IAA6007: Computer Architecture
Ch.1. Digital Logic Circuits

Wooil Kim
Dept. of Computer Science & Engineering
Incheon National University

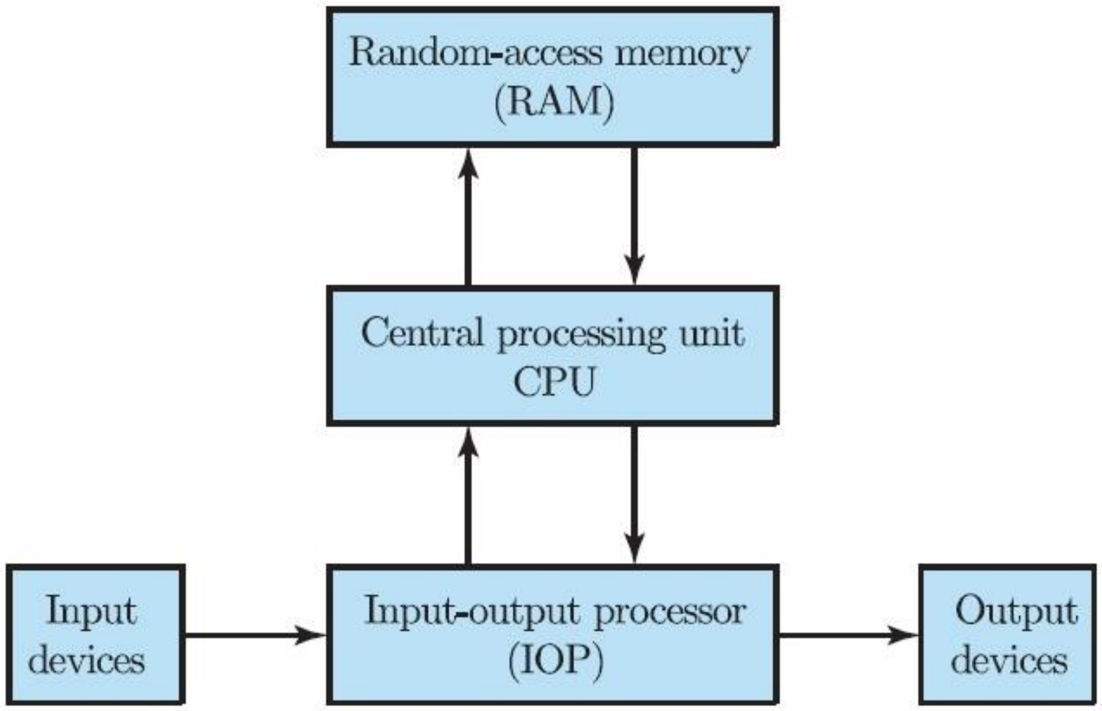
2019 Fall

Outline

- Digital computers
- Logic gates
- Boolean algebra
- Map simplification
- Combinational circuits
- Flip-flops
- Sequential circuits

1-1. Digital computers

- Digital computers use the binary number system, which has two digits 0 and 1
- Groups of bits are used to represent binary numbers, symbols, decimal digits or letters of the alphabet


Random-access memory (RAM)
Central processing unit CPU
Input devices
Input-output processor (IOP)
Output devices

1-2. Logic gates

- Two different voltages represent binary values 0 and 1
  - Ex) a signal of 3 volts represent 1, and 0.5 volt represents 0
- Gates – circuits which perform operation on binary values

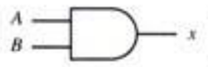
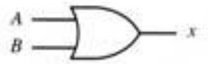
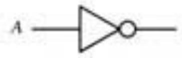
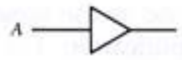
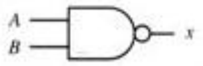
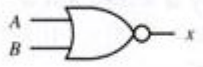
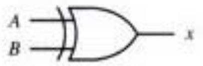
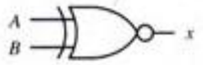
Name	Graphic symbol	Algebraic function	Truth table
AND		$x = A \cdot B$ or $x = AB$	A B | x 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$x = A + B$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$x = A'$	A | x 0 | 1 1 | 0
Buffer		$x = A$	A | x 0 | 0 1 | 1
NAND		$x = (AB)'$	A B | x 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$x = (A + B)'$	A B | x 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$x = A \oplus B$ or $x = A'B + AB'$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR or equivalence		$x = (A \oplus B)'$ or $x = A'B' + AB$	A B | x 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Figure 1-2 Digital logic gates.

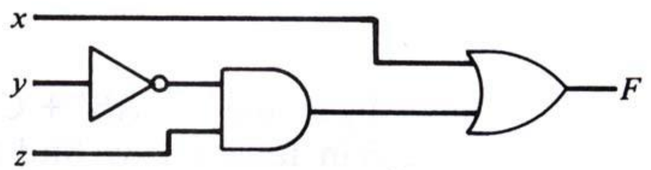
1-3. Boolean algebra

- Boolean function
  - Algebraic expression
  - Ex) F = x + y'z, where x, y, z are binary variables
  - A Boolean function may be expressed with a truth table
  - A Boolean function can be transformed from an algebraic expression into a logic diagram

Figure 1-3 Truth table and logic diagram for $F = x + y'z$.

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(a) Truth table



(b) Logic diagram

1-3. Boolean algebra

- A Boolean function can be expressed algebraically in many different ways
  - One may obtain a simpler expression that requires fewer gates

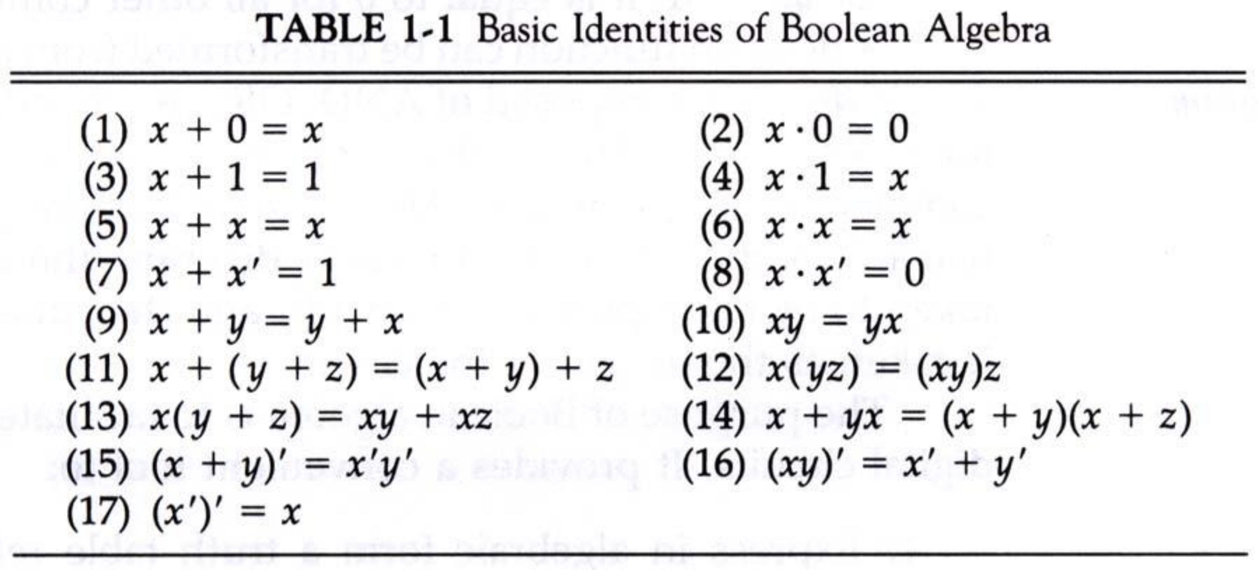


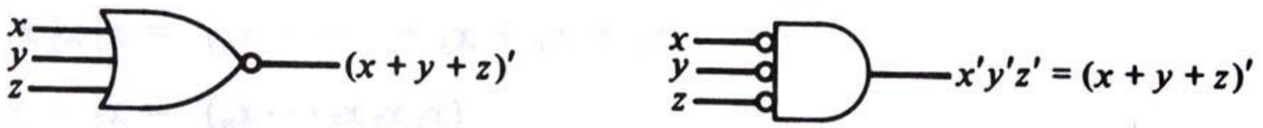
TABLE 1-1 Basic Identities of Boolean Algebra

(1) $x + 0 = x$	(2) $x \cdot 0 = 0$
(3) $x + 1 = 1$	(4) $x \cdot 1 = x$
(5) $x + x = x$	(6) $x \cdot x = x$
(7) $x + x' = 1$	(8) $x \cdot x' = 0$
(9) $x + y = y + x$	(10) $xy = yx$
(11) $x + (y + z) = (x + y) + z$	(12) $x(yz) = (xy)z$
(13) $x(y + z) = xy + xz$	(14) $x + yz = (x + y)(x + z)$
(15) $(x + y)' = x'y'$	(16) $(xy)' = x' + y'$
(17) $(x')' = x$	

1-3. Boolean algebra

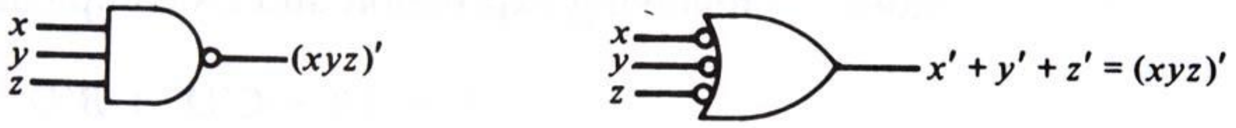
- DeMorgan's theorem
  - (x + y + z)' = x'y'z', (xyz)' = x' + y' + z'

Figure 1-4 Two graphic symbols for NOR gate.



(a) OR-invert (b) invert-AND

Figure 1-5 Two graphic symbols for NAND gate.

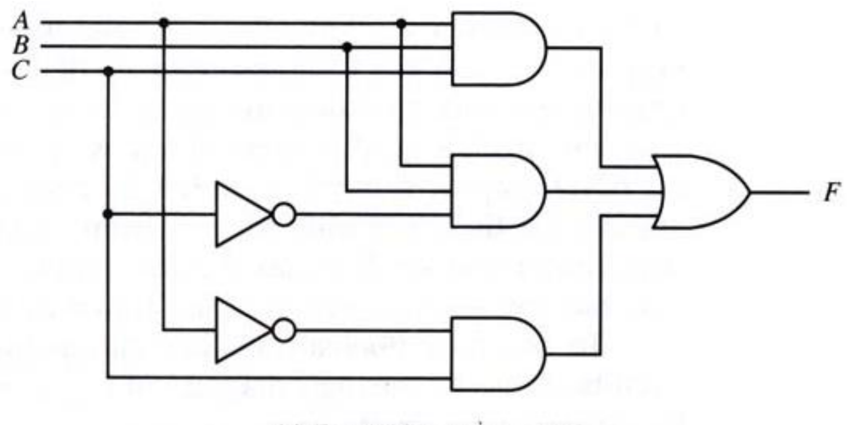


(a) AND-invert (b) invert-OR

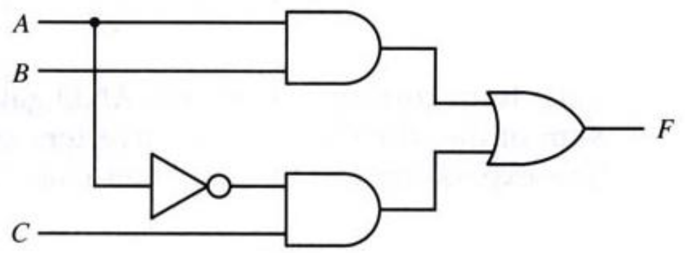
1-3. Boolean algebra

- Ex) F = ABC + ABC’ + A’C
  - F = ABC + ABC’ + A’C = AB(C + C’) + A’C

    = AB + A’C



(a) $F = ABC + ABC' + A'C$



(B) $F = AB + A'C$

Figure 1-6 Two logic diagrams for the same Boolean function.

1-3. Boolean algebra

- Complement of a function
  - Can be derived from DeMorgan’s theorem
  - Ex) F = AB + C’D’ + B’D
  - F’ = (AB + C’D’ + B’D)’

    = (AB)’(C’D’)’(B’D)’

    = (A’ + B’)(C + D)(B + D’)

  - Also can be obtained by taking duals and complementing each literal
    - Interchange OR and AND
  - F = AB + C’D’ + B’D
  - F’ = (A’ + B’)(C + D)(B + D’)

1-4. Map simplification

- Minterm: each combination of the variables
  - x, y -> xy, x'y, xy', x'y'
  - n variables -> 2^n minterms
- Sum of Products (SOP)
  - Boolean expression ORing AND terms
  - F = y' + xy + x'y'z'
- Product of Sums (POS)
  - Boolean expression ANDing OR terms
  - F = y'(x + y)(x' + y' + z')

1-4. Map simplification

- Canonical SOP form
  - Boolean function expressed as sum of minterms
  - A Boolean function may be converted to a canonical SOP form
  - Ex) $F(x, y, z) = y' + xy + x'yz'$

$$= y'(x + x') + xy(z + z') + x'yz'$$

$$= xy' + x'y' + xyz + xyz' + x'yz'$$

$$= xy'(z + z') + x'y'(z + z') + xyz + xyz' + x'yz'$$

$$= xy'z + xy'z' + x'y'z + x'y'z' + xyz + xyz' + x'yz'$$

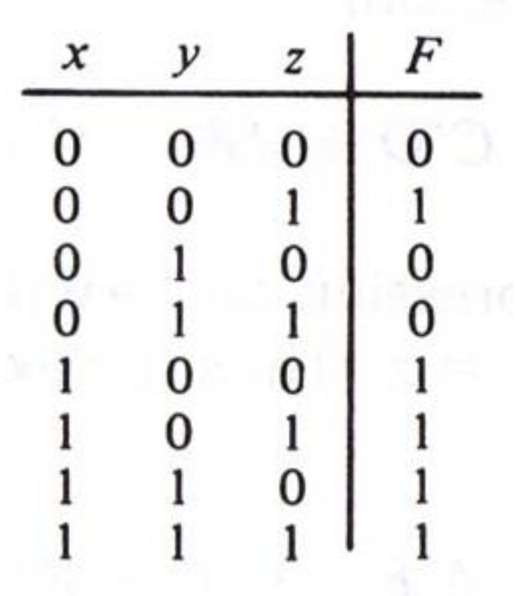
1-4. Map simplification

- A Boolean function may be expressed by listing the decimal equivalent of those minterms that produce 1
  - Ex) F(x, y, z) = xy'z + xy'z' + x'y'z + x'y'z' + xyz + xyz' + x'yz'

    $= \Sigma\ (0, 1, 2, 4, 5, 6, 7)$

- Ex) $F(x, y, z) = \Sigma\ (1, 4, 5, 6, 7)$

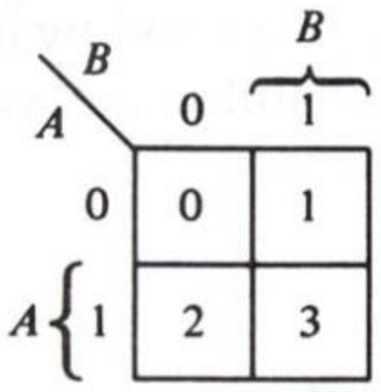
  = x'y'z + xy'z' + xy'z + xyz' + xyz



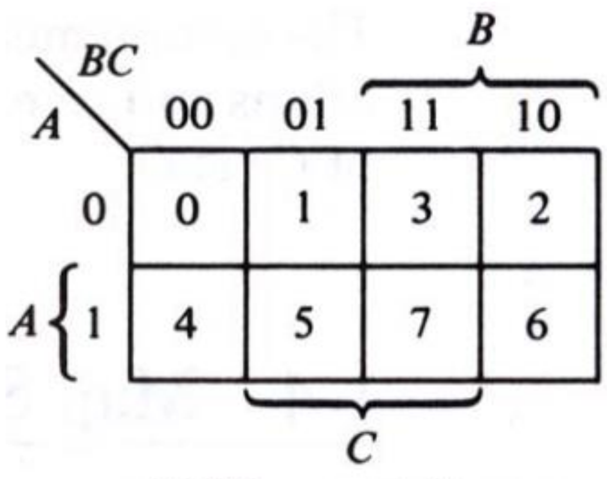
x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

1-4. Map simplification

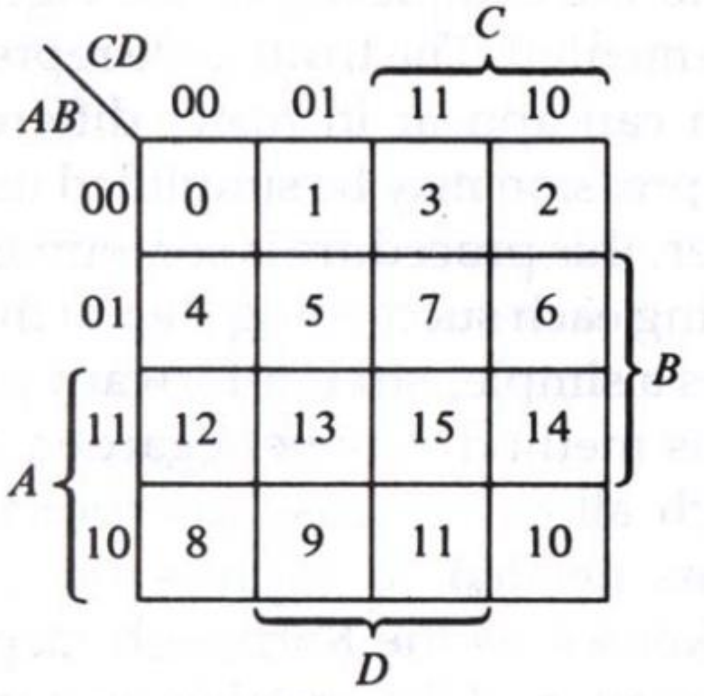
- Karnaugh map (K-map)



(a) Two-variable map



(b) Three-variable map



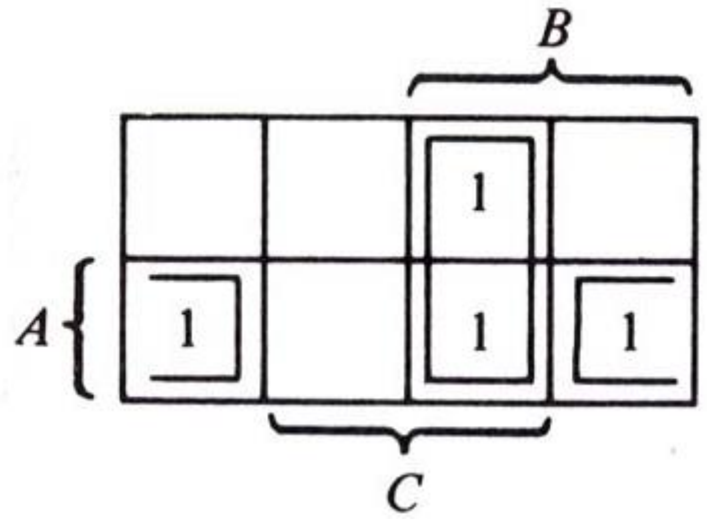
(c) Four-variable map

1-4. Map simplification

- Step 1: Insert 1’s in those squares where the function is 1
- Step 2: Combine the squares containing 1’s in groups of adjacent squares
  - Must contain a number of squares that is an integral power of 2 (e.g., 2, 4, 8, …)
  - Groups of combined adjacent squares may share one or more
- Step 3: Take OR of those terms
  - Each group of squares represents an algebraic term

1-4. Map simplification

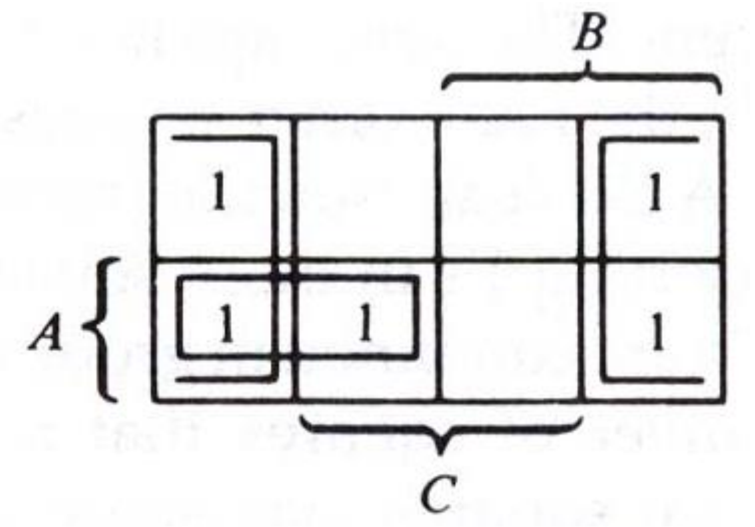
- Ex) F(A, B, C) = Σ(3, 4, 6,7)



  - F = BC + AC’

1-4. Map simplification

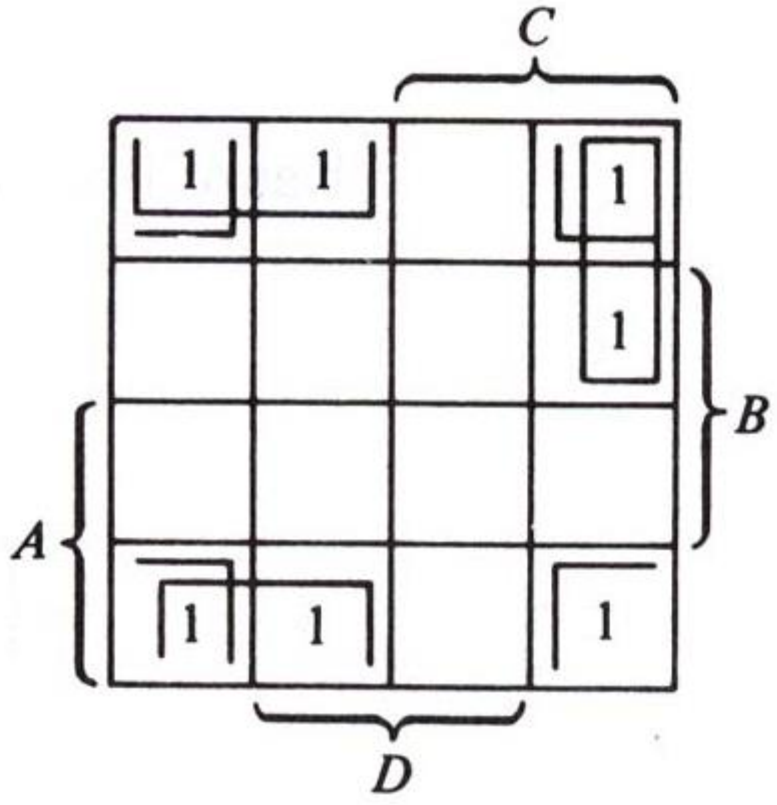
- Ex) F(A, B, C) = Σ(0, 2, 4, 5, 6)



  - F = C' + AB'

1-4. Map simplification

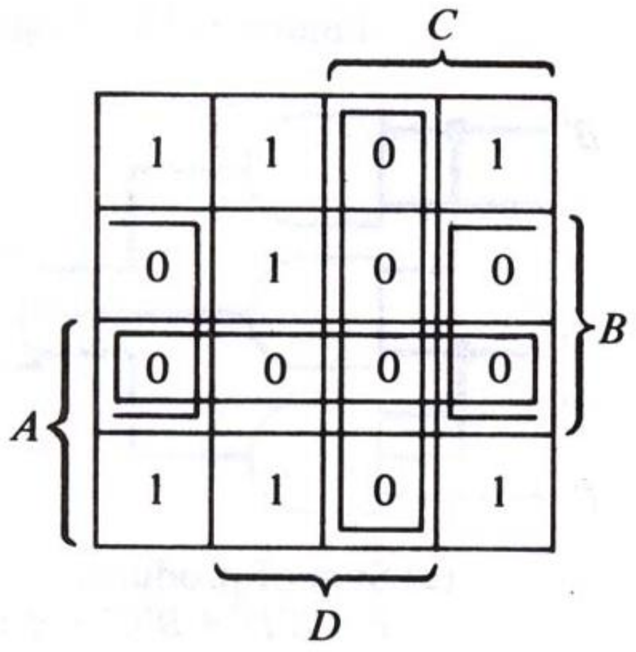
- Ex) F(A, B, C, D) = Σ(0. 1. 2. 6. 8. 9. 10)



  - F = B’D’ + B’C’+ A’CD’

1-4. Map simplification

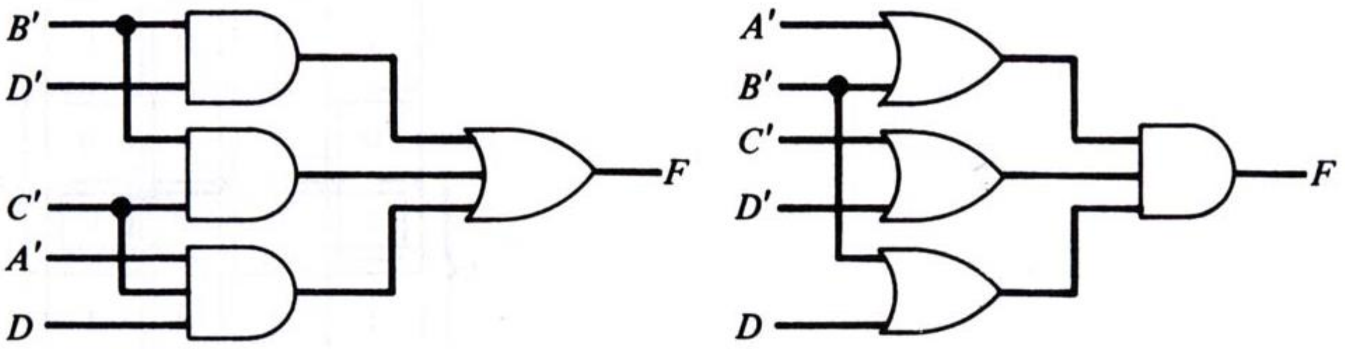
- Product of sums simplification
  - Ex) F(A, B, C, D) = Σ(0, 1, 2, 5, 8, 9, 10)
- Sum of products (SOP)
  - F = B'D' + B'C' + A'C'D
- Combine squares of 0
  - F' = AB + CD + BD'
- Product of Sums (POS)
  - By DeMorgan's theorem
  - F = (A' + B')(C' + D')(B' + D)



1-4. Map simplification

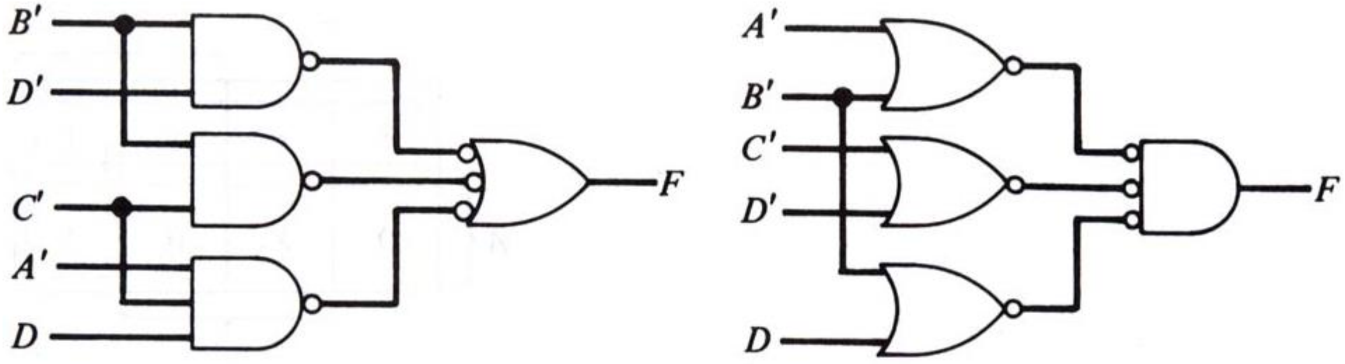
- Product of sums

Figure 1-12 Logic diagrams with AND and OR gates.



(a) Sum of products:
$F = B'D' + B'C' + A'C'D$

(b) Product of sums:
$F = (A' + B')(C' + D')(B' + D)$



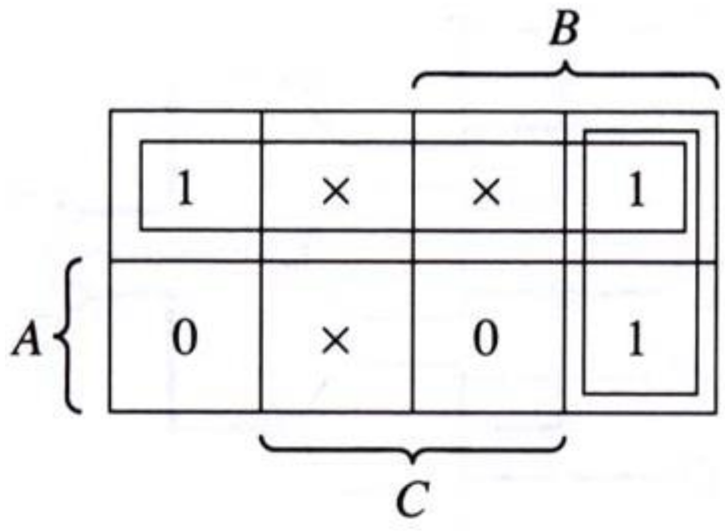
(a) With NAND gates

(b) With NOR gates

Figure 1-13 Logic diagrams with NAND or NOR gates.

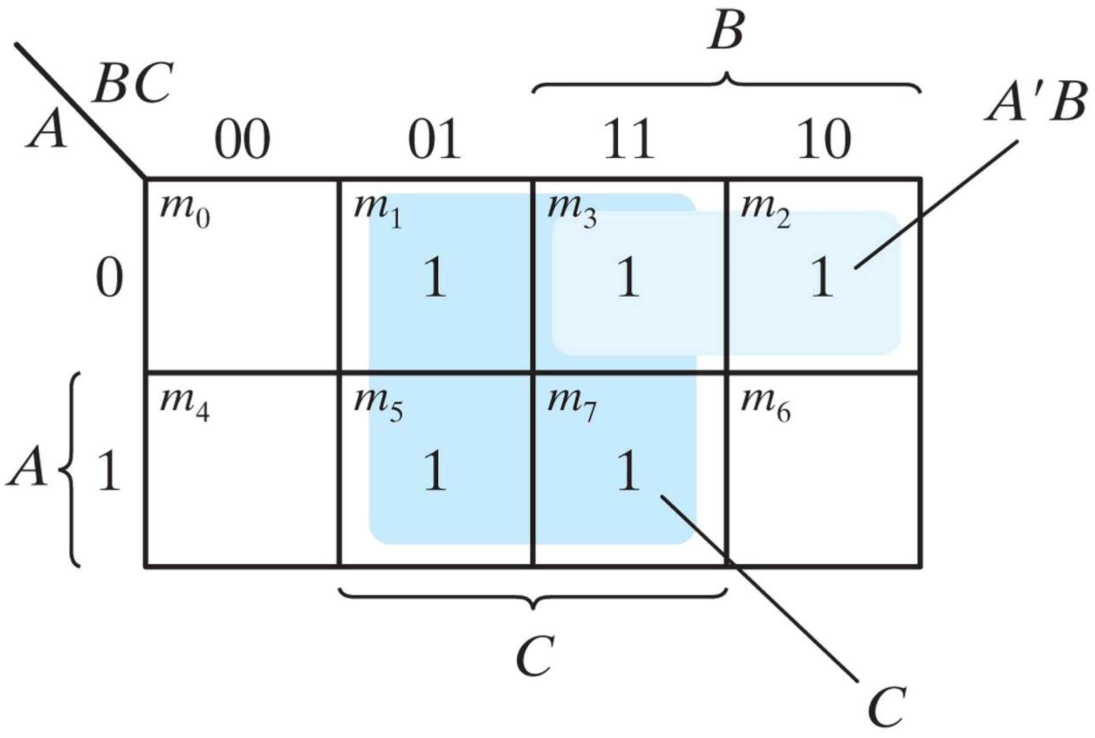
1-4. Map simplification

- Don’t care conditions
  - Not matter if function produce 0 or 1
- Ex) F(A, B, C) = Σ(0, 2, 6)
  - d(A, B, C) = Σ(1, 3, 5); don’t care condition
  - F = A’ + BC’
  - F = A’C’ + BC’; not good



1-4. Map simplification

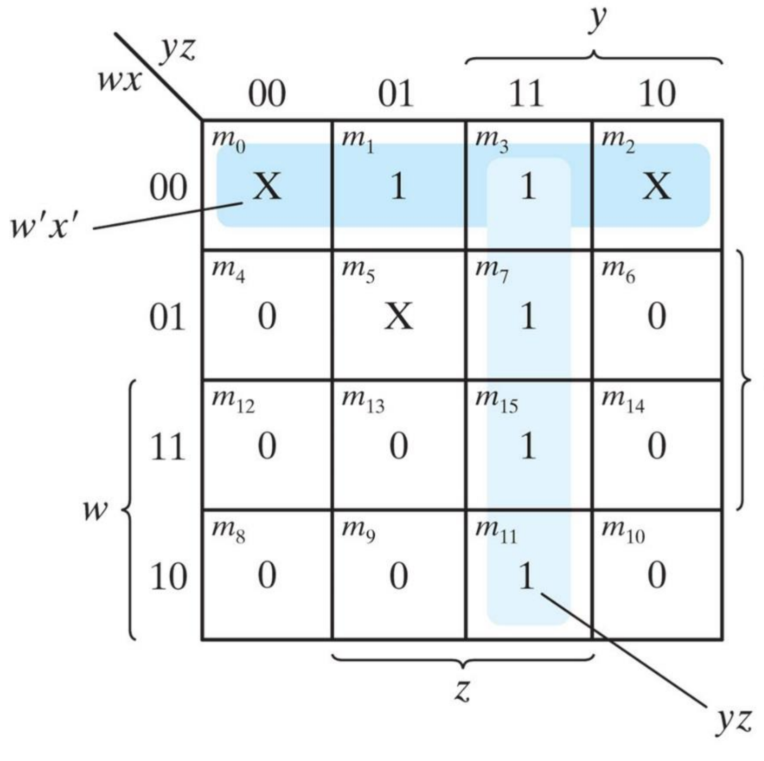
- Ex) Simplify the Boolean function, F = A'C + A'B + AB'C + BC



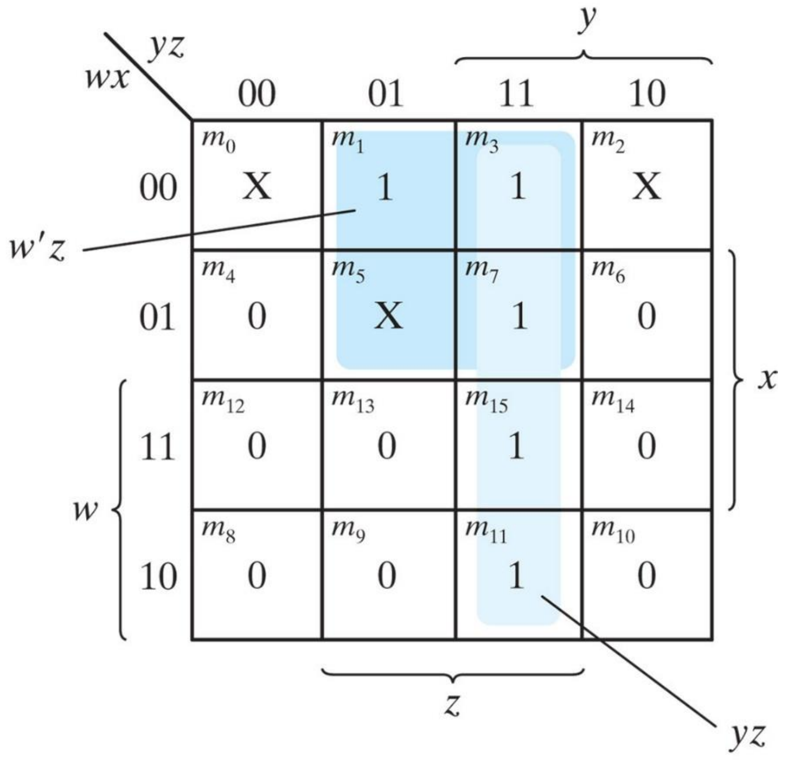
- F = C + A'B

1-4. Map simplification

- Ex) Simplify the Boolean function
- F(w, x, y, z) = Σ(1, 3, 7, 11, 15)
- Don’t-care conditions, d(w, x, y, z) = Σ(0, 2, 5)



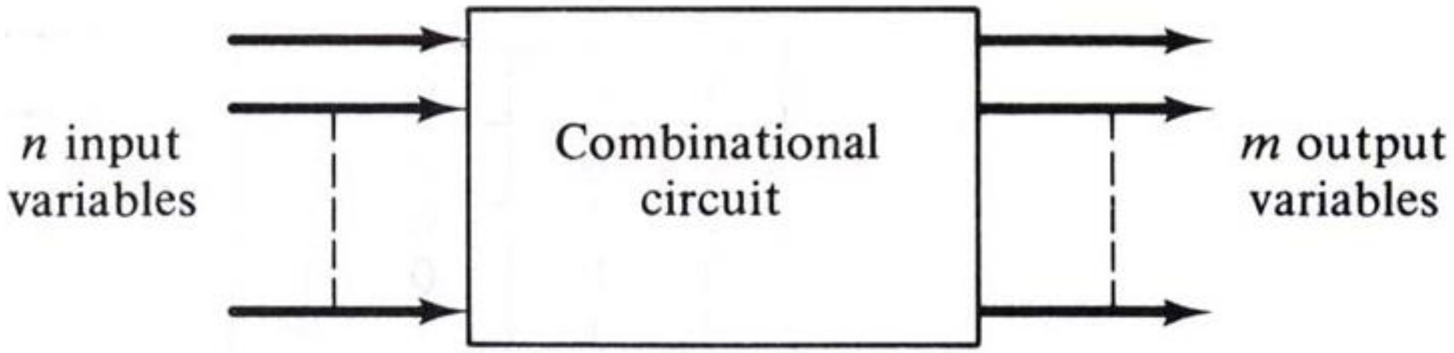
(a) $F = yz + w'x'$



(b) $F = yz + w'z$

1-5. Combinational circuits

- Connected arrangement f logic gates with a set of inputs and outputs
- Can be described by a truth table
  - Shows binary relationship between n input variables and m output variables

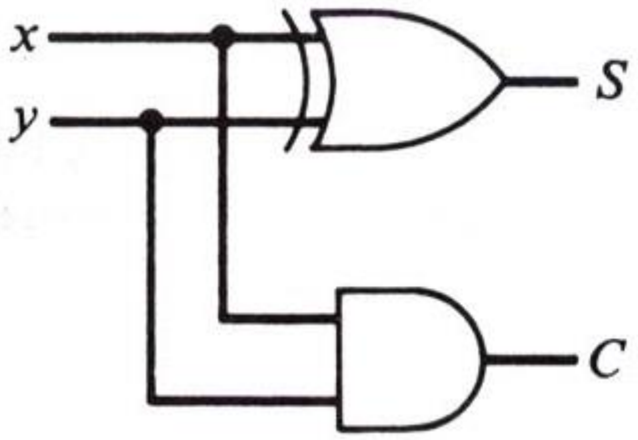


1-5. Combinational circuits

- Half adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(a) Truth table



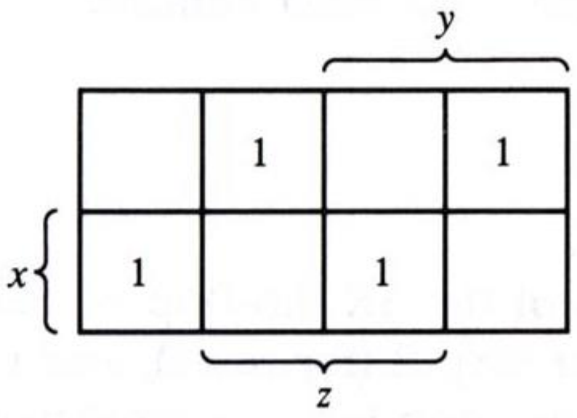
(b) Logic diagram

- $S = x'y + xy' = x \oplus y$
- $C = xy$

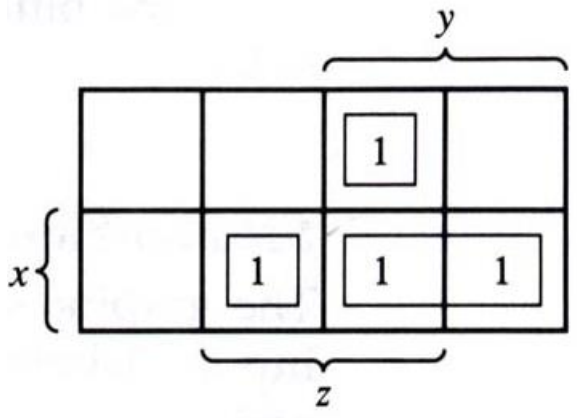
1-5. Combinational circuits

- Full adder

Inputs			Outputs	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



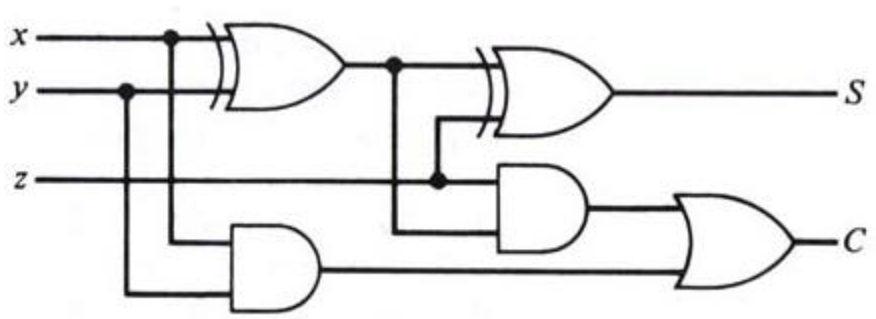
$$S = x'y'z + x'yz' + xy'z' + xyz$$
$$= x \oplus y \oplus z$$



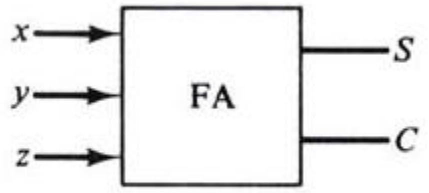
$$C = xy + xz + yz$$
$$= xy + (x'y + xy')z$$

1-5. Combinational circuits

- Full adder
  - $S = x \oplus y \oplus z$
  - $C = xy + (x'y + xy')z = xy + (x \oplus y)z$



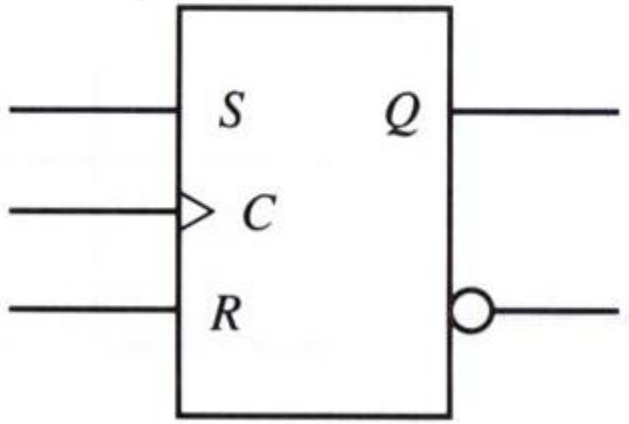
(a) Logic diagram



(b) Block diagram

1-6. Flip-flop

- SR flip-flop



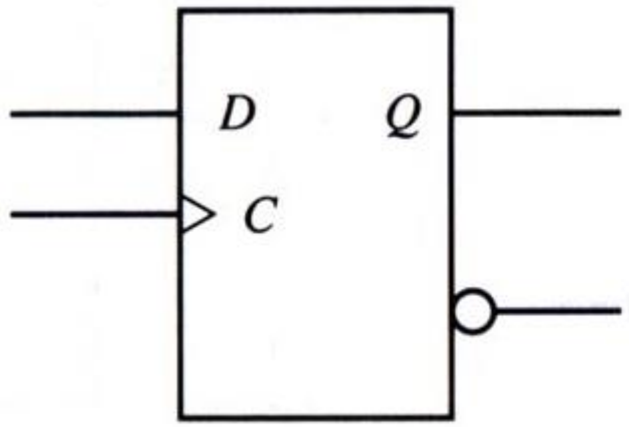
(a) Graphic symbol

S	R	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	?	Indeterminate

(b) Characteristic table

1-6. Flip-flop

- D flip-flop



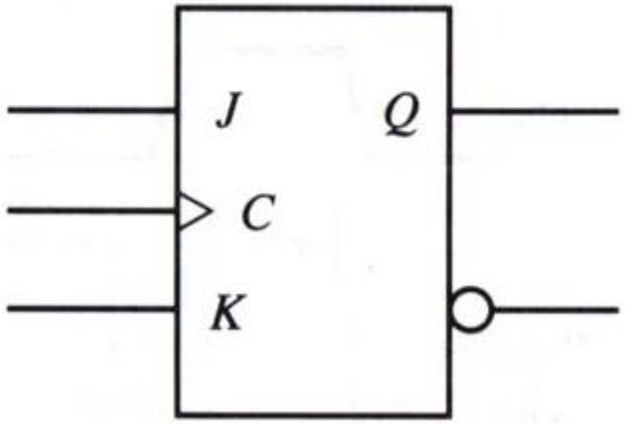
(a) Graphic symbol

D	$Q(t+1)$	
0	0	Clear to 0
1	1	Set to 1

(b) Characteristic table

1-6. Flip-flop

- JK flip-flop



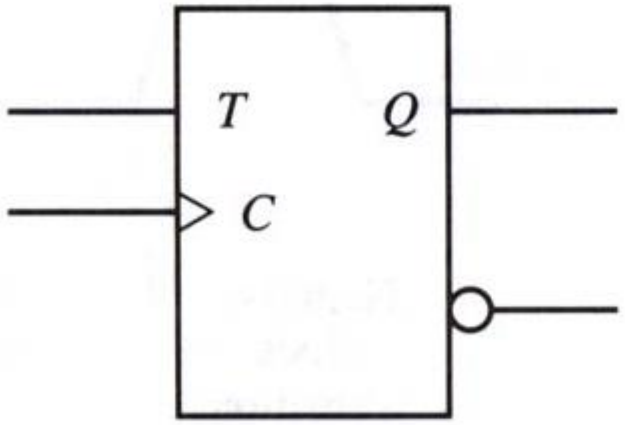
(a) Graphic symbol

J	K	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	$Q'(t)$	Complement

(b) Characteristic table

1-6. Flip-flop

- T flip-flop



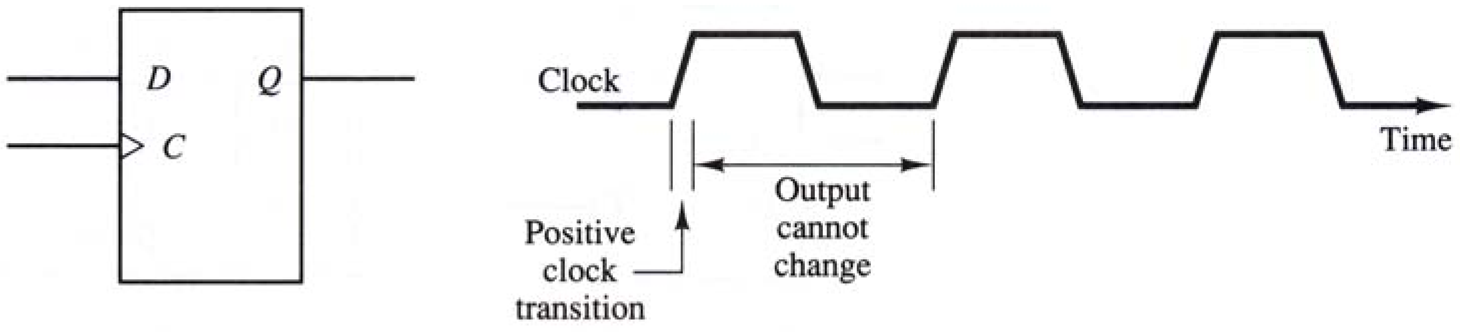
(a) Graphic symbol

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

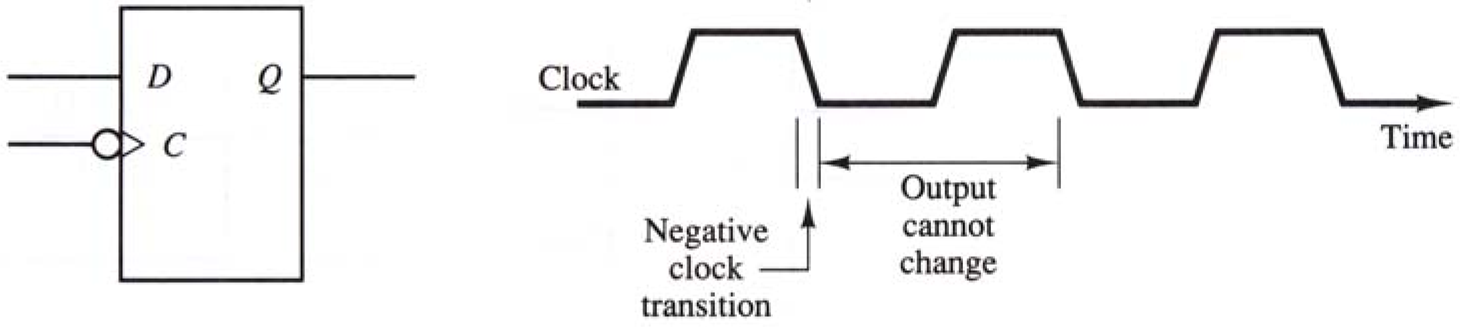
(b) Characteristic table

1-6. Flip-flop

- Edge-triggered flip-flop
  - State change is synchronized during a clock pulse transition



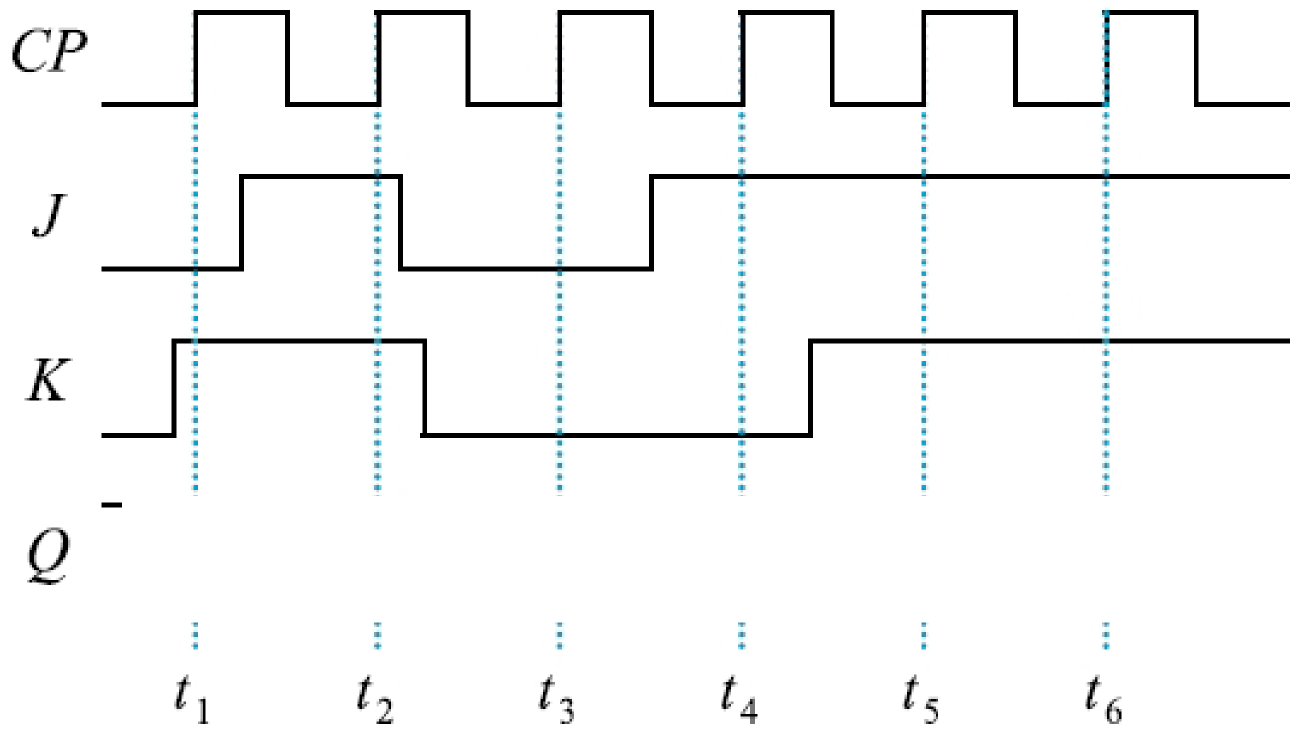
(a) Positive-edge-triggered *D* flip-flop.



(b) Negative-edge-triggered *D* flip-flop.

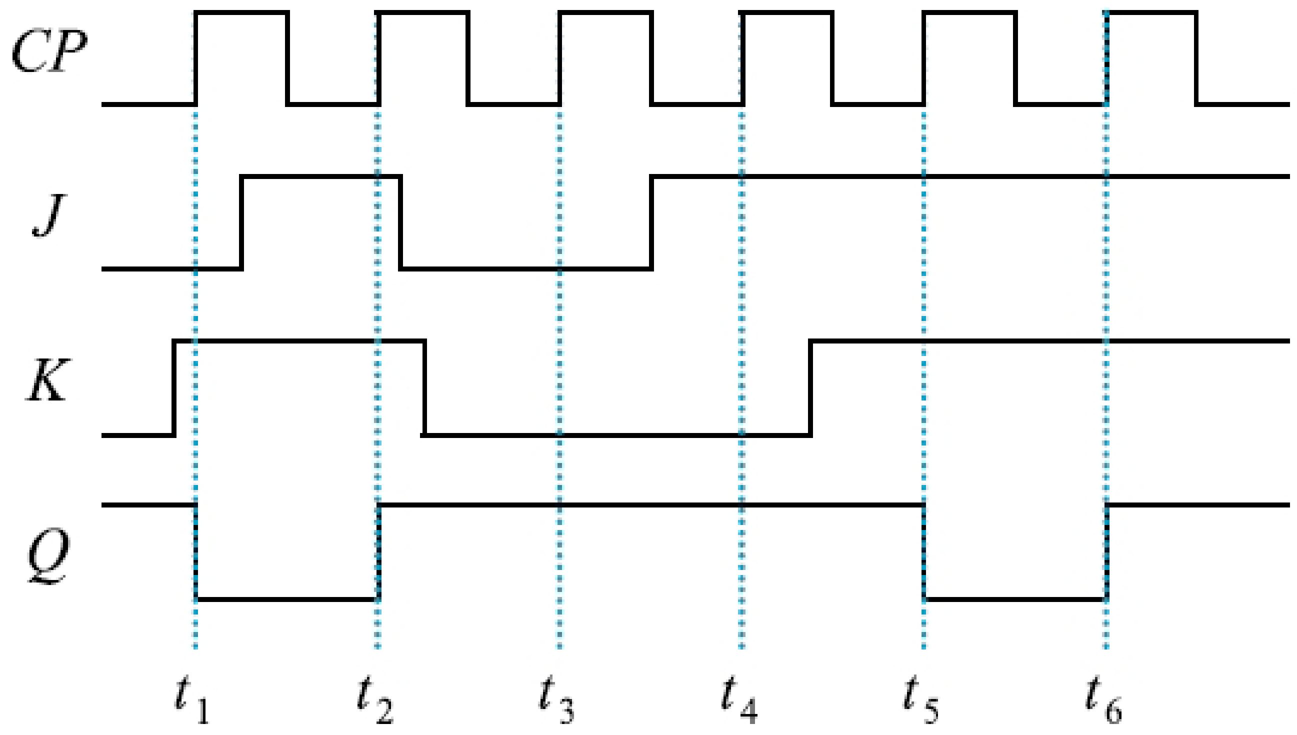
1-6. Flip-flop

- Example of edge-triggered flip-flop
  - Positive-edge-triggered JK flip-flop
  - Q: Initially set to 1



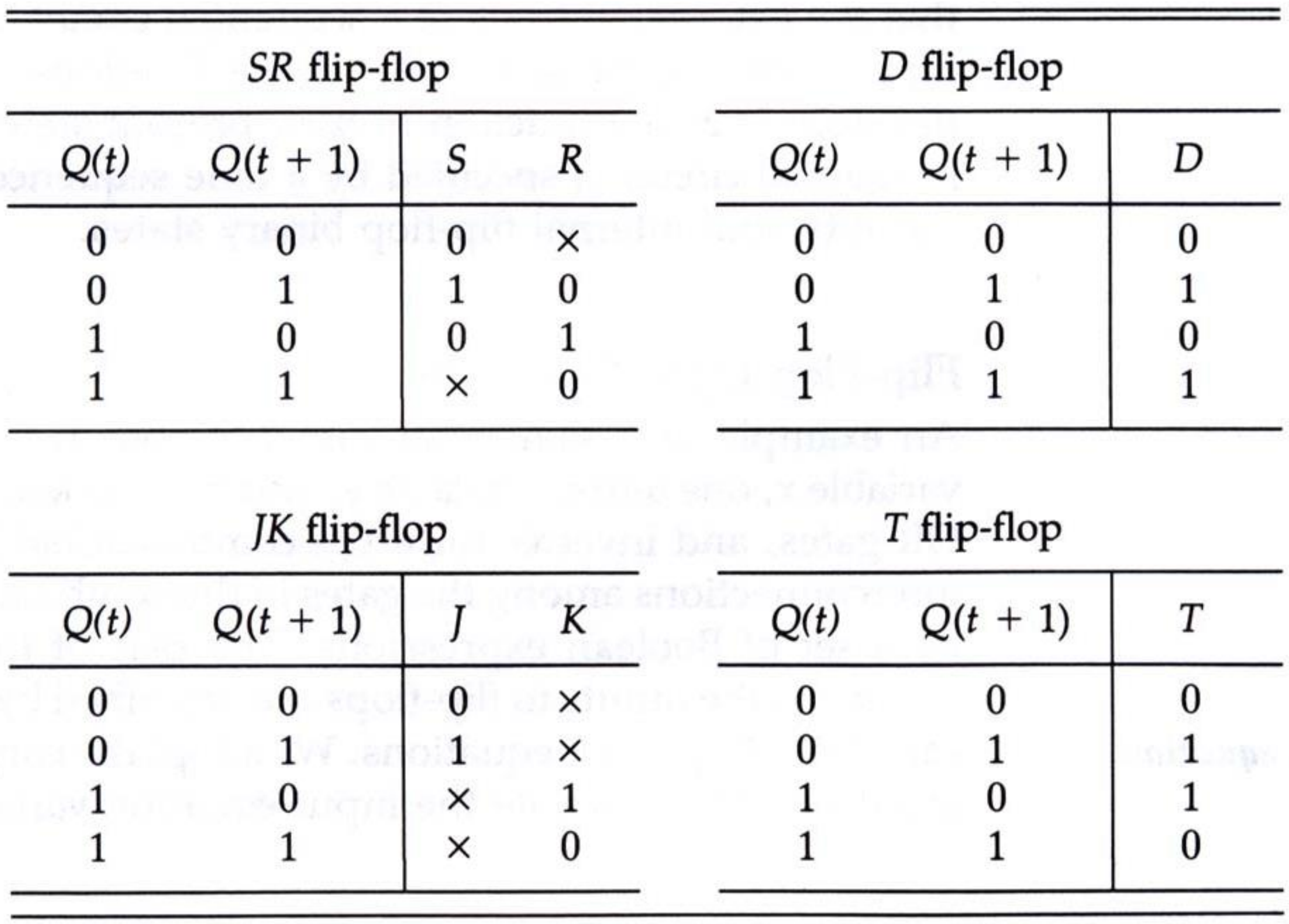
1-6. Flip-flop

- Example of edge-triggered flip-flop
  - Positive-edge-triggered JK flip-flop
  - Q: Initially set to 1



1-6. Flip-flop

- Excitation table



SR flip-flop

$Q(t)$	$Q(t+1)$	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

D flip-flop

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

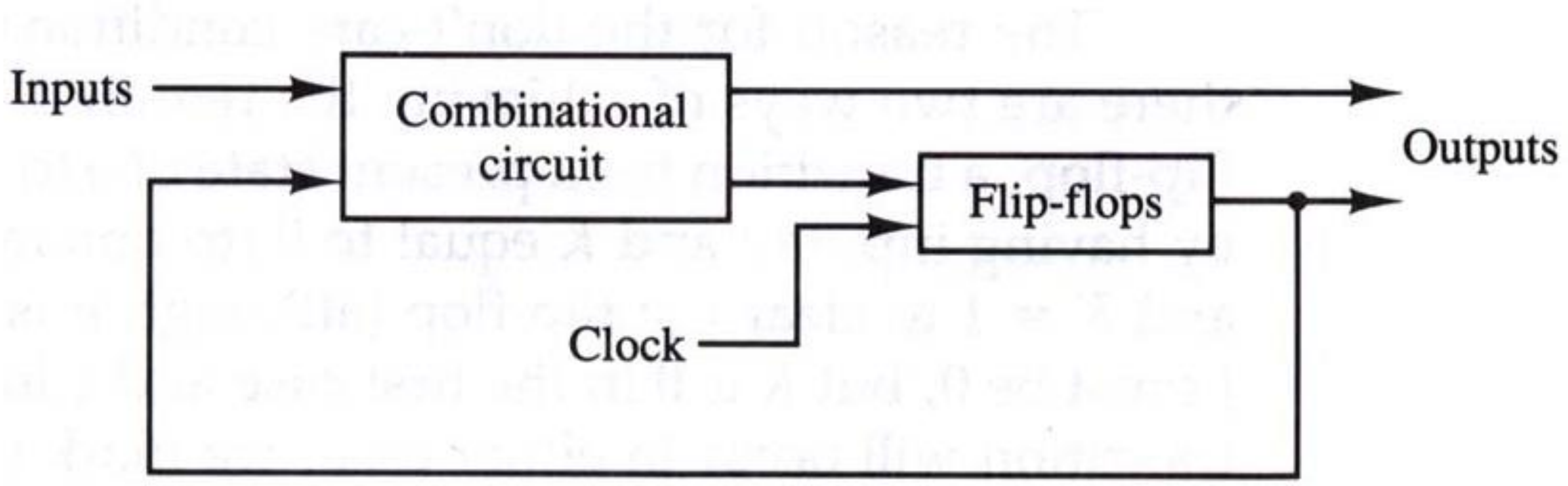
JK flip-flop

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

T flip-flop

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

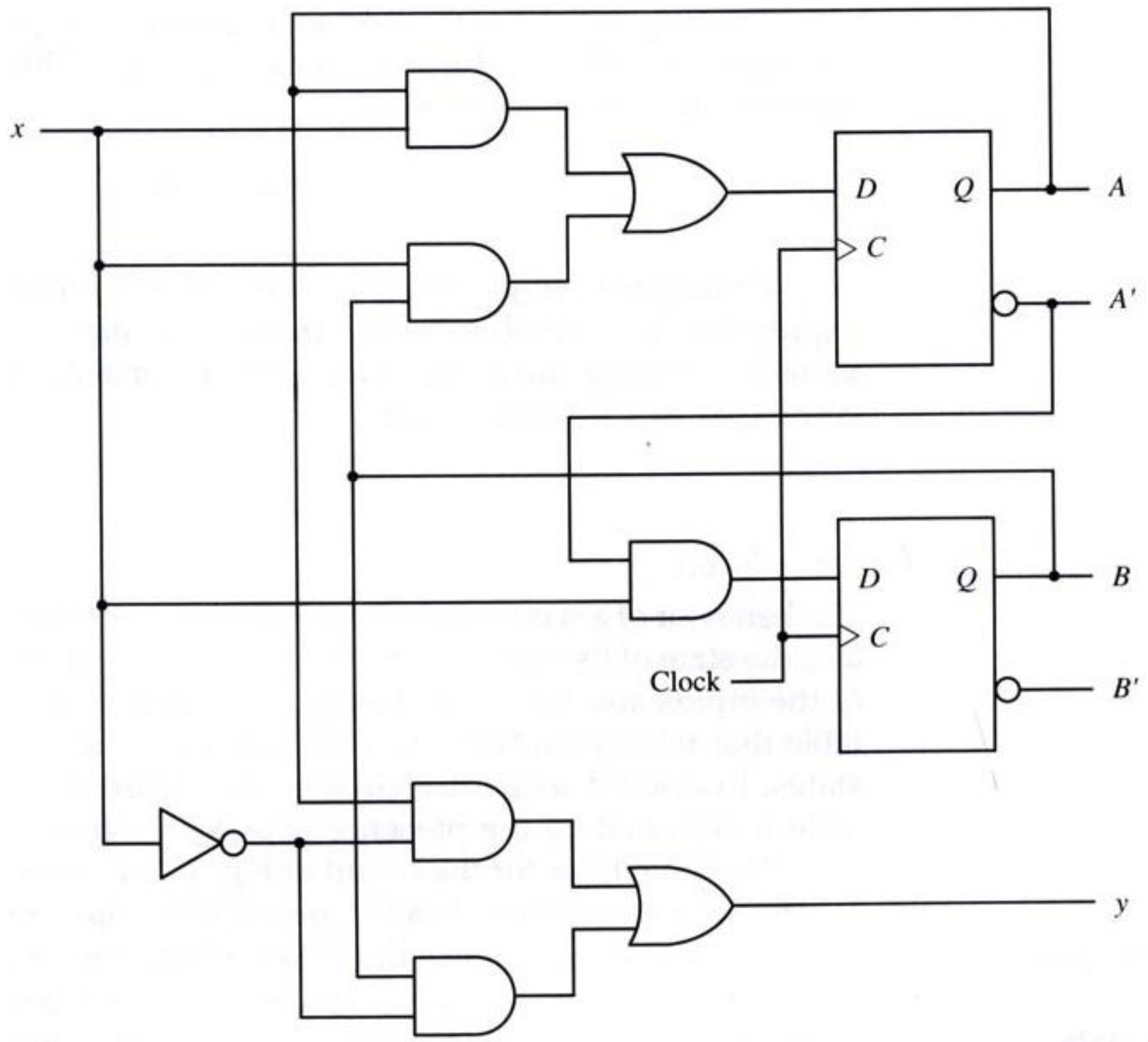
1-7. Sequential circuits



- Analysis
  - Drive Boolean functions for combinational circuits
    - Flip-flops’ inputs and outputs
  - Find state table
  - Obtain state diagram

1-7. Sequential circuit

- Analysis of sequential circuit

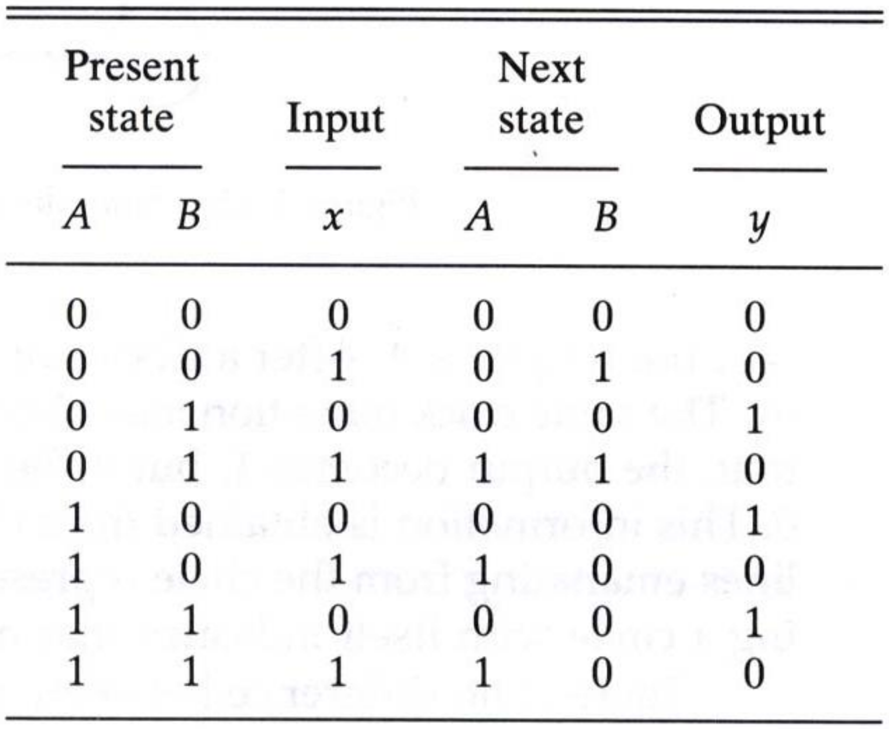


1-7. Sequential circuit

- Flip-flop input equations
  - $D_A = Ax + Bx$, $D_B = A'x$
- Combinational circuit output equation
  - $y = Ax' + Bx'$

1-7. Sequential circuit

- State table



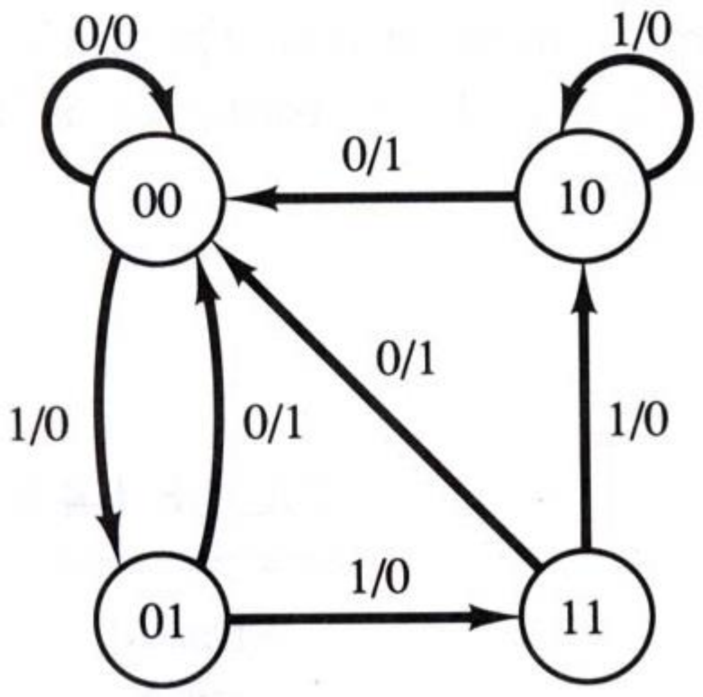
Present state		Input	Next state		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

1-7. Sequential circuit

- State table

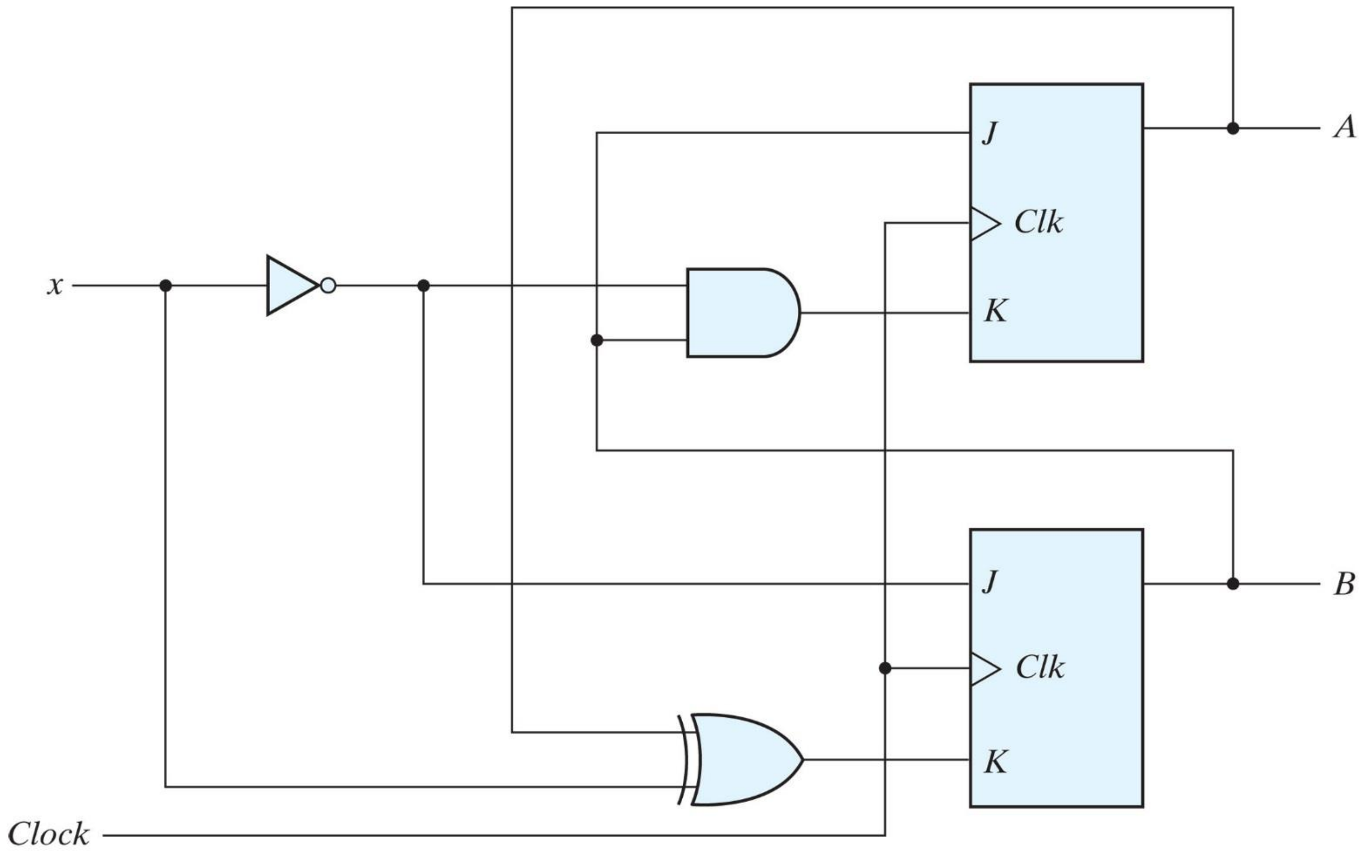
Present state		Input	Next state		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

- State diagram



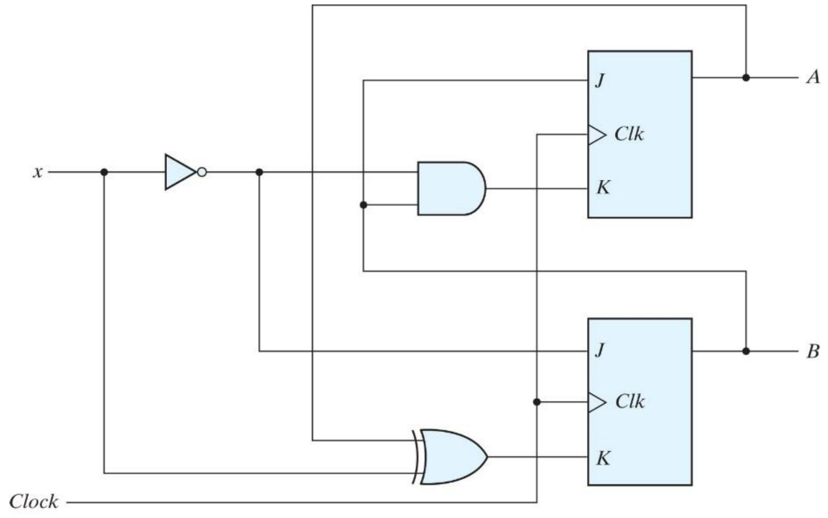
Ex) Analysis with JK flip-flops

- State equation is not the same as the input equation



Analysis with JK flip-flops

- State equation is not the same as the input equation
- Have to refer characteristic table or characteristic equation
- Input equations
  - $J_A = B \quad K_A = Bx'$
  - $J_B = x' \quad K_B = A'x + Ax'$



Analysis with JK flip-flops

- State table and state diagram

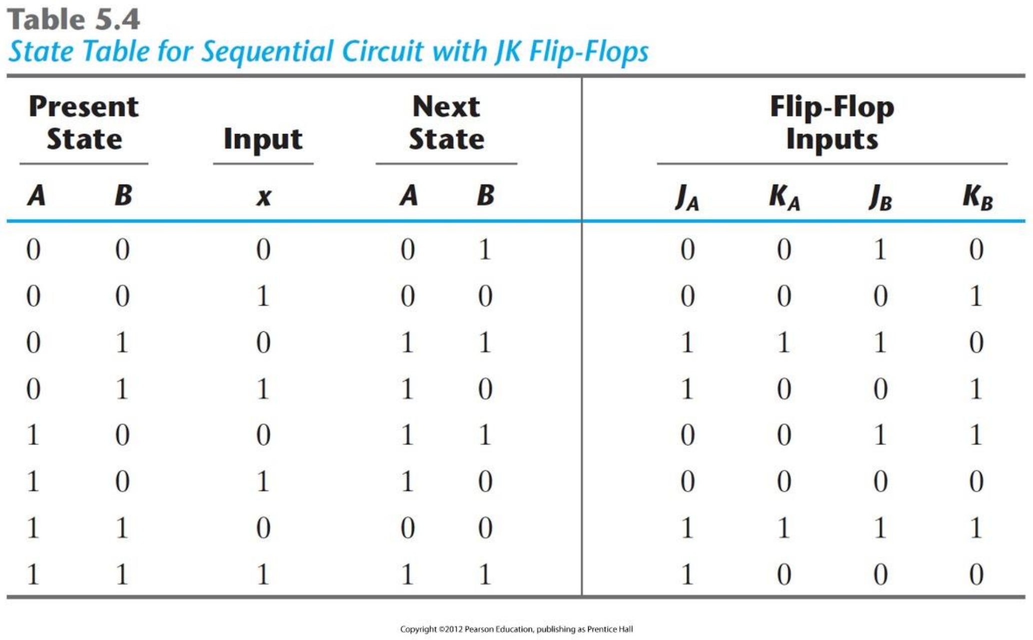


Table 5.4
State Table for Sequential Circuit with JK Flip-Flops

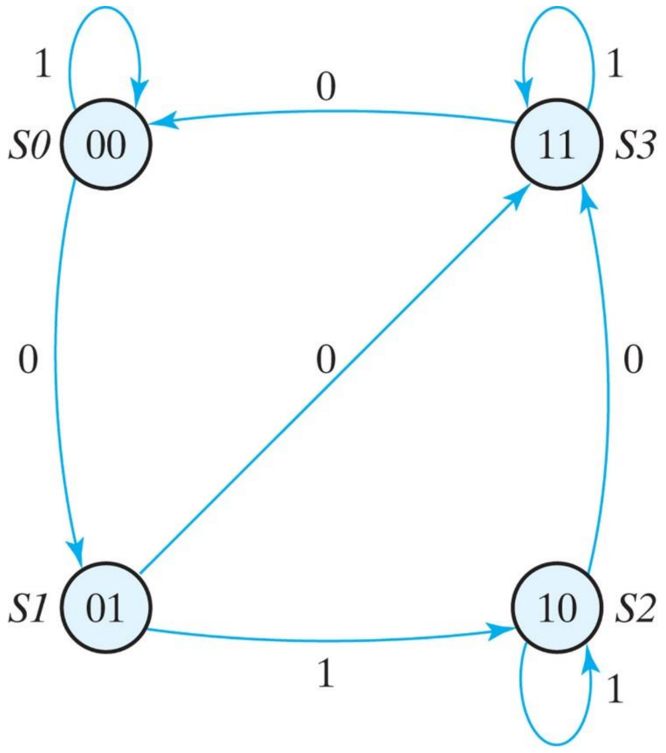
Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

Analysis with JK flip-flops

- State table and state diagram

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

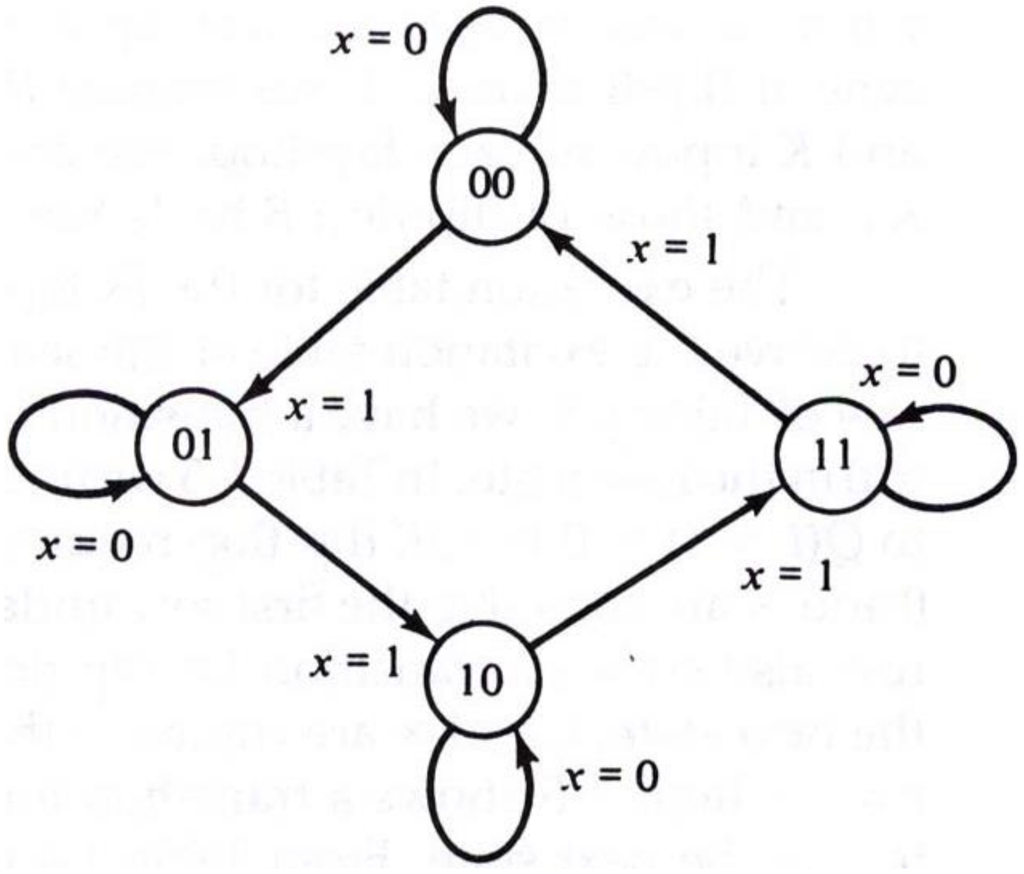


1-7. Sequential circuit

- Design example
  - Binary counter
- Design procedure
  - (1) translate circuit specification -> (2) state diagram - > (3) state table -> (4) logic circuit diagram
- Binary counter
  - Sequence of repeated binary states 00, 01, 10, 11 when external input x = 1
  - State remains unchanged when x = 0
  - Need two flip-flops representing two bits

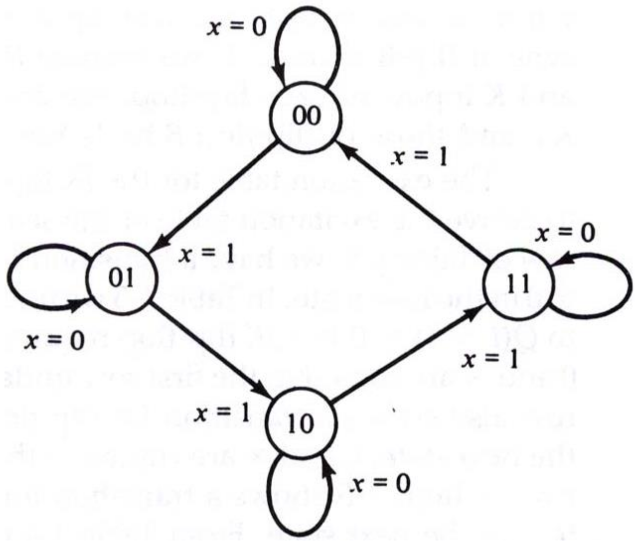
1-7. Sequential circuit

- State diagram



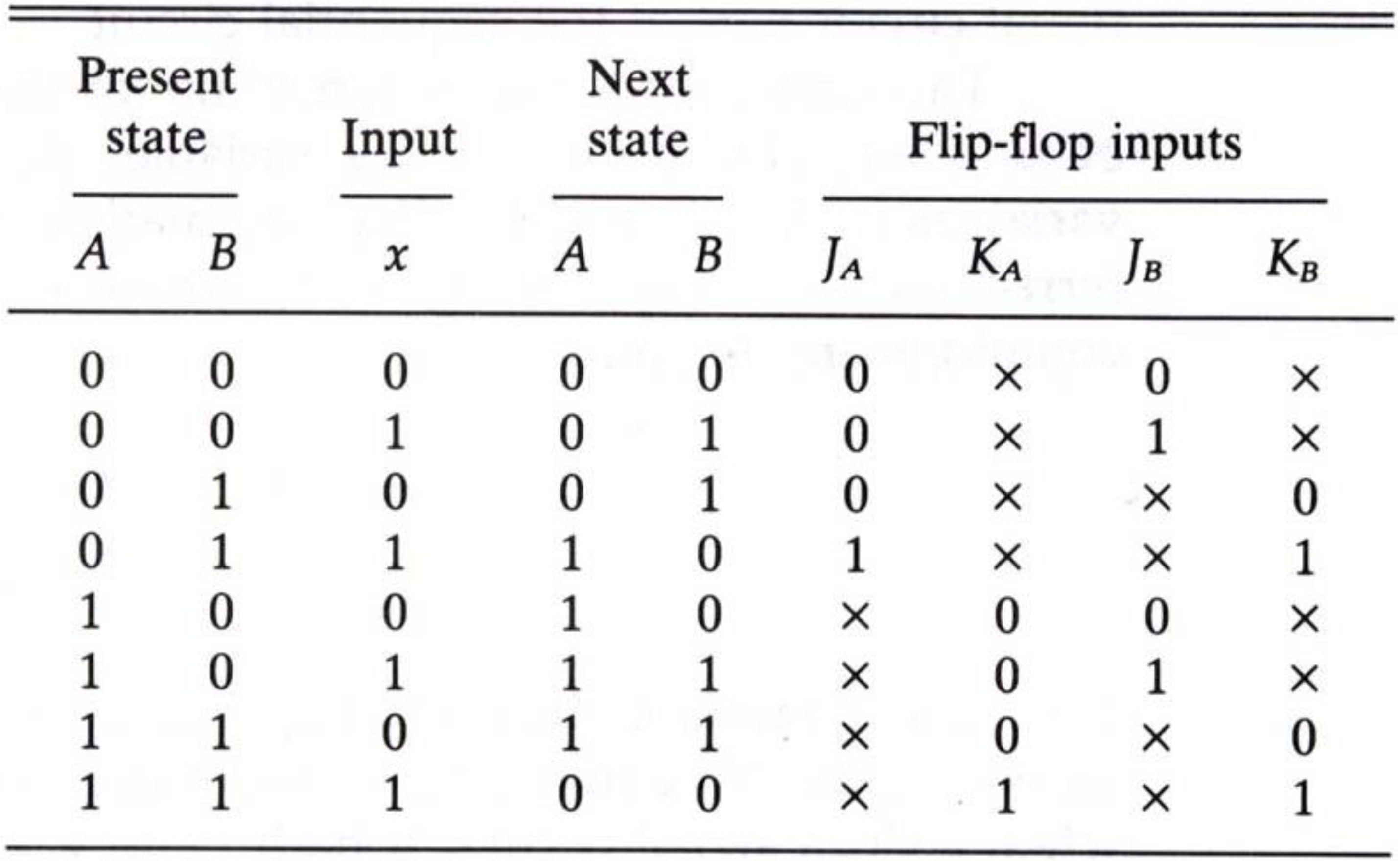
1-7. Sequential circuit

- State diagram



1-7. Sequential circuit

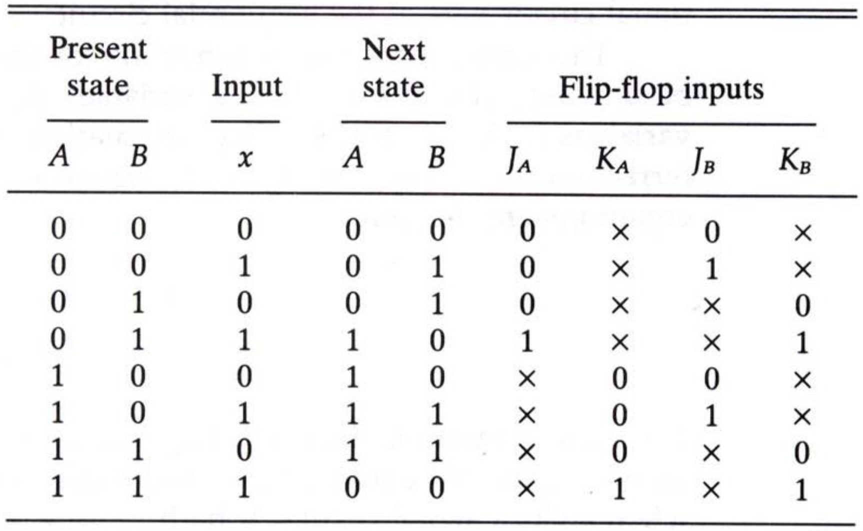
- State table



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	×	0	×
0	0	1	0	1	0	×	1	×
0	1	0	0	1	0	×	×	0
0	1	1	1	0	1	×	×	1
1	0	0	1	0	×	0	0	×
1	0	1	1	1	×	0	1	×
1	1	0	1	1	×	0	×	0
1	1	1	0	0	×	1	×	1

1-7. Sequential circuit

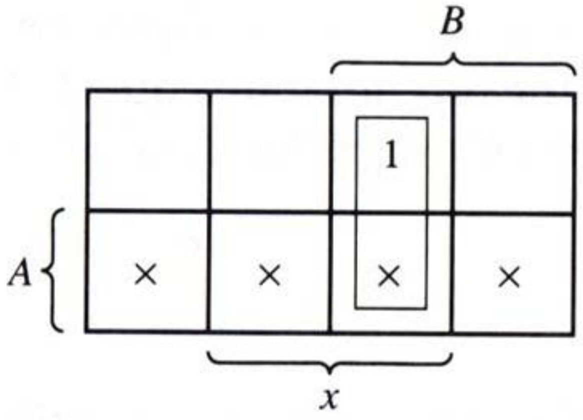
- State table



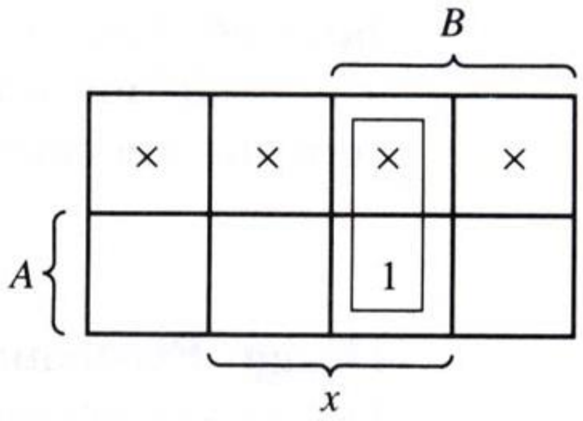
Present state		Input	Next state		Flip-flop inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	×	1	×
0	1	0	0	1	0	×	×	0
0	1	1	1	0	1	×	×	1
1	0	0	1	0	×	0	0	×
1	0	1	1	1	×	0	1	×
1	1	0	1	1	×	0	×	0
1	1	1	0	0	×	1	×	1

1-7. Sequential circuit

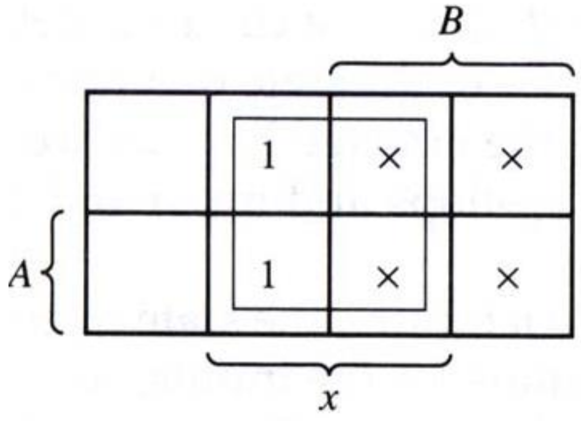
- K-maps



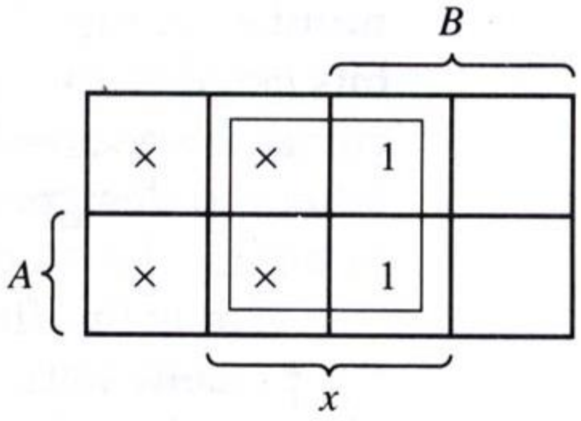
$J_A = Bx$



$K_A = Bx$



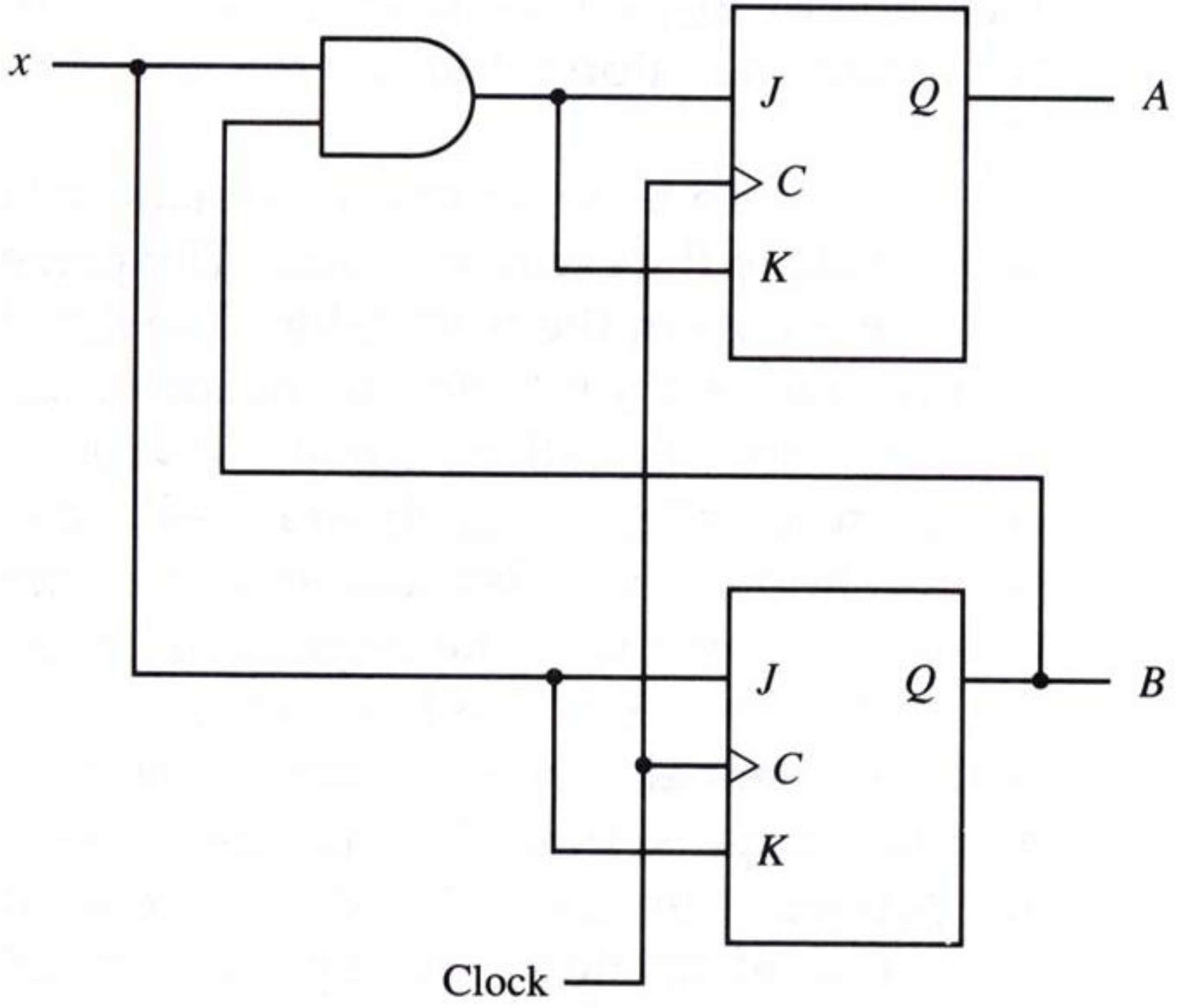
$J_B = x$



$K_B = x$

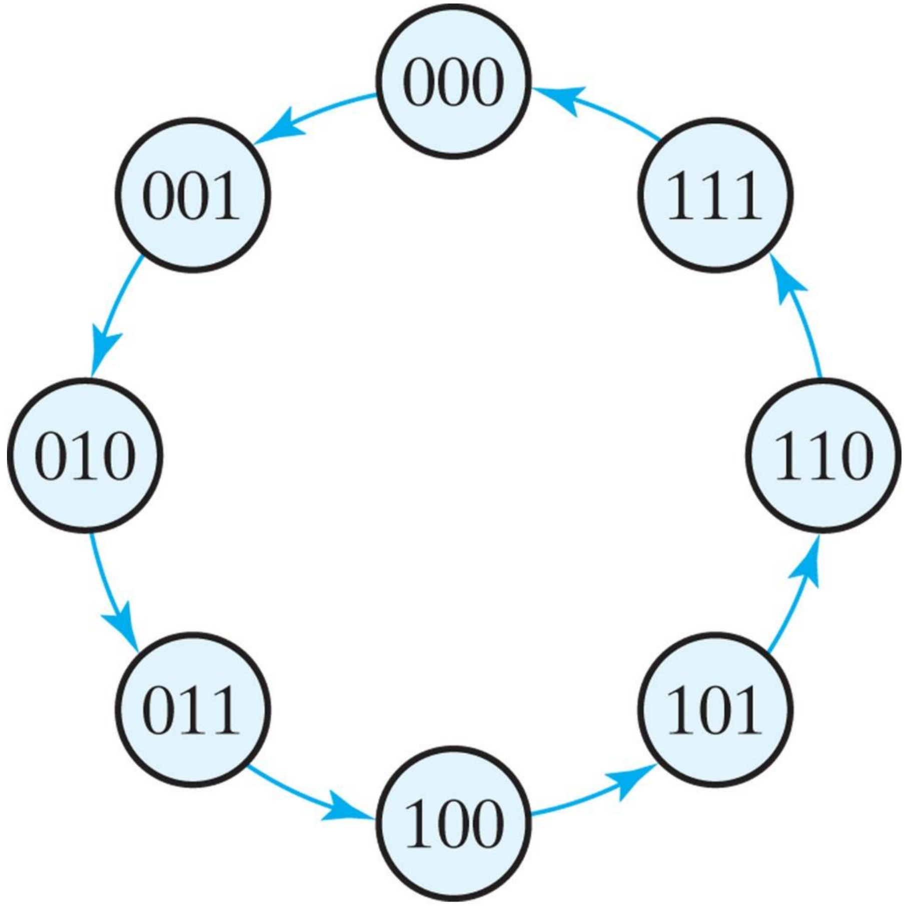
1-7. Sequential circuit

- Logic diagram



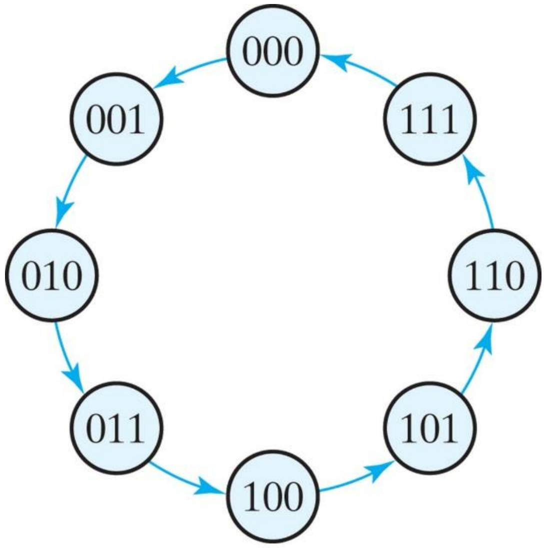
Ex) Synthesis using T flip-flops

- 3-bit binary counter
  - 3-bit counter has 3 flip-flops and can count from 0 to 2^n-1 (n=3)



Synthesis using T flip-flops

- 3-bit binary counter
  - 3-bit counter has 3 flip-flops and can count from 0 to 2^n-1 (n=3)



Synthesis using T flip-flops

- 3-bit binary counter
  - 3-bit counter has 3 flip-flops and can count from 0 to 2^n-1 (n=3)

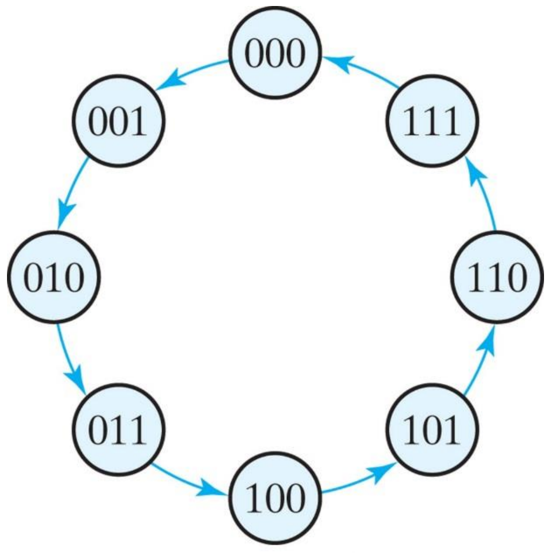


Table 5.14
State Table for Three-Bit Counter

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	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

- State table

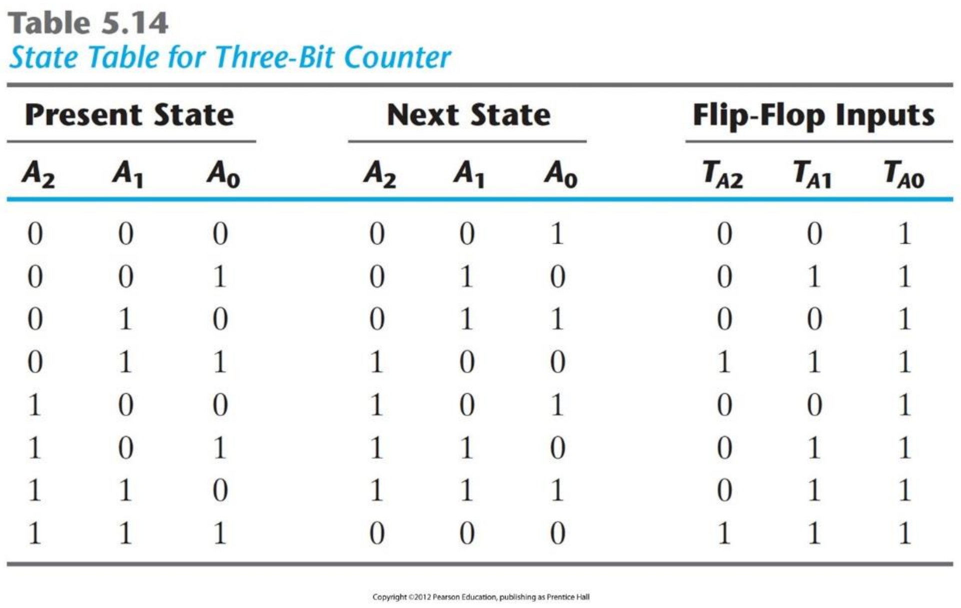
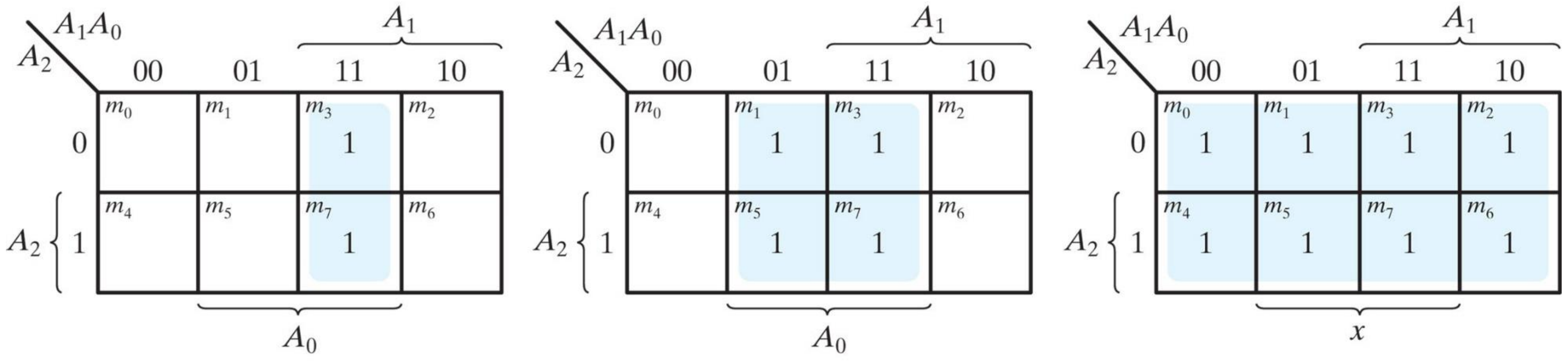


Table 5.14
State Table for Three-Bit Counter

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	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

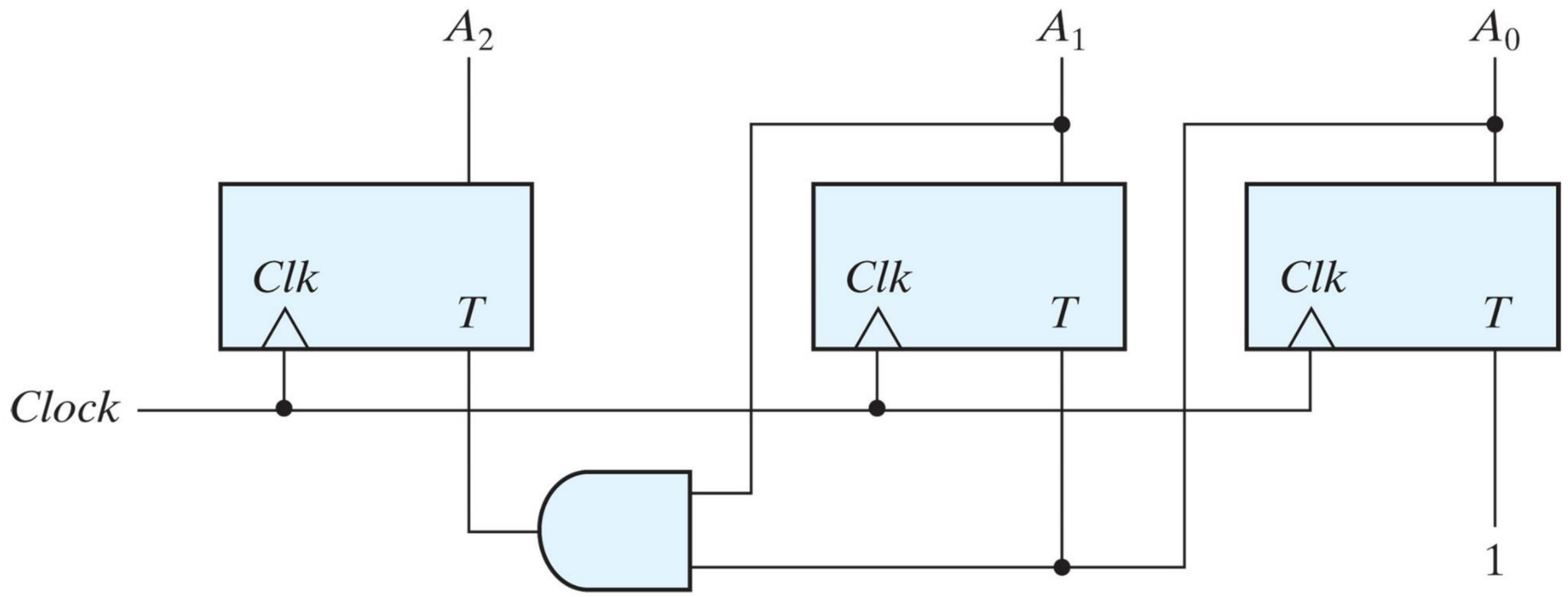
- K-maps



$T_{A2} = A_1A_0$ $T_{A1} = A_0$ $T_{A0} = 1$

Synthesis using T flip-flops

- Logic diagram



Problems

- 1-2, 1-4, 1-7, 1-8 a, b, 1-9 a, b
- 1-10 a, 1-13, 1-16, 1-19, 1-20