Digital Fundamentals

Thomas L. Floyd

Logic Gates /
Boolean Algebra and Logic Simplification
Chapter 4

Boolean Variables

variable : 0 or 1.

complement : inverse of a variable

complement of A is $\overline{A}$.

literal : variable or its complement

Boolean Addition

Addition (Sum) : OR operation

Sum = 0 iff all of literals are 0

Sum = 1 iff any of literal is 1

Determine *A*, *B*, and *C* that satisfies $\overline{A} + B + \overline{C} = 0$

Each literal must equal 0; therefore $A = 1$, $B = 0$ and $C = 1$.

Boolean Multiplication

Multiplication : AND operation

Multiplication = 0 iff any of literal is 0

Multiplication = 1 iff all of literals are 1

Determine A, B and C if the product term of $\overline{A} \cdot \overline{B} \cdot C = 1$?

Each literal must = 1; therefore $A = 0$, $B = 0$ and $C = 1$.

Commutative Laws

The **commutative laws** apply to both addition and multiplication

In terms of the result, the order in which variables are ORed makes no difference.

$$\boxed{A + B = B + A}$$

For multiplication, the commutative law states:

In terms of the result, the order in which variables are ANDed makes no difference.

$$\boxed{AB = BA}$$

Associative Laws

The **associative laws** also apply to both addition and multiplication. For addition,

When ORing more than two variables, the result is the same regardless of the grouping of the variables.

$$\boxed{A + (B + C) = (A + B) + C}$$

For multiplication

When ANDing more than two variables, the result is the same regardless of the grouping of the variables.

$$\boxed{A(BC) = (AB)C}$$

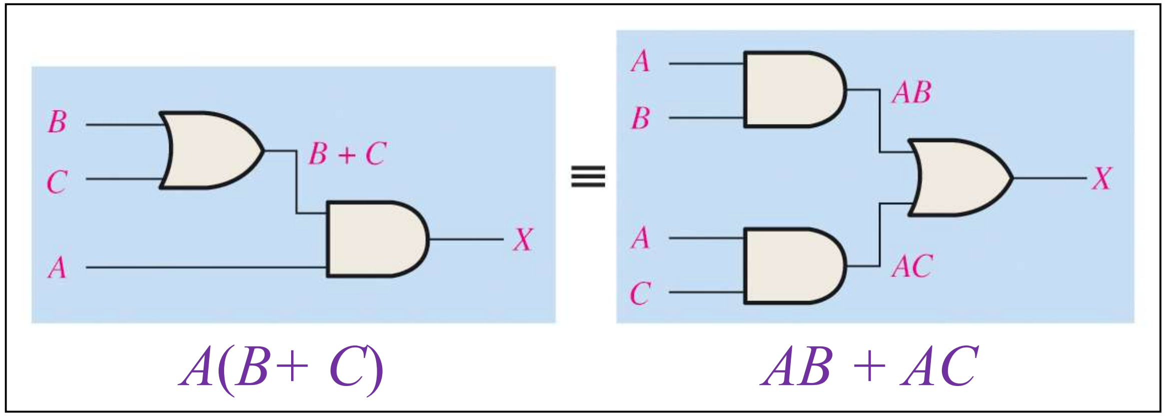
How about NAND?

Distributive Law

The **distributive law** is the *factoring law*

$$AB + AC = A(B + C)$$

The distributive law can be illustrated with equivalent circuits:



Rules of Boolean Algebra

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $A + \overline{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \overline{A} = 0$
9. $\overline{\overline{A}} = A$
10. $A + AB = A$
11. $A + \overline{A}B = A + B$
12. $(A + B)(A + C) = A + BC$

Rules of Boolean Algebra

Rule 1: $A + 0 = A$

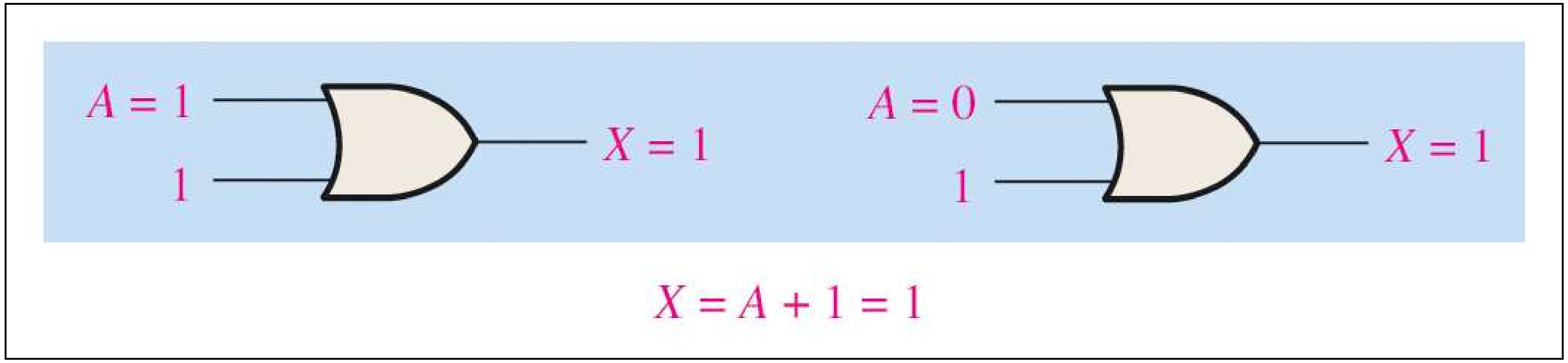
When $A = 1$, the input causes the output to go to $X = 1$.
When $A = 0$, the 0 inputs cause the output to go to $X = 0$.
In either case, the value of X equals the value of A.

$X = A + 0 = A$

Rules of Boolean Algebra

Rule 2: $A + 1 = 1$

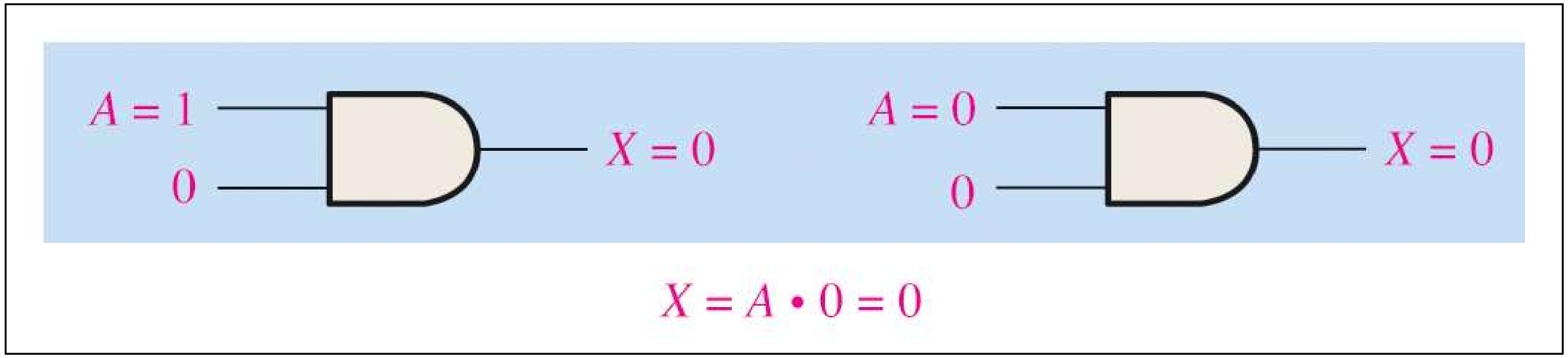
When $A = 1$, the inputs cause the output to go to $X = 1$.
When $A = 0$, the 1 input caused the output to go to $X = 1$.
In either case, the value of X equals one (1).



Rules of Boolean Algebra

Rule 3: $A \cdot 0 = 0$

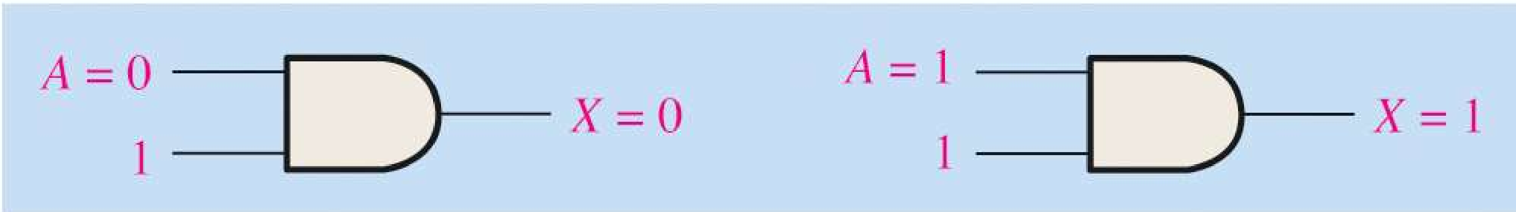
When either input to an AND gate equals 0, the output from the gate has a value of $X = 0$, regardless of the value at the other input.



Rules of Boolean Algebra

Rule 4: $A \cdot 1 = A$

When one input to an AND gate equals 1, the output from the gate has a value of $X = A$. As shown, $X = 1$ when $A = 1$ and $X = 0$ when $A = 0$.

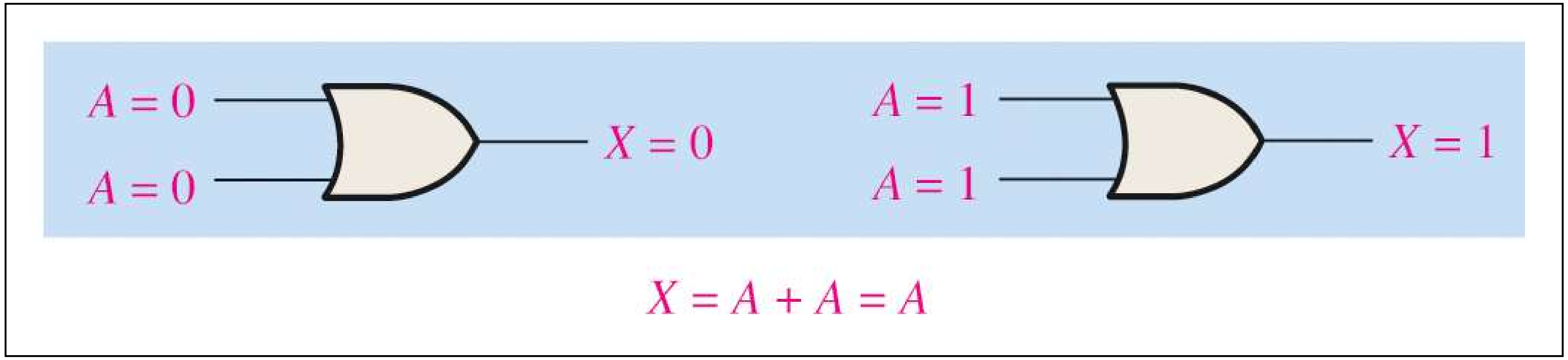


$X = A \cdot 1 = A$

Rules of Boolean Algebra

Rule 5: $A + A = A$

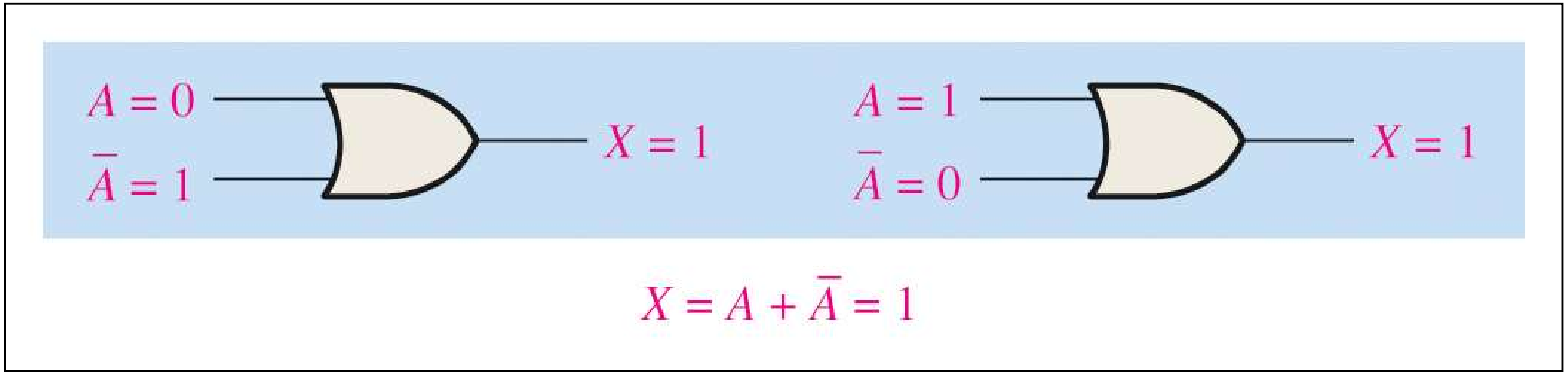
When the inputs to an OR gate are equal, the output equals the value at the inputs. When both inputs equal 1, the gate output is $X = 1$. When both inputs equal 0, the gate output is $X = 0$.



Rules of Boolean Algebra

Rule 6: $A + \overline{A} = 1$

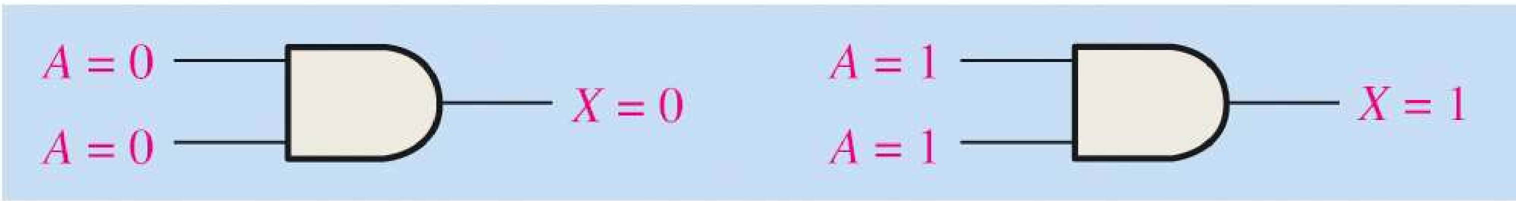
When the inputs to an OR gate are unequal (complements), one of the two always equals 1. When either input equals 1, the gate output is $X = 1$. Therefore, the output from the OR gate equals 1 whenever the inputs are unequal (complementary).



Rules of Boolean Algebra

Rule 7: $\boldsymbol{A \cdot A = A}$

When the inputs to an AND gate are equal, the gate output also equals that value. Thus, $X = 1$ when both inputs equal 1 and $X = 0$ when both inputs equal 0.

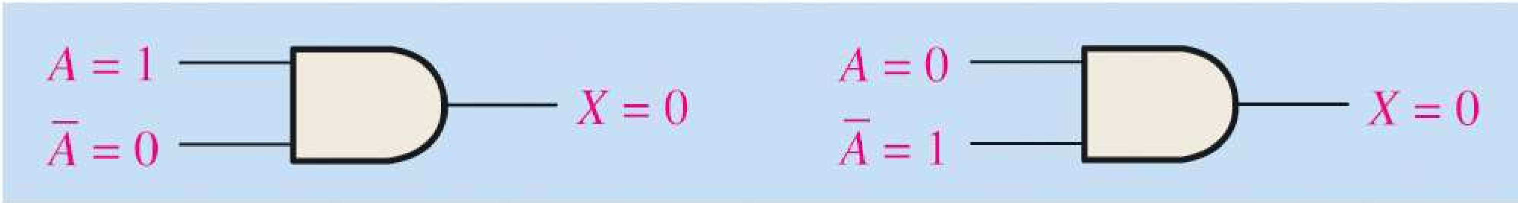


$X = A \cdot A = A$

Rules of Boolean Algebra

Rule 8: $A \cdot \overline{A} = 0$

When the inputs to an AND gate are unequal (complements), one of the two always equals 0. When either input equals 0, the gate output is $X = 0$. Therefore, the output from the AND gate equals 0 whenever the inputs are unequal (complementary).

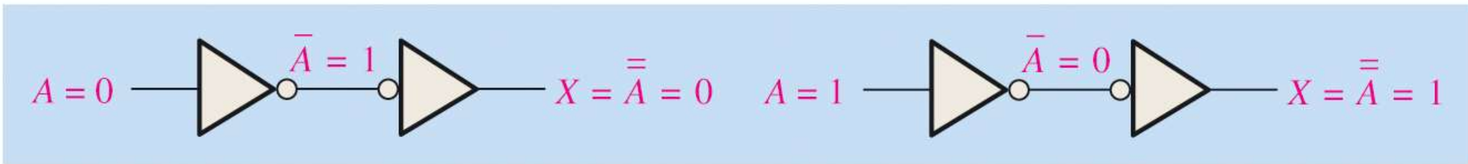


$X = A \cdot \overline{A} = 0$

Rules of Boolean Algebra

Rule 9: $\overline{\overline{A}} = A$

When a value is inverted, it is the complement of the original value. When inverted a second time, it returns to its original value. Thus, A = 0 inverted twice equals 0 and A = 1 inverted twice equals 1.



$X = \overline{\overline{A}} = A$

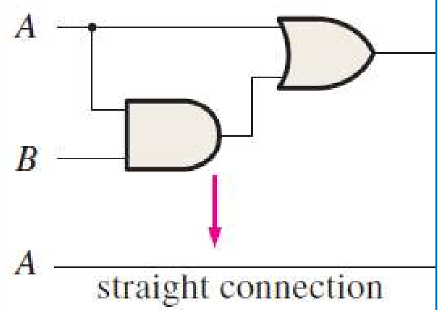
Rules of Boolean Algebra

Rule 10: $A + AB = A$

The circuit and truth table (below) can be used to demonstrate this rule. The truth table shows the outputs from the circuit for every possible combination of A and B. In each case, the output from the OR gate equals the value of A. Thus, $A + AB$ always equals the value of A.

A	B	AB	$A + AB$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

equal



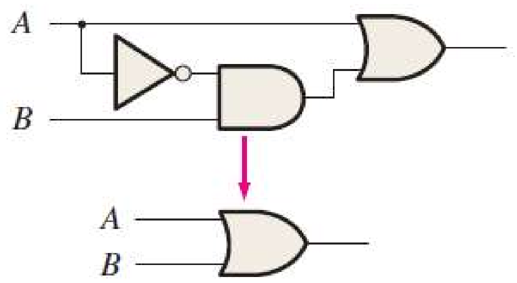
Rules of Boolean Algebra

Rule 11: $A + \overline{A}B = A + B$

The circuits and truth table (below) demonstrate this rule. The truth table shows the outputs from the two circuits are equal for every possible combination of *A* and *B*. As such, the two functions ($A + \overline{A}B$) and ($A + B$) are equal.

A	B	$\overline{A}B$	$A + \overline{A}B$	$A + B$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

equal



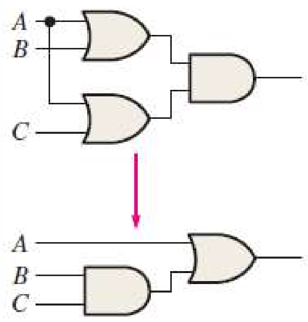
Rules of Boolean Algebra

Rule 12: $(A + B)(A + C) = A + BC$

The circuits and truth table (below) demonstrate this rule. The truth table shows the outputs from the two circuits are equal for every possible combination of *A* and *B*. As such, the two functions $(A + B)(A + C)$ and $(A + BC)$ are equal.

A	B	C	$A+B$	$A+C$	$(A+B)(A+C)$	BC	$A+BC$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

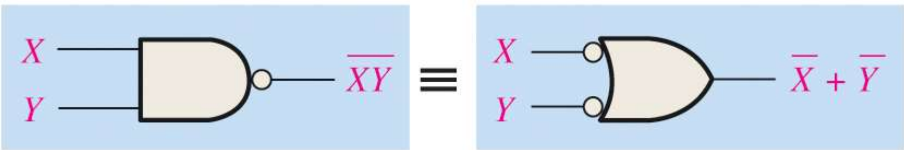
equal



DeMorgan’s Theorem

The complement of two or more ANDed variables equals the OR of the individual variable complements.

$$\overline{XY} = \overline{X} + \overline{Y}$$



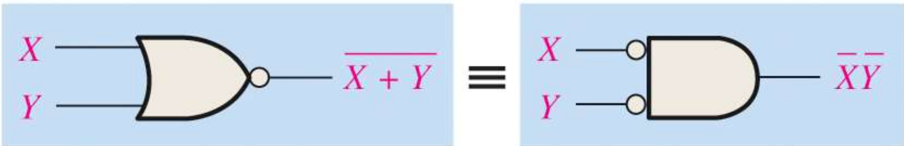
NAND   Negative-OR

INPUTS		OUTPUT	
X	Y	$\overline{XY}$	$\overline{X} + \overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

DeMorgan’s Theorem

The complement of two or more ORed variables equals the AND of the individual variable complements.

$$\overline{X+Y}=\overline{X}\cdot\overline{Y}$$



NOR   Negative-AND

INPUTS		OUTPUT	
X	Y	$\overline{X+Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

DeMorgan's Theorem

The complement of two or more ORed variables equals the AND of the individual variable complements.

$$\boxed{\overline{X+Y+Z} = \overline{X}\cdot\overline{Y}\cdot\overline{Z}}$$

$$\boxed{\overline{X\cdot Y\cdot Z} = \overline{X}+\overline{Y}+\overline{Z}}$$

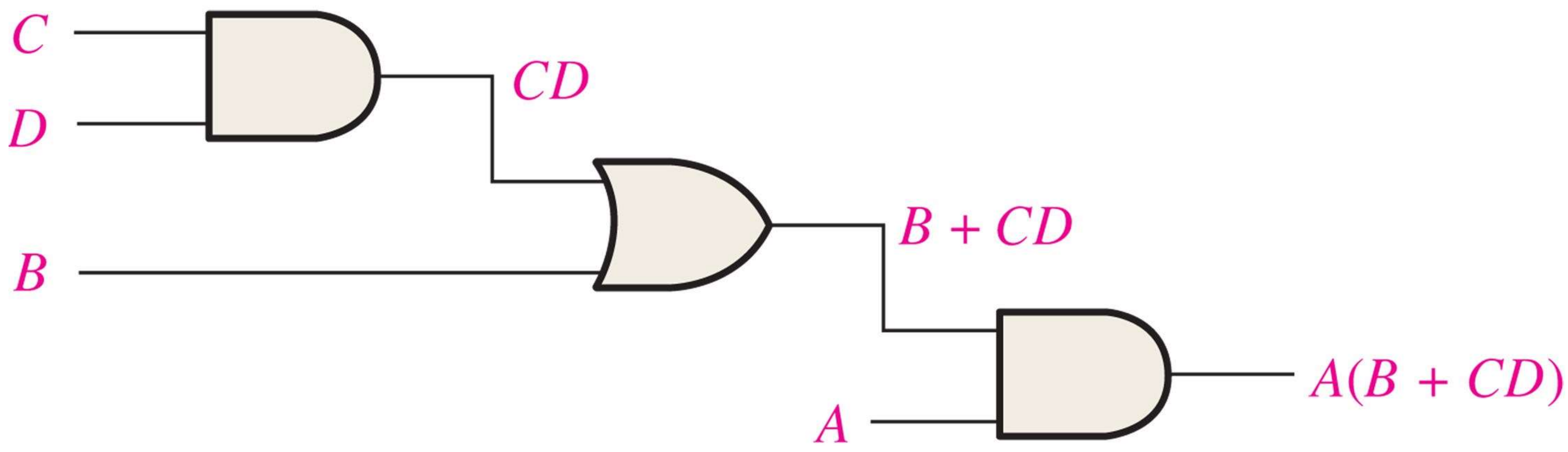
DeMorgan's Theorem: Application

$$\overline{\overline{A+B\overline{C}}+D(\overline{E+\overline{F}})}=?$$

$$\overline{(AB+C)(A+BC)}=\ ?$$

Boolean Expression for a Logic Circuit

Derive Boolean expression for a given combinatorial logic circuit.



Constructing a Truth Table

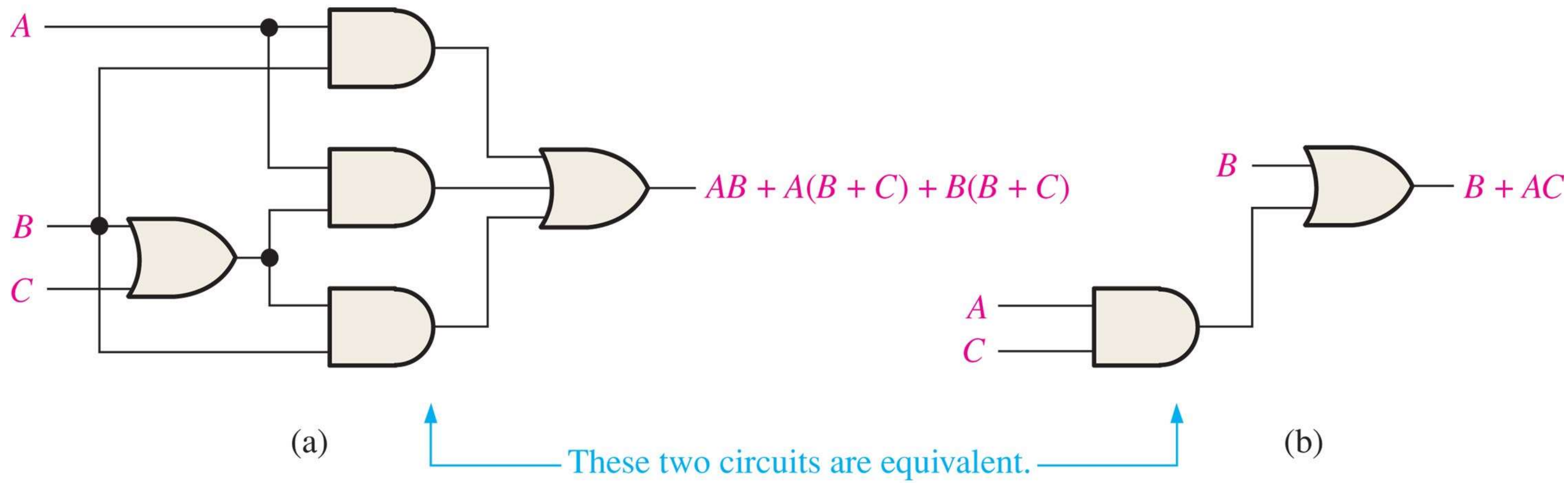
TABLE 4–5

Truth table for the logic circuit in Figure 4–18.

Inputs				Output
A	B	C	D	$A(B + CD)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Logic Simplification

$$AB + A(B+C) + B(B+C) = B + AC$$



Logic Simplification

$$(\overline{A}+B)C+ABC= ?$$

Sum-of-Products (SOP) Form

Two or more product terms are summed by Boolean addition

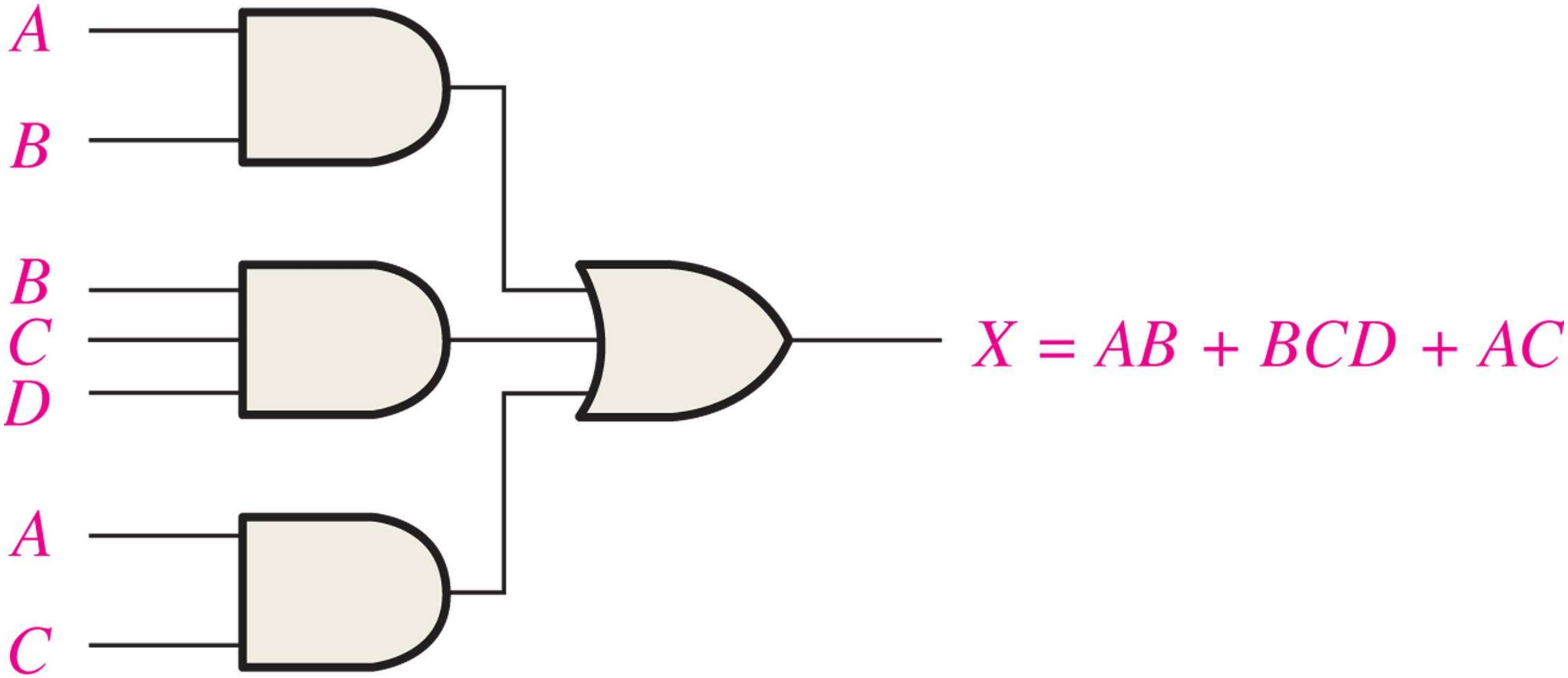
$$AB + BCD + AC \quad ✓$$

$$\overline{A}B + \overline{A}B\overline{C} + AC \quad ✓$$

$$\overline{A}B + \overline{ABC} + AC \quad ✗$$

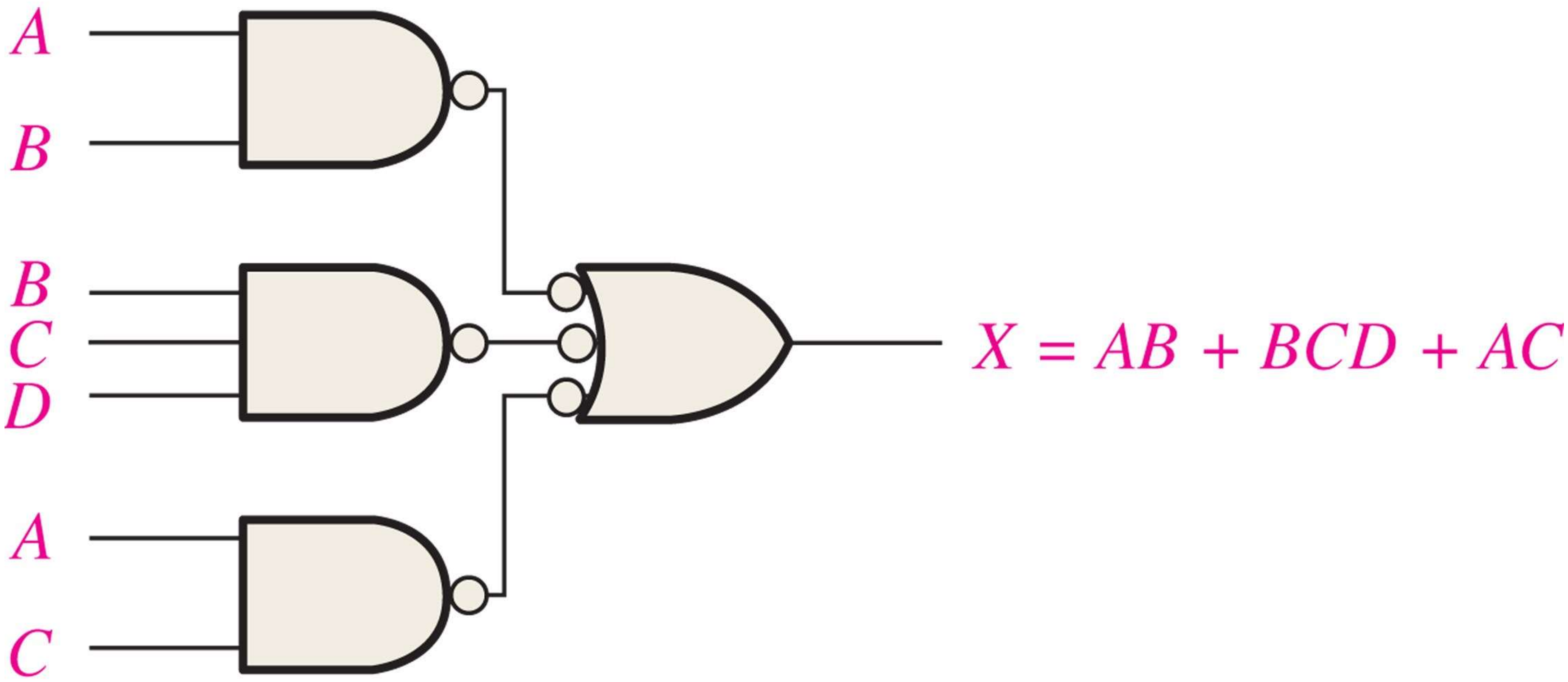
Sum-of-Products (SOP) Form

$$AB + BCD + AC$$



Sum-of-Products (SOP) Form

$$AB + BCD + AC$$



Standard SOP

All the variables in the domain appear in each product term in the expression (use $D + \overline{D} = 1$)

$$A\overline{B}C + \overline{A}\,\overline{B} + AB\overline{C}D$$

$$\downarrow$$

$$A\overline{B}CD + A\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD$$
$$+\overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}D$$

Binary Representation

SOP expression is equal to 1 only if one or more of the product terms in the expression is equal to 1

$$ABCD \rightarrow 1 \cdot 1 \cdot 1 \cdot 1$$

$$A\overline{B}\overline{C}D \rightarrow 1 \cdot \overline{0} \cdot \overline{0} \cdot 1$$

$$\overline{A}\overline{B}\overline{C}\overline{D} \rightarrow \overline{0} \cdot \overline{0} \cdot \overline{0} \cdot \overline{0}$$

SOP to Truth Table

TABLE 4–6

Inputs			Output	
A	B	C	X	Product Term
0	0	0	0	
0	0	1	1	$\overline{A}\,\overline{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

Product-of-Sums (POS) Form

Two or more product terms are summed by Boolean addition

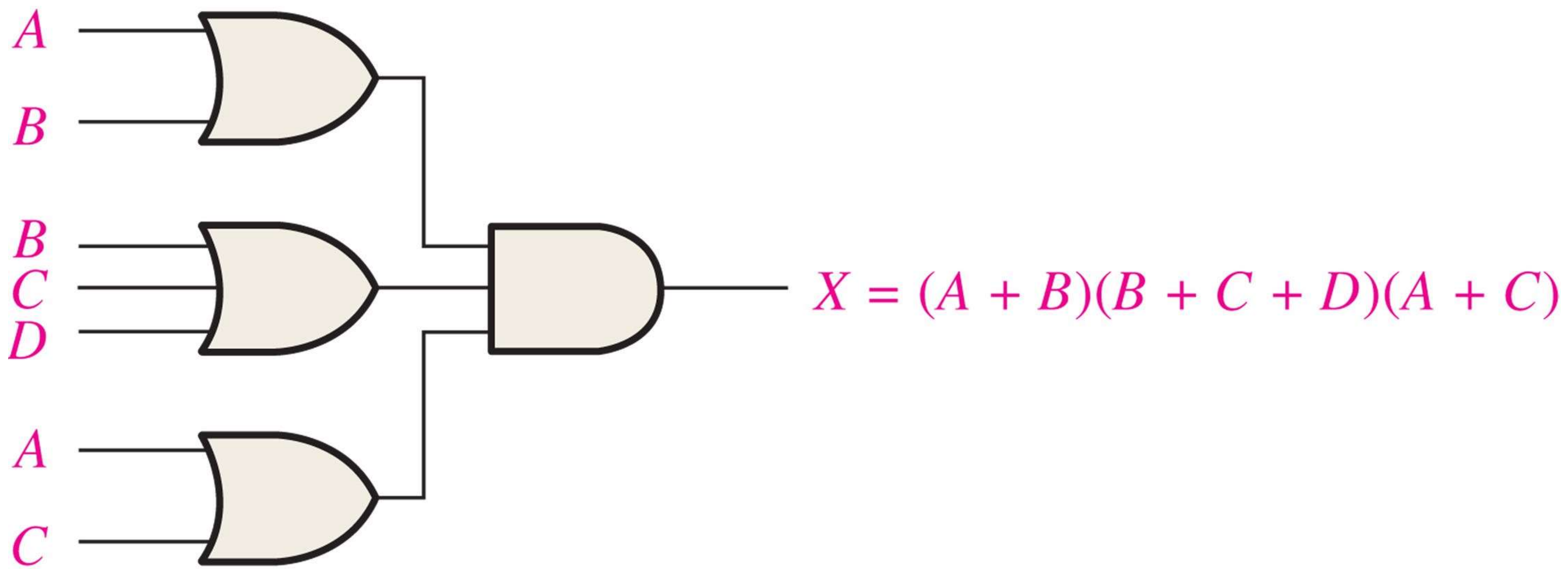
$$(\overline{A}+B)(A+\overline{B}+C) \quad ✓$$

$$(A+B)(A+\overline{B}+C)(\overline{A}+C) \quad ✓$$

$$(A+B)(\overline{A+B+C})(\overline{A}+C) \quad ✗$$

Product-of-Sums (POS) Form

$$(A+B)(B+C+D)(A+C)$$



Standard POS

All the variables in the domain appear in each sum term in the expression
(use $D\overline{D} = 0$ and $A + BC = (A + B)(A + C)$)

$$(A+\overline{B}+C)(A+B+\overline{D})$$

$$\downarrow$$

$$(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})$$

$$(A+B+C+\overline{D})(A+B+\overline{C}+\overline{D})$$

Binary Representation

POS expression is equal to 0 only if one or more of the sum terms in the expression is equal to 0

$$A + B + C + D \rightarrow 0 + 0 + 0 + 0$$

$$A + \overline{B} + \overline{C} + D \rightarrow 0 + \overline{1} + \overline{1} + 0$$

$$\overline{A} + \overline{B} + \overline{C} + \overline{D} \rightarrow \overline{1} + \overline{1} + \overline{1} + \overline{1}$$

POS to Truth Table

TABLE 4–7

Inputs			Output	
A	B	C	X	Sum Term
0	0	0	0	$(A + B + C)$
0	0	1	1	
0	1	0	0	$(A + \overline{B} + C)$
0	1	1	0	$(A + \overline{B} + \overline{C})$
1	0	0	1	
1	0	1	0	$(\overline{A} + B + \overline{C})$
1	1	0	0	$(\overline{A} + \overline{B} + C)$
1	1	1	1	

SOP to Truth Table

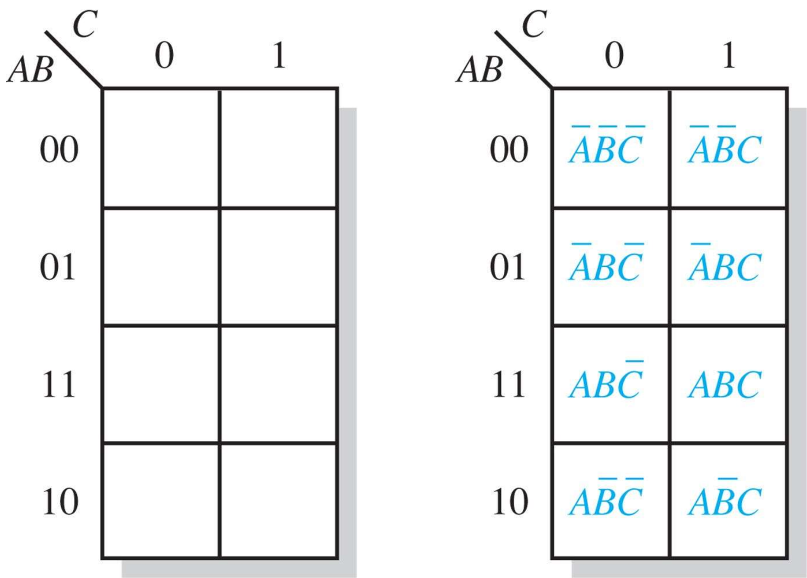
TABLE 4–6

Inputs			Output	
A	B	C	X	Product Term
0	0	0	0	
0	0	1	1	$\overline{A}\,\overline{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

Karnaugh Map

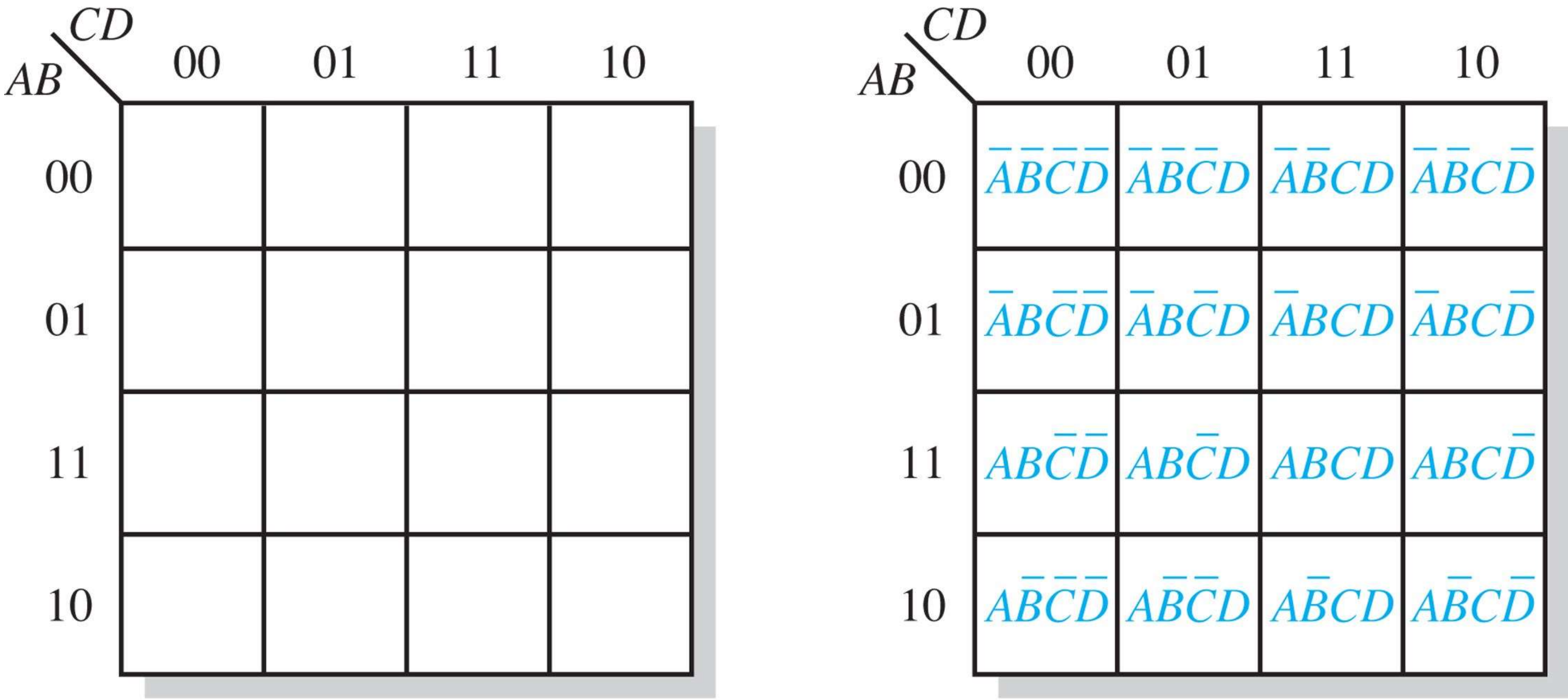
Karnough map provides a systematic method for simplifying Boolean expressions.

Karnough map is similar to a truth table.



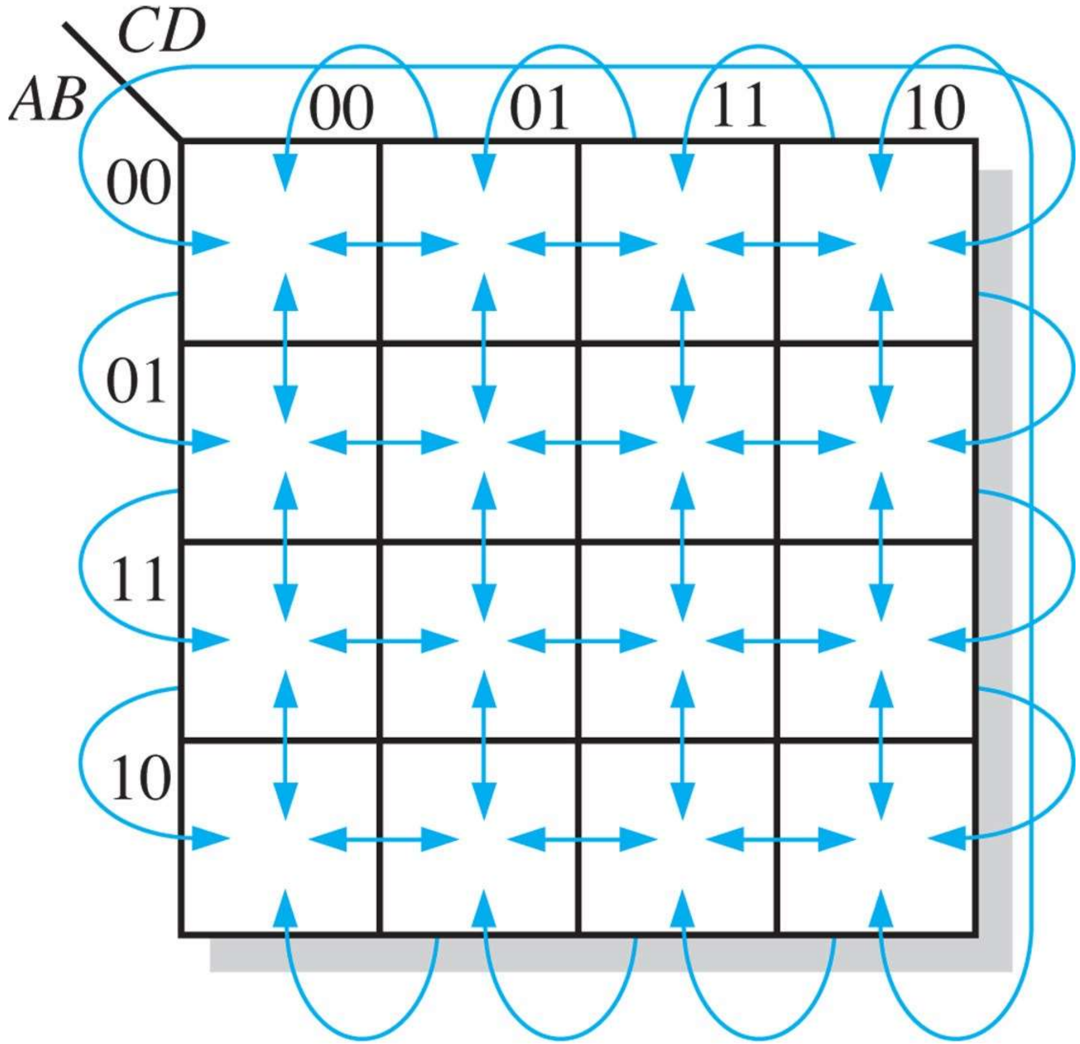
(a) (b)

4-Variable Karnaugh Map

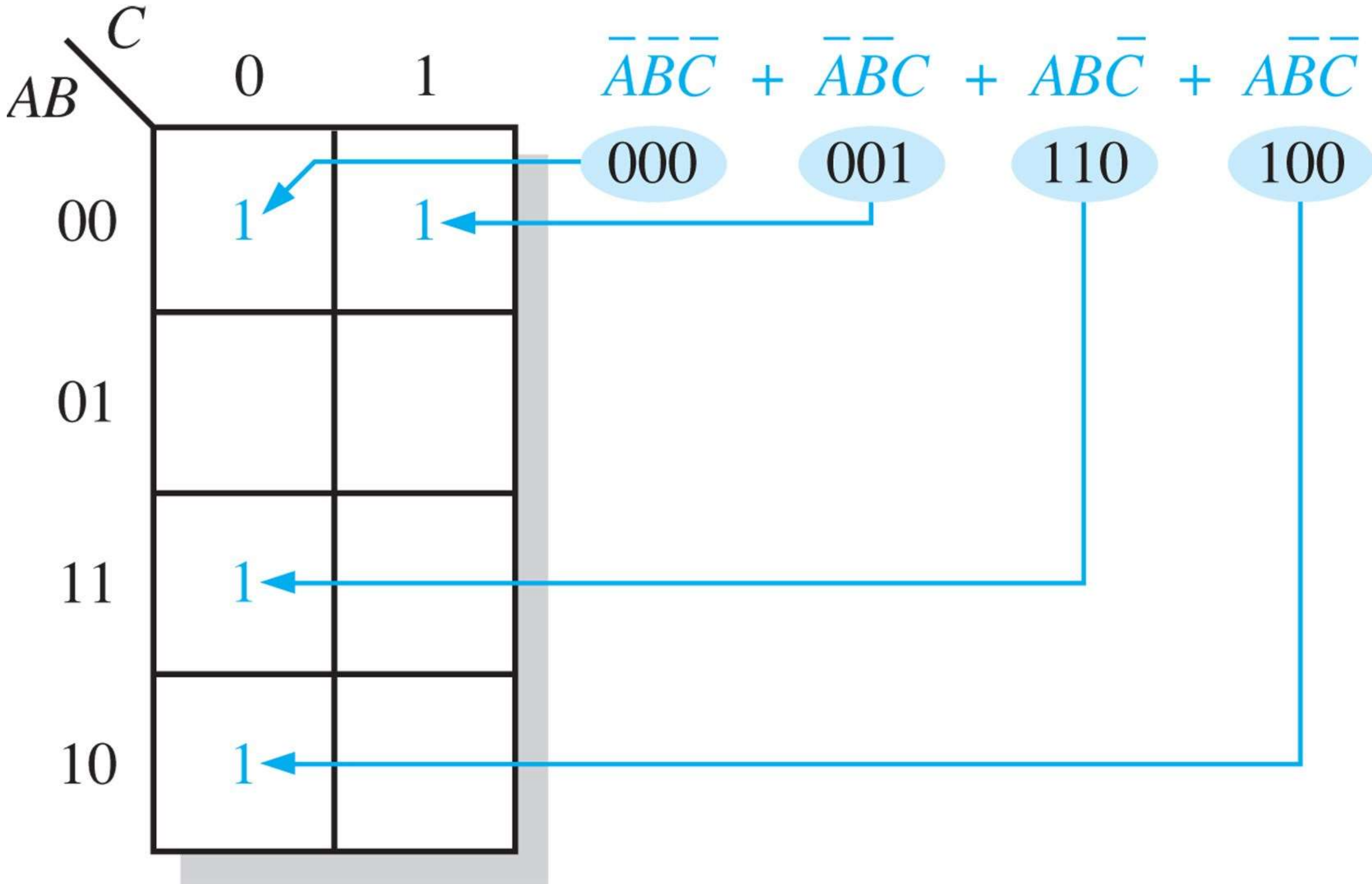


(a) (b)

4-Variable Karnaugh Map

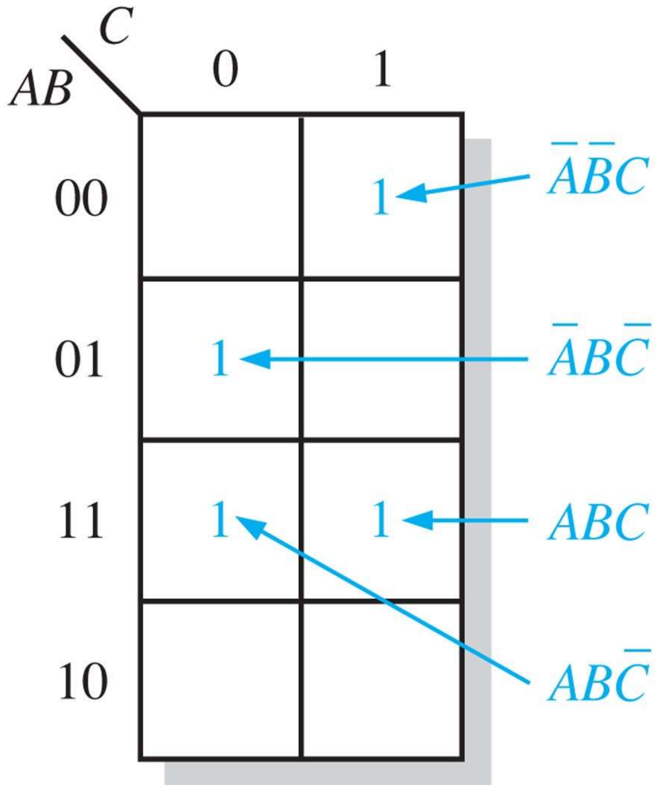


Mapping a Standard SOP



Mapping a Standard SOP

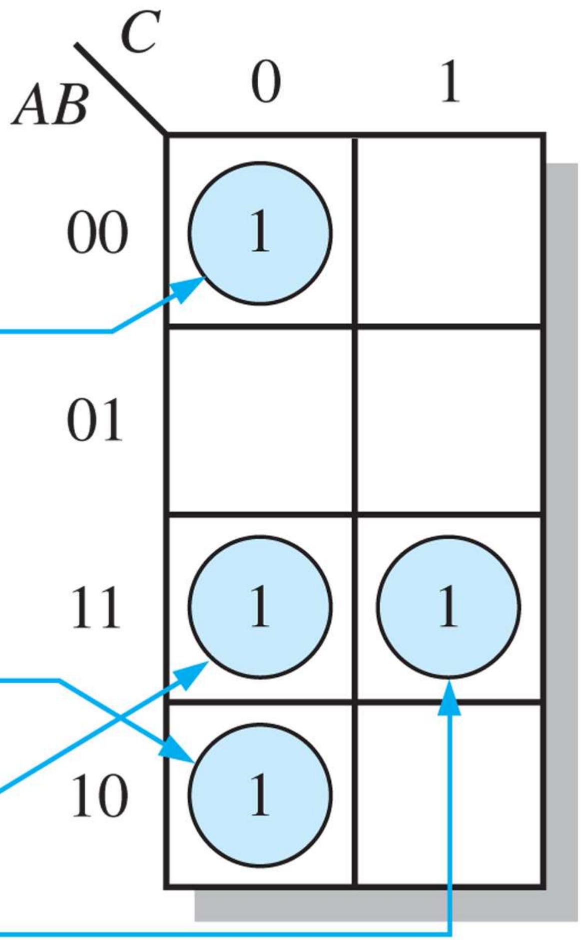
$$\overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + AB\,\overline{C} + ABC$$



Mapping from Truth Table

$$X = \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + AB\overline{C} + ABC$$

Inputs			Output
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



Mapping a SOP

$$\overline{A} + A\overline{B} + AB\overline{C}$$

AB \ C	0	1
00	1	1
01	1	1
11	1	
10	1	1

Mapping a SOP

$$\overline{B}\,\overline{C} + A\overline{B} + AB\overline{C} + A\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}CD$$

$AB \backslash CD$	00	01	11	10
00	1	1		
01				
11	1	1		
10	1	1	1	1

Karnough Map Simplification

AB \ C	0	1
00	1	
01		1
11	1	1
10		

(a)

AB \ C	0	1
00	1	1
01	1	
11		1
10	1	1

(b)

AB \ CD	00	01	11	10
00	1	1		
01	1	1	1	1
11				
10		1	1	

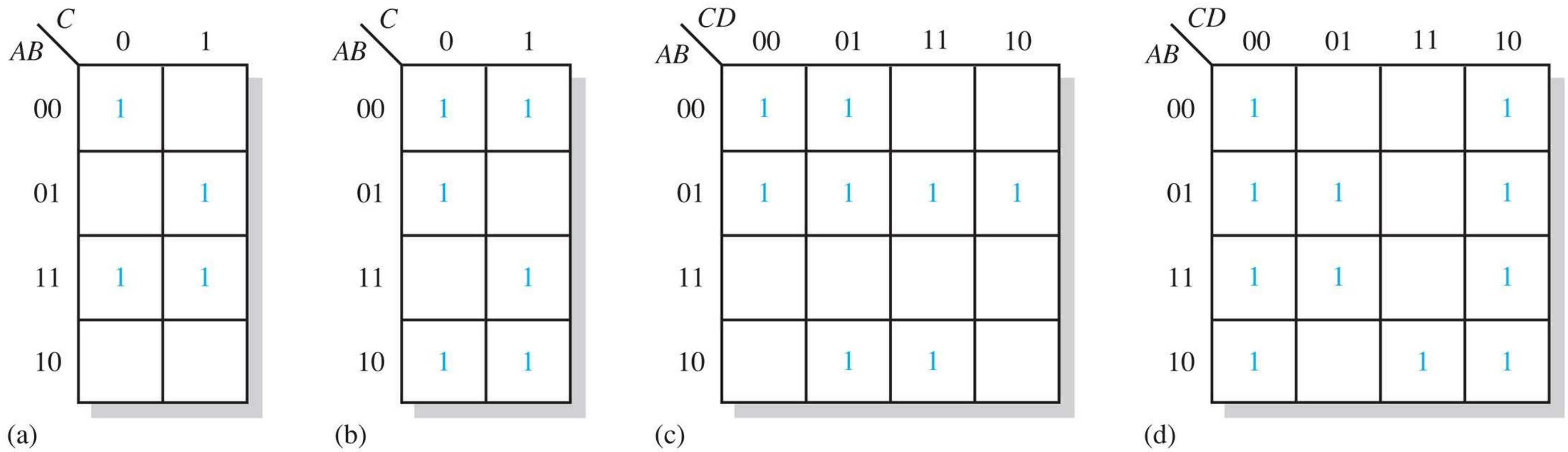
(c)

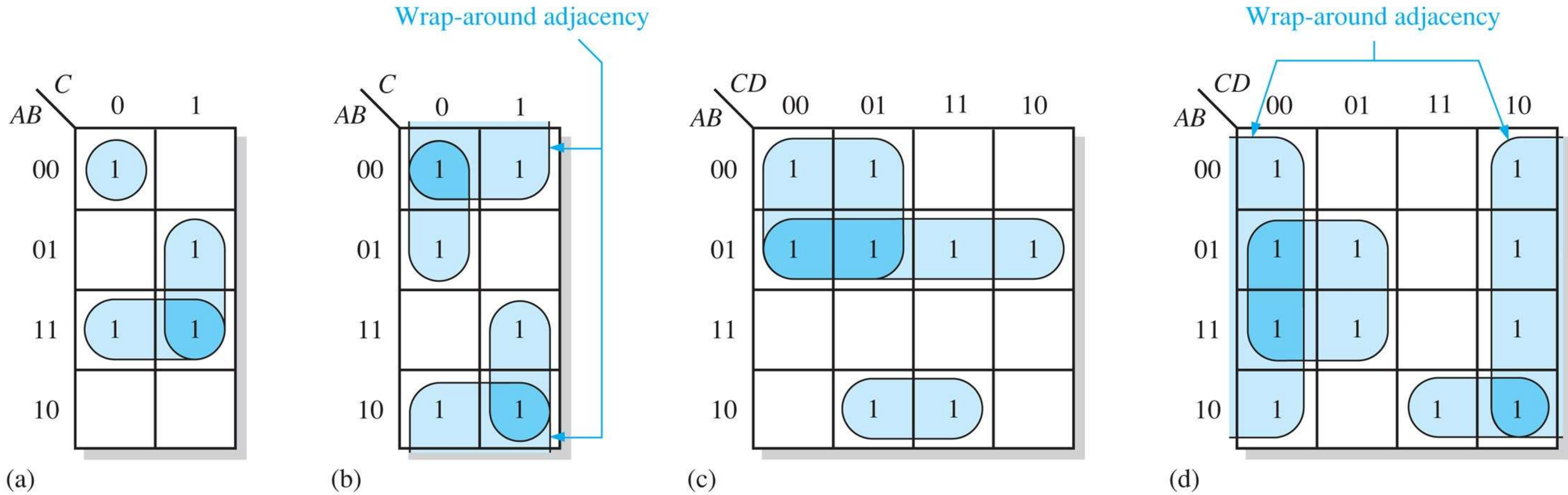
AB \ CD	00	01	11	10
00	1			1
01	1	1		1
11	1	1		1
10	1		1	1

(d)

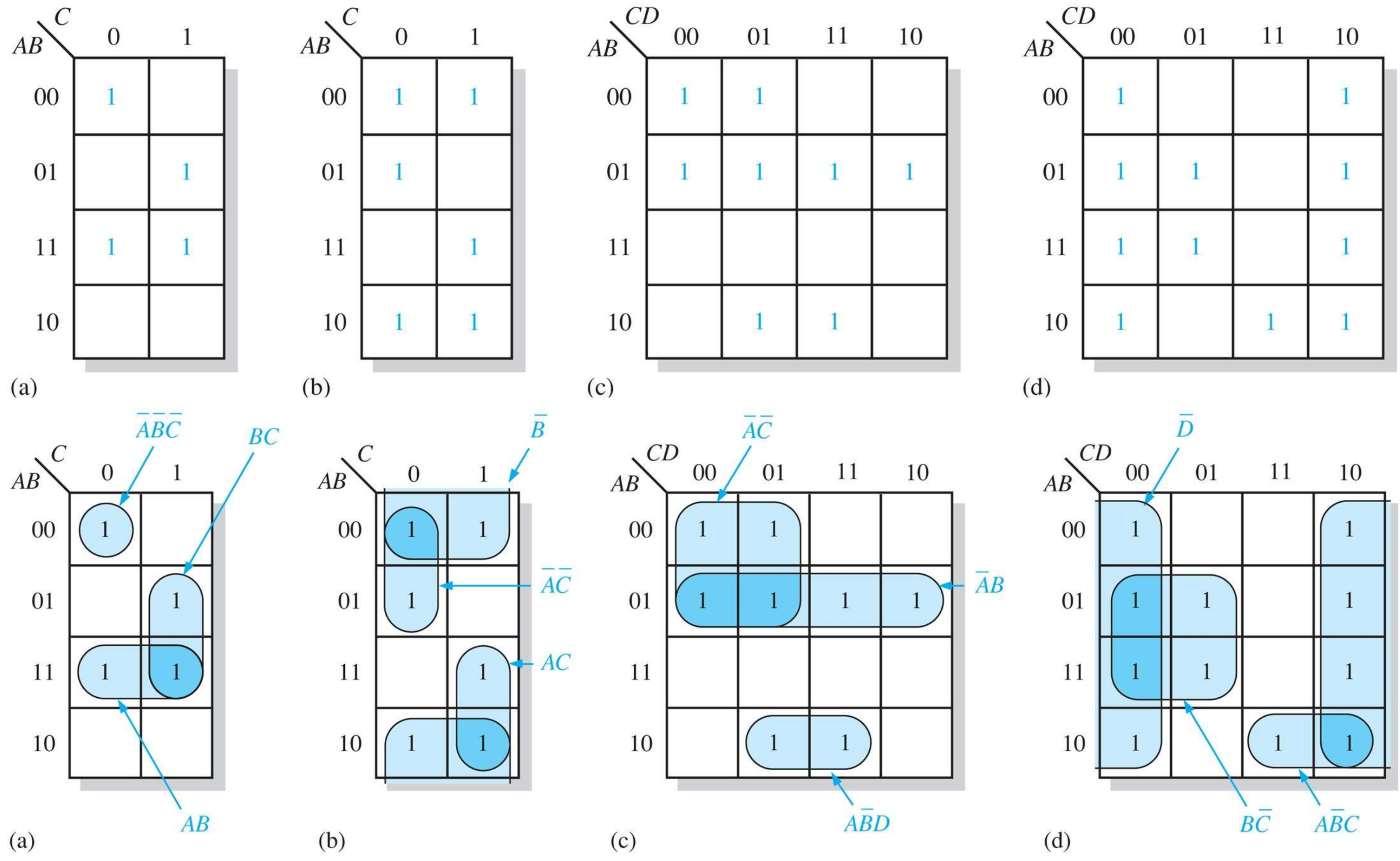
1. Group must contain 1, 2, 4, 8, 16 cells
2. Each cell in a group must be adjacent to each other
3. Always include the largest possible number of 1s

Karnough Map Simplification





Karnough Map Simplification

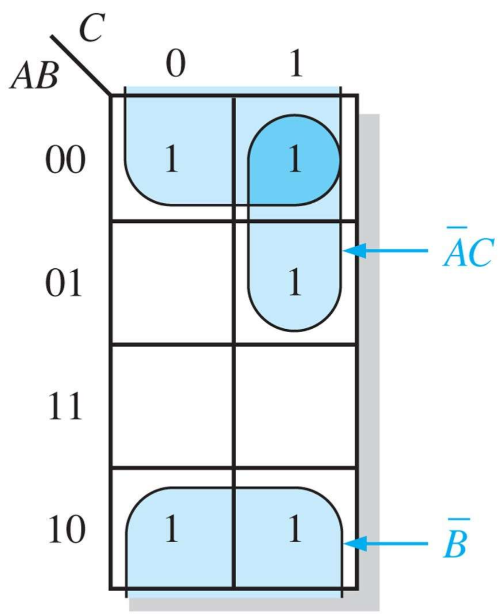


Karnough Map Example

$$A\overline{B}C+\overline{A}BC+\overline{A}\,\overline{B}C+\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}$$

$$\downarrow$$

$$\overline{B}+\overline{A}C$$



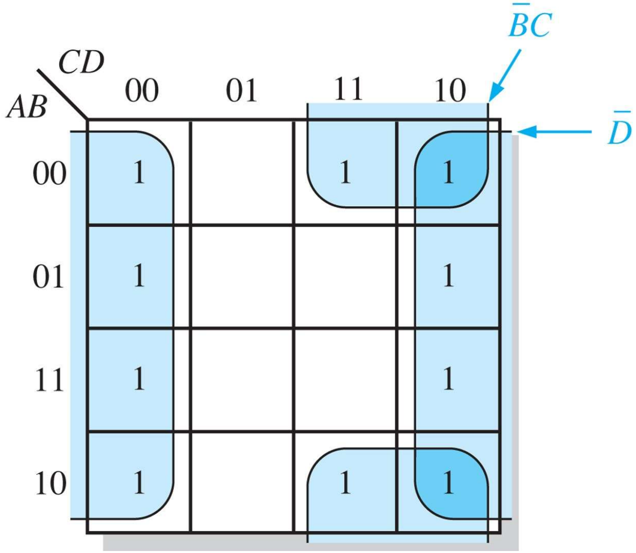
Karnough Map Example

$$\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}CD + A\overline{B}CD$$

$$+ \overline{A}\,\overline{B}C\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$$

$$\downarrow$$

$$\overline{D} + \overline{B}C$$

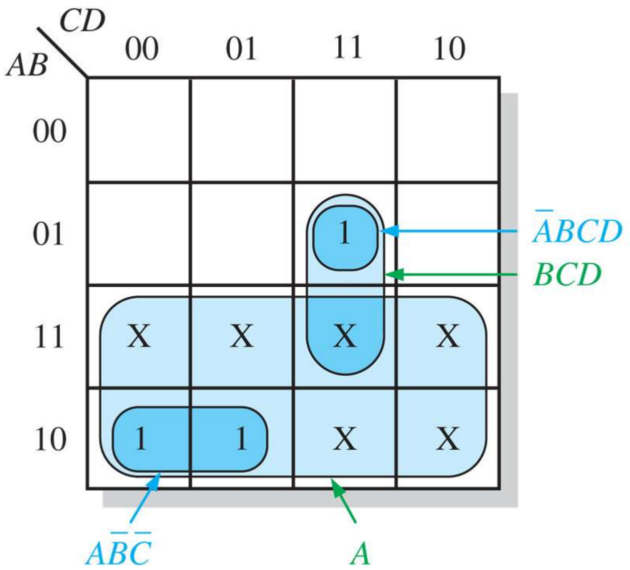


Don’t Care Terms

Inputs				Output
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

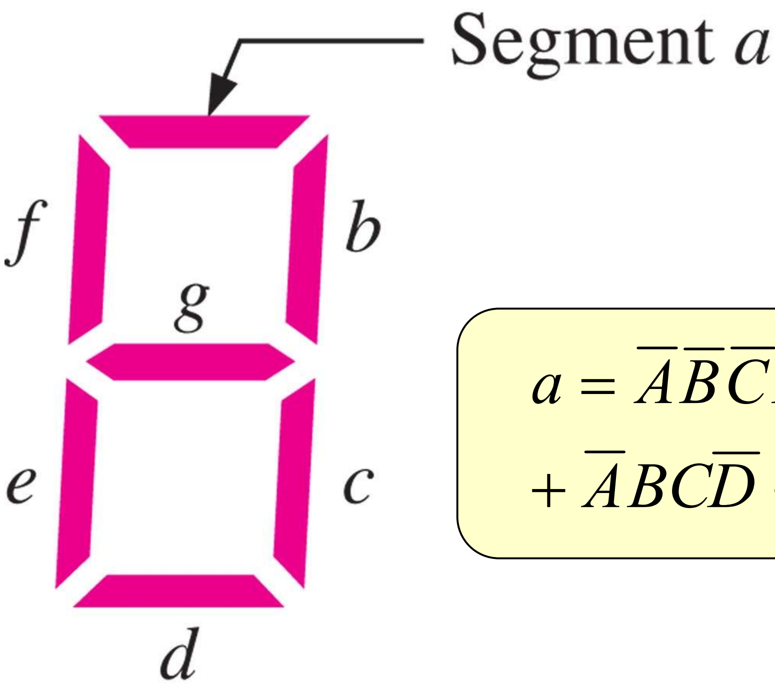
Don’t cares

(a) Truth table



(b) Without “don’t cares” $Y = A\bar{B}\bar{C} + \bar{A}BCD$
With “don’t cares” $Y = A + BCD$

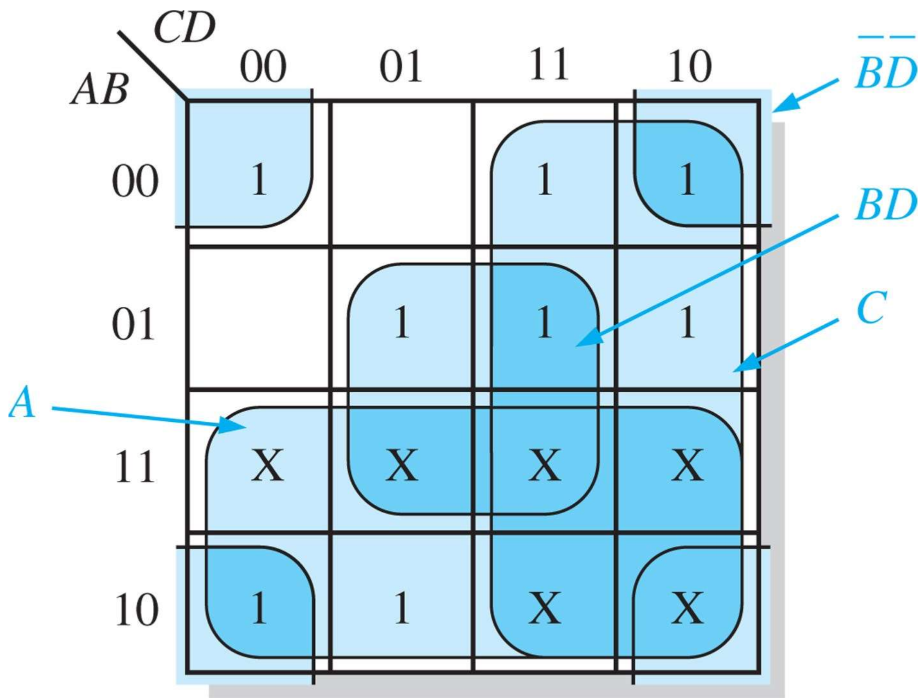
7-Segment Display



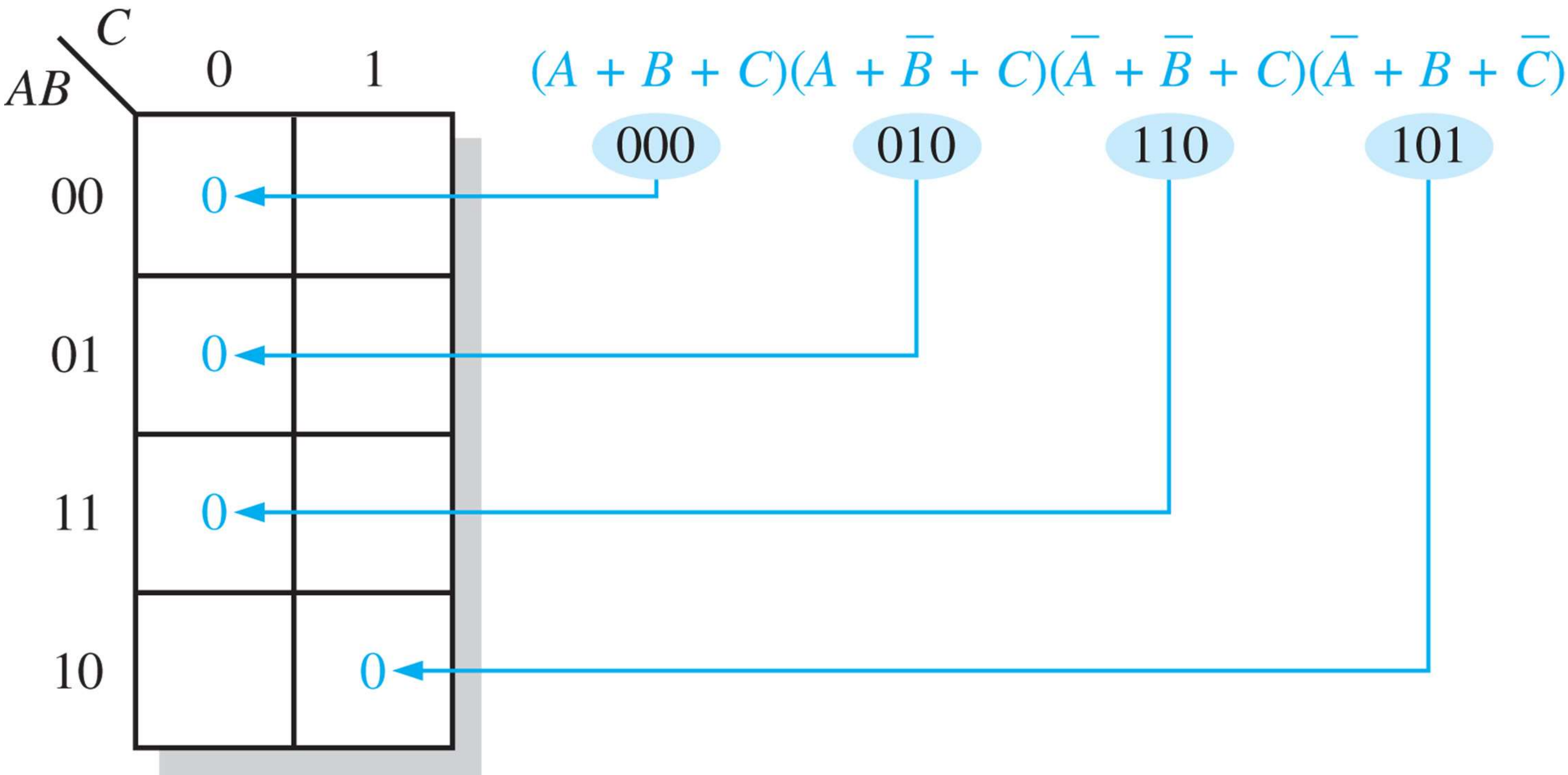
$$a = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}D$$
$$+ \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D$$

7-Segment Display

$$a = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}D$$
$$+ \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D$$

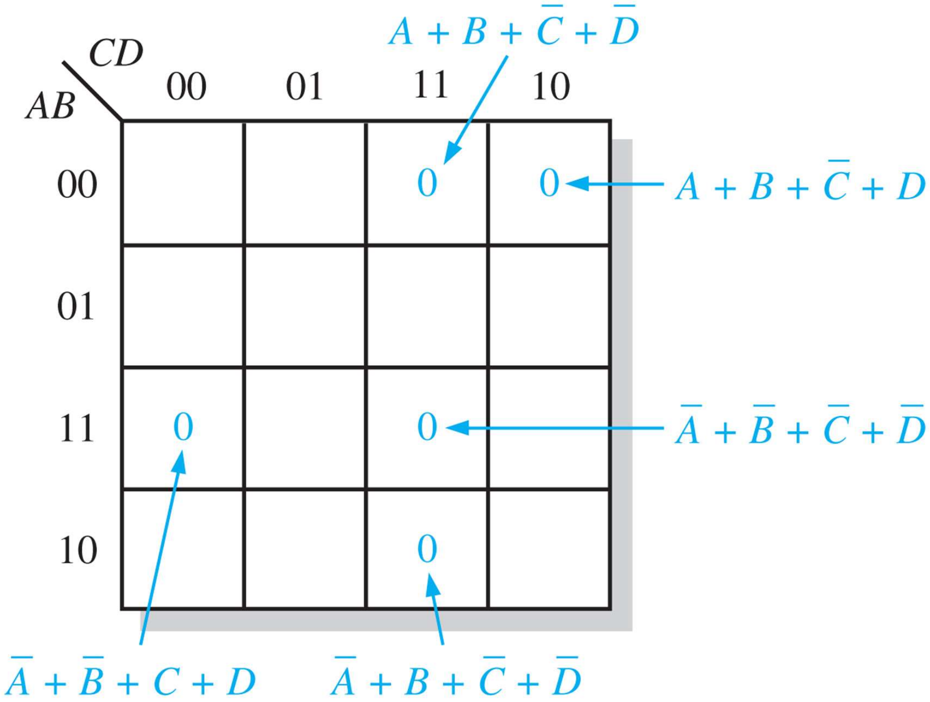


Mapping a Standard POS



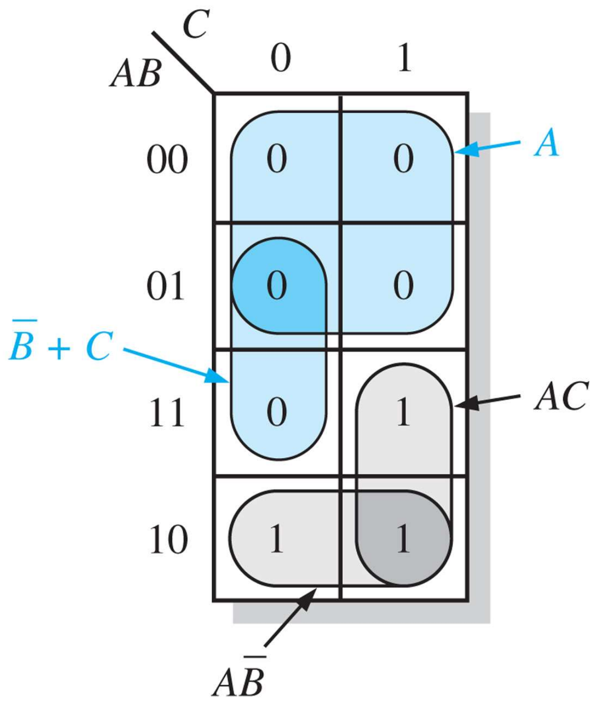
Mapping a Standard POS

$$(\overline{A}+\overline{B}+C+D)(\overline{A}+B+\overline{C}+\overline{D})(A+B+\overline{C}+D)$$
$$(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+B+\overline{C}+\overline{D})$$



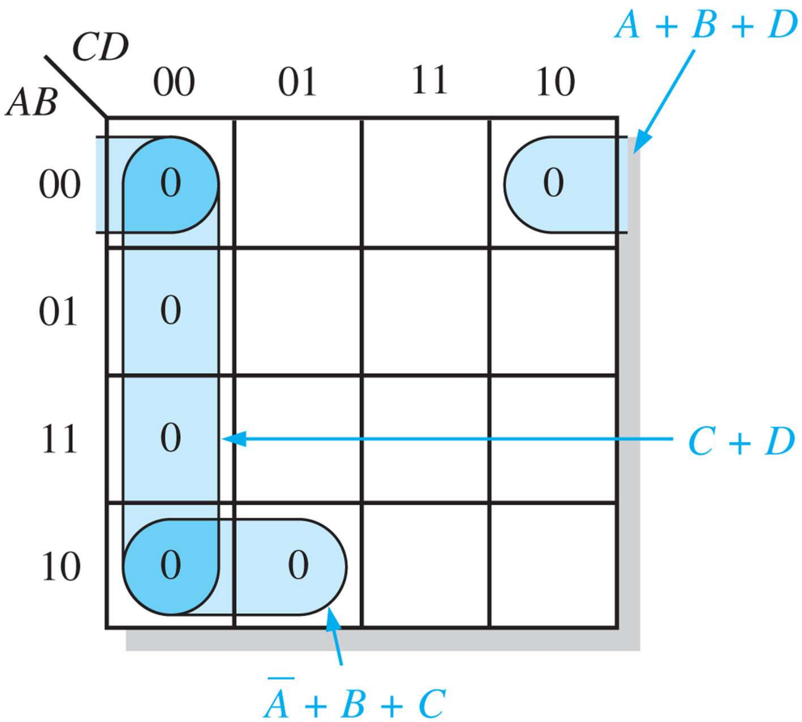
Simplification of POS

$$(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)$$
$$(A+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)$$



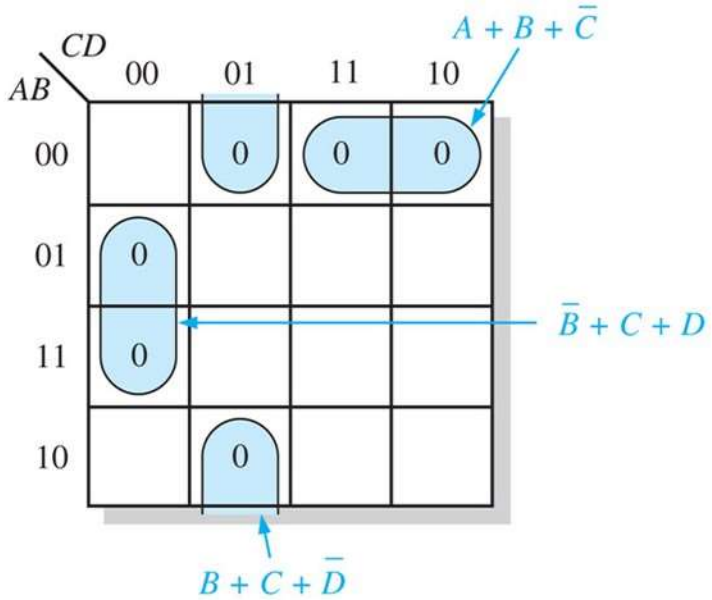
Simplification of POS

$$(B+C+D)(A+B+\overline{C}+D)(\overline{A}+B+C+\overline{D})$$
$$(A+\overline{B}+C+D)(\overline{A}+\overline{B}+C+D)$$



Convert between SOP and POS

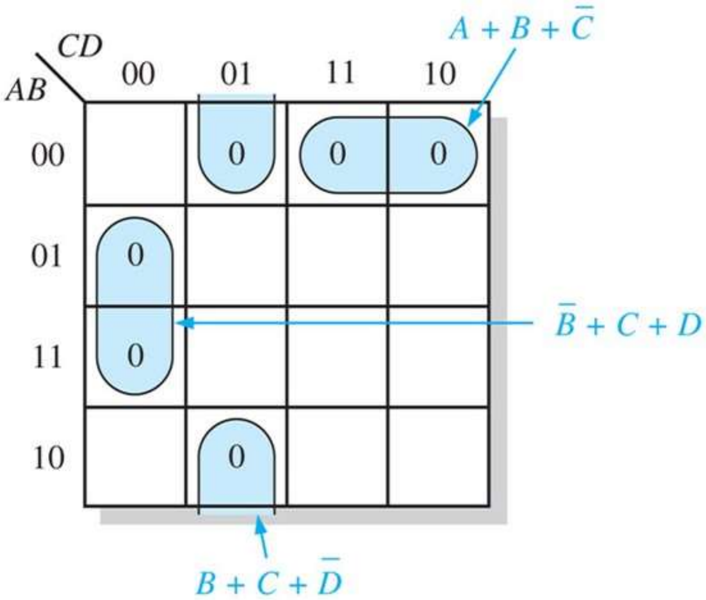
$$(\overline{A}+\overline{B}+C+D)(A+\overline{B}+C+D)(A+B+C+\overline{D})$$
$$(A+B+\overline{C}+\overline{D})(\overline{A}+B+C+\overline{D})(A+B+\overline{C}+D)$$



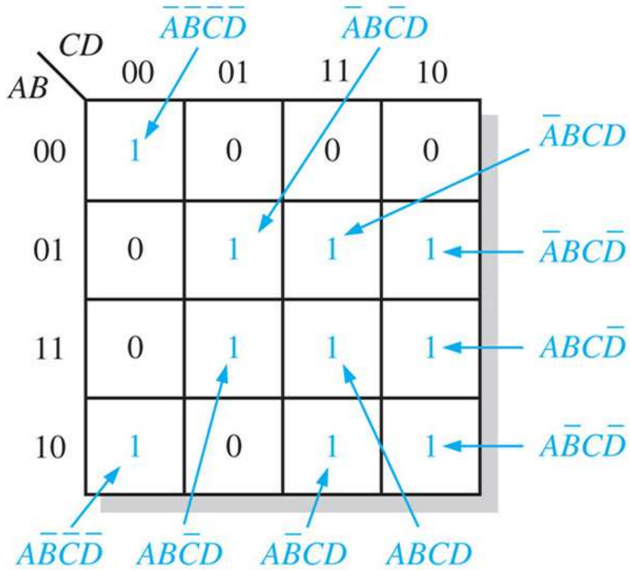
(a) Minimum POS: $(A+B+\overline{C})(\overline{B}+C+D)(B+C+\overline{D})$

Convert between SOP and POS

$$(\overline{A}+\overline{B}+C+D)(A+\overline{B}+C+D)(A+B+C+\overline{D})$$
$$(A+B+\overline{C}+\overline{D})(\overline{A}+B+C+\overline{D})(A+B+\overline{C}+D)$$



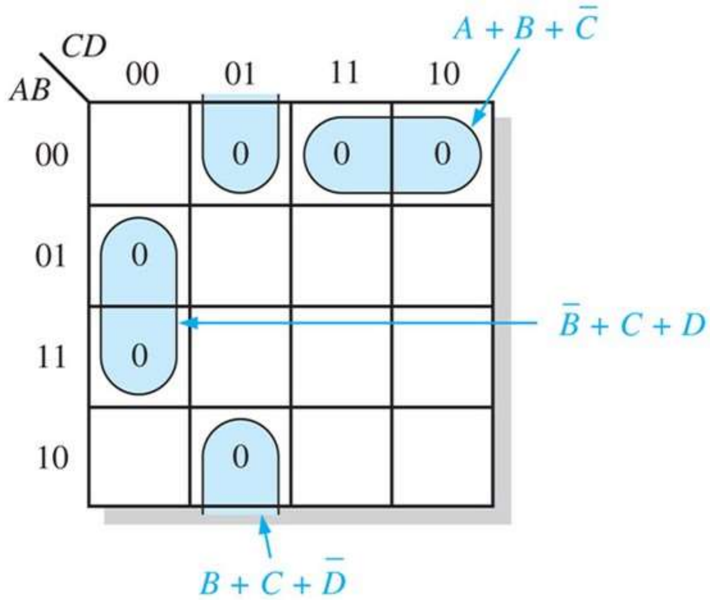
(a) Minimum POS: $(A + B + \overline{C})(\overline{B} + C + D)(B + C + \overline{D})$



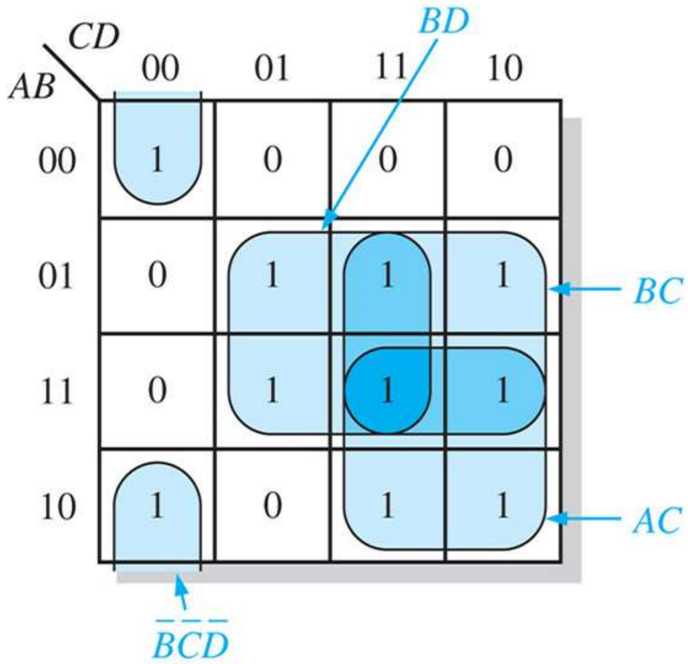
(b) Standard SOP:
$\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D} +$
$A\overline{B}\overline{C}\overline{D} + AB\overline{C}D + A\overline{B}CD + ABCD$

Convert between SOP and POS

$$(\overline{A}+\overline{B}+C+D)(A+\overline{B}+C+D)(A+B+C+\overline{D})$$
$$(A+B+\overline{C}+\overline{D})(\overline{A}+B+C+\overline{D})(A+B+\overline{C}+D)$$



(a) Minimum POS: $(A+B+\overline{C})(\overline{B}+C+D)(B+C+\overline{D})$



(c) Minimum SOP: $AC+BC+BD+\overline{B}\overline{C}\overline{D}$

Quine-McCluskey Method

TABLE 4–9

ABCD	*X*	Minterm
0000	0	
0001	1	m_1
0010	0	
0011	1	m_3
0100	1	m_4
0101	1	m_5
0110	0	
0111	0	
1000	0	
1001	0	
1010	1	m_{10}
1011	0	
1100	1	m_{12}
1101	1	m_{13}
1110	0	
1111	1	m_{15}

Quine-McCluskey Method

TABLE 4–10

Number of 1s	Minterm	*ABCD*
1	m_1	0001
	m_4	0100
2	m_3	0011
	m_5	0101
	m_{10}	1010
	m_{12}	1100
3	m_{13}	1101
4	m_{15}	1111

Quine-McCluskey Method

TABLE 4–11

Number of 1s in Minterm	Minterm	ABCD	First Level
1	m_1	0001 ✓	(m_1, m_3) 00x1
	m_4	0100 ✓	(m_1, m_5) 0x01
2	m_3	0011 ✓	(m_4, m_5) 010x
	m_5	0101 ✓	(m_4, m_{12}) x100
	m_{10}	1010	(m_5, m_{13}) x101
	m_{12}	1100 ✓	(m_{12}, m_{13}) 110x
3	m_{13}	1101 ✓	(m_{13}, m_{15}) 11x1
4	m_{15}	1111 ✓	

Quine-McCluskey Method

TABLE 4–12

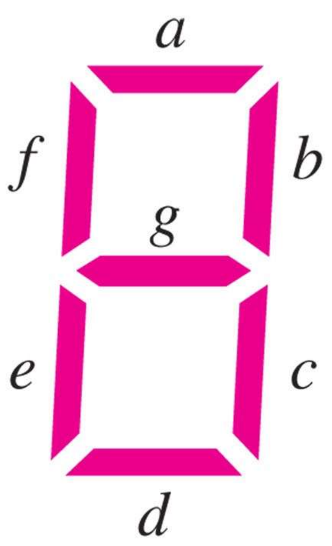
First Level	Number of 1s in First Level	Second Level
(m_1, m_3) 00x1	1	$(m_4, m_5, m_{12}, m_{13})$ x10x
(m_1, m_5) 0x01		$(m_4, m_5, m_{12}, m_{13})$ x10x
(m_4, m_5) 010x ✓		
(m_4, m_{12}) x100 ✓		
(m_5, m_{13}) x101 ✓	2	
(m_{12}, m_{13}) 110x ✓		
(m_{13}, m_{15}) 11x1	3	

Quine-McCluskey Method

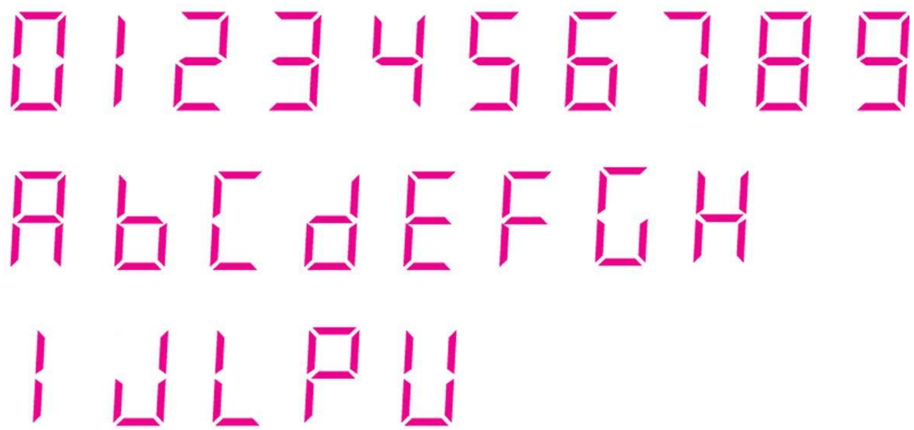
TABLE 4–13

	Minterms							
Prime Implicants	m_1	m_3	m_4	m_5	m_{10}	m_{12}	m_{13}	m_{15}
$B\overline{C}$ (m_4, m_5, m_{12}, m_{13})			✓	✓		✓	✓	
$\overline{A}\overline{B}D$ (m_1, m_3)	✓	✓						
$\overline{A}\overline{C}D$ (m_1, m_5)	✓			✓				
ABD (m_{13}, m_{15})							✓	✓
$A\overline{B}C\overline{D}$ (m_{10})					✓			

7-Segment Display

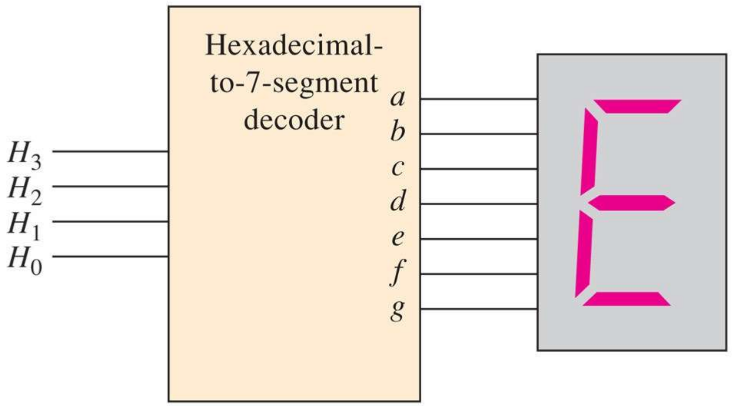


(a) Segment arrangement



(b) Formation of the ten digits and certain letters

7-Segment Display



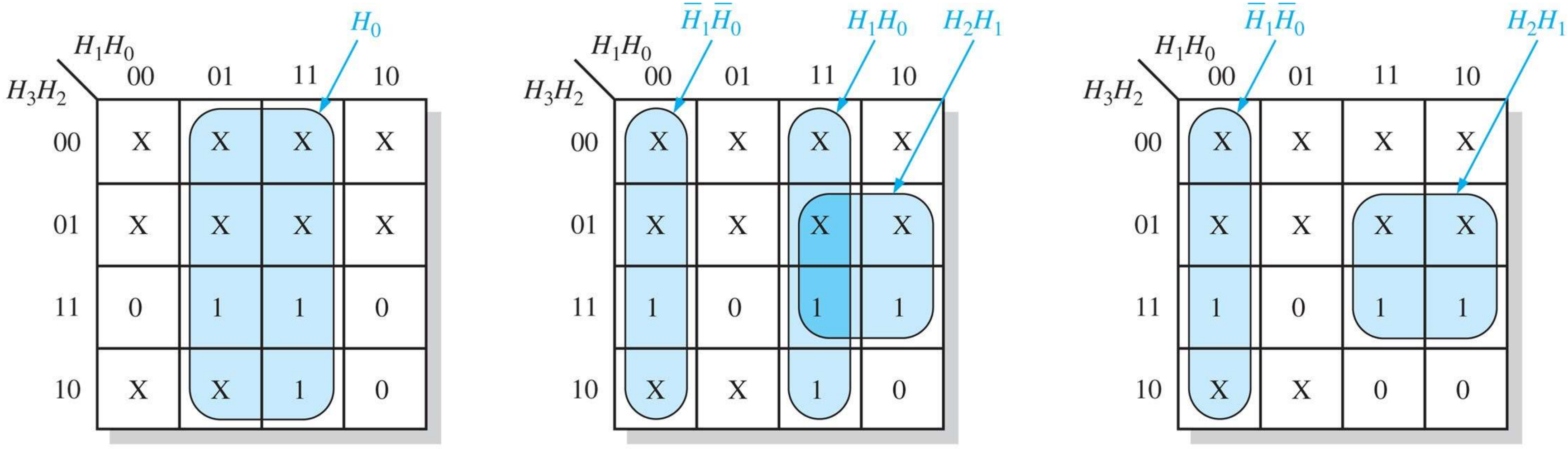
(a)

Letter	Hexadecimal Inputs				Segment Ouputs						
	H_3	H_2	H_1	H_0	*a*	*b*	*c*	*d*	*e*	*f*	*g*
A	1	0	1	0	0	0	0	1	0	0	0
b	1	0	1	1	1	1	0	0	0	0	0
C	1	1	0	0	0	1	1	0	0	0	1
d	1	1	0	1	1	0	0	0	0	1	0
E	1	1	1	0	0	1	1	0	0	0	0
F	1	1	1	1	1	1	1	1	1	1	1

(b)

1. Display the type of Vitamin (A, b, C, d, E)
2. F is not used as an Input

7-Segment Display



$a = H_0$

(a)

$b = \overline{H}_1\overline{H}_0 + H_1H_0 + H_2H_1$

(b)

$c = \overline{H}_1\overline{H}_0 + H_2H_1$

(c)