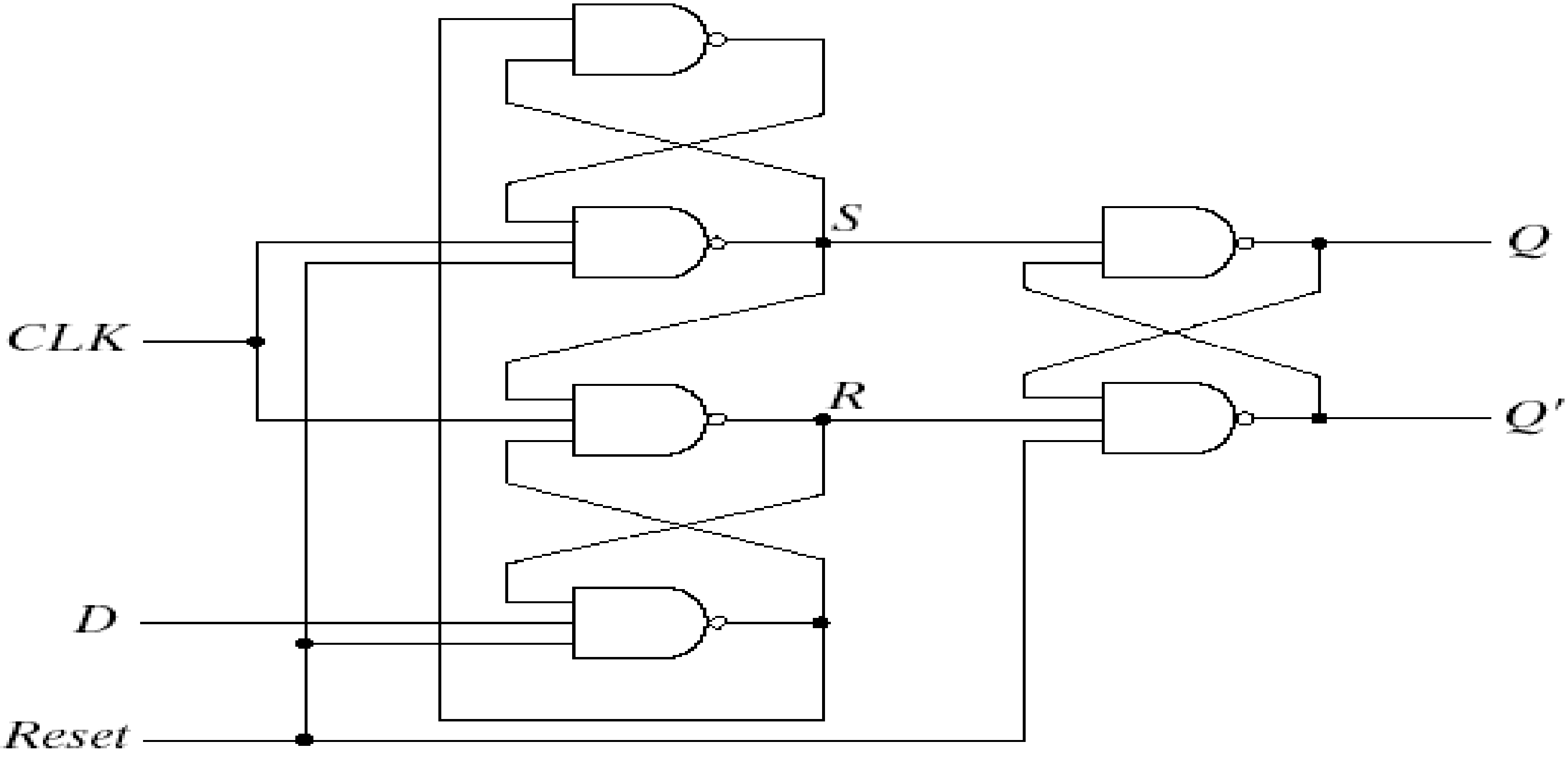
$$Q(t + 1) = T \oplus Q = TQ` + T`Q$$

Direct Inputs

Some flip-flops have asynchronous inputs that are used to force the flip-flop to a particular state independent of the clock. The input that sets the flip-flop to 1 is called present or direct set. The input that clears the flip-flop to 0 is called clear or direct reset. When power is turned on a digital system, the state of the flip-flops is unknown. The direct inputs are useful for bringing all flip-flops in the system to a known starting state prior to the clocked operation.

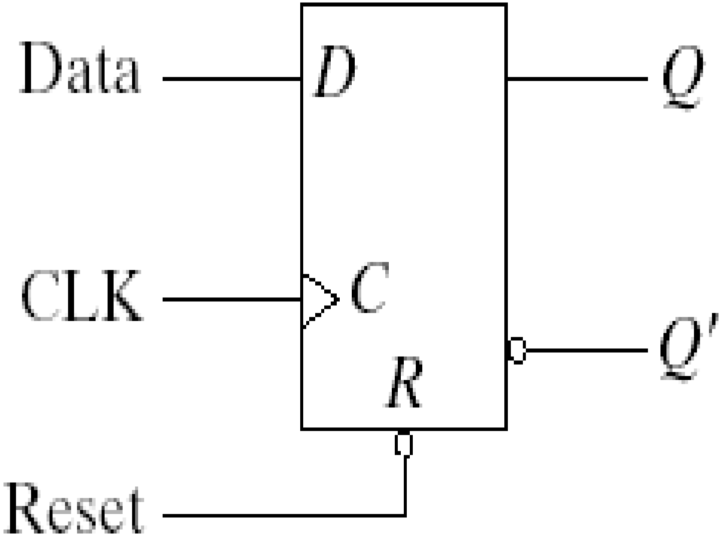
D Flip-Flop with Asynchronous Reset

A positive-edge-triggered D flip-flop with asynchronous reset is shown in Fig(a).



(a) Circuit diagram

D Flip-Flop with Asynchronous Reset



(b) Graphic symbol

R	C	D	Q	Q'
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

(b) Function table

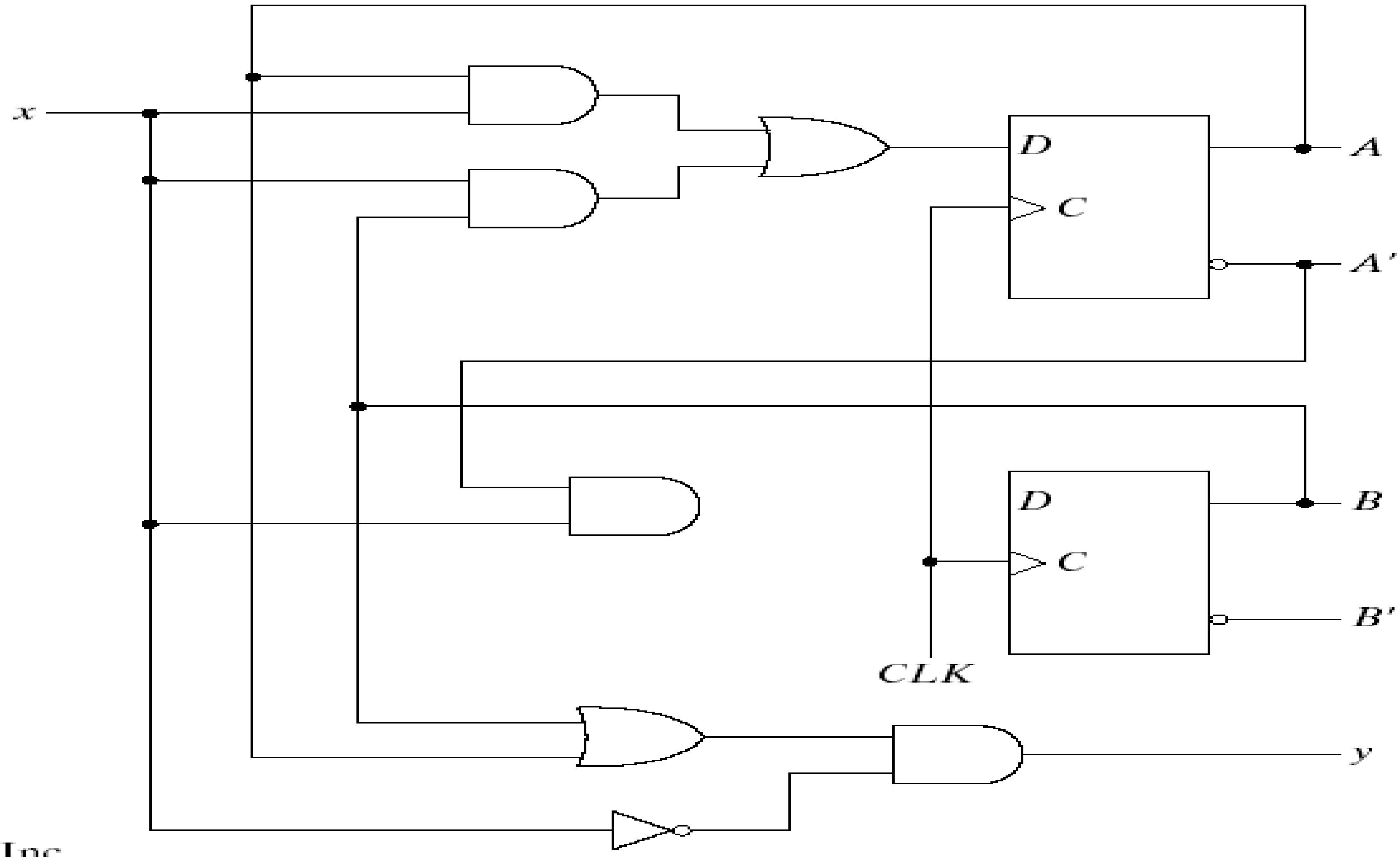
5-4 Analysis of Clocked Sequential Circuits

The analysis of a sequential circuit consists of obtaining a table or a diagram for the time sequence of inputs, outputs, and internal states. It is also possible to write Boolean expressions that describe the behavior of the sequential circuit. These expressions must include the necessary time sequence, either directly or indirectly.

State Equations

The behavior of a clocked sequential circuit can be described algebraically by means of state equations. A state equation specifies the next state as a function of the present state and inputs. Consider the sequential circuit shown in Fig. 5-15. It consists of two D flip-flops A and B, an input x and an output y.

Fig.5-15 Example of Sequential Circuit



Inc

State Equation

$$A(t+1) = A(t)\, x(t) + B(t)\, x(t)$$

$$B(t+1) = A`(t)\, x(t)$$

A state equation is an algebraic expression that specifies the condition for a flip-flop state transition. The left side of the equation with (t+1) denotes the next state of the flip-flop one clock edge later. The right side of the equation is Boolean expression that specifies the present state and input conditions that make the next state equal to 1.

$$Y(t) = (A(t) + B(t))\, x(t)`$$

State Table

The time sequence of inputs, outputs, and flip-flop states can be enumerated in a state table (sometimes called transition table).

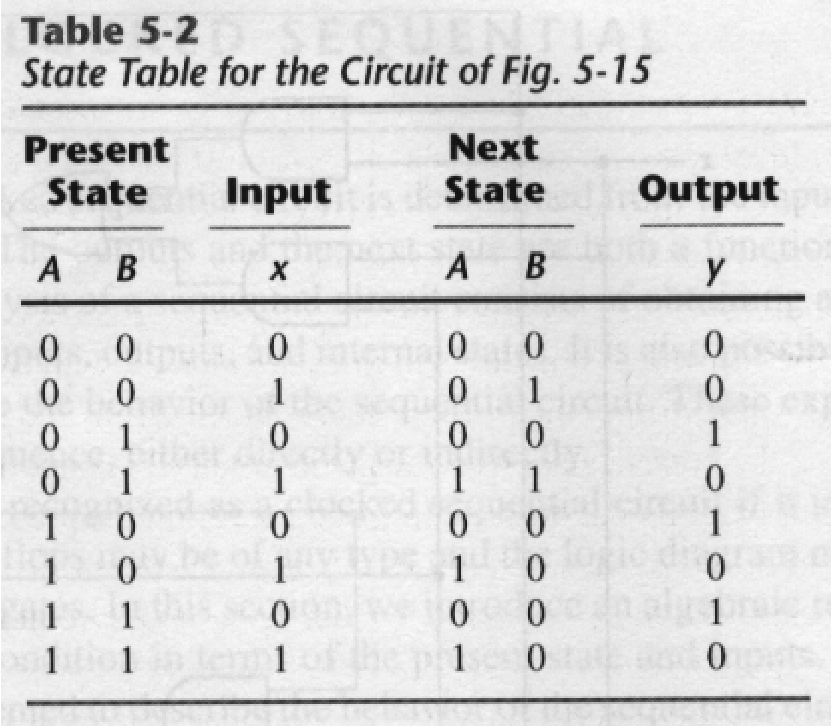


Table 5-2
State Table for the Circuit of Fig. 5-15

Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

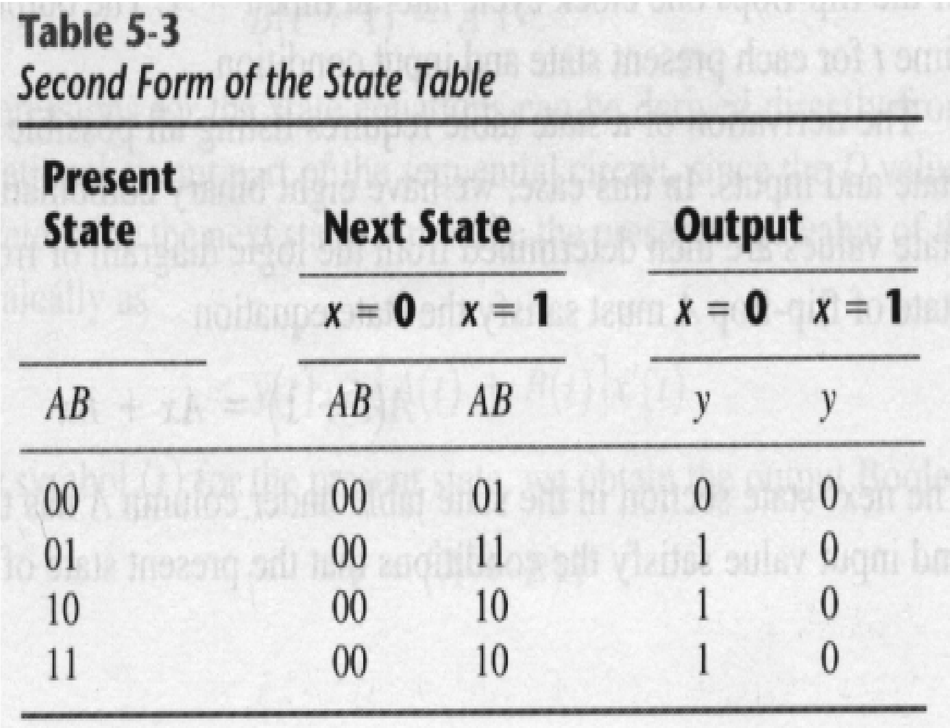


Table 5-3
Second Form of the State Table

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
AB	*AB*	*AB*	*y*	*y*
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

State Diagram

The information available in a state table can be represented graphically in the form of a state diagram. In this type of diagram, a state is represented by a circle, and the transitions between states are indicated by directed lines connecting the circles.

1/0 : means input =1
output=0

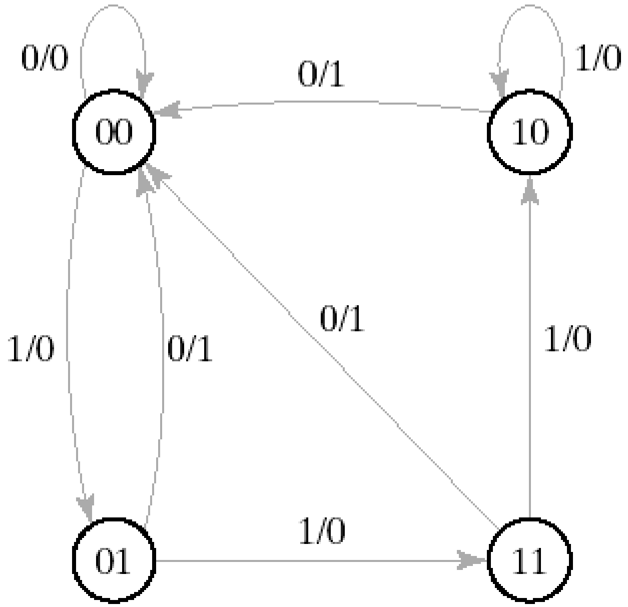


Fig. 5-16 State Diagram of the Circuit of Fig. 5-15

Flip-Flop Input Equations

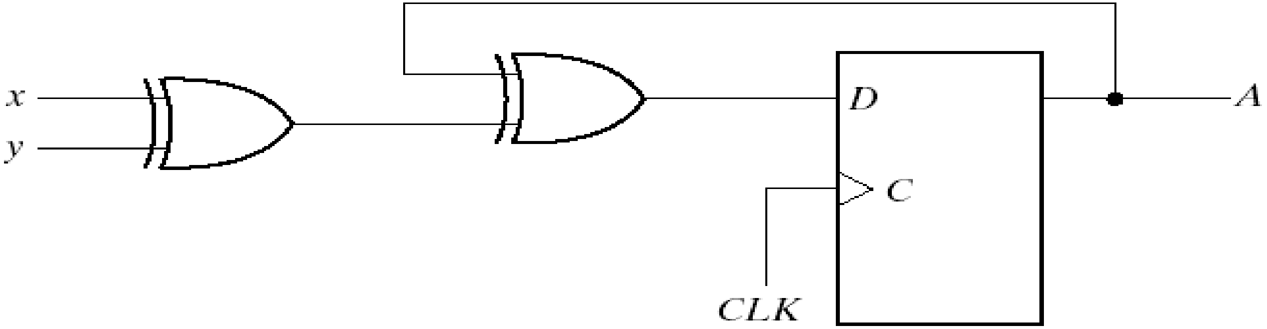
The part of the combinational circuit that generates external outputs is descirbed algebraically by a set of Boolean functions called output equations. The part of the circuit that generates the inputs to flip-flops is described algebraically by a set of Boolean functions called flip-flop input equations. The sequential circuit of Fig. 5-15 consists of two D flip-flops A and B, an input x, and an output y. The logic diagram of the circuit can be expressed algebraically with two flip-flop input equations and an output equation:

$$D_A = Ax + Bx$$
$$D_B = A`x$$
$$y = (A + B)x`$$

Analysis with D Flip-Flop

The circuit we want to analyze is described by the input equation $D_A = A \oplus x \oplus y$

The D_A symbol implies a D flip-flop with output A. The x and y variables are the inputs to the circuit. No output equations are given, so the output is implied to come from the output of the flip-flop.

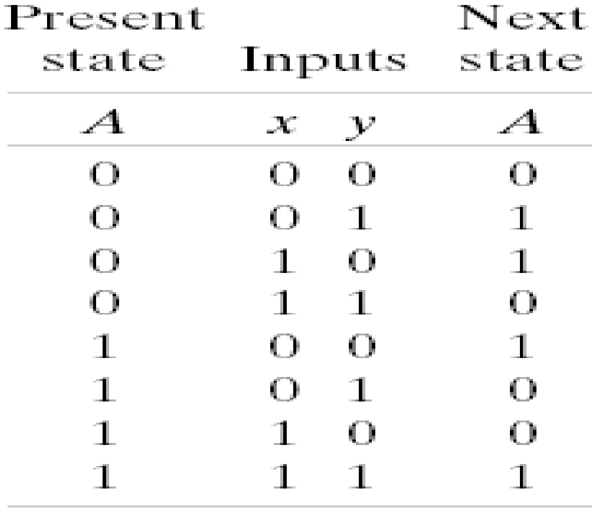


(a) Circuit diagram

Analysis with D Flip-Flop

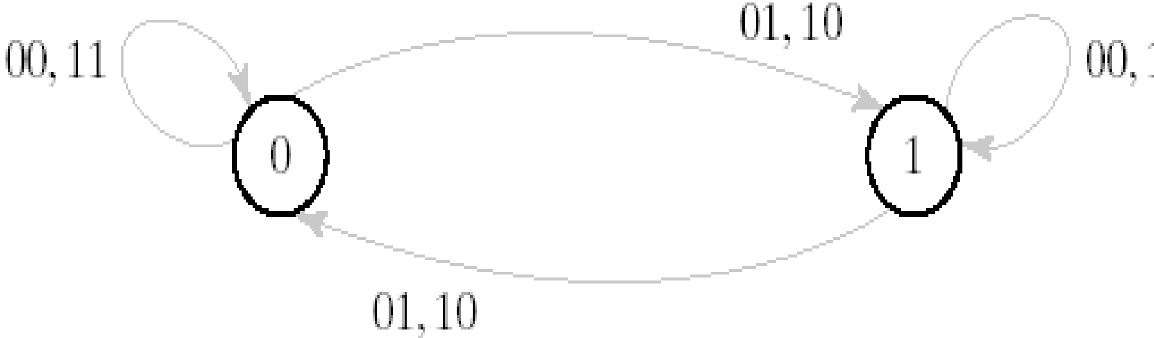
The binary numbers under Axy are listed from 000 through 111 as shown in Fig. 5-17(b). The next state values are obtained from the state equation $A(t+1) = A \oplus x \oplus y$

The state diagram consists of two circles-one for each state as shown in Fig. 5-17(c)



Present state	Inputs		Next state
A	*x*	*y*	*A*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) State table



(c) State diagram

Analysis with JK Flip-Flops

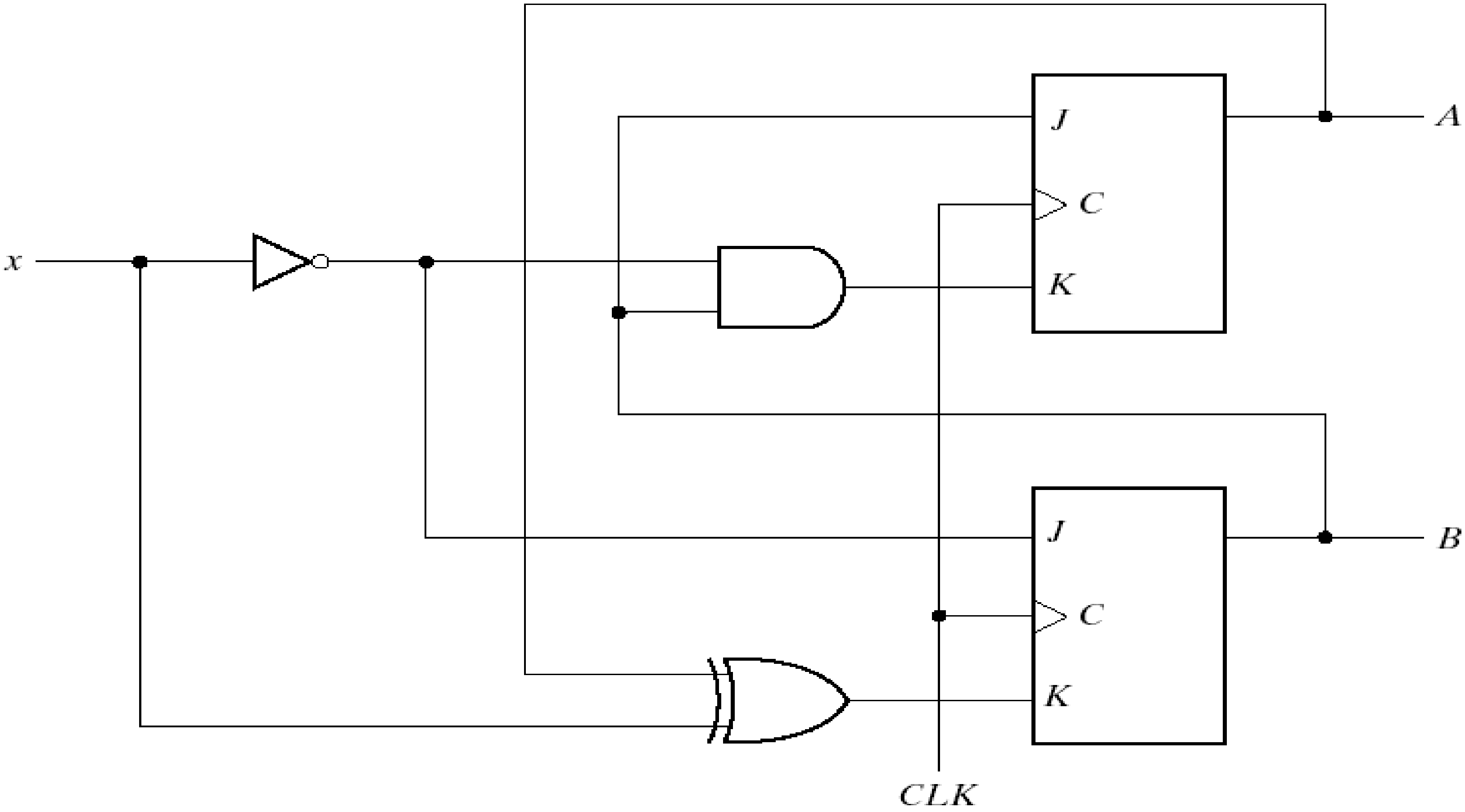


Fig. 5-18 Sequential Circuit with *JK* Flip-Flop

Analysis with JK Flip-Flop

The circuit can be specified by the flip-flop input equations

$J_A = B \qquad K_A = Bx`$

$J_B = x` \qquad K_B = A`x + Ax` = A \oplus x$

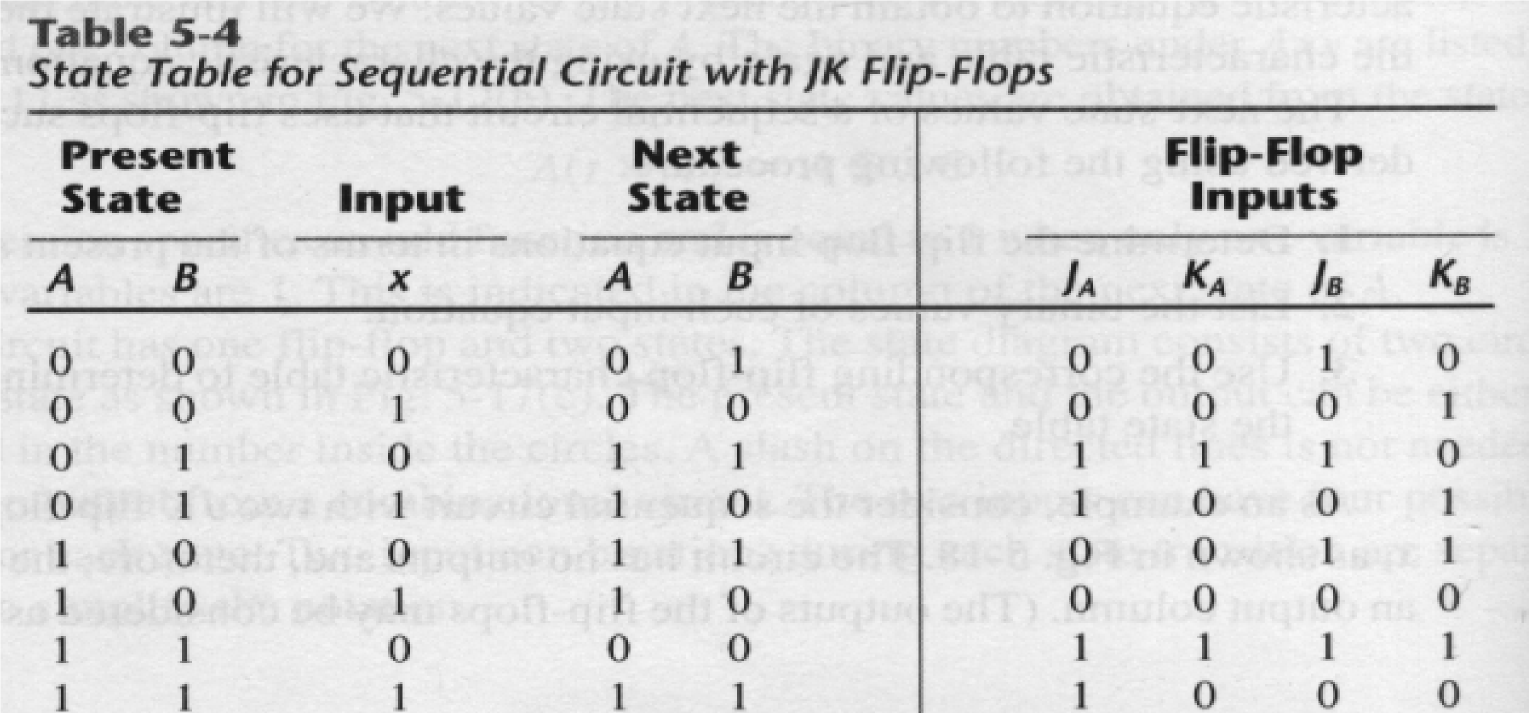


Table 5-4
State Table for Sequential Circuit with JK Flip-Flops

Present State		Input	Next State		Flip-Flop Inputs			
A	*B*	*x*	*A*	*B*	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

Analysis with JK Flip-Flops

$$A(t + 1) = JA` + K`A$$
$$B(t + 1) = JB` + K`B$$

Substituting the values of JA and KA from the input equations, we obtain the state equation for A:

$$A(t + 1) = BA` + (Bx`)`A = A`B + AB` + Ax$$

The state equation provides the bit values for the column under next state of A in the state table. Similarly, the state equation for flip-flop B can be derived from the characteristic equation by substituting the values of J_B and K_B:

$$B(t + 1) = x`B` + (A \oplus x)`B = B`x` + ABx + A`Bx`$$

Analysis with JK Flip-Flops

The state diagram of the sequential circuit is shown in Fig. 5-19.

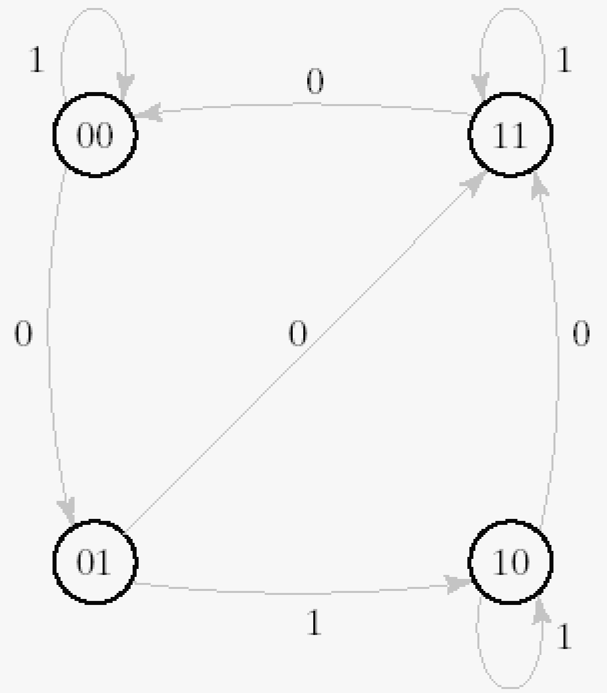
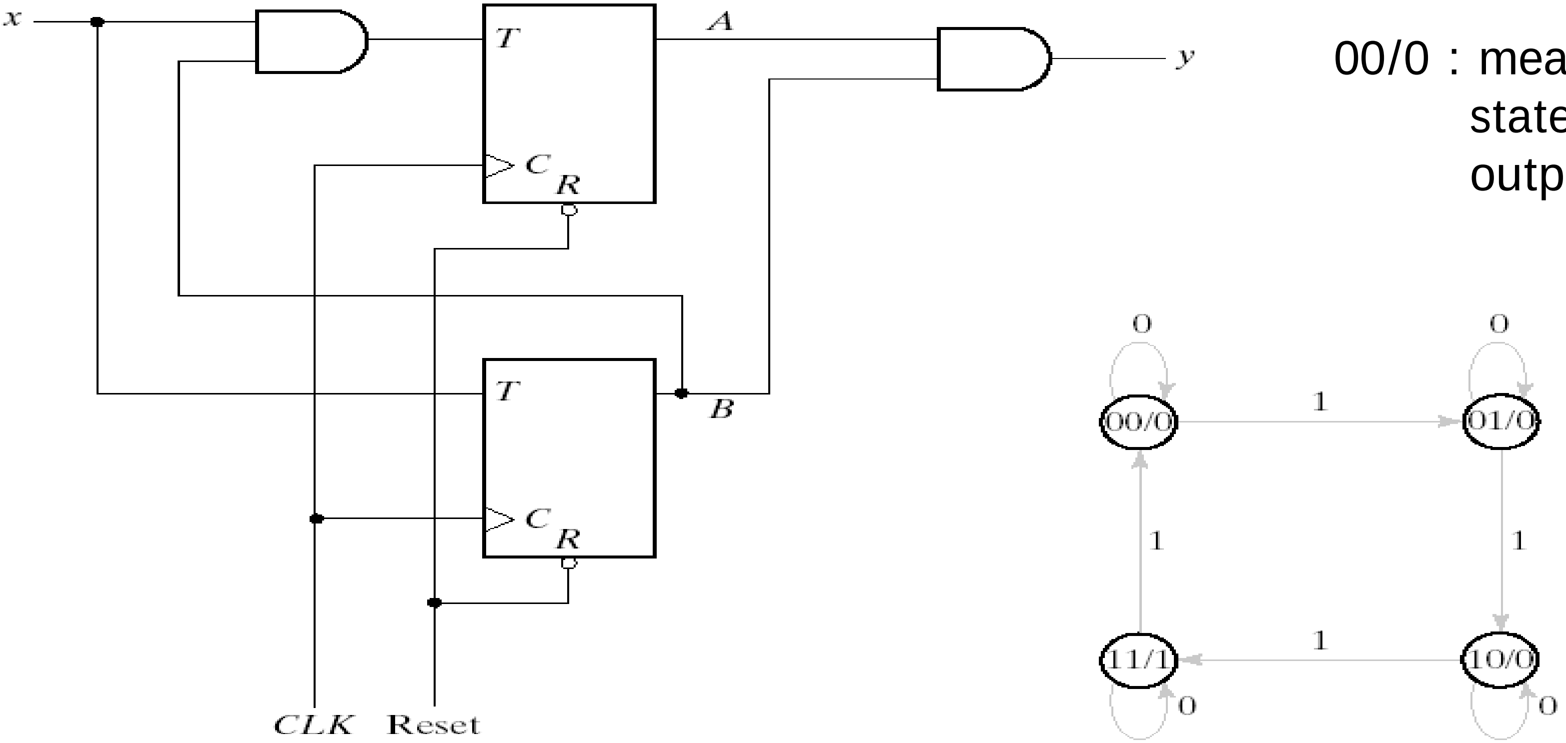


Fig. 5-19 State Diagram of the Circuit of Fig. 5-18

Analysis With T Flip-Flops

Characteristic equation

$$Q(t+1) = T \oplus Q = T`Q + TQ`$$



(a) Circuit diagram

(b) State diagram

00/0 : means
state is 00
output is 0

Analysis With T Flip-Flops

Consider the sequential circuit shown in Fig. 5-20. It has two flip-flops A and B, one input x, and one output y. It can be described algebraically by two input equations and an output equation:

$T_A = Bx$

$T_B = x$

$y = AB$

$A(t+1)=(Bx)'A+(Bx)A'$

$=AB'+Ax'+A'Bx$

$B(t+1)=x\oplus B$

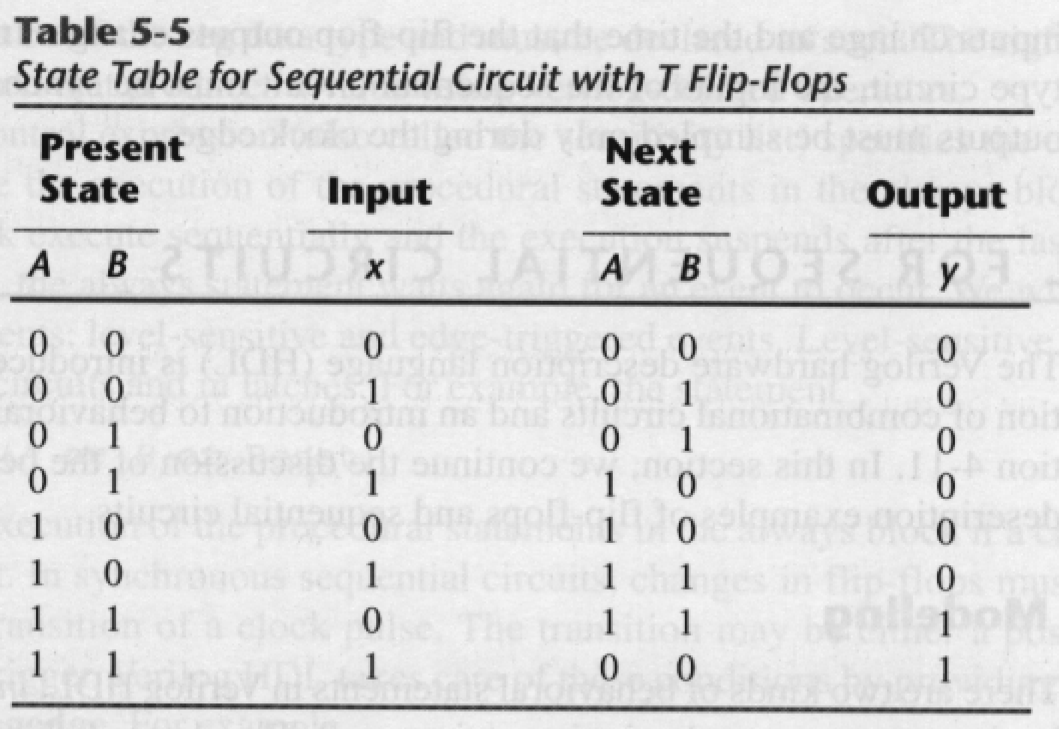


Table 5-5

State Table for Sequential Circuit with T Flip-Flops

Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

Mealy and Moore Models

The most general model of a sequential circuit has inputs, outputs, and internal states. It is customary to distinguish between two models of sequential circuits: the Mealy model and the Moore model. They differ in the way the output is generated. In the Mealy model, the output is a function of both the present state and input. In the Moore model, the output is a function of the present state only. When dealing with the two models, some books and other technical sources refer to a sequential circuit as a finite state machine abbreviated FSM. The Mealy model of a sequential circuit is referred to as a Mealy FSM or Mealy machine. The Moore model is refereed to as a Moore FSM or Moore machine.

5-5 HDL For Sequential Circuit

The Verilog hardware description language (HDL) is introduced in Section 3-9. The description of combinational circuits and an introduction to behavioral modeling is presented in Section 4-11. In this section, we continue the discussion of the behavioral modeling and present description examples of flip-flops and sequential circuits.

Behavioral Modeling

There are two kinds of behavioral statements in Verilog HDL: **initial** and **always**.

```
initial
   begin
         clock = 1`b0;
         repeat  (30)
         #10 clock =  ~clock;
   end
```

```
initial
   begin
         clock = 1`b0;
         #300  $finish;
   end
always
         #10 clock =  ~clock;
```

Behavioral Modeling

The **always** statement can be controlled by delays that wait for a certain time or by certain conditions to become true or by events to occur.

always @(event control expression)
procedural assignment statements.

always @(A **or** B **or** Reset)

always @(**posedge** clock or **negedge** reset)

Flip-Flops and Latches

HDL Example 5-1

```
//Description of D latch (See Fig. 5-6)
module D_latch (Q, D, control);
      output Q;
      input D, control;
      reg Q;
      always @(control or D)
      if (control) Q = D;         //Same as: if (control ==1)
endmodule
```

Flip-Flops and Latches

HDL Example 5-2

```
//D flip-flop
module D_FF (Q, D, CLK);
    output Q;
    input D, CLK;
    reg Q;
    always @(posedge CLK)
       Q = D;
endmodule

// D flip-flop with asynchronous reset.
module DFF (Q, D, CLK, RST);
       output Q;
       input D, CLK, RST;
       reg Q;
       always @(posedge CLK or negedge RST)
          if (~RST) Q = 1`b0;     // Same as: if (RST ==0)
          else Q = D;
endmodule
```

Flip-Flops and Latches

HDL Example 5-3

```
//T flip-flop from D flip-flop and gates
module TFF (Q, T, CLK, RST);
    output Q;
    input T, CLK, RST;
    reg DT;
    assign DT = Q ^ T;
//Instantiate the D flip-flop
  DFF TF1 (Q, DT, CLK, RST);
Endmodule

// JK flip-flop from D flip-flop and gates
module JKFF (Q, J, K, CLK, RST);
       output Q;
       input J, K, CLK, RST;
       wire JK;
       assign JK = (J&~Q) | (~K & Q);
// Instantiate D flipflop
   DFF JK1 (Q, JK, CLK, RST);
endmodule
```

Flip-Flops and Latches

```
module DFF (Q, D, CLK, RST);
        output Q;
        input D, CLK, RST;
        reg Q;
        always @(posedge CLK or negedge RST)
           if (~RST) Q = 1`b0;     // Same as: if (RST==0)
           else Q = D;
endmodule
```

Flip-Flops and Latches

HDL Example 5-4

```
// Functional description of JK flip-flop
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
```

State Diagram

HDL Example 5-5

```
//Mealy state diagram (Fig. 5-16)
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;   //Initialize to state S0
         else Prstate = Nxtstate;  //Clock operations
      always @ (Prstate or x)
            case (Prstate)
```

State Diagram

```
      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)       //Evaluate output
    case (Prestate)
       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
```

State Diagram

HDL Example 5-6

```
//Moore state diagram (Fig. 5-19)
module  Moore_md1 (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;   //Initialize to 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;                //Output of flip-flops
endmodule
```

Structural Description

HDL Example 5-7

```
//Structural description of sequential circuit
//See Fig. 5-20 (a)
module Tcircuit (x, y, A, B, CLK, RST);
    input x, CLK, RST;
    output y, A, B;
    wire TA, TB;
//Flip-flop input equations
    assign TB = x,
           TA = x & B;
//Output equation
    assign y = A &B;
//Instantiate T flip-flops
    T_FF BF (B, TB, CLK, RST);
    T_FF AF (A, TA, CLK, RST);
endmodule
```

Structural Description

```
//T flip-flop
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
```

Structural Description

```
//Stimulus for testing sequential circuit
module testTcircuit;
    reg x, CLK, RST;    //inputs for circuit
    wire y, A, B;         //output from circuit
Tcircuit TC (x, y, A, B, CLK, RST); //instantiate circuit
 initial
    begin
          RST = 0;
          CLK = 0;
    #5  RST = 1;
          repeat (16)
    #5  CLK = ~CLK;
     end
 initial
    begin
           x = 0;
    #15  x = 1;
           repeat (8)
    #10  x = ~x;
    end
endmodule
```

Structural Description

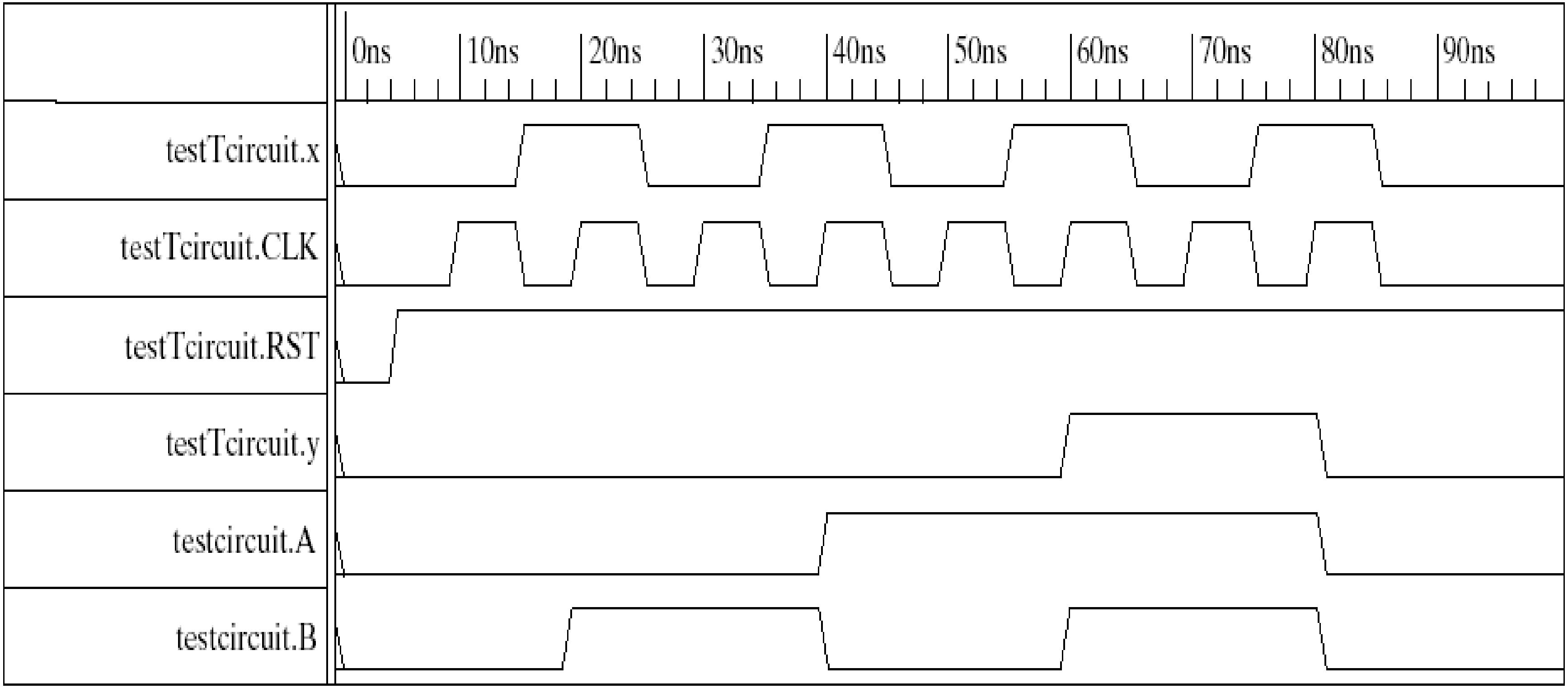


Fig. 5-21 Simulation Output of HDL Example 5-7

5-6 State Reduction and Assignment

The analysis of sequential circuits starts from a circuit diagram and culminates in a state table or diagram. The design of a sequential circuit starts from a set of specifications and culminates discusses certain properties of sequential circuits that may be used to reduce the number of gates and flip-flops during the design.

State Reduction

The reduction of the number of flip-flops in a sequential circuit is referred to as the state-reduction problem. State-reduction algorithms are concerned with procedures for reducing the number of states in a state table, while keeping the external input-output requirements unchanged. Since m flip-flops produce 2^m states, a reduction in the number of states may result in a reduction in the number of flip-flops. An unpredictable effect in reducing the number of flip-flops is that sometimes the equivalent circuit may require more combinational gates.

State Reduction

Example：

state	a	a	b	c	d	e	f	f	g	f	g	a
input	0	1	0	1	0	1	1	0	1	0	0	
output	0	0	0	0	0	1	1	0	1	0	0	

Initial point

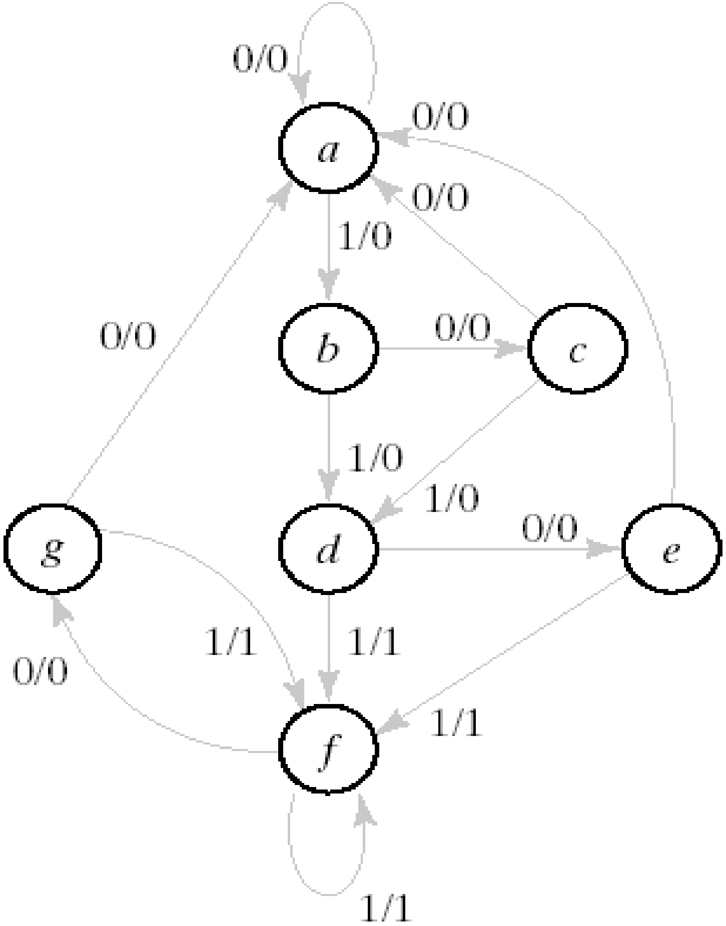


Fig. 5-22 State Diagram

State Reduction

We now proceed to reduce the number of states for this example. First, we need the state table; it is more convenient to apply procedures for state reduction using a table rather than a diagram. The state table of the circuit is listed in Table 5-6 and is obtained directly from the state diagram.

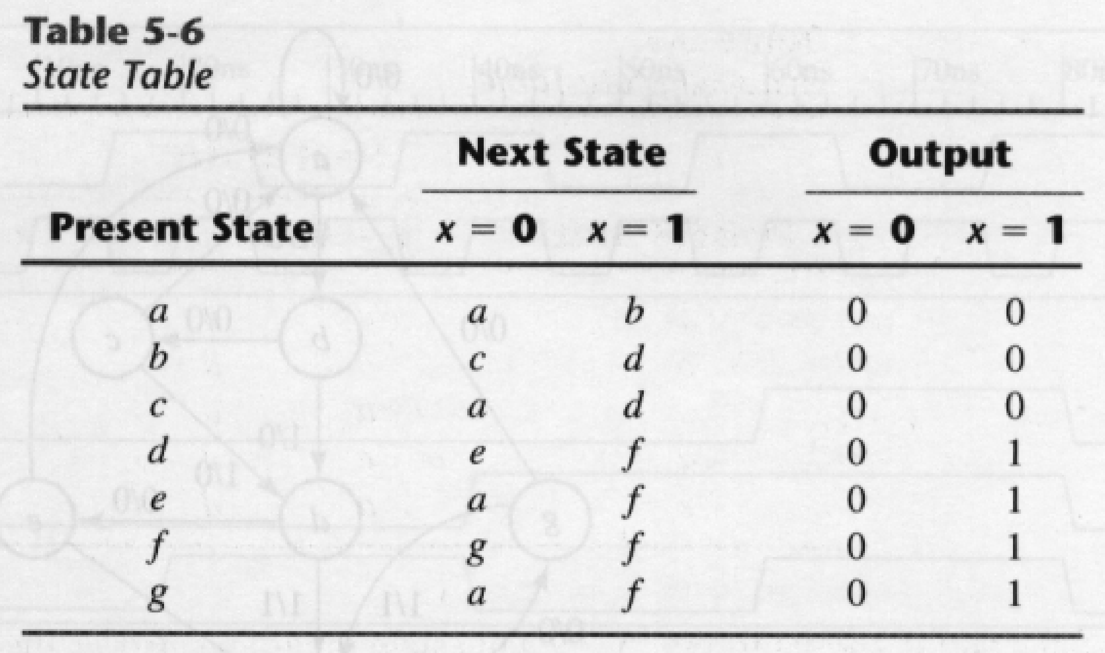


Table 5-6
State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

State Reduction

States g and e are two such states: they both go to states a and f and have outputs of 0 and 1 for x=0 and x=1, respectively. Therefore, states g and e are equivalent and one of these states can be removed. The procedure of removing a state and replacing it by its equivalent is demonstrated in Table 5-7. The row with present g is removed and state g is replaced by state e each time it occurs in the next-state columns.

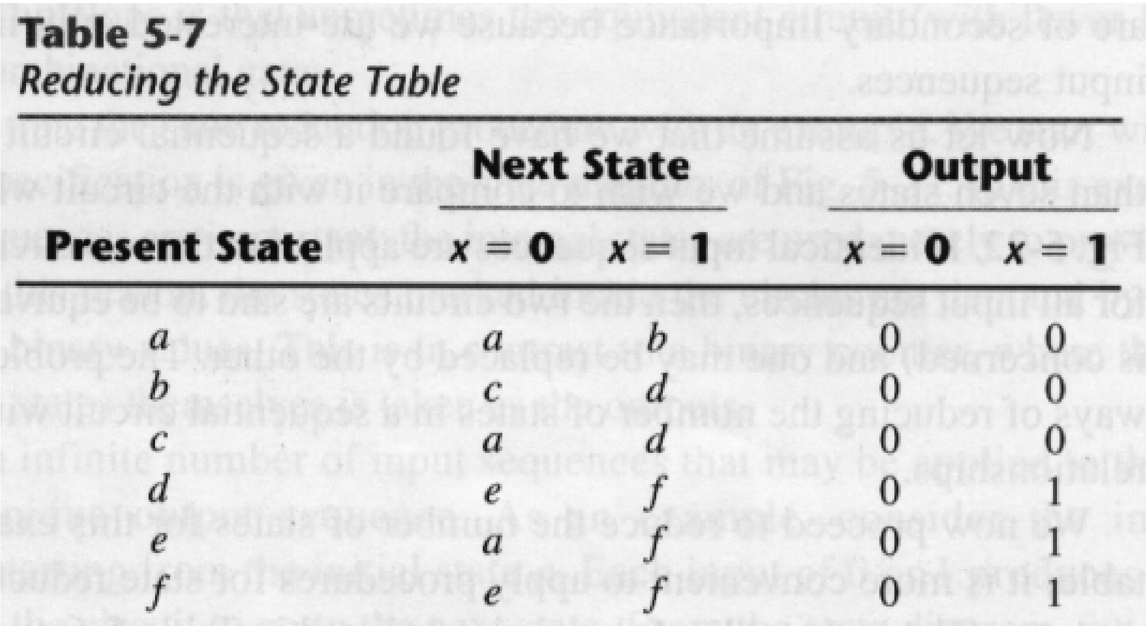


Table 5-7
Reducing the State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*f*	0	1
e	*a*	*f*	0	1
f	*e*	*f*	0	1

State Reduction

Present state f now has next states e and f and outputs 0 and 1 for x=0 and x=1, respectively. The same next states and outputs appear in the row with present state d. Therefore, states f and d are equivalent and state f can be removed and replaced by d. The final reduced table is shown in Table 5-8. The state diagram for the reduced table consists of only five states and is shown in Fig. 5-23.

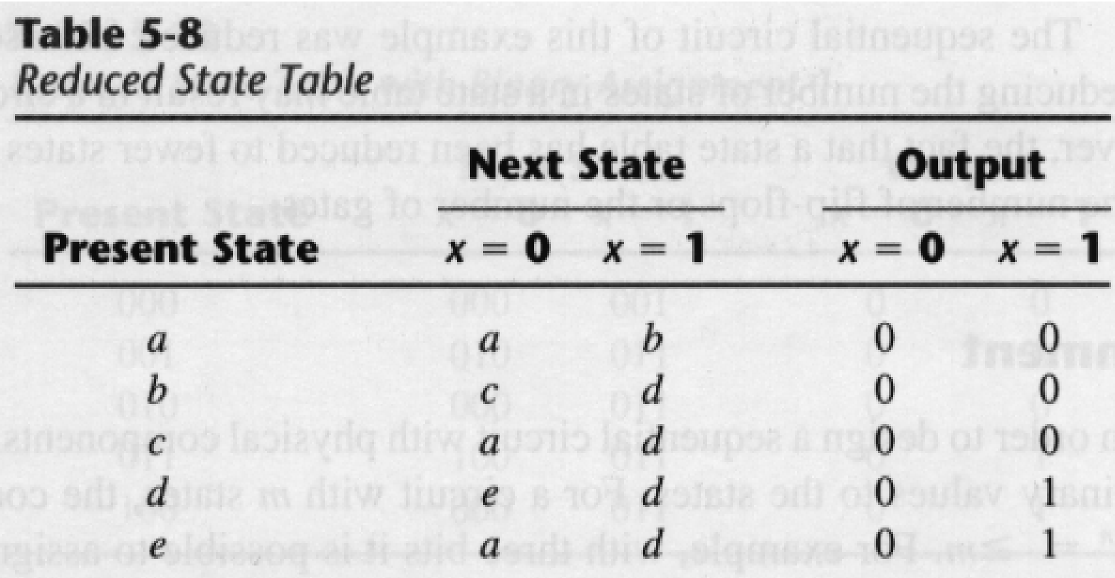
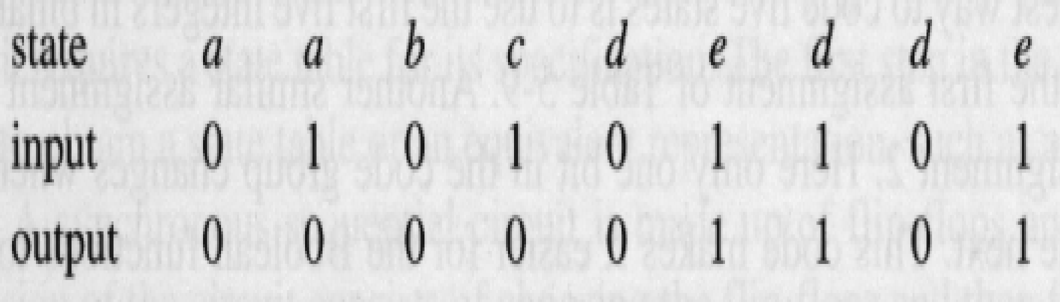


Table 5-8
Reduced State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*d*	0	1
e	*a*	*d*	0	1

State Reduction



state	*a*	*a*	*b*	*c*	*d*	*e*	*d*	*d*	*e*
input	0	1	0	1	0	1	1	0	1
output	0	0	0	0	0	1	1	0	1

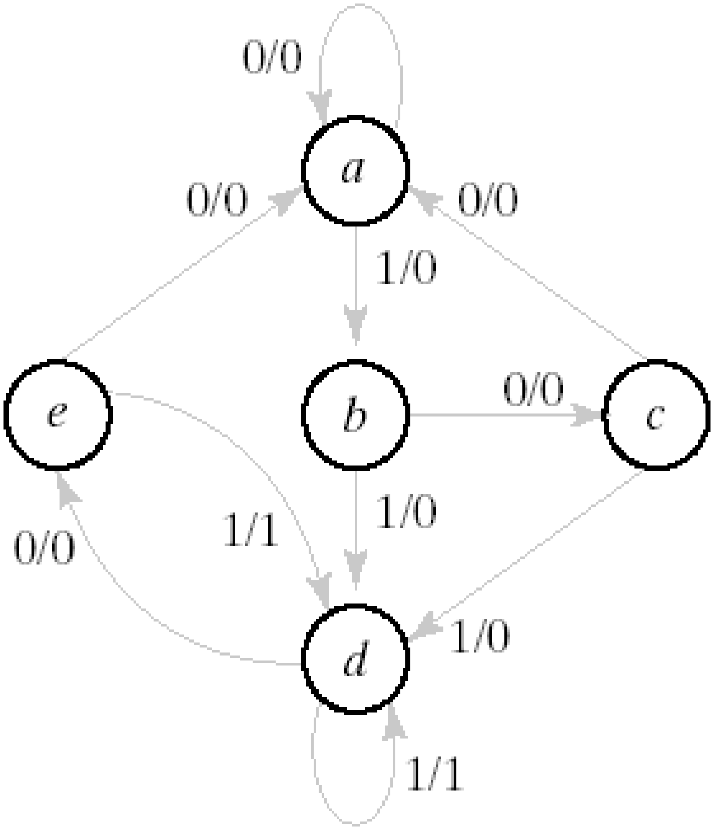


Fig. 5-23 Reduced State Diagram

State Assignment

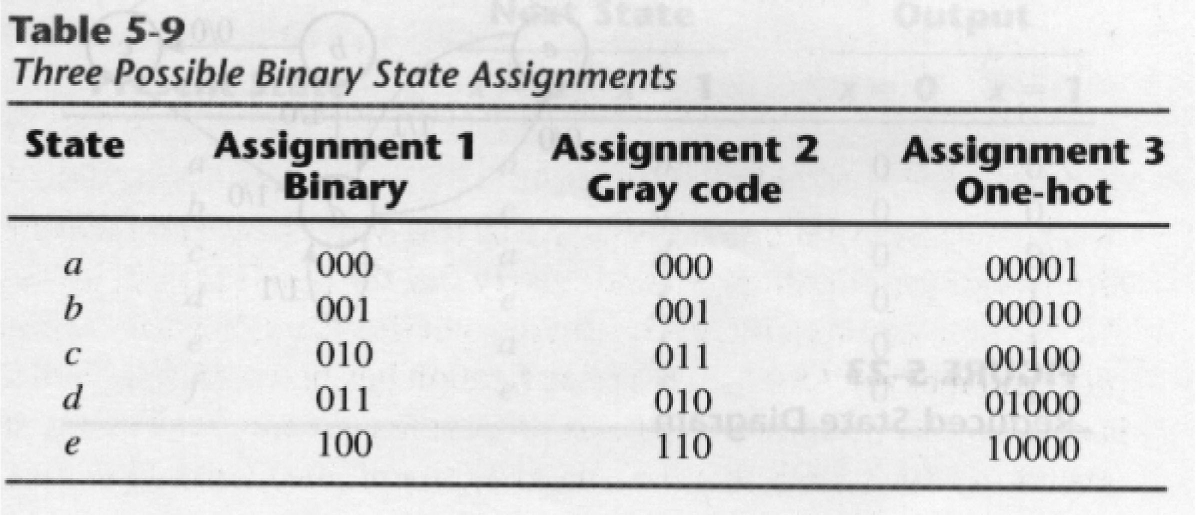


Table 5-9
Three Possible Binary State Assignments

State	Assignment 1 Binary	Assignment 2 Gray code	Assignment 3 One-hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

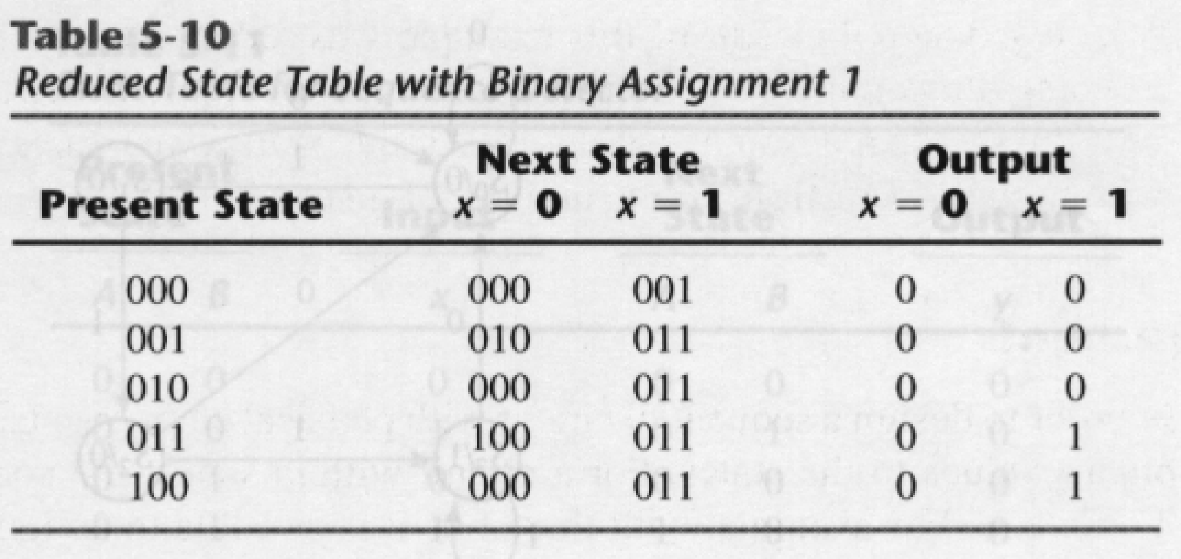


Table 5-10
Reduced State Table with Binary Assignment 1

	Next State		Output	
Present State	*x* = 0	*x* = 1	*x* = 0	*x* = 1
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

5-7 Design Procedure

The procedure for designing synchronous sequential circuits can be summarized by a list of recommended steps.

1. From the word description and specifications of the desired operation, derive a state diagram for the circuit.
2. Reduce the number of states if necessary.
3. Assign binary values to the states.
4. Obtain the binary-coded state table.
5. Choose the type of flip-flops to be used.
6. Derive the simplified flip-flop input equations and output equations.
7. Draw the logic diagram.

Design Procedure

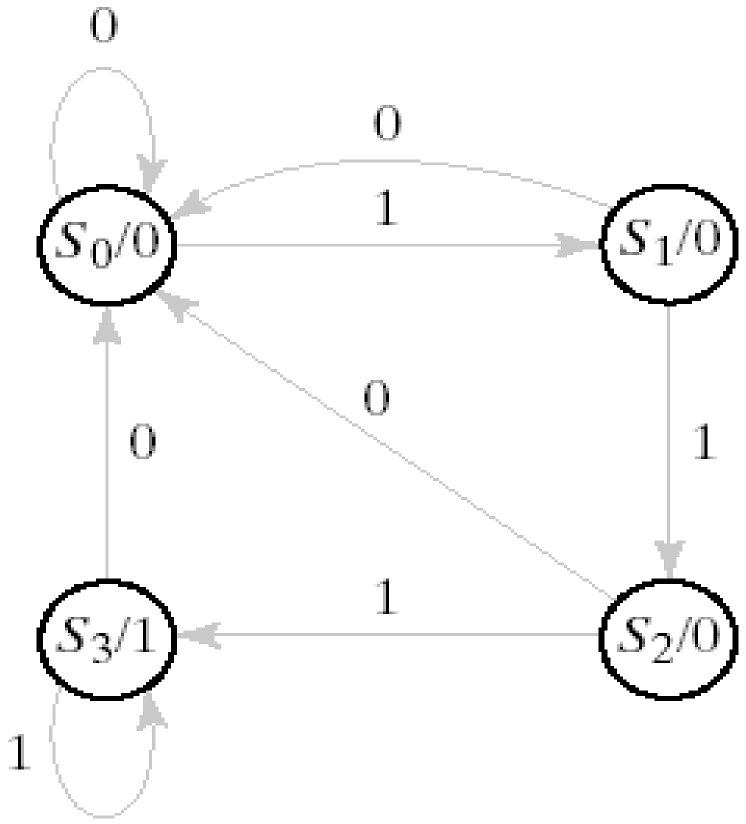


Fig. 5-24 State Diagram for Sequence Detector

Synthesis Using D Flip-Flops

$$A(t + 1) = D_A(A, B, x) = ?\ (3, 5, 7)$$

$$B(t + 1) = D_B(A, B, x) = ?\ (1, 5, 7)$$

$$y(A, B, x) = ?\ (6, 7)$$

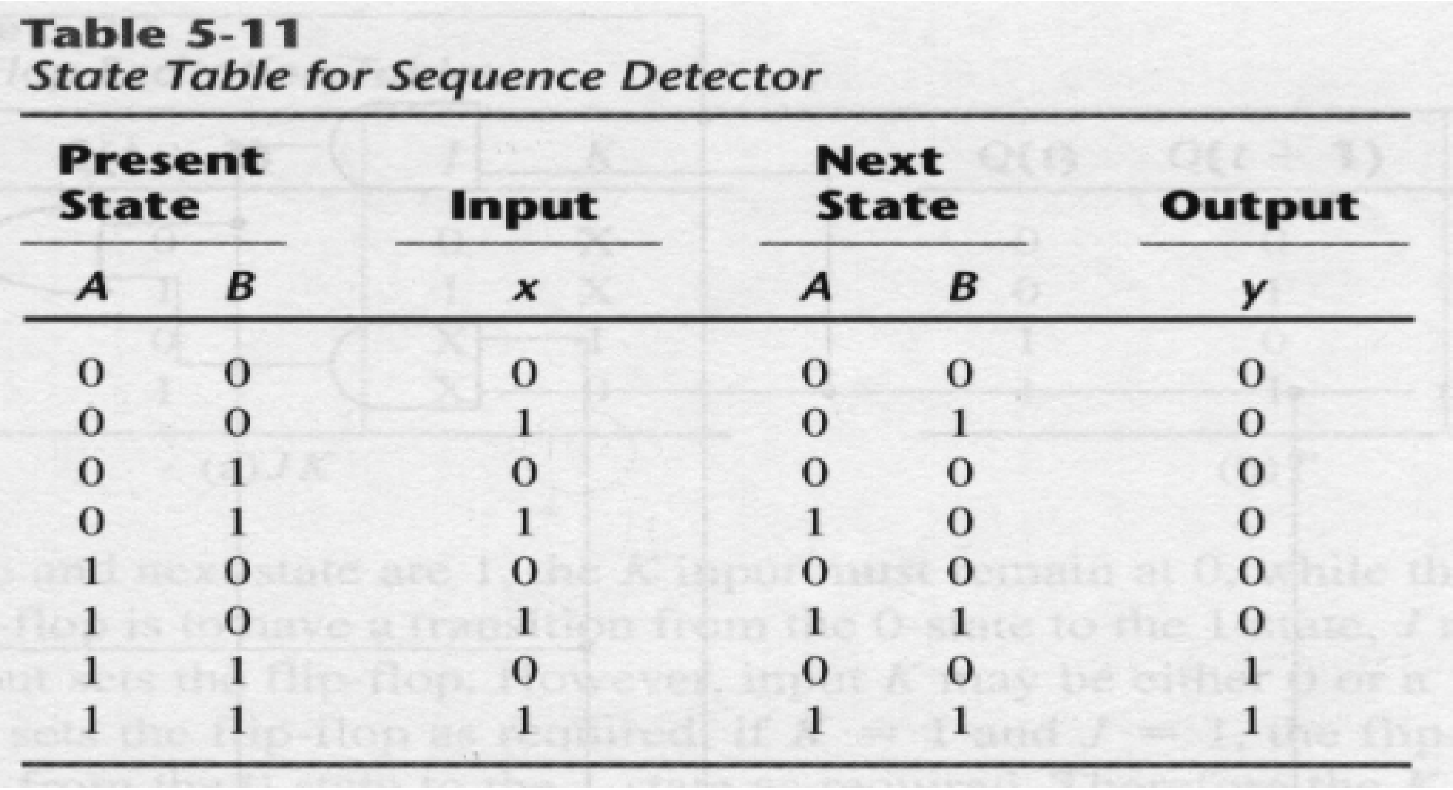


Table 5-11

State Table for Sequence Detector

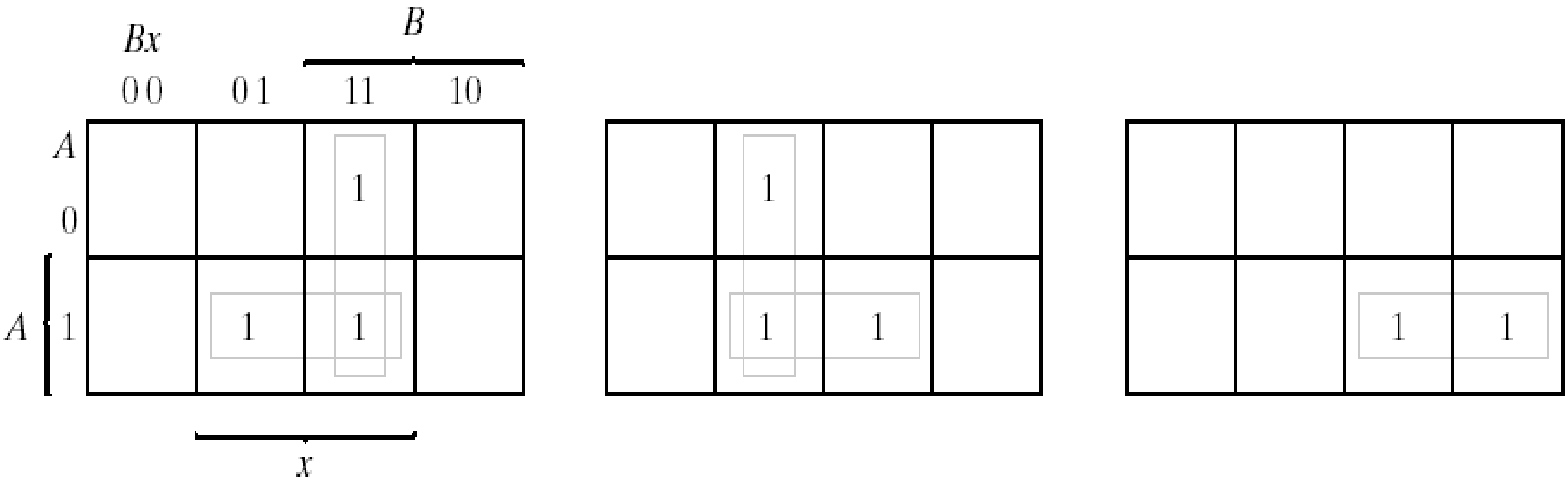
Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Synthesis Using D Flip-Flops

D_A = Ax +Bx

D_B = Ax + B`x

y = AB



$D_A = Ax + Bx$ $D_B = Ax + B'x$ $y = AB$

Fig. 5-25 Maps for Sequence Detector

Synthesis Using D Flip-Flops

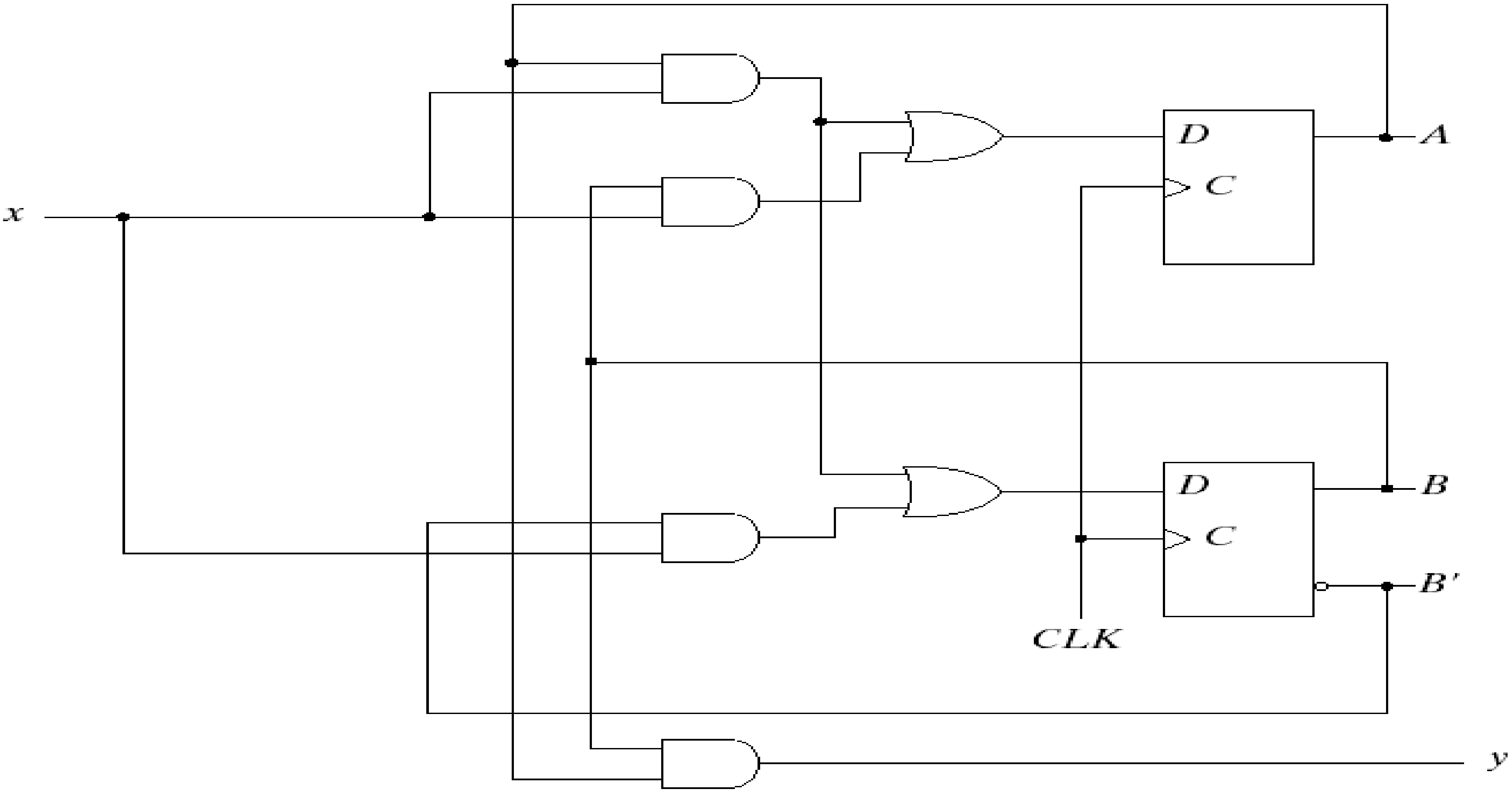


Fig. 5-26 Logic Diagram of Sequence Detector

Synthesis Using JK Flip-Flops

Different from Table 5-11 !!

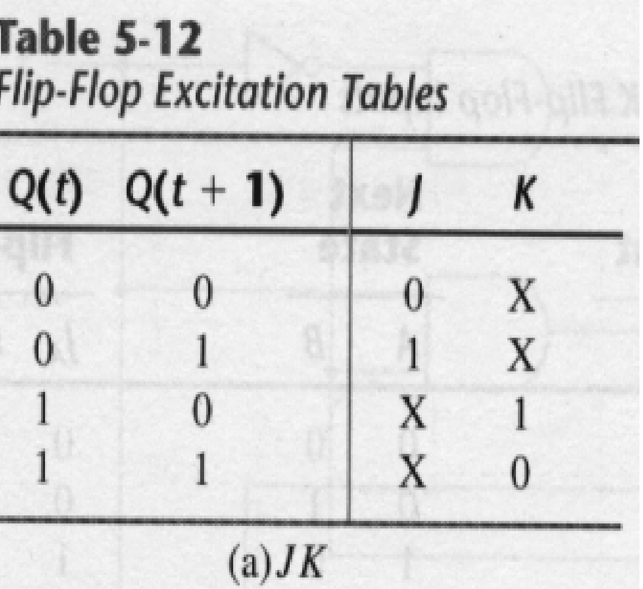


Table 5-12

Flip-Flop Excitation Tables

$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(a) JK

Ref. Table 5-1

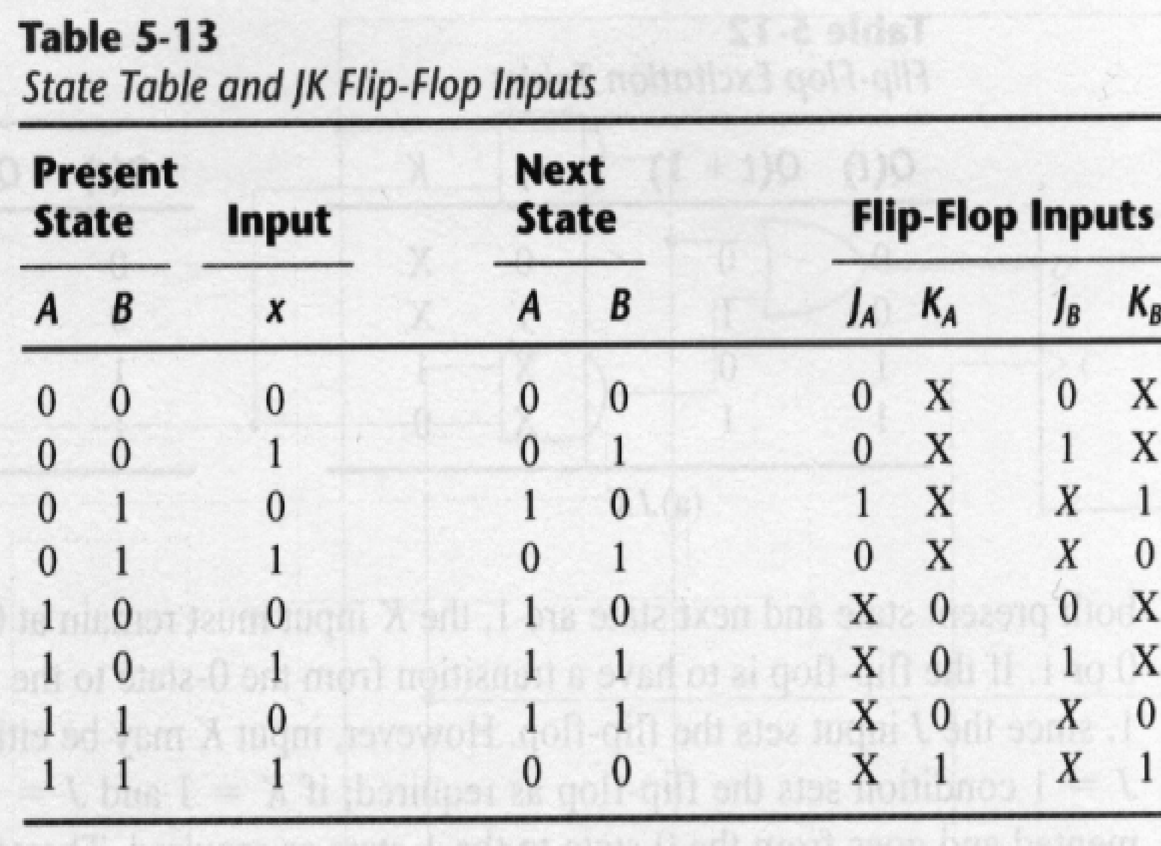
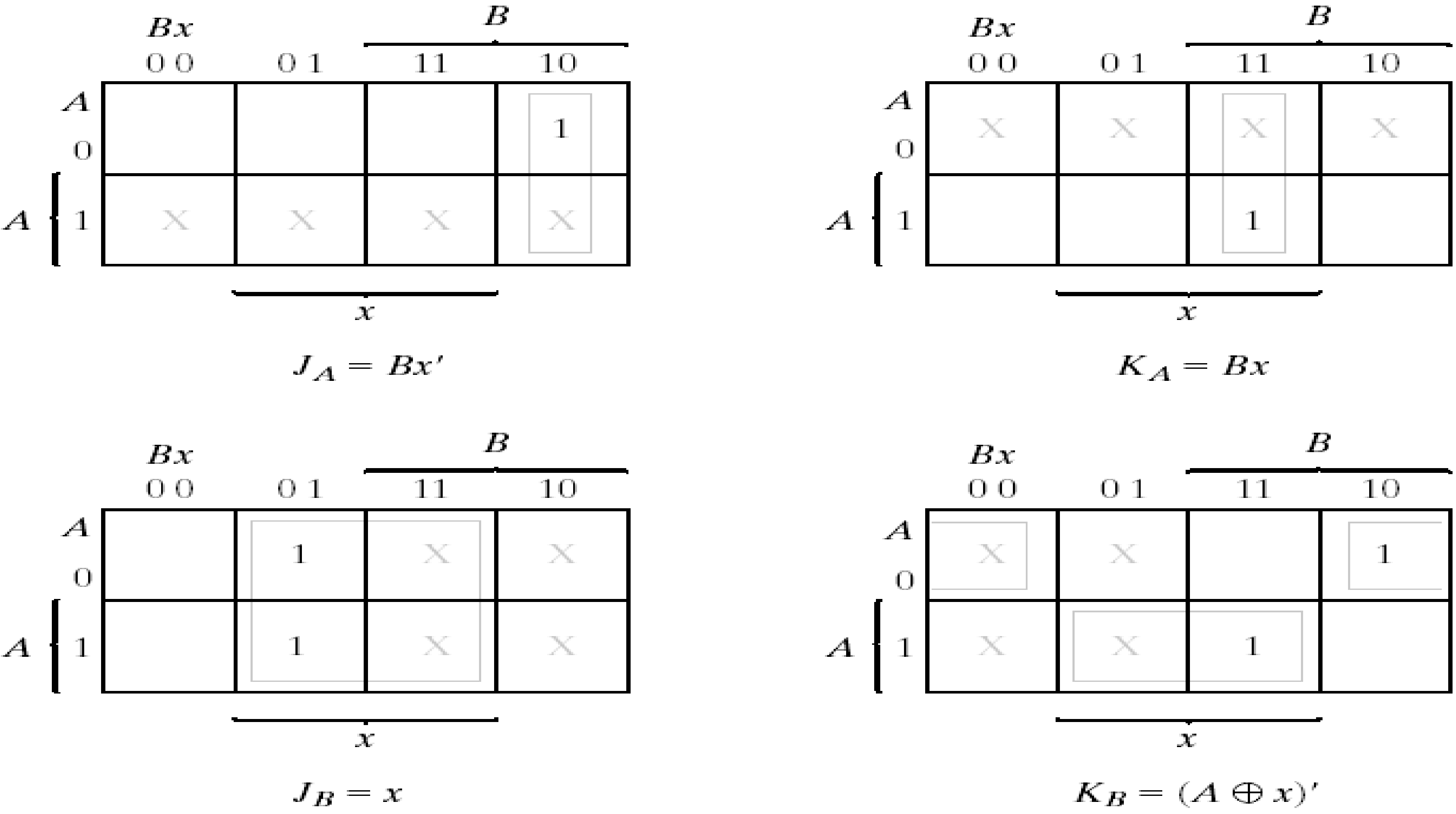


Table 5-13

State Table and JK Flip-Flop Inputs

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Synthesis Using JK Flip-Flops



A \ Bx	00	01	11	10
0				1
1	X	X	X	X

$J_A = Bx'$

A \ Bx	00	01	11	10
0	X	X	X	X
1			1	

$K_A = Bx$

A \ Bx	00	01	11	10
0		1	X	X
1		1	X	X

$J_B = x$

A \ Bx	00	01	11	10
0	X	X		1
1	X	X	1	

$K_B = (A \oplus x)'$

Fig. 5-27 Maps for *J* and *K* Input Equations

Synthesis Using JK Flip-Flops

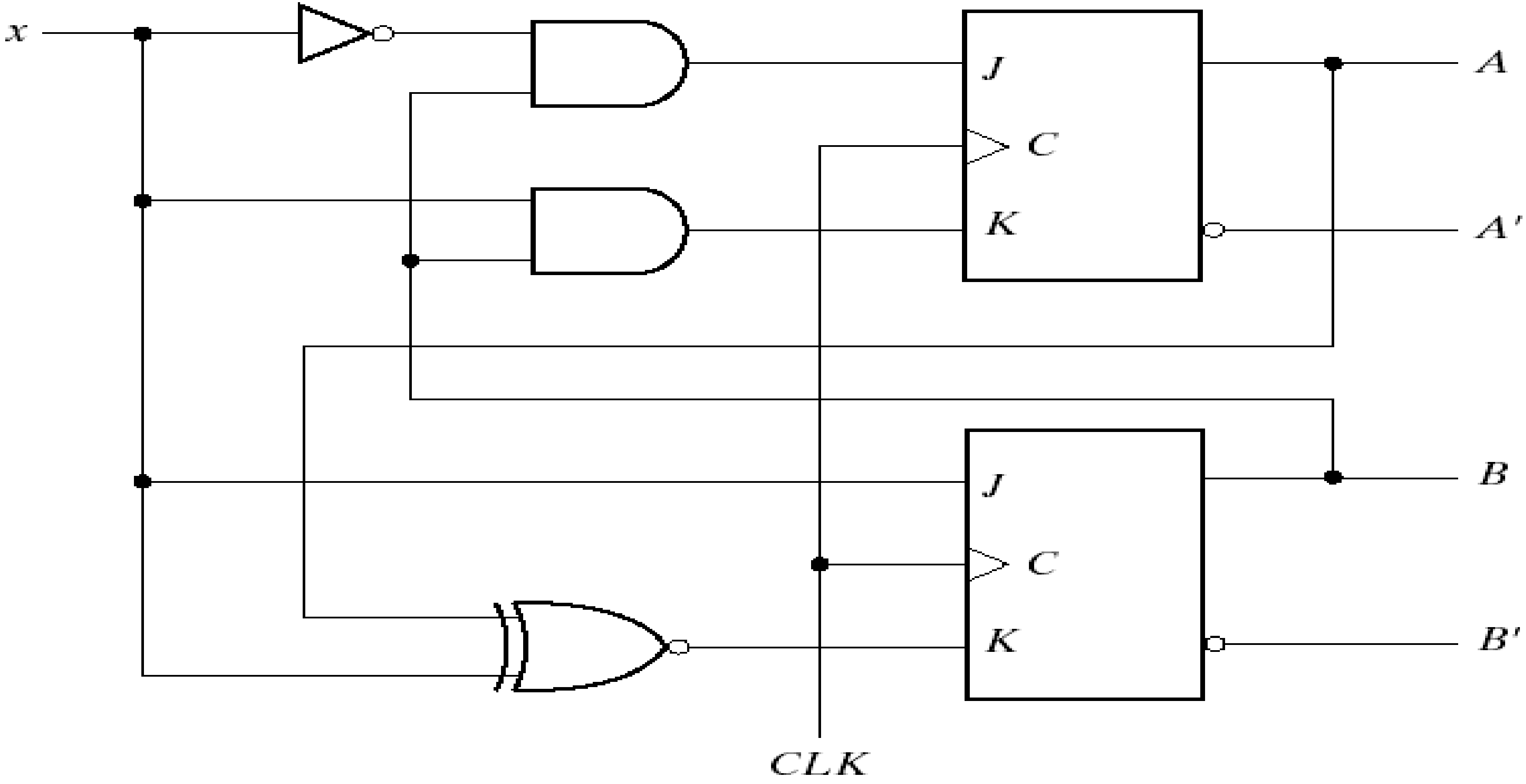
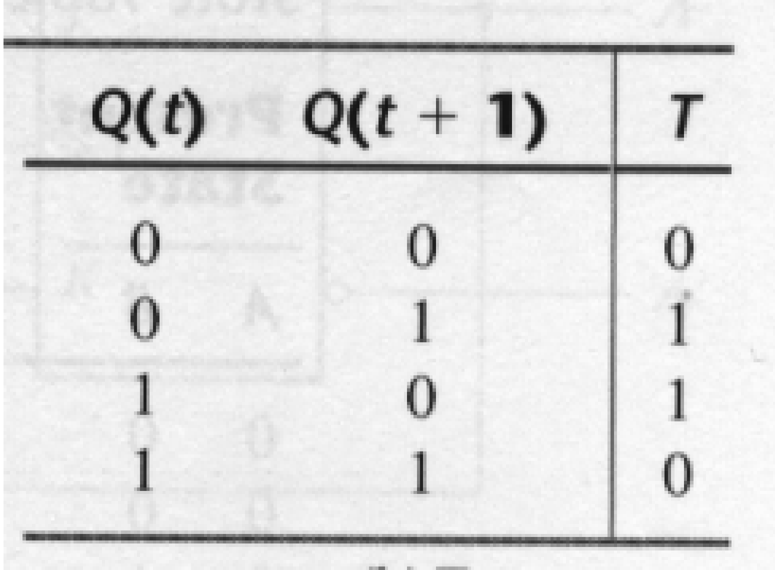


Fig. 5-28 Logic Diagram for Sequential Circuit with *JK* Flip-Flops

Synthesis Using T Flip-Flops

The synthesis using T flip-flops will be demonstrated by designing a binary counter. An n-bit binary counter consists of n flip-flops that can count in binary from 0 to 2^n-1. The state diagram of a 3-bit counter is shown in Fig. 5-29.



$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

(b)T

Ref. Table 5-1

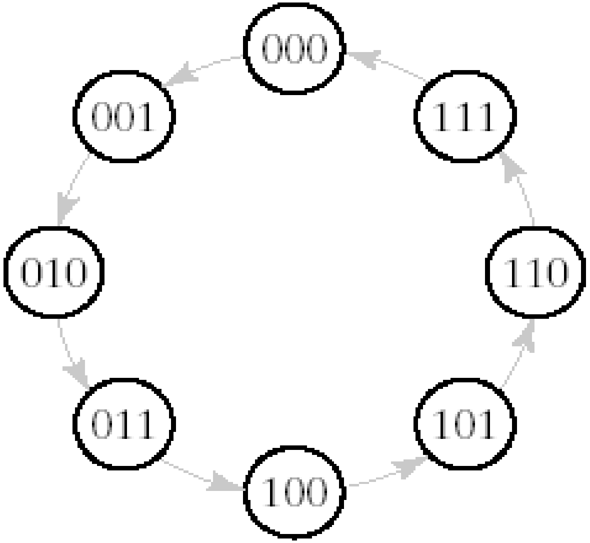


Fig. 5-29 State Diagram of 3-Bit Binary Counter

Synthesis Using T Flip-Flops

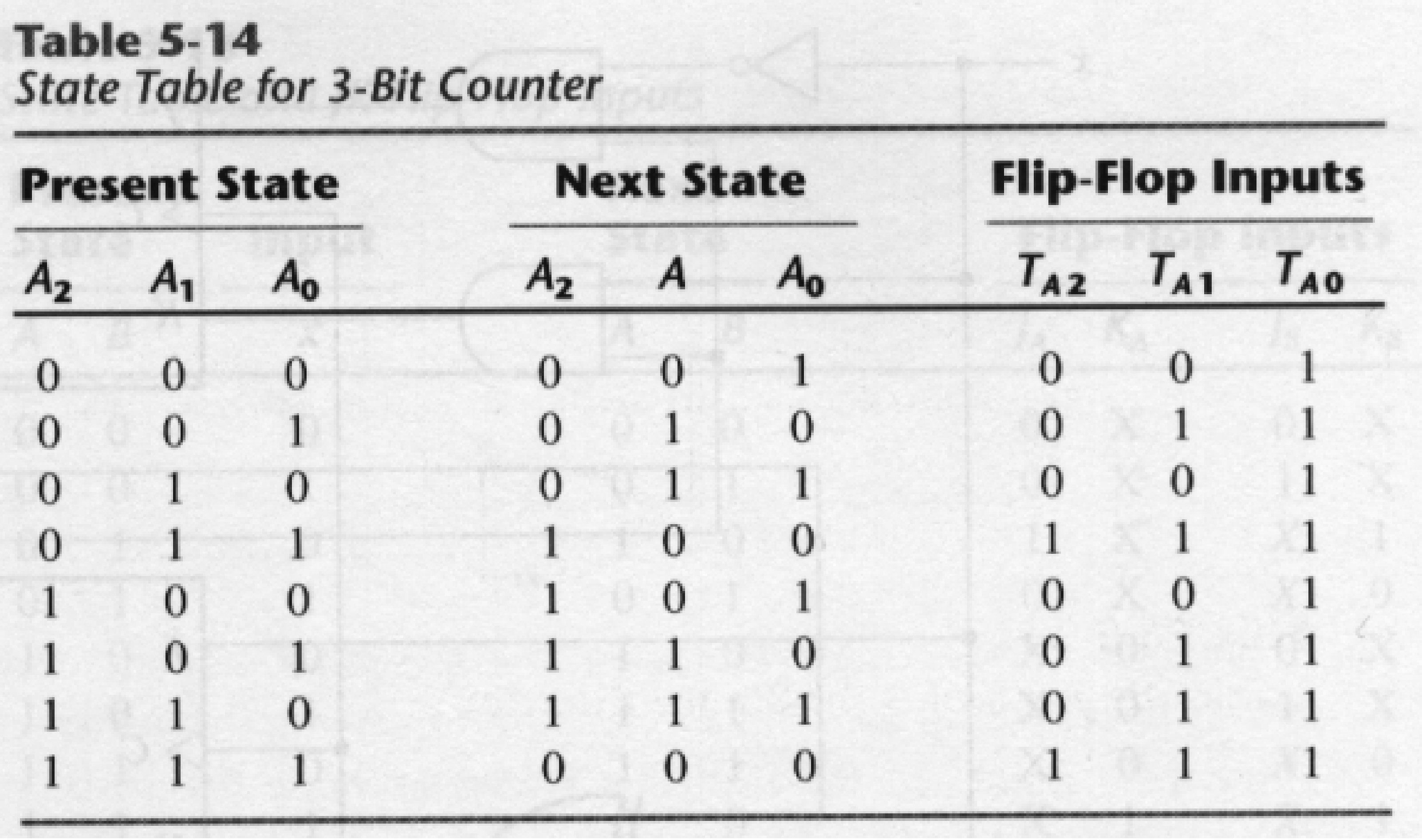
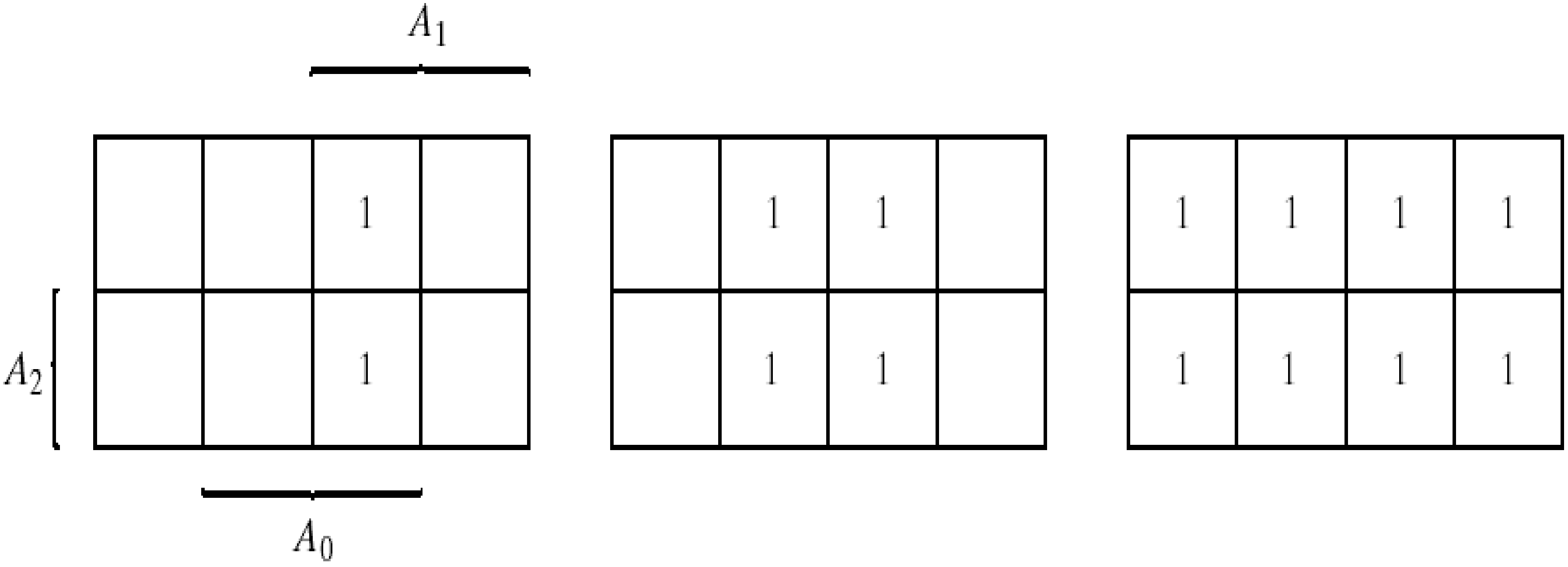


Table 5-14
State Table for 3-Bit Counter

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

Synthesis Using T Flip-Flops



		1	
		1	

$T_{A2} = A_1 A_0$

	1	1	
	1	1	

$T_{A1} = A_0$

1	1	1	1
1	1	1	1

$T_{A0} = 1$

Fig. 5-30 Maps for 3-Bit Binary Counter

Synthesis Using T Flip-Flops

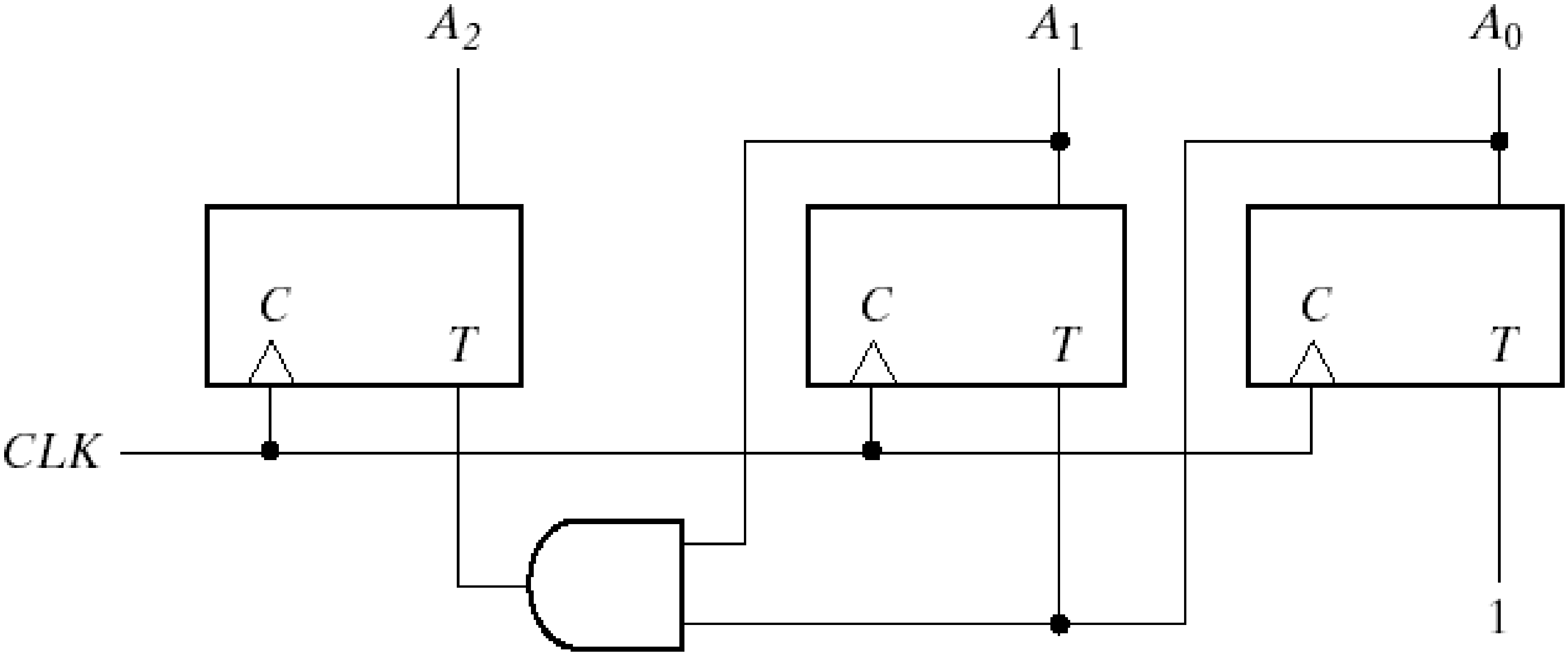


Fig. 5-31 Logic Diagram of 3-Bit Binary Counter