# CHAPTER 5
# SEQUENTIAL LOGIC

INEL 4205 Logic Circuits

Spring 2012

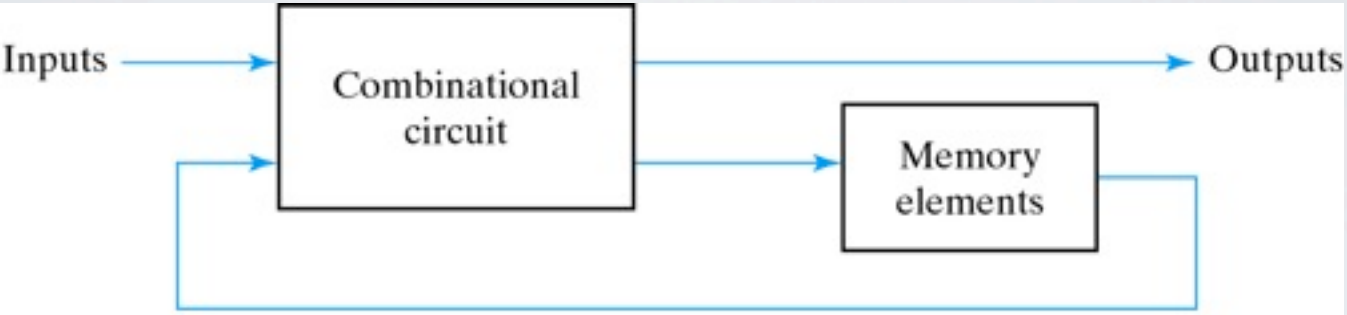
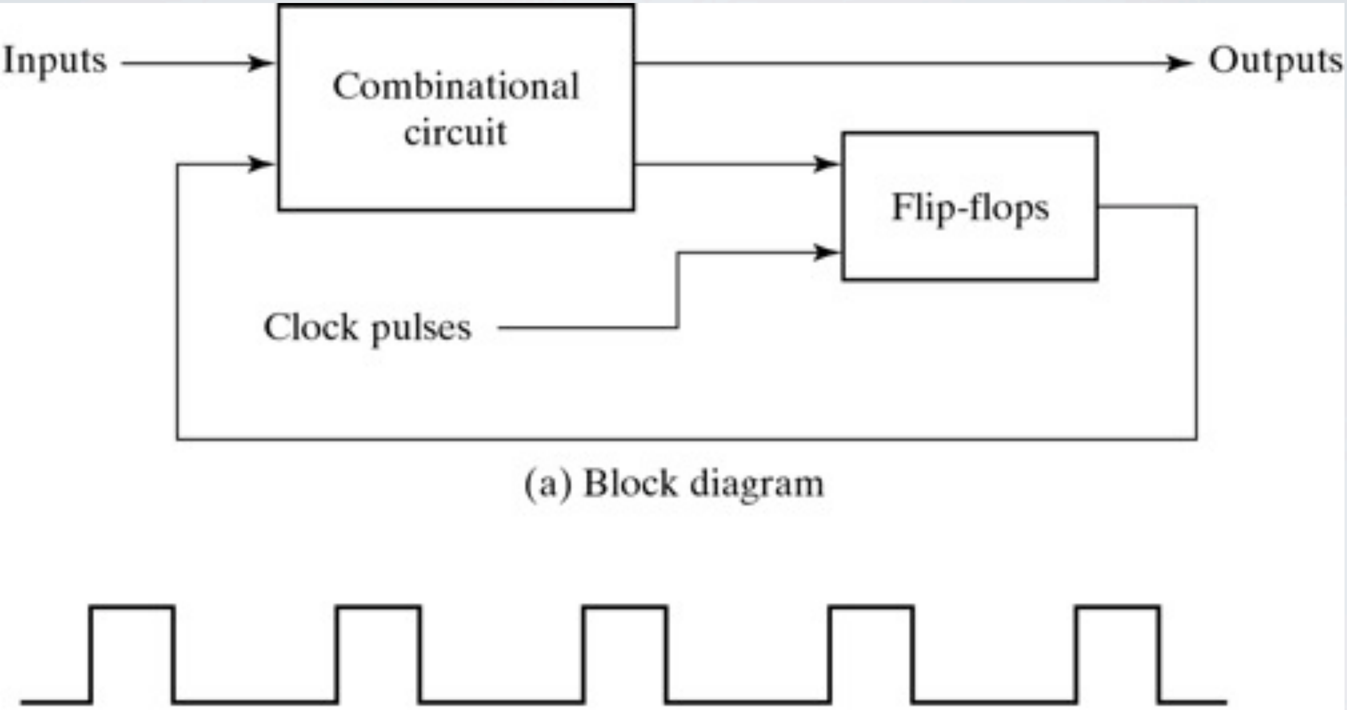


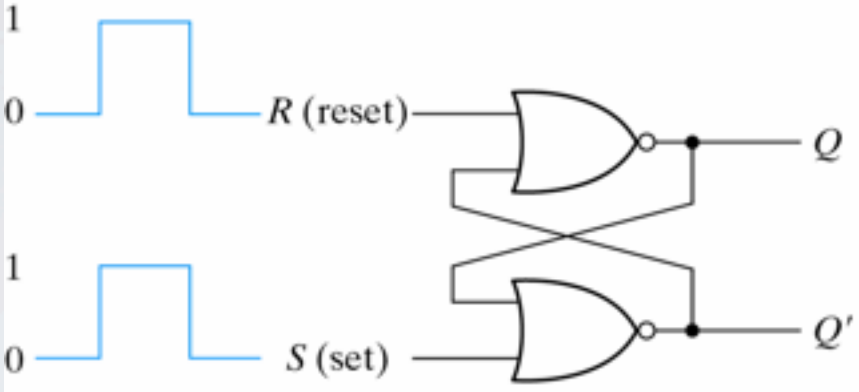
Fig. 5-1 Block Diagram of Sequential Circuit



(a) Block diagram

(b) Timing diagram of clock pulses

Fig. 5-2 Synchronous Clocked Sequential Circuit

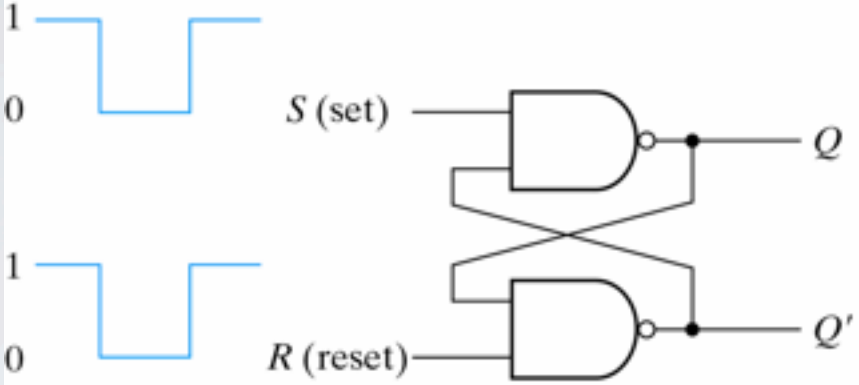


(a) Logic diagram

| $S$ | $R$ | $Q$ | $Q'$ | |
|---|---|---|---|---|
| 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

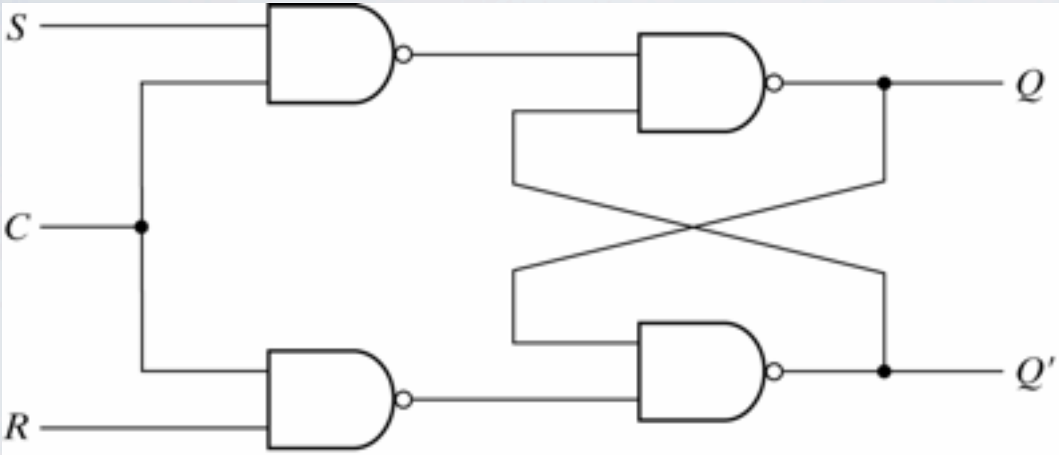


(a) Logic diagram

| $S$ | $R$ | $Q$ | $Q'$ | |
|---|---|---|---|---|
| 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

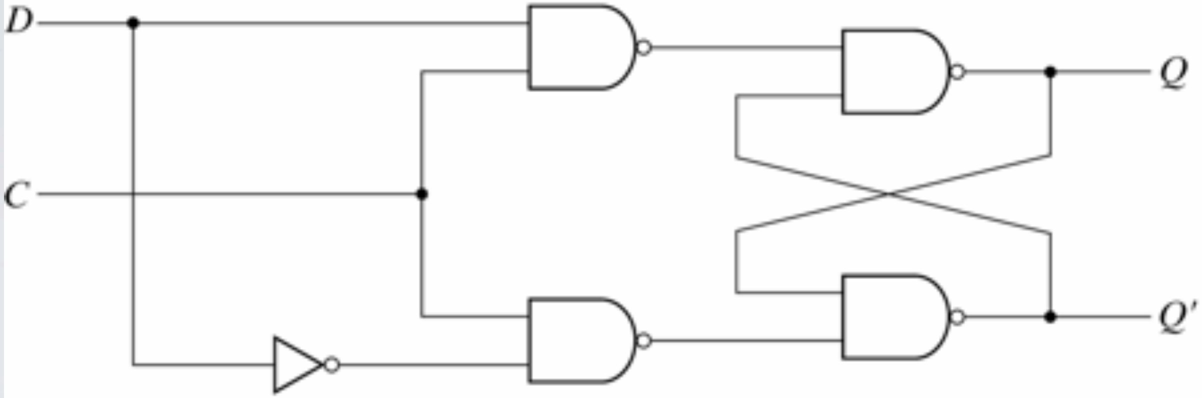


(a) Logic diagram

| 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

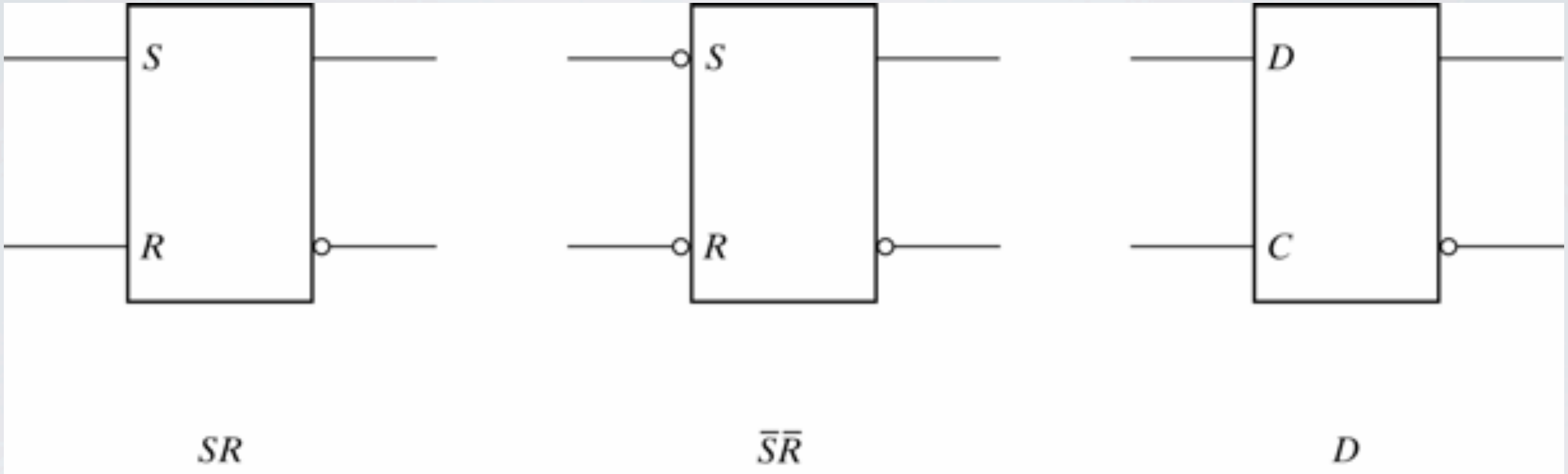


(a) Logic diagram

| $C$ | $D$ | 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

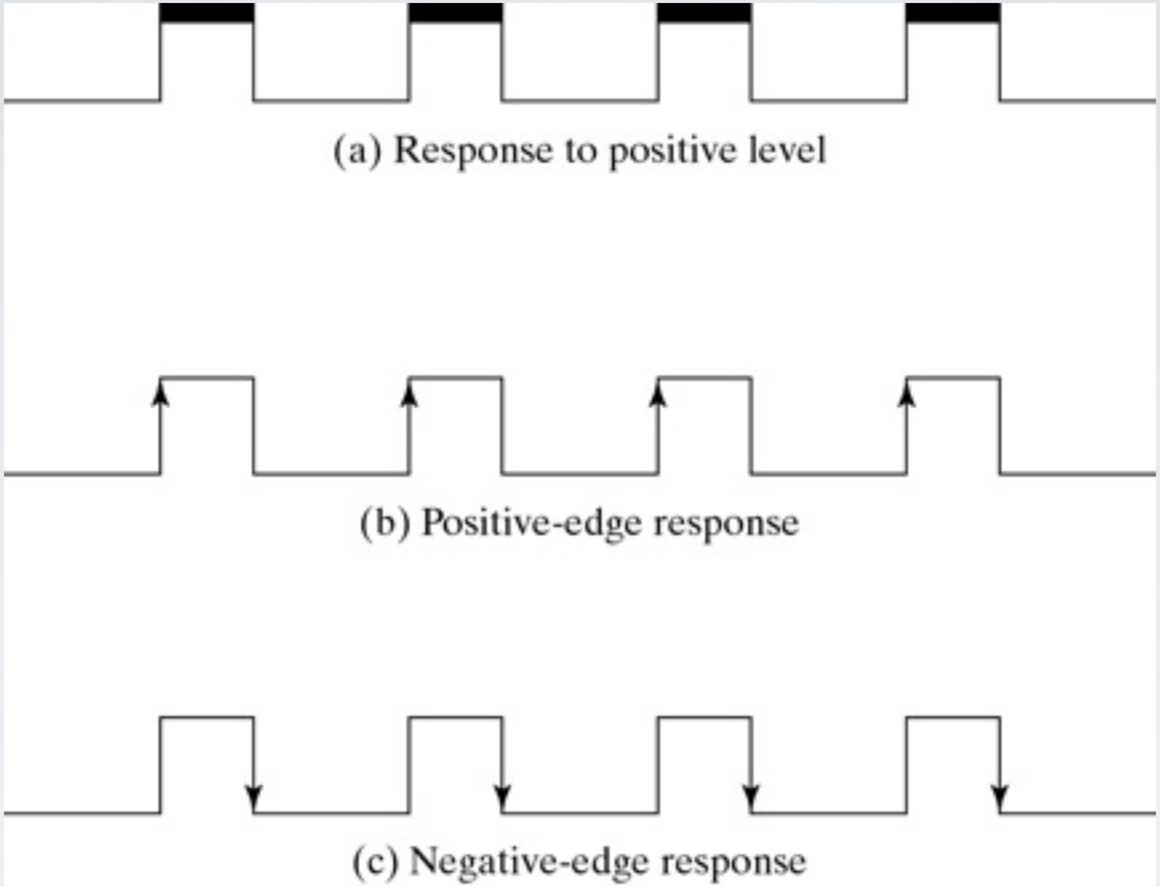


SR

$\overline{S}\overline{R}$

D

Fig. 5-7 Graphic Symbols for Latches



(a) Response to positive level

(b) Positive-edge response

(c) Negative-edge response

Fig. 5-8 Clock Response in Latch and Flip-Flop

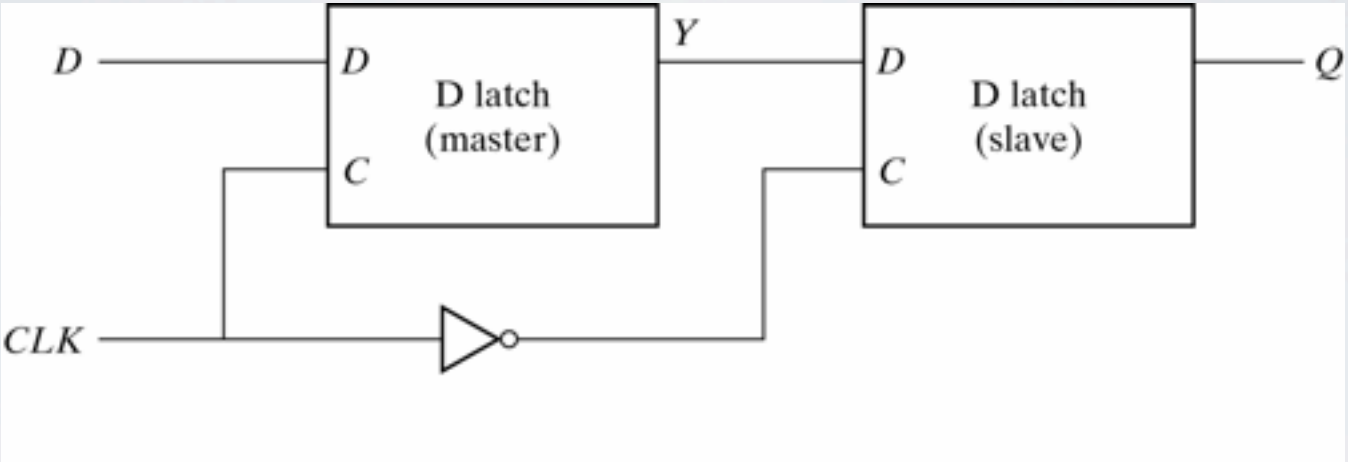
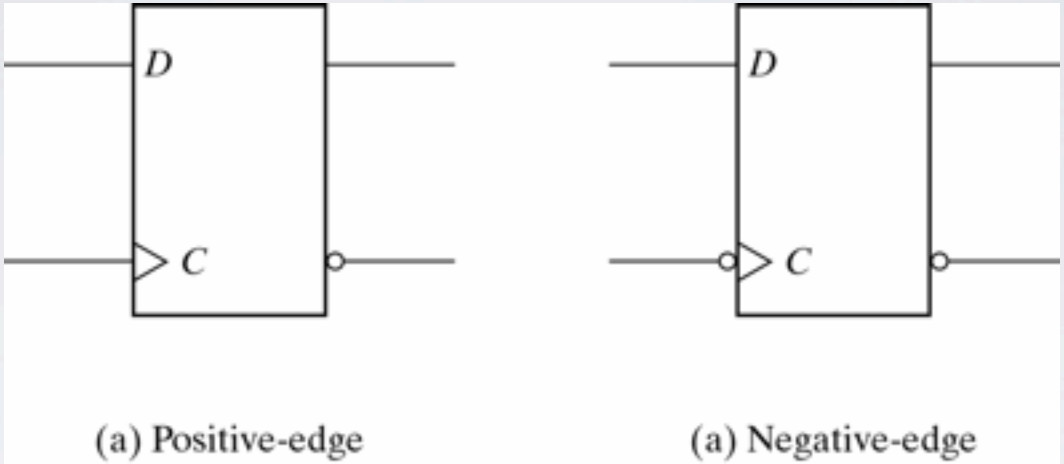


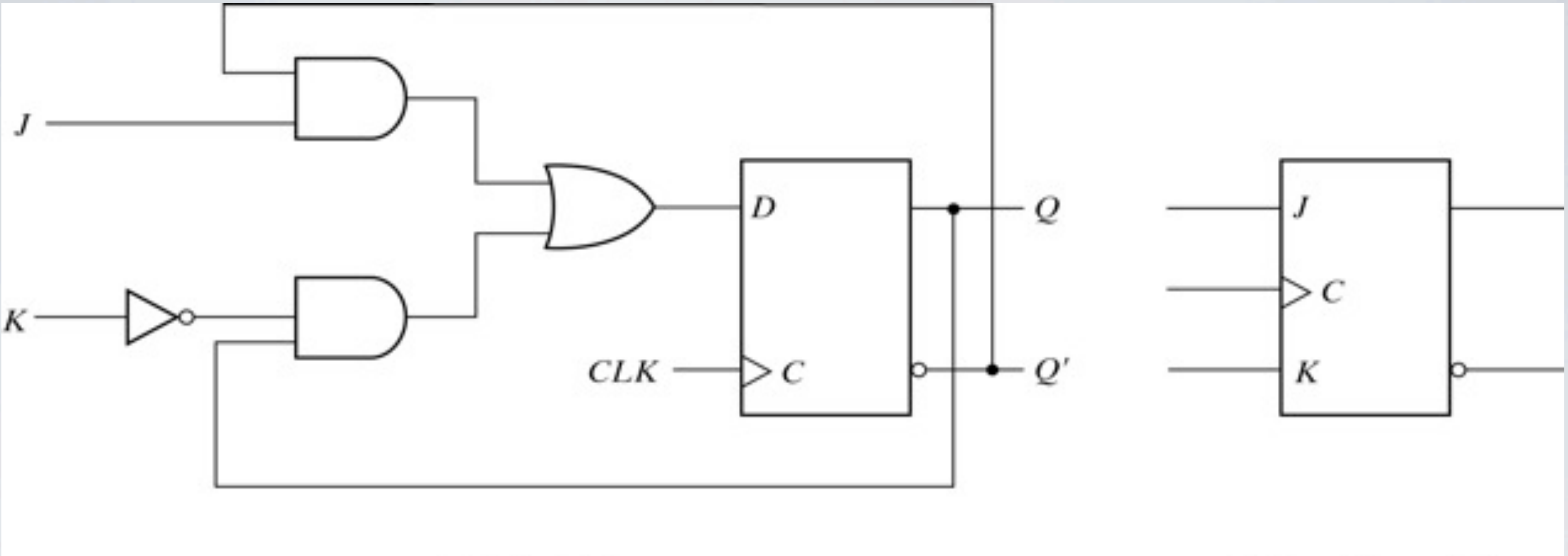
Fig. 5-9 Master-Slave *D* Flip-Flop



(a) Positive-edge

(a) Negative-edge

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



(a) Circuit diagram

(b) Graphic symbol

Fig. 5-12 *JK* Flip-Flop

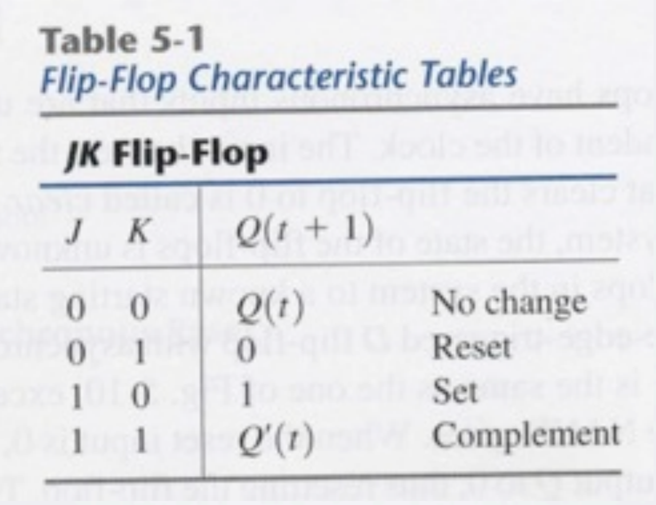
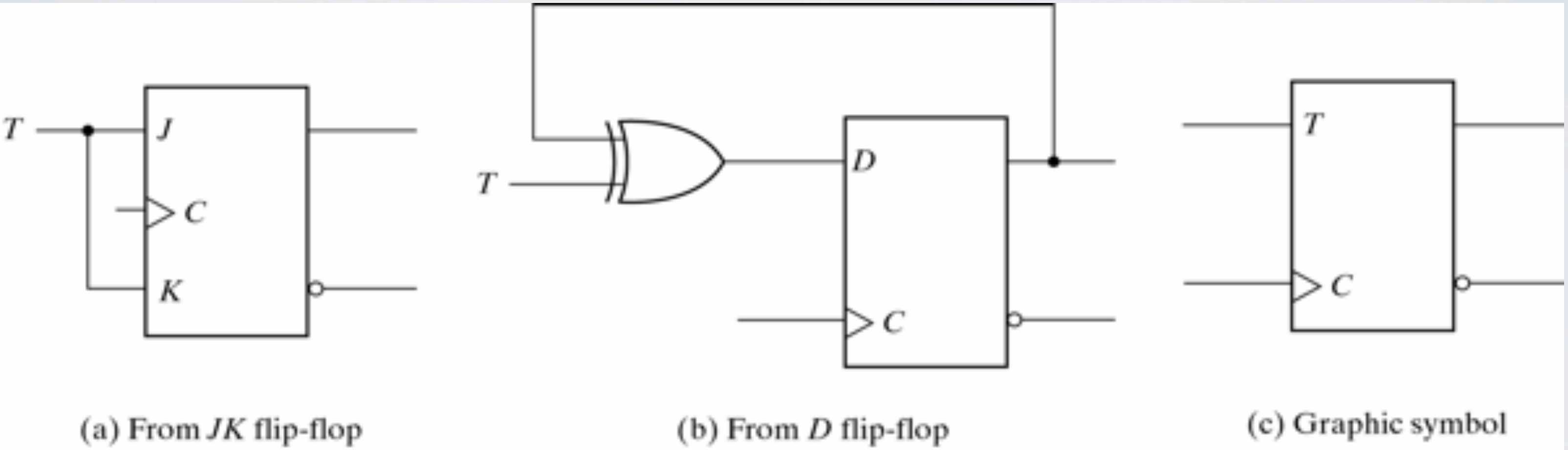


**Table 5-1**
*Flip-Flop Characteristic Tables*

***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 |



(a) From *JK* flip-flop

(b) From *D* flip-flop

(c) Graphic symbol

Fig. 5-13 T Flip-Flop

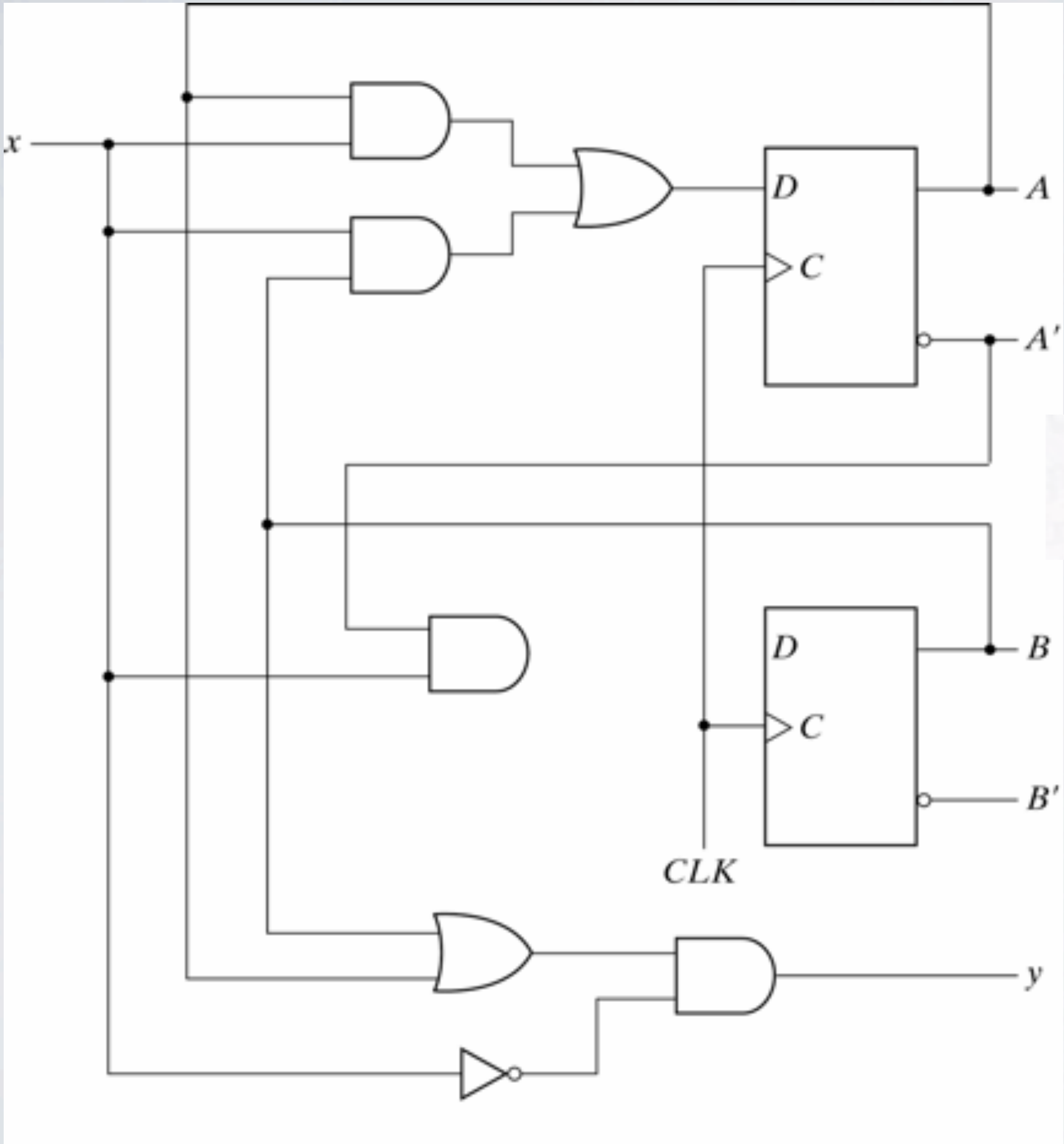


Fig. 5-15 Example of Sequential Circuit

$$A(t+1) = Ax + Bx$$
$$B(t+1) = A'x$$

$$y = (A + B)x'$$

$$A(t+1) = Ax + Bx$$
$$B(t+1) = A'x$$

State equations or transition equations

$$y = (A + B)x'$$

Output boolean equation

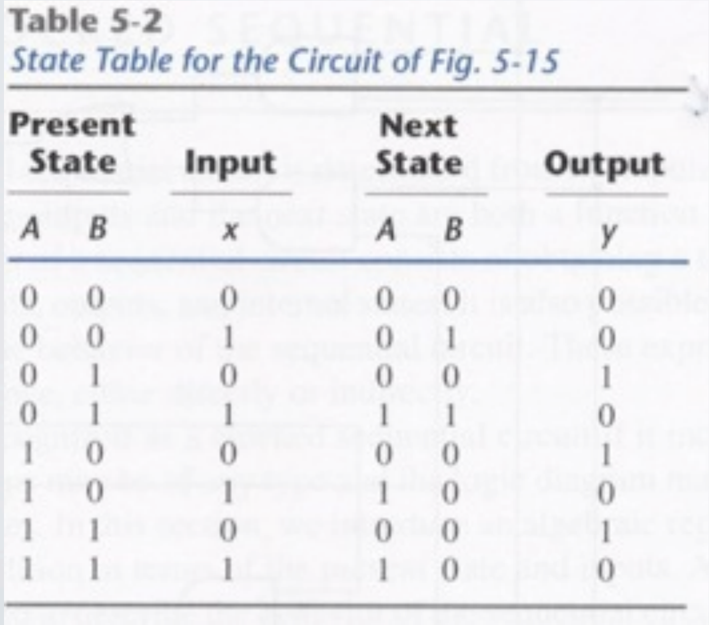


**Table 5-2**
*State Table for the Circuit of Fig. 5-15*

| 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 |

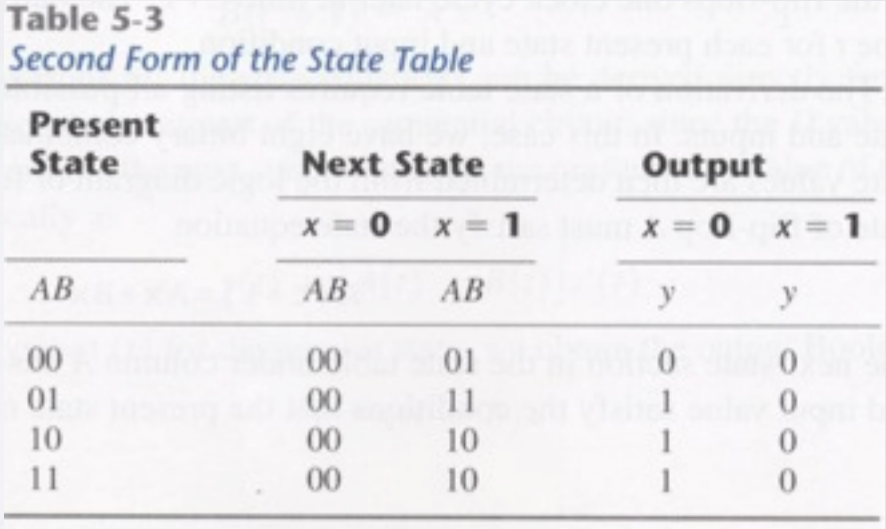


**Table 5-3**
*Second Form of the State Table*

| 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 |

/0 represents the output during the present state with the given input

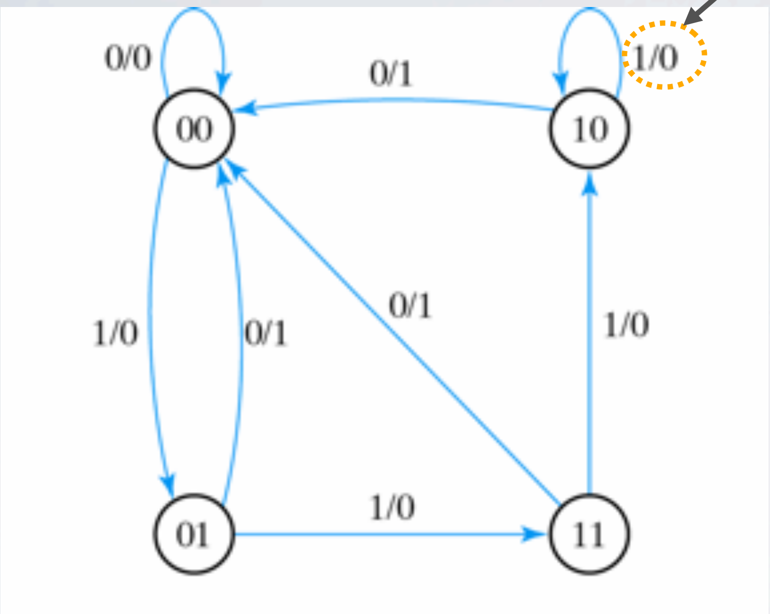
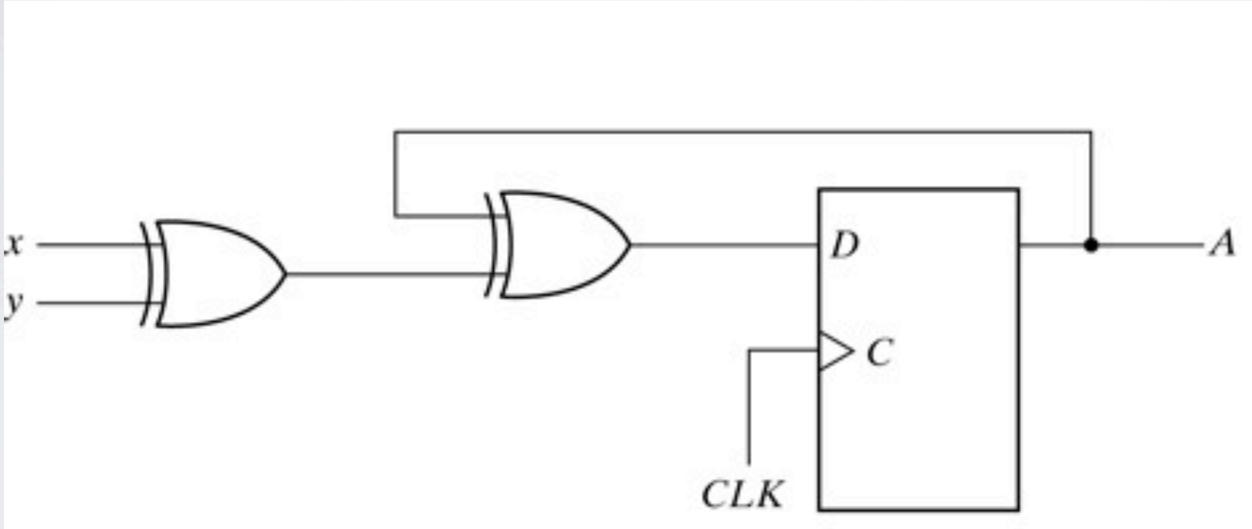


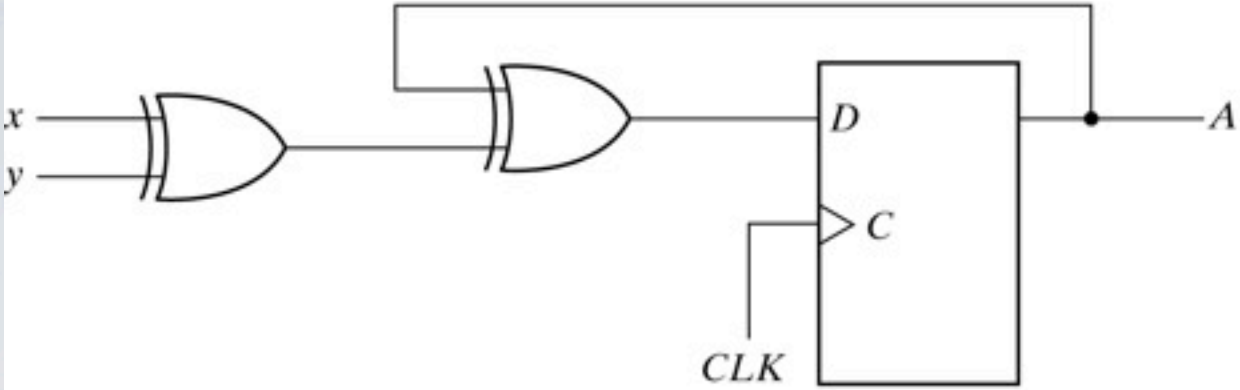
Fig. 5-16 State Diagram of the Circuit of Fig. 5-15

Mealy Finite State Machine (FSM) – output is a function of present state and input

## Example:



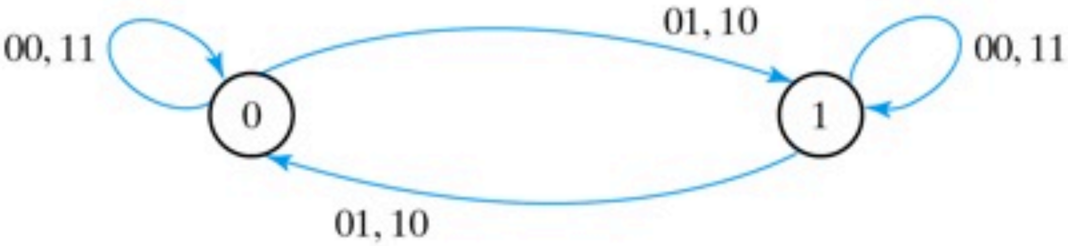
1. Find the state table
2. Draw the state diagram



(a) Circuit diagram

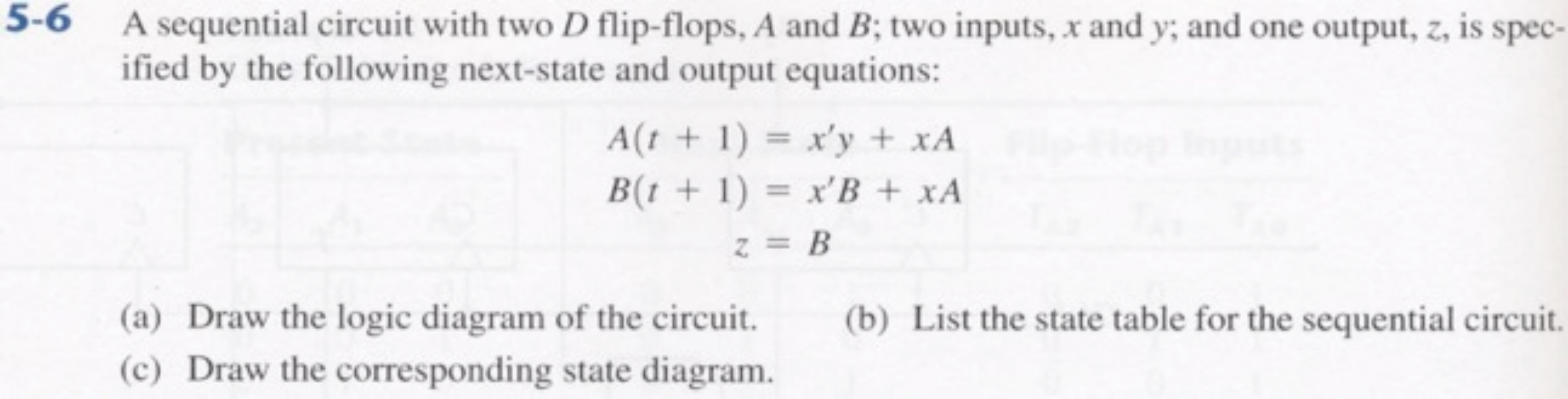
| Present state | Inputs | | Next state |
|---|---|---|---|
| $A$ | $x$ | $y$ | $A$ |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

(b) State table



(c) State diagram

Fig. 5-17 Sequential Circuit with *D* Flip-Flop



**5-6** A sequential circuit with two $D$ flip-flops, $A$ and $B$; two inputs, $x$ and $y$; and one output, $z$, is specified by the following next-state and output equations:

$$A(t+1) = x'y + xA$$
$$B(t+1) = x'B + xA$$
$$z = B$$

(a) Draw the logic diagram of the circuit. (b) List the state table for the sequential circuit.

(c) Draw the corresponding state diagram.

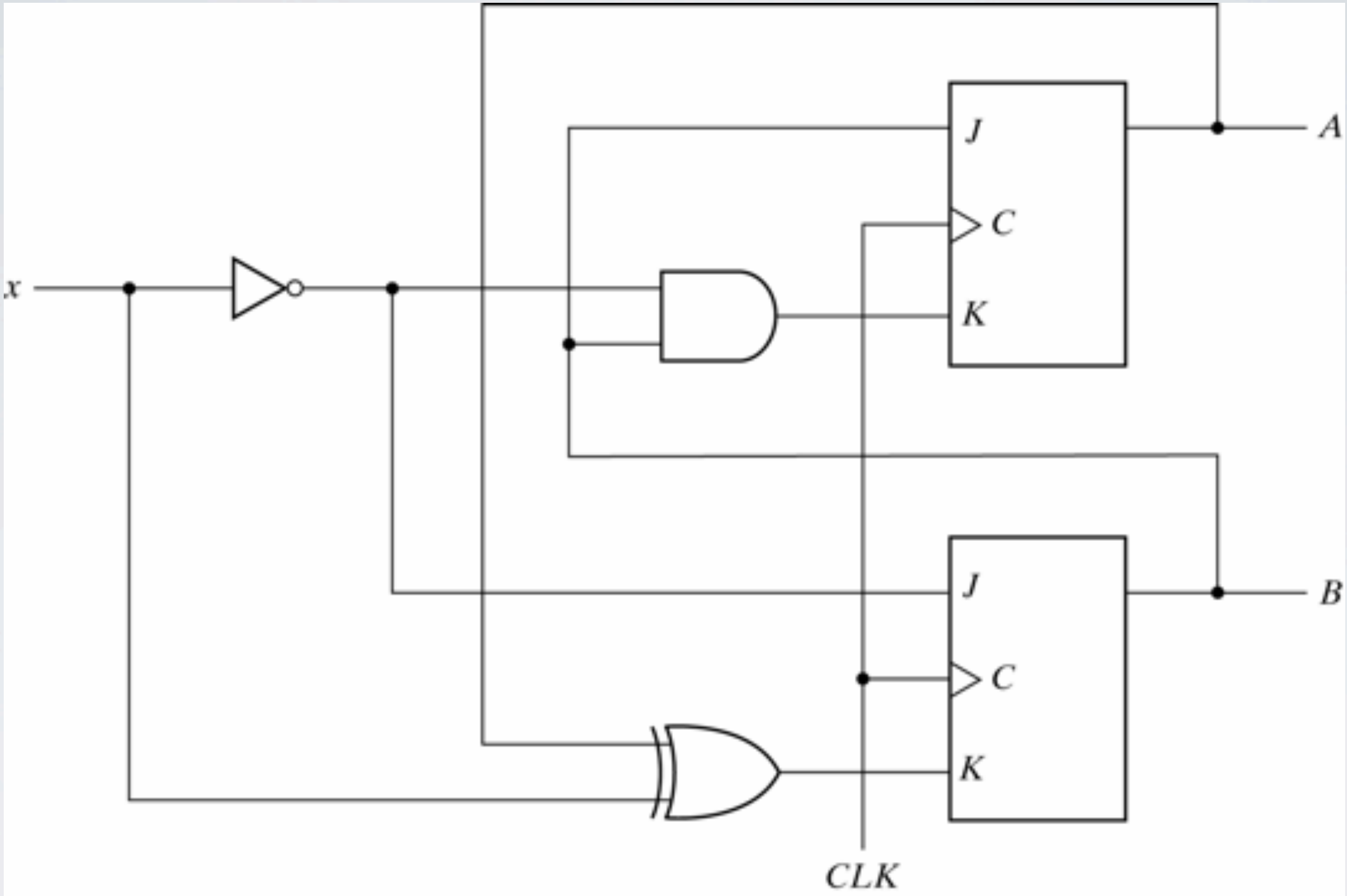


Fig. 5-18 Sequential Circuit with *JK* Flip-Flop

flip-flop input equations

$$J_A = B \qquad K_A = Bx'$$

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

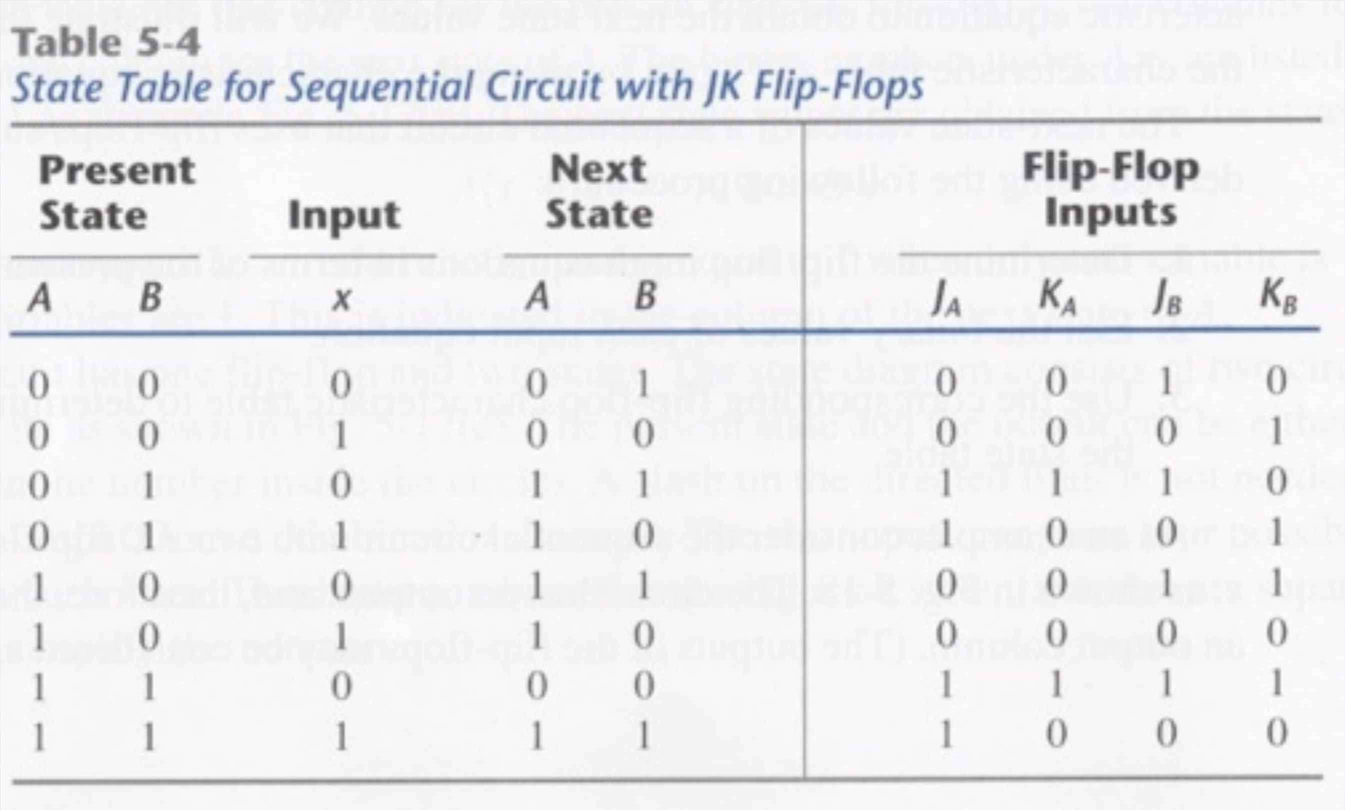


**Table 5-4**
*State Table for Sequential Circuit with JK Flip-Flops*

| 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 |

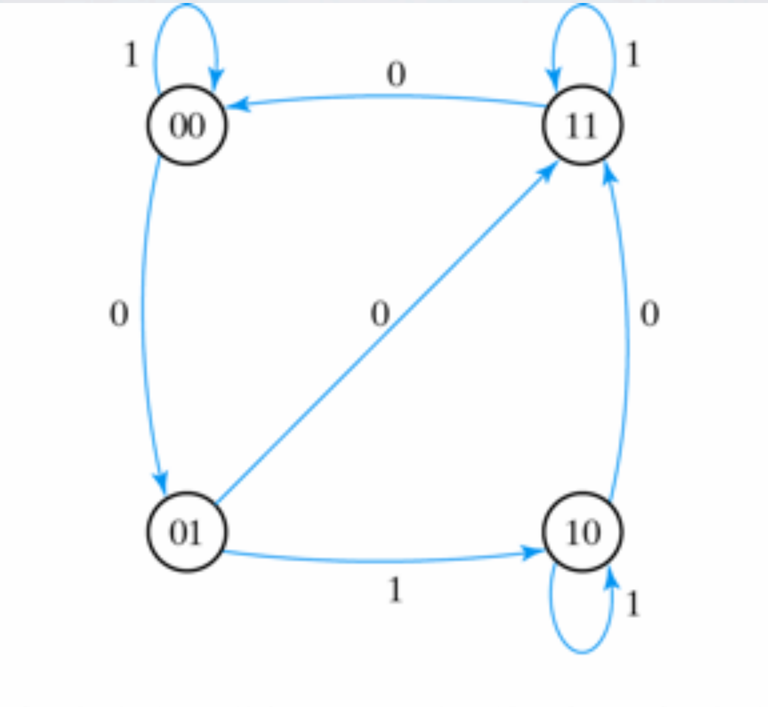
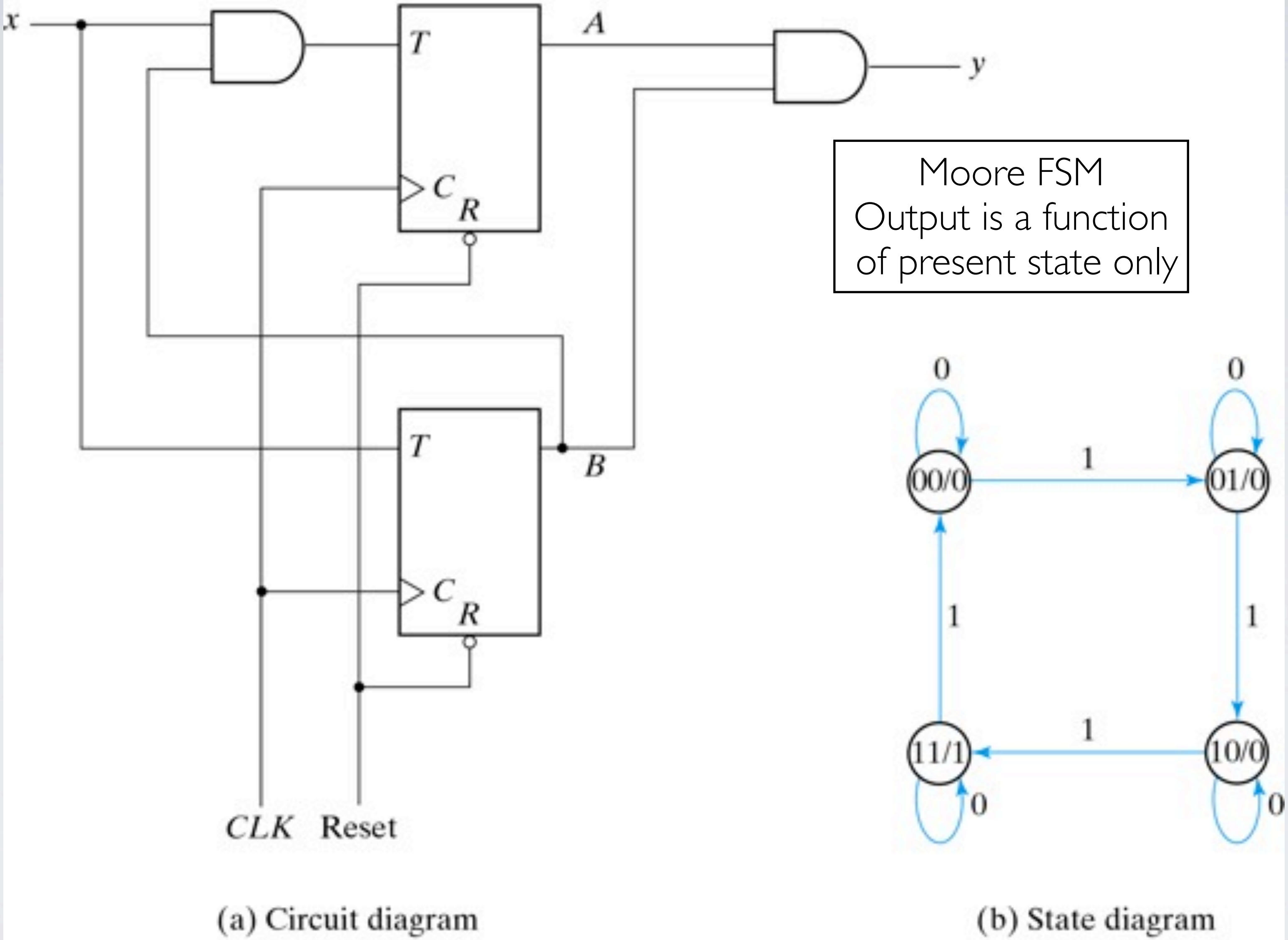


Fig. 5-19 State Diagram of the Circuit of Fig. 5-18

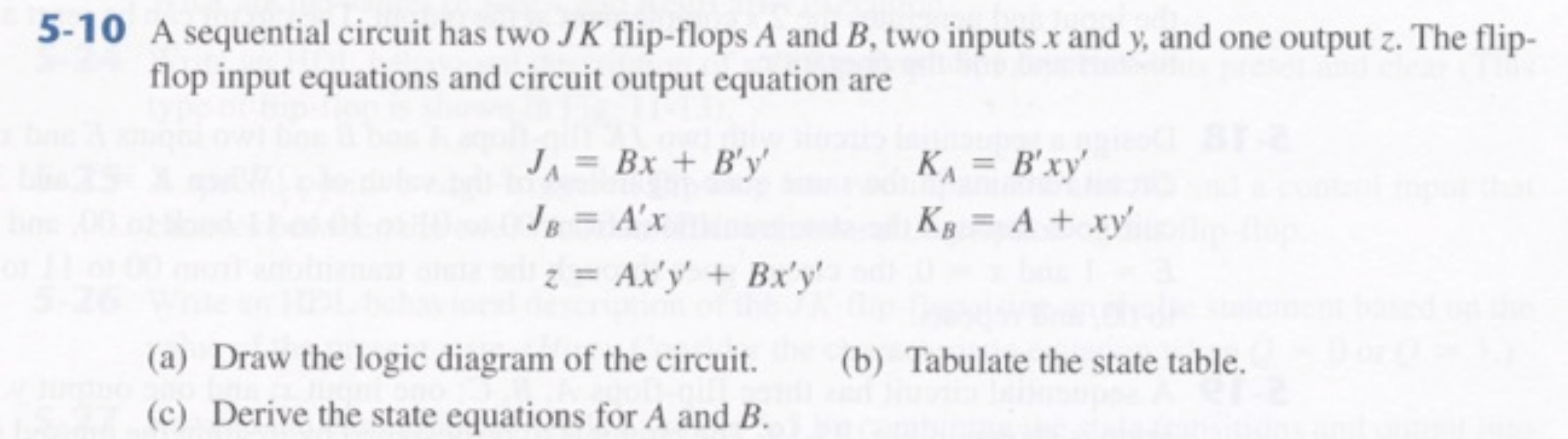


(a) Circuit diagram

(b) State diagram

Moore FSM
Output is a function
of present state only

Fig. 5-20 Sequential Circuit with *T* Flip-Flops



**5-10** A sequential circuit has two *JK* flip-flops *A* and *B*, two inputs *x* and *y*, and one output *z*. The flip-flop input equations and circuit output equation are

$$J_A = Bx + B'y' \qquad K_A = B'xy'$$

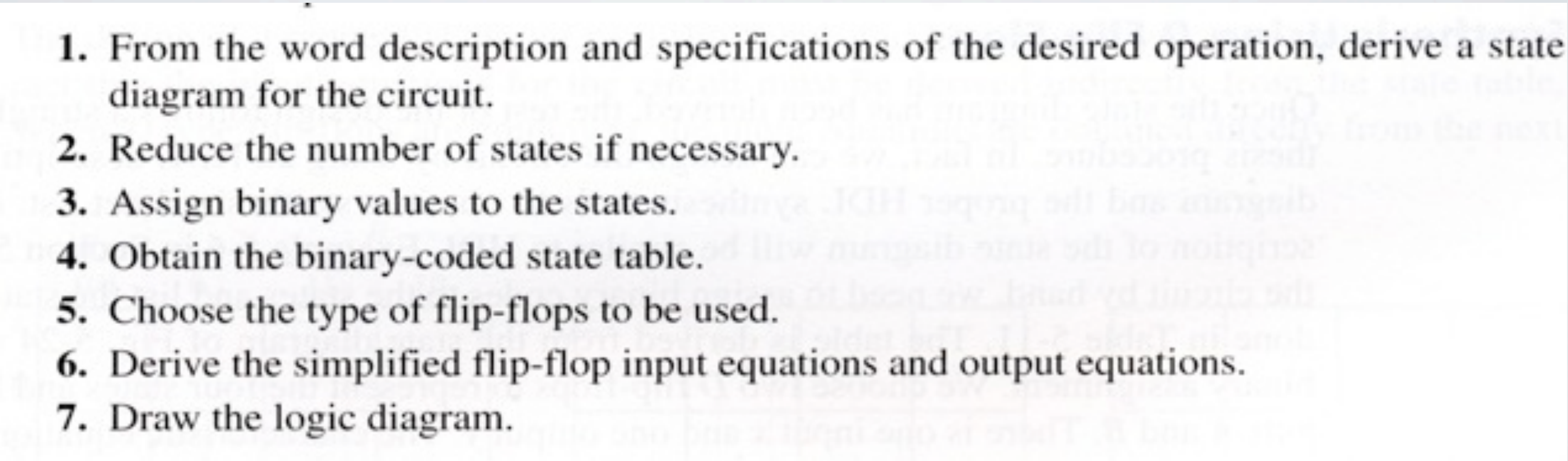
$$J_B = A'x \qquad K_B = A + xy'$$

$$z = Ax'y' + Bx'y'$$

(a) Draw the logic diagram of the circuit. (b) Tabulate the state table.

(c) Derive the state equations for *A* and *B*.

# Design Procedure



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.

# State reduction

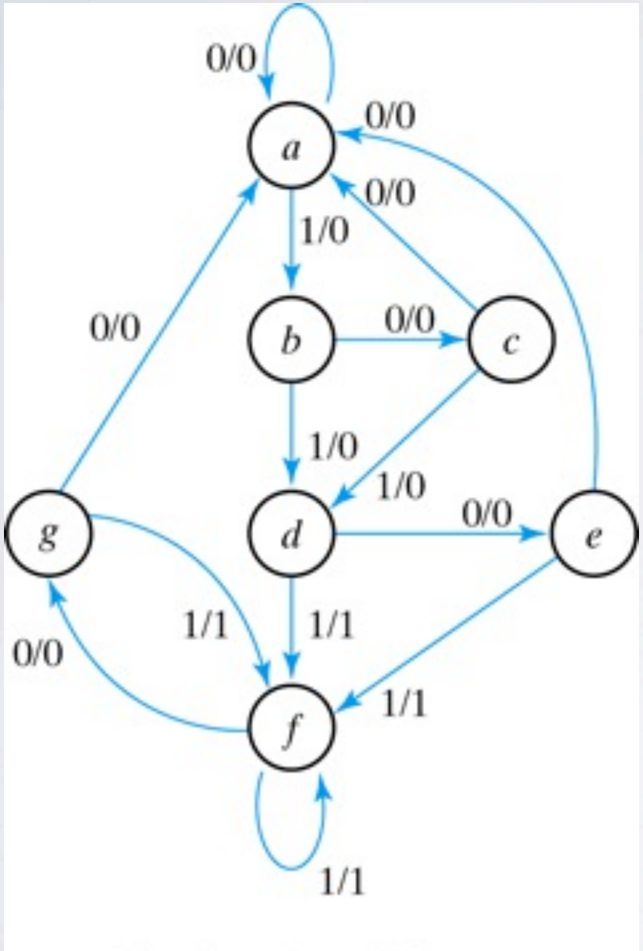


Fig. 5-22 State Diagram

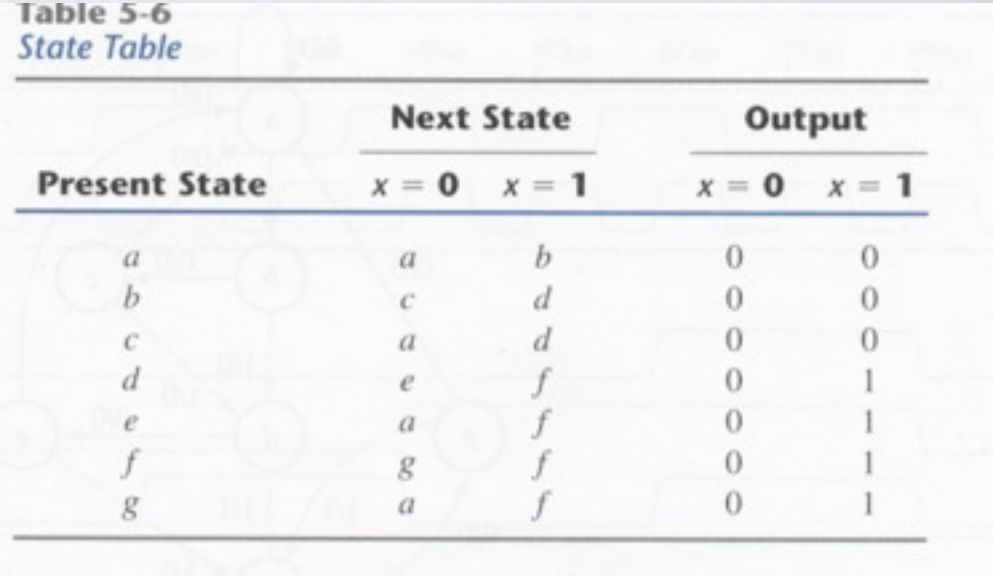
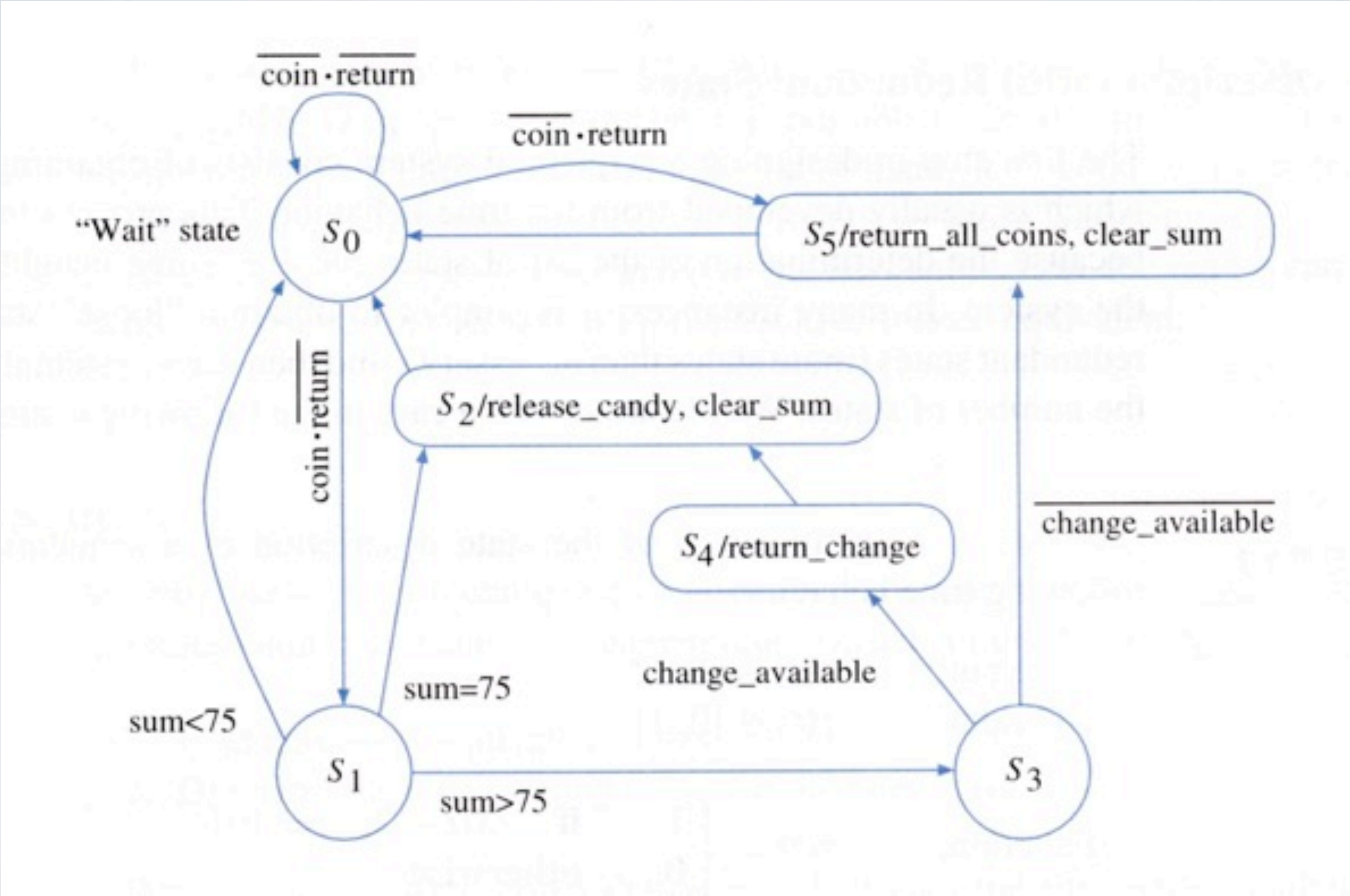


Table 5-6
*State Table*

| Present State | Next State |  | Output |  |
|---|---|---|---|---|
|  | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| *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 |

States are application-dependant.
The names given here (a,b,c,d,…)
are arbitrary.
It is assumed that only the output
response to a given sequence of
inputs is important.


$\overline{\text{coin}}\cdot\overline{\text{return}}$
$\overline{\text{coin}}\cdot\text{return}$
"Wait" state
$S_0$
$S_5$/return_all_coins, clear_sum
$\text{coin}\cdot\overline{\text{return}}$
$S_2$/release_candy, clear_sum
$\overline{\text{change\_available}}$
$S_4$/return_change
change_available
sum=75
sum<75
$S_1$
sum>75
$S_3$

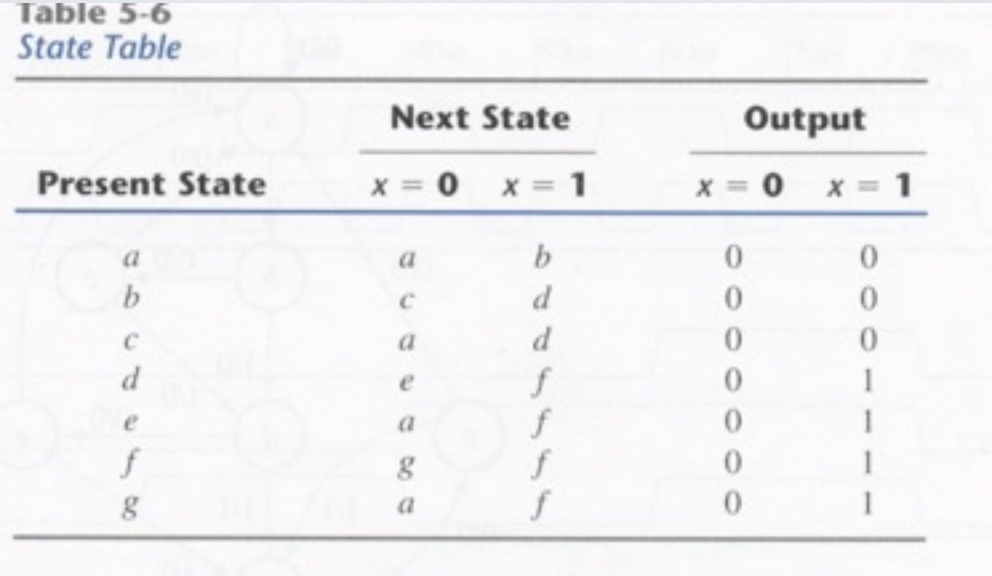
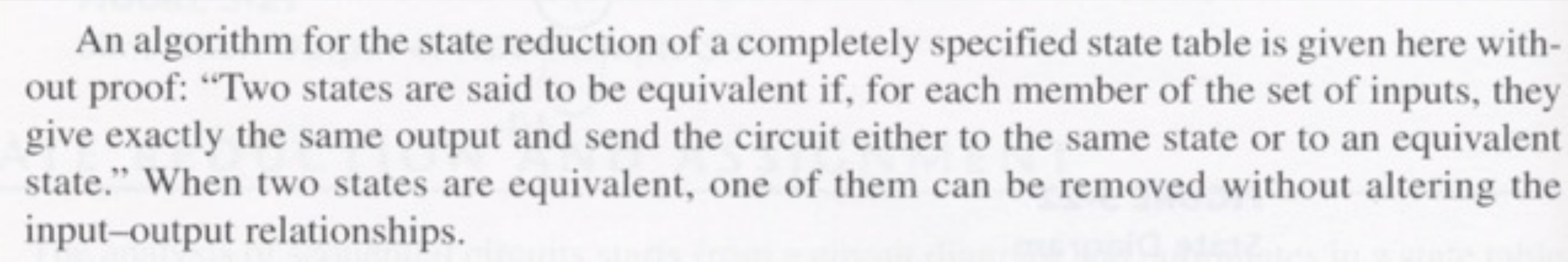


Table 5-6
*State Table*

| | 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 |



An algorithm for the state reduction of a completely specified state table is given here without proof: "Two states are said to be equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit either to the same state or to an equivalent state." When two states are equivalent, one of them can be removed without altering the input–output relationships.

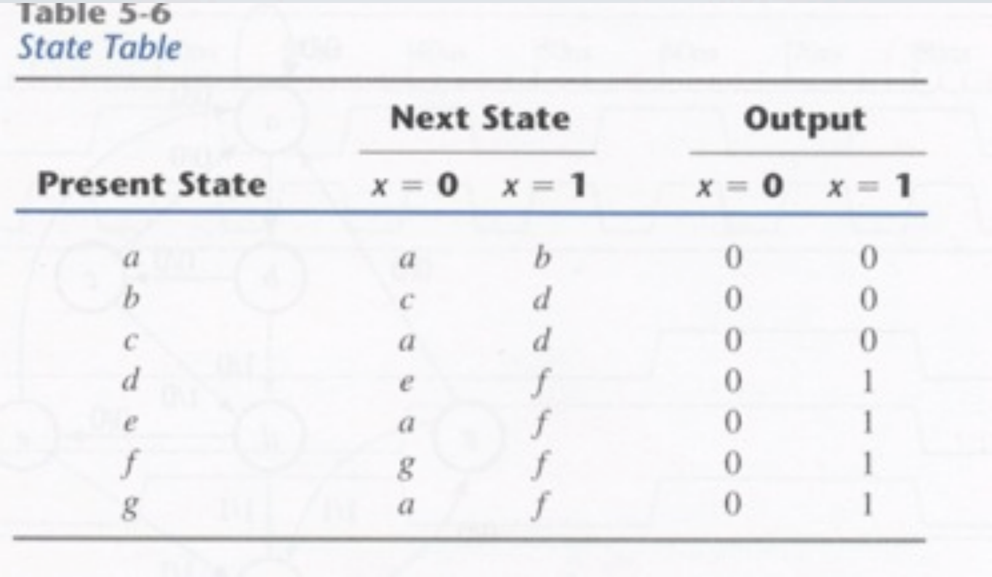


Table 5-6
*State Table*

| Present State | Next State |  | Output |  |
|---|---|---|---|---|
|  | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| $a$ | $a$ | $b$ | 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 |

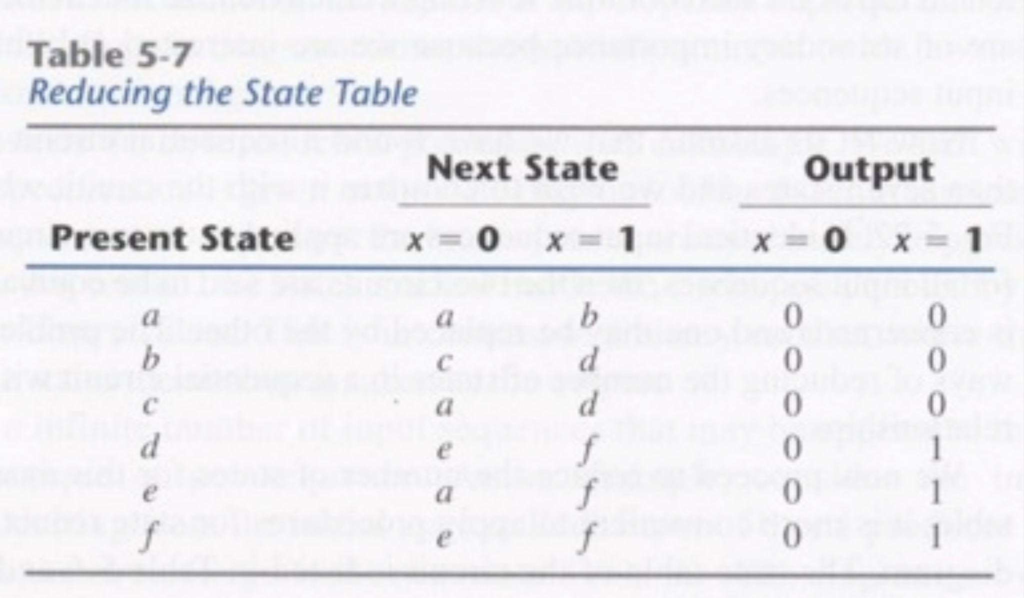


Table 5-7
*Reducing the State Table*

| Present State | Next State |  | Output |  |
|---|---|---|---|---|
|  | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| $a$ | $a$ | $b$ | 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 |

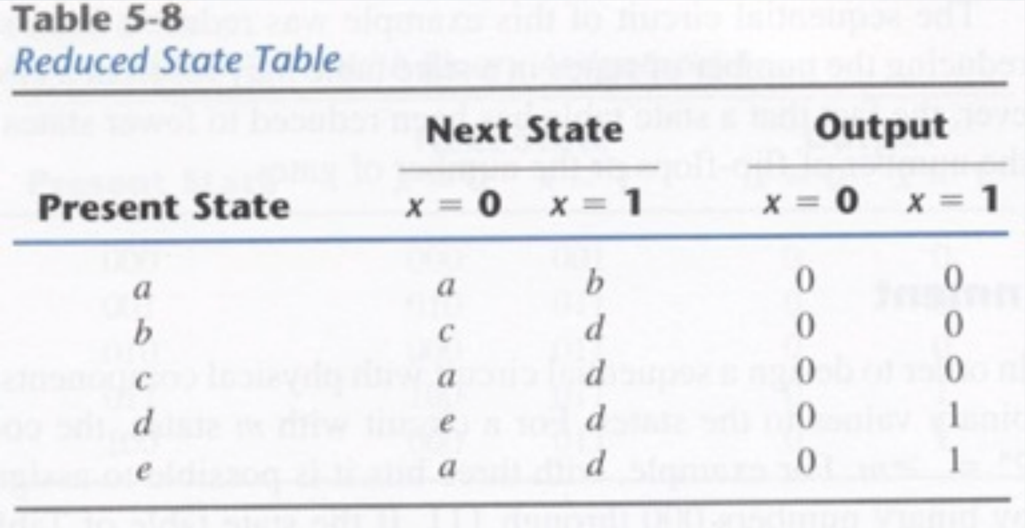


Table 5-8
*Reduced State Table*

| | 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 |

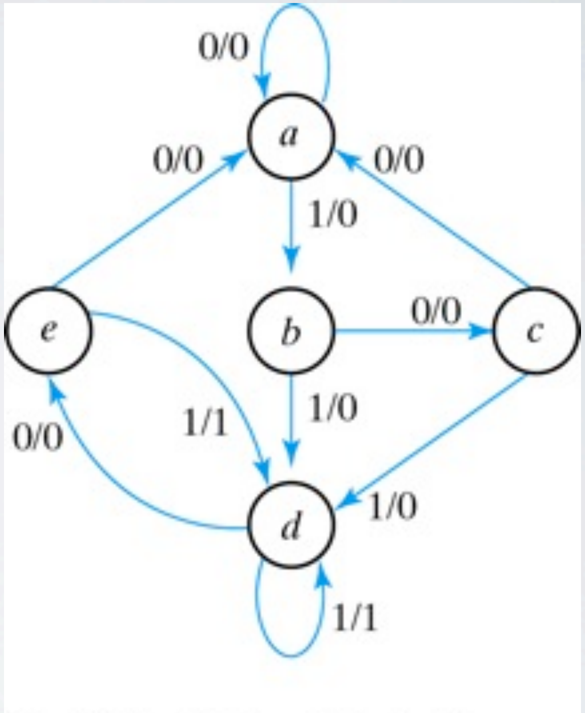


Fig. 5-23 Reduced State Diagram

# State Assignment

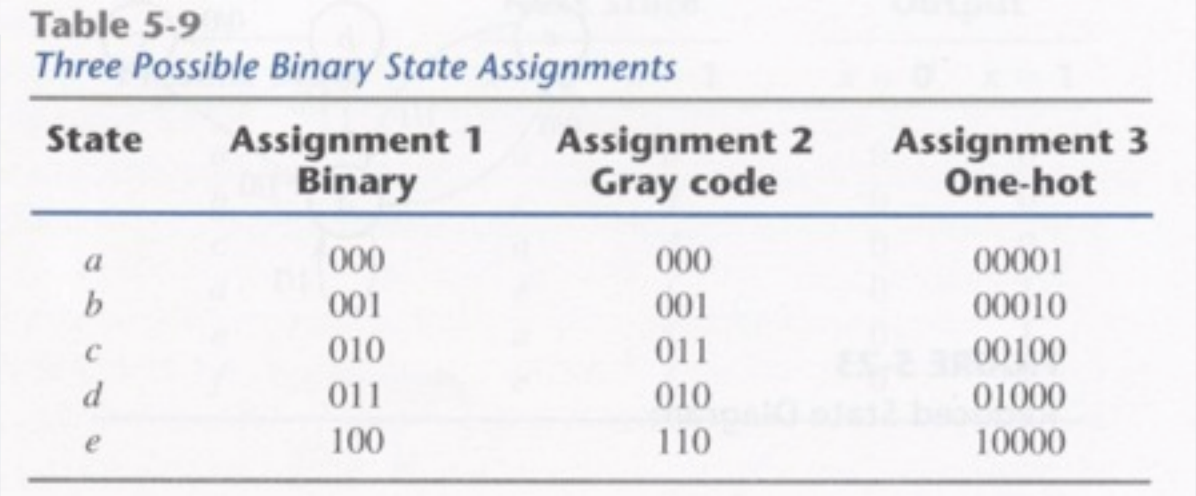


Table 5-9

*Three Possible Binary State Assignments*

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

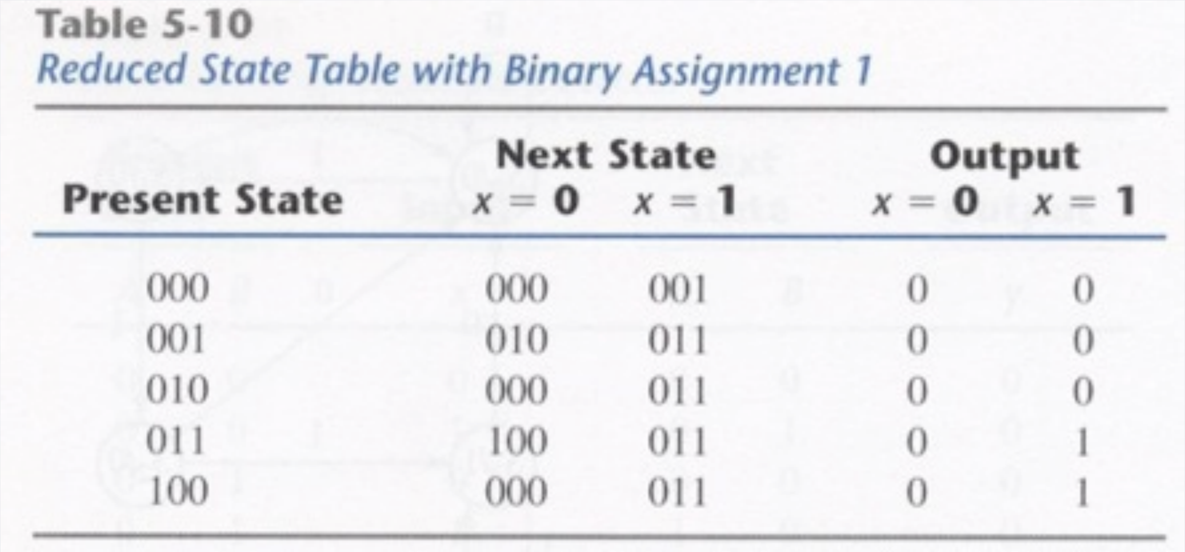


Table 5-10

*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 |

Sequence detector: circuit that detects 3 consecutive 1's in a string of bits coming through the input line

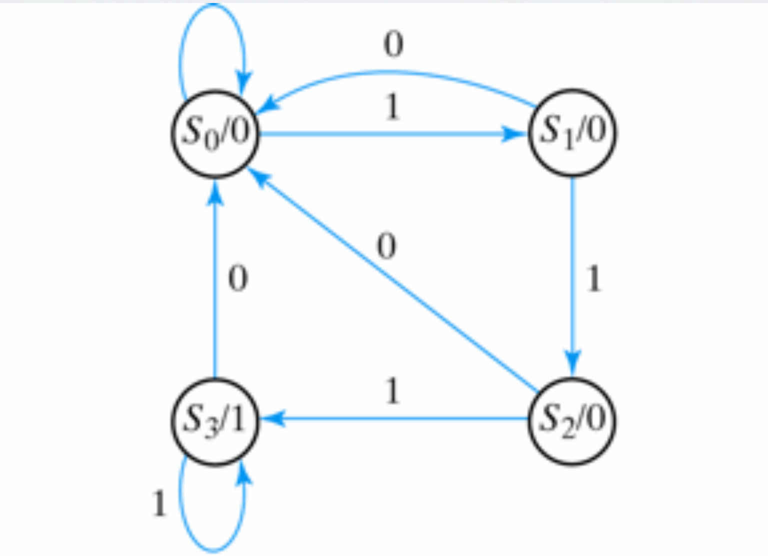
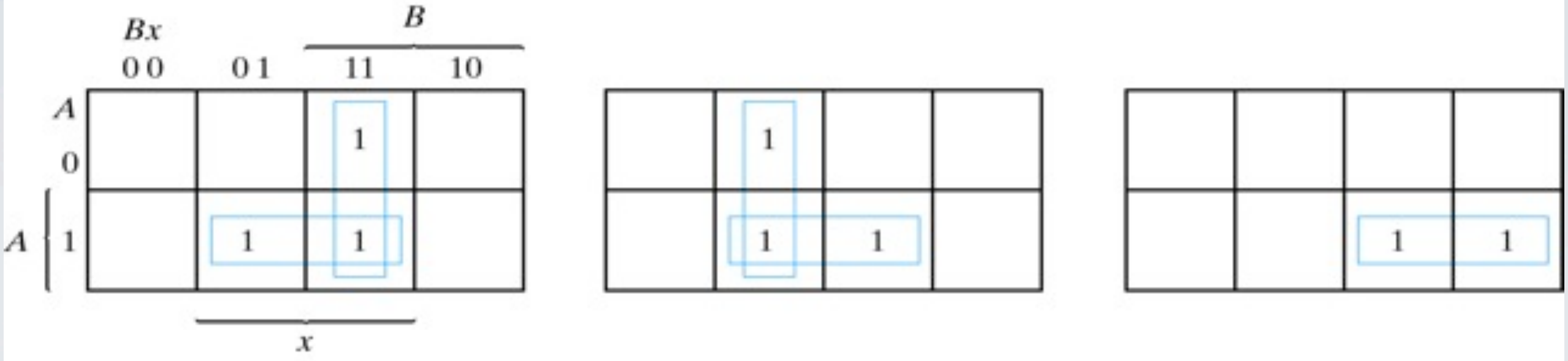


Fig. 5-24 State Diagram for Sequence Detector



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

Fig. 5-25 Maps for Sequence Detector

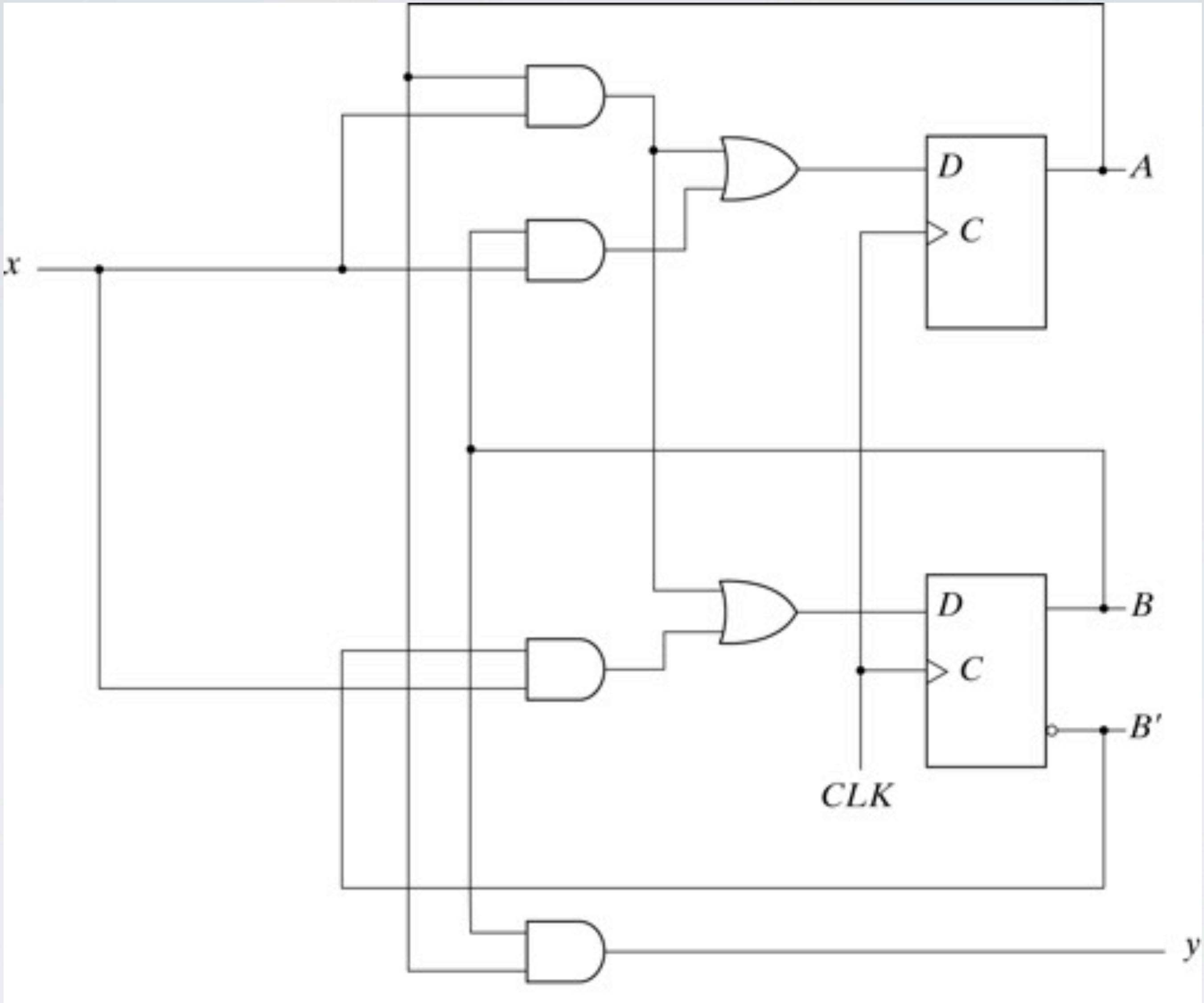


Fig. 5-26 Logic Diagram of Sequence Detector

# Using JK or T flip-flops

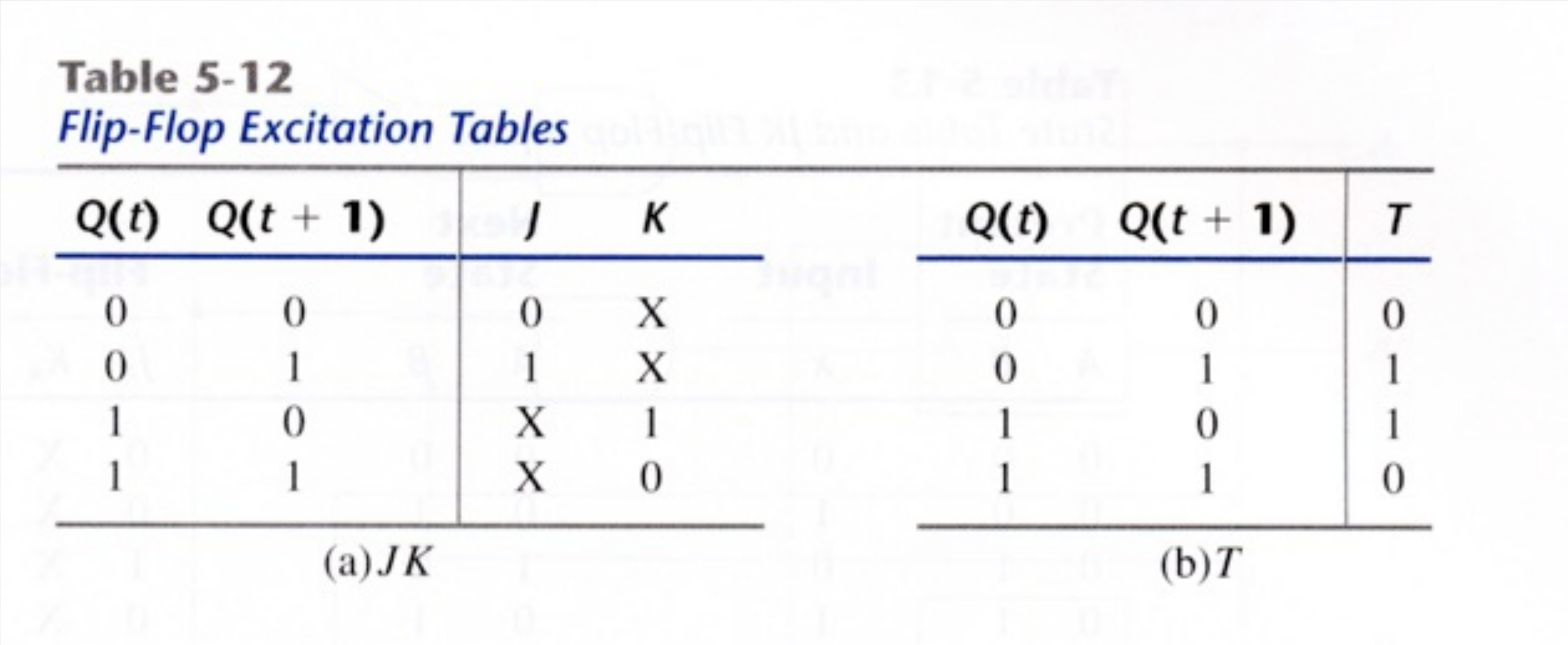


**Table 5-12**

***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) $JK$

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

(b) $T$

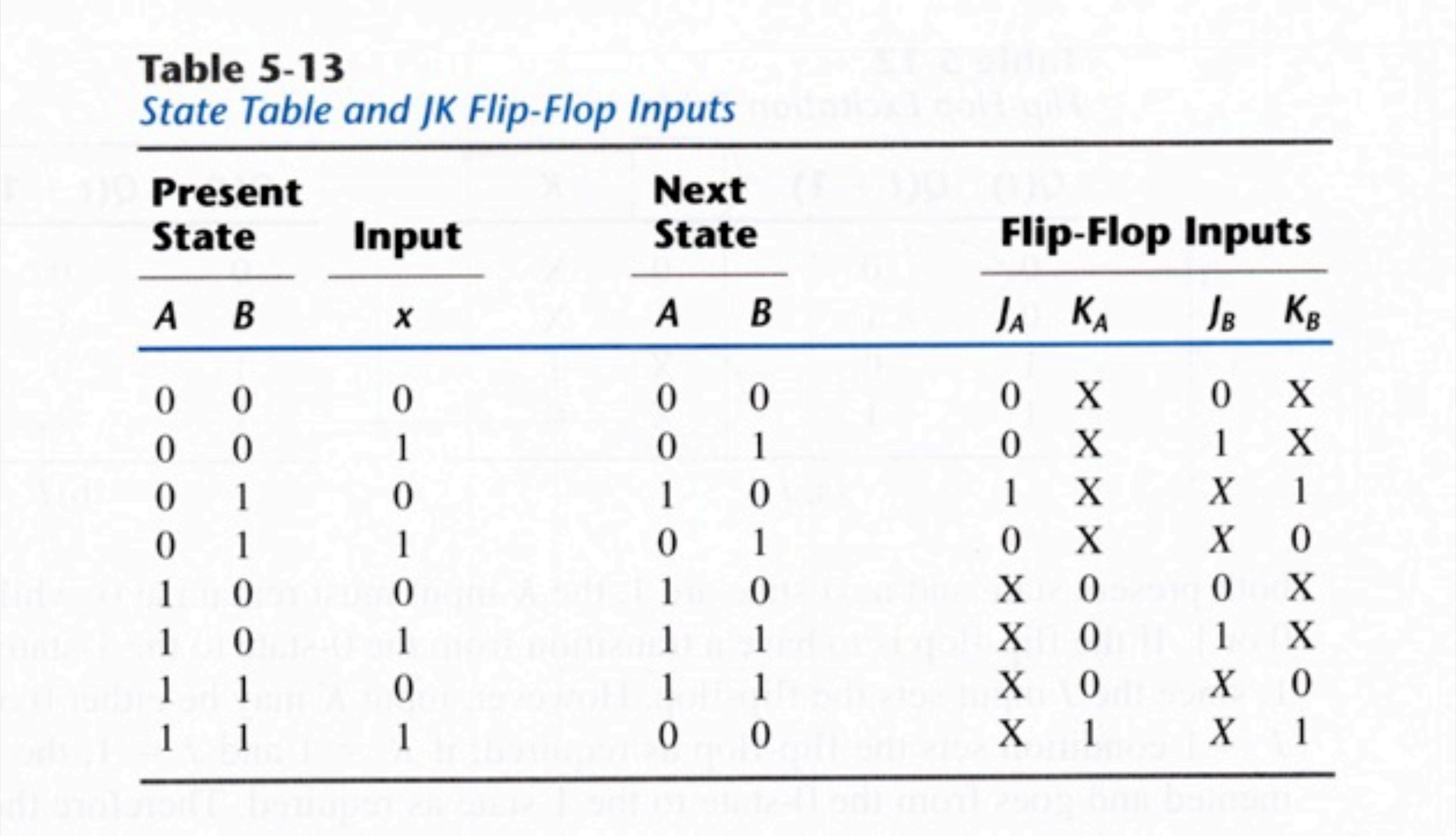
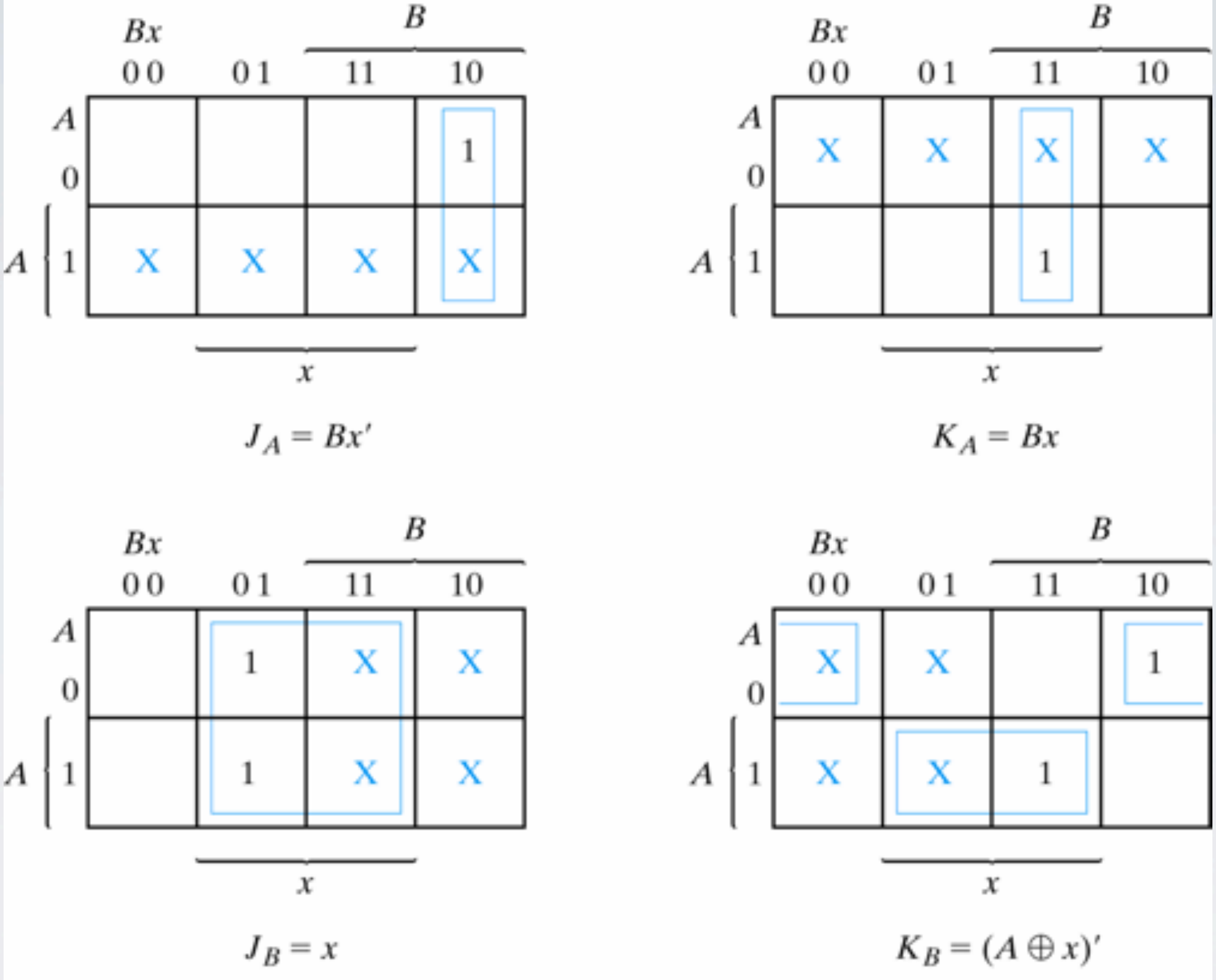


**Table 5-13**
*State Table and JK Flip-Flop Inputs*

| 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 |



| 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

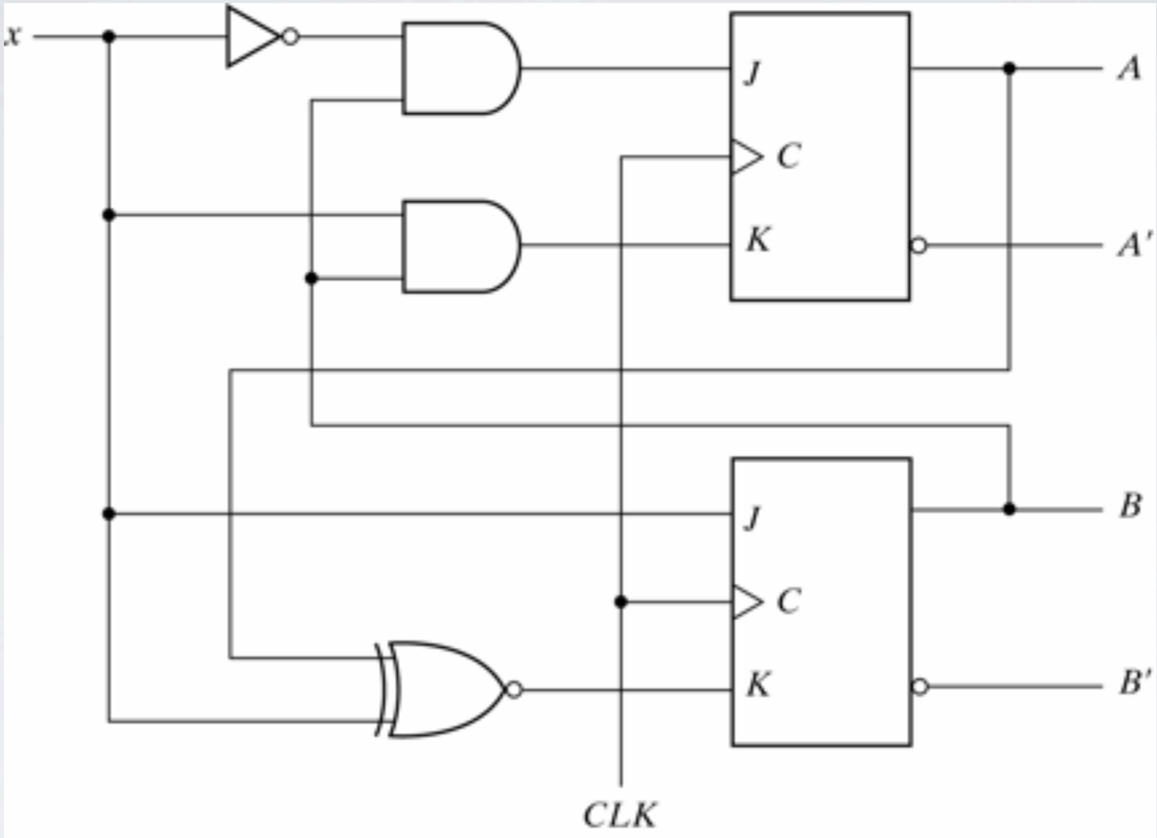
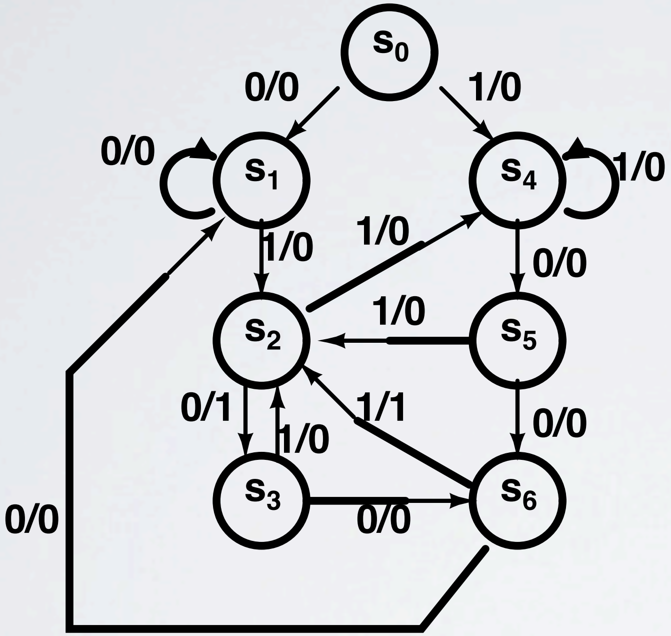


Fig. 5-28 Logic Diagram for Sequential Circuit with *JK* Flip-Flops

# STATE ASSIGNMENT GUIDELINES

- assign neighboring codes if states have the same
  - next state (G1)
  - previous state (G2)
  - outputs (G3)
- prioritize state combinations for which G1, G2, G3 apply more than once

# Sequence detector for 010 or 1001



| present | next<br>x=0 | | output<br>x=0 | |
|---|---|---|---|---|
| $s_0$ | $s_1$ | $s_4$ | 0 | 0 |
| $s_1$ | $s_1$ | $s_2$ | 0 | 0 |
| $s_2$ | $s_3$ | $s_4$ | 1 | 0 |
| $s_3$ | $s_6$ | $s_2$ | 0 | 0 |
| $s_4$ | $s_5$ | $s_4$ | 0 | 0 |
| $s_5$ | $s_6$ | $s_2$ | 0 | 0 |
| $s_6$ | $s_1$ | $s_2$ | 0 | 1 |

$s_3$ & $s_5$ are equivalent

# Sequence detector for 010 or 1001 (cont)

| present | next x=0 | | output x=0 | |
|---|---|---|---|---|
| $s_0$ | $s_1$ | $s_4$ | 0 | 0 |
| $s_1$ | $s_1$ | $s_2$ | 0 | 0 |
| $s_2$ | $s_3$ | $s_4$ | 1 | 0 |
| $s_3$ | $s_6$ | $s_2$ | 0 | 0 |
| $s_4$ | $s_3$ | $s_4$ | 0 | 0 |
| $s_6$ | $s_1$ | $s_2$ | 0 | 1 |

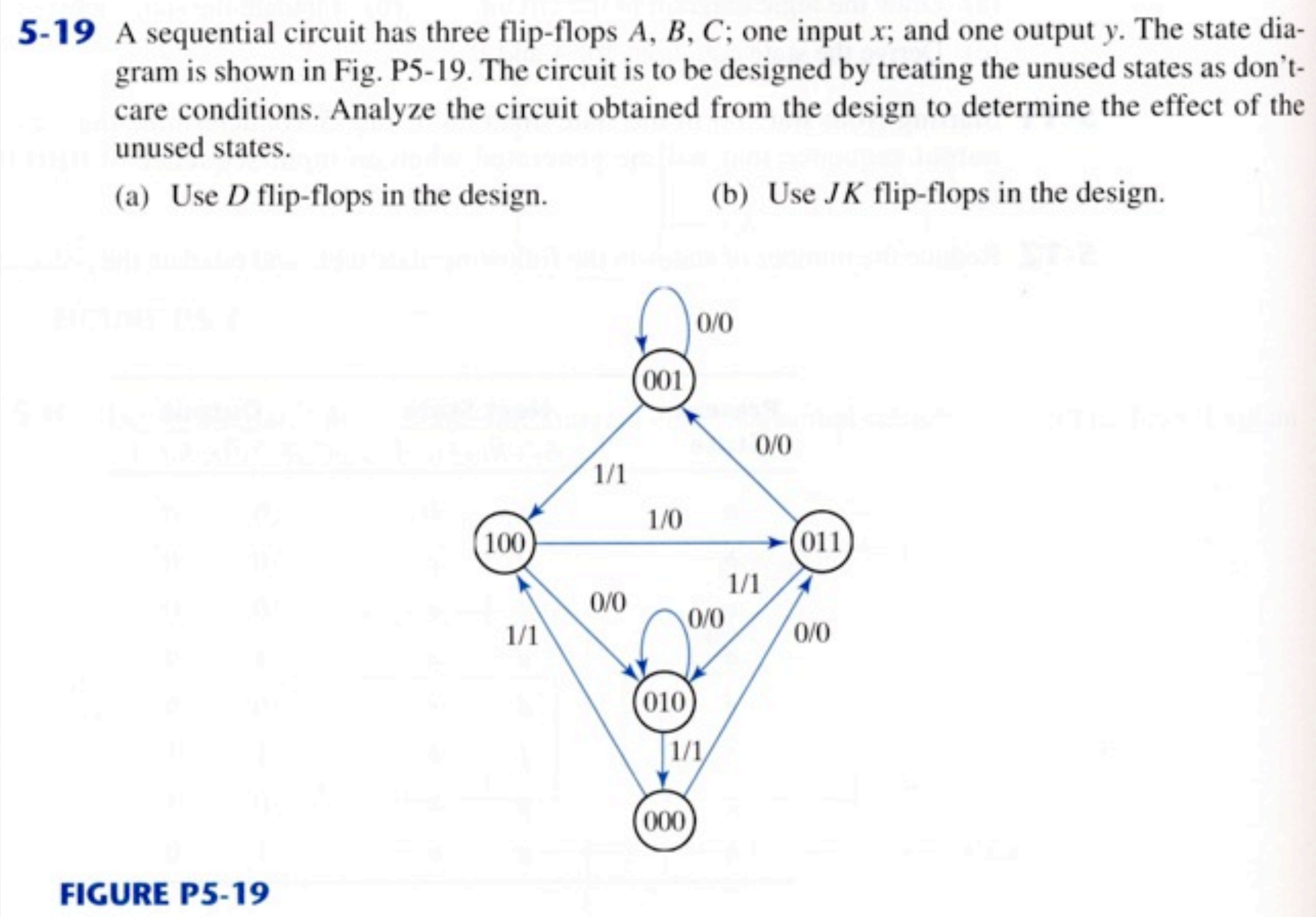
- (s0,s1,s6), (s2, s4), (s0,s2,s4), (s1,s3,s6) g1
- (s1,s2), (s3,s4) g2 × 2
- (s0, s1, s3, s4) g3

| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | s0 | s1 | s6 | X |
| 1 | s4 | s2 | X | s3 |

one possibility

# Sequence detector for 010 or 1001 (cont)

| present | | Next | | output | |
|---|---|---|---|---|---|
| | | x=0 | x=1 | x=0 | x=1 |
| s0 | 000 | 001 | 100 | 0 | 0 |
| s1 | 001 | 001 | 101 | 0 | 0 |
| s2 | 101 | 110 | 100 | 1 | 0 |
| s3 | 110 | 011 | 101 | 0 | 0 |
| s4 | 100 | 110 | 100 | 0 | 0 |
| s6 | 011 | 001 | 101 | 0 | 1 |

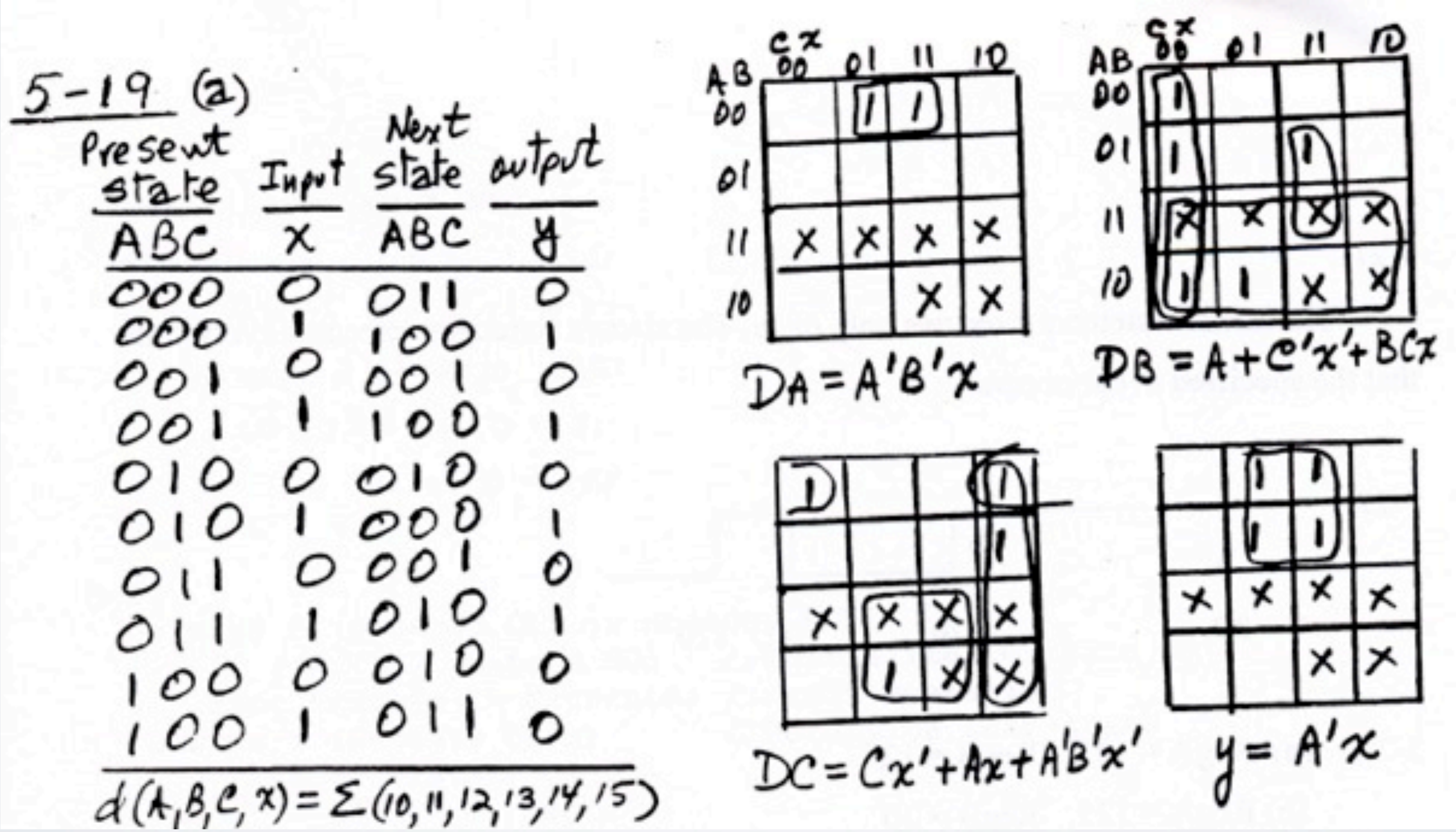


**5-19** A sequential circuit has three flip-flops $A$, $B$, $C$; one input $x$; and one output $y$. The state diagram is shown in Fig. P5-19. The circuit is to be designed by treating the unused states as don't-care conditions. Analyze the circuit obtained from the design to determine the effect of the unused states.

(a) Use $D$ flip-flops in the design. (b) Use $JK$ flip-flops in the design.

0/0
001
0/0
1/1
1/0
100
011
1/1
0/0
0/0
1/1
0/0
010
1/1
000

**FIGURE P5-19**



5-19 (a)

| Present state ABC | Input x | Next state ABC | output y |
|---|---|---|---|
| 000 | 0 | 011 | 0 |
| 000 | 1 | 100 | 1 |
| 001 | 0 | 001 | 0 |
| 001 | 1 | 100 | 1 |
| 010 | 0 | 010 | 0 |
| 010 | 1 | 000 | 1 |
| 011 | 0 | 001 | 0 |
| 011 | 1 | 010 | 1 |
| 100 | 0 | 010 | 0 |
| 100 | 1 | 011 | 0 |

d(A,B,C,x) = Σ(10,11,12,13,14,15)

| AB \ Cx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 |  | 1 | 1 |  |
| 01 |  |  |  |  |
| 11 | X | X | X | X |
| 10 |  |  | X | X |

DA = A'B'x

| AB \ Cx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 |  |  |  |
| 01 | 1 |  | 1 |  |
| 11 | X | X | X | X |
| 10 | 1 | 1 | X | X |

DB = A + C'x' + BCx

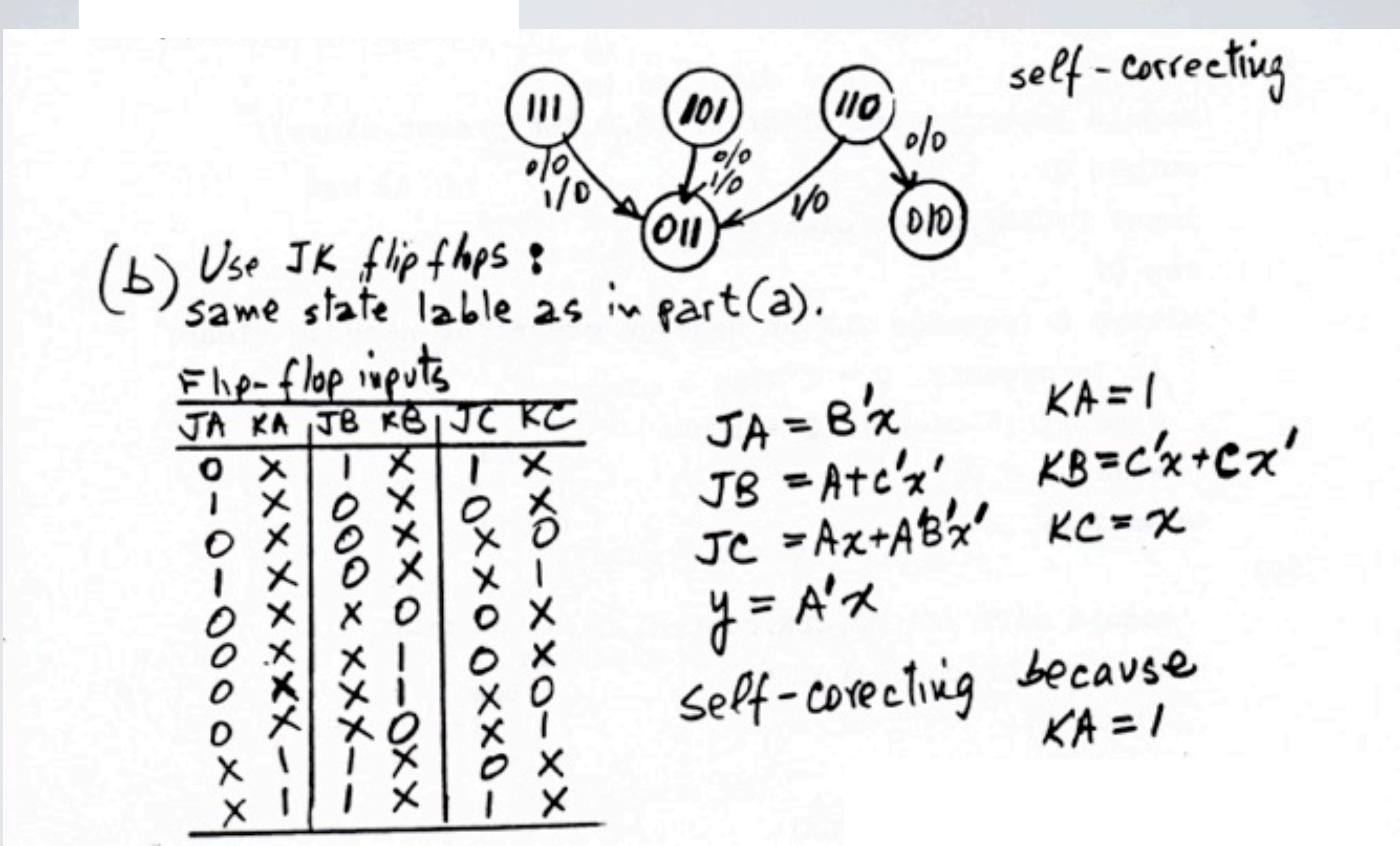
| | | | |
|---|---|---|---|
| 1 |  |  | 1 |
|  |  |  | 1 |
| X | X | X | X |
|  | 1 | X | X |

DC = Cx' + Ax + A'B'x'

| | | | |
|---|---|---|---|
|  | 1 | 1 |  |
|  | 1 | 1 |  |
| X | X | X | X |
|  |  | X | X |

y = A'x

Sunday, April 8, 12



self-correcting

111
101
110
011
010
0/0
1/0
0/0
1/0
1/0
0/0

(b) Use JK flip flops:
same state lable as in part(a).

Flip-flop inputs

| JA | KA | JB | KB | JC | KC |
|---|---|---|---|---|---|
| 0 | X | 1 | X | 1 | X |
| 1 | X | 0 | X | 0 | X |
| 0 | X | 0 | X | X | 0 |
| 1 | X | 0 | X | X | 1 |
| 0 | X | X | 0 | 0 | X |
| 0 | X | X | 1 | 0 | X |
| 0 | X | X | 1 | X | 0 |
| 0 | X | X | 0 | X | 1 |
| X | 1 | 1 | X | 0 | X |
| X | 1 | 1 | X | 1 | X |

$JA = B'x$ $\quad KA = 1$

$JB = A + C'x'$ $\quad KB = C'x + Cx'$

$JC = Ax + AB'x'$ $\quad KC = x$

$y = A'x$

Self-corecting because $KA = 1$

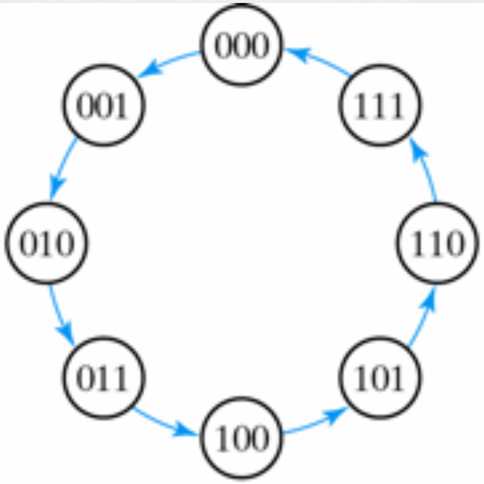
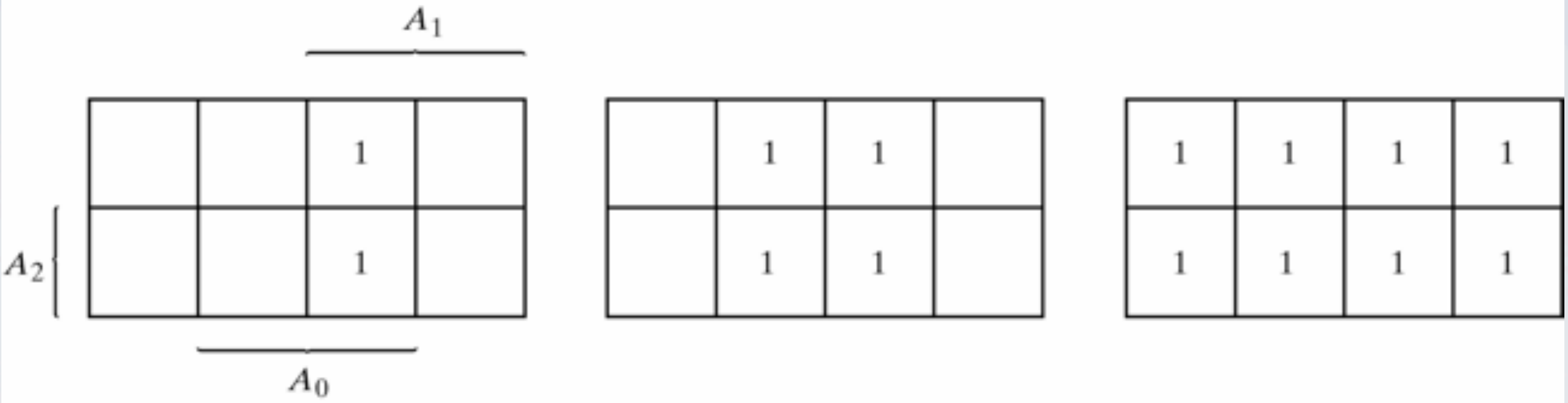


Fig. 5-29 State Diagram of 3-Bit Binary Counter



|  |  |  |  |
|---|---|---|---|
|  |  | 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

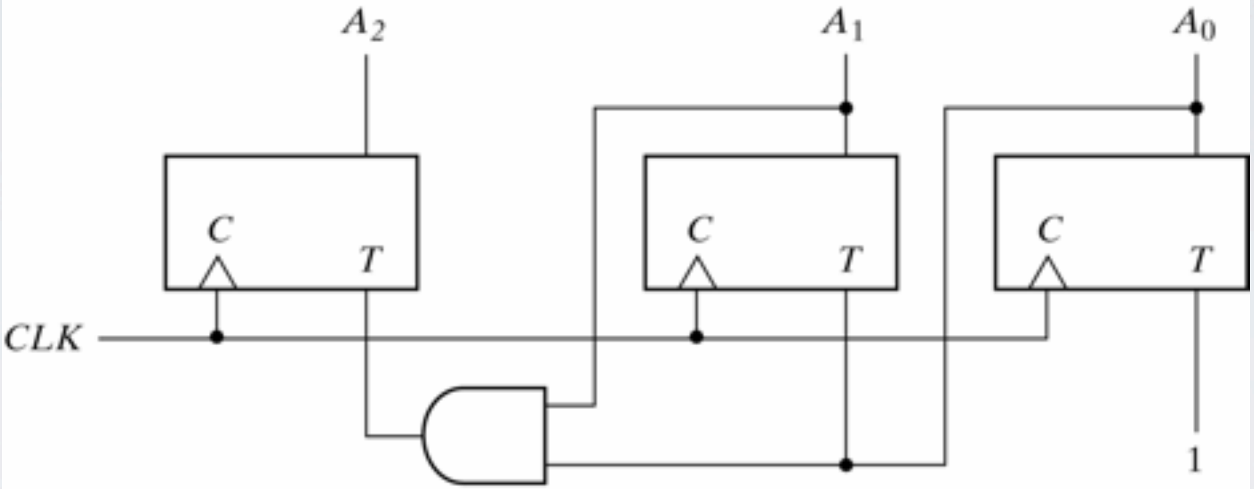


Fig. 5-31 Logic Diagram of 3-Bit Binary Counter

# EXERCISE

- draw the state diagram for a circuit that detects the sequence “0101” (left-to-right) using
  - a moore finite state machine (FSM)
  - a mealy fsm

# EXERCISE

- for a clocked synchronous state machine with two inputs, X and Y, and one output, Z, The output should be 1 if the number of 1 inputs on X and Y since reset is a multiple of 4, and 0 otherwise. draw the state diagram for a
    - moorE machine
    - Mealy machine

# EXERCISE

- design a circuit to detect the sequence $d_0d_1d_2d_3D_4$=01101, where $D_0$ is the first bit to arrive at input "x". The output "y" should be a logic-1 for a full clock cycle following detection of the sequence.
  - Draw a state diagram
  - assign binary states
  - Write a state table
  - find the combinational circuit's logic expressions if d, JK and

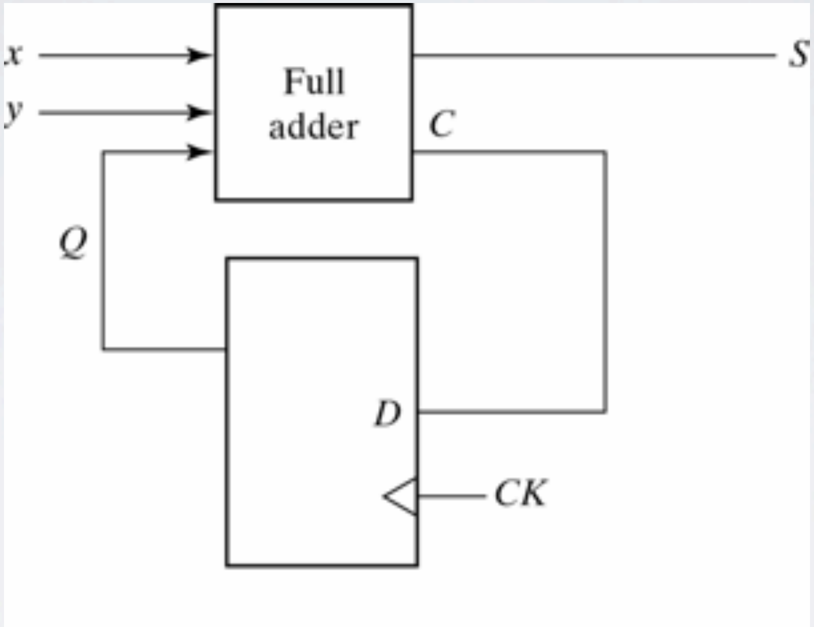


Fig. P5-7

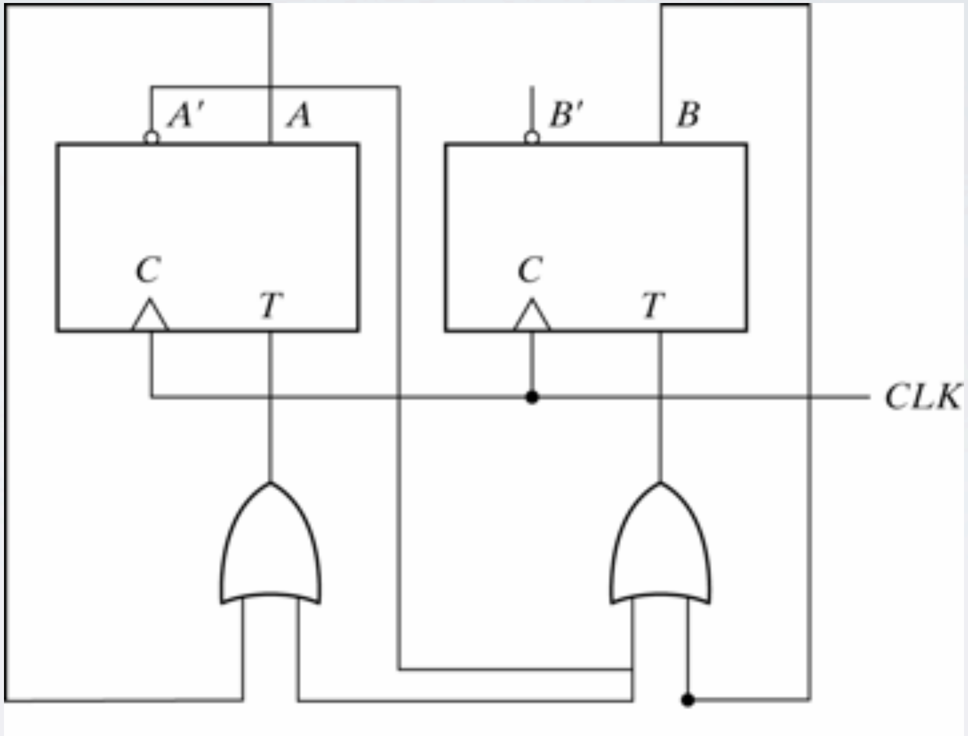


Fig. P5-8

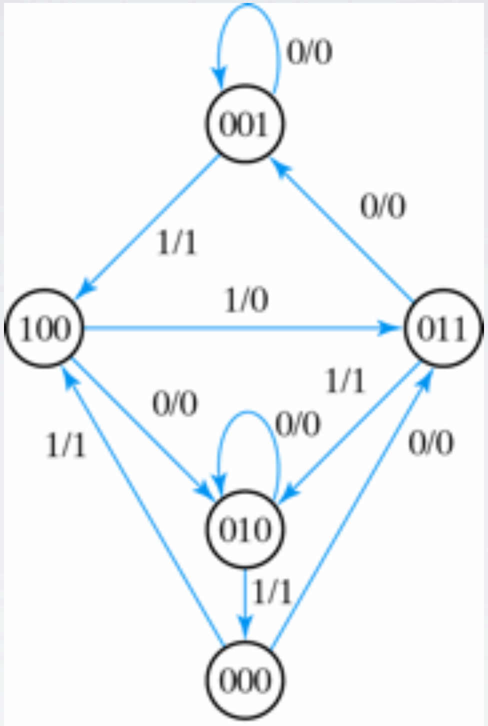


Fig. P5-19