



Author	Thomas L. Floyd
Year	2009
Title of Article/Chapter	Boolean Algebra And Logic Simplification
Title of Journal/Book	Digital Fundamentals
Vol/part/pages	173-208
Publisher	Pearson Prentice Hall

This Digital Copy has been made under the terms of a CLA licence which allows you to:

Access and download a copy

Print out a copy

ISBN/ISSN: 9780138146467

Learning | **Resources**

4-1 BOOLEAN OPERATIONS AND EXPRESSIONS

Boolean algebra is the mathematics of digital systems. A basic knowledge of Boolean algebra is indispensable to the study and analysis of logic circuits. In the last chapter, Boolean operations and expressions in terms of their relationship to NOT, AND, OR, NAND, and NOR gates were introduced.

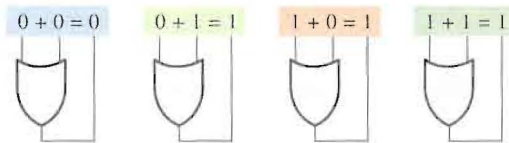
After completing this section, you should be able to

- ♦ Define *variable*
- ♦ Define *literal*
- ♦ Identify a sum term
- ♦ Evaluate a sum term
- ♦ Identify a product term
- ♦ Evaluate a product term
- ♦ Explain Boolean addition
- ♦ Explain Boolean multiplication

Variable, *complement*, and *literal* are terms used in Boolean algebra. A **variable** is a symbol (usually an italic uppercase letter or word) used to represent an action, a condition, or data. Any single variable can have only a 1 or a 0 value. The **complement** is the inverse of a variable and is indicated by a bar over the variable (overbar). For example, the complement of the variable A is \bar{A} . If $A = 1$, then $\bar{A} = 0$. If $A = 0$, then $\bar{A} = 1$. The complement of the variable A is read as “not A ” or “ A bar.” Sometimes a prime symbol rather than an overbar is used to denote the complement of a variable; for example, B' indicates the complement of B . In this book, only the overbar is used. A **literal** is a variable or the complement of a variable.

Boolean Addition

Recall from Chapter 3 that **Boolean addition** is equivalent to the OR operation and the basic rules are illustrated with their relation to the OR gate as follows:



In Boolean algebra, a **sum term** is a sum of literals. In logic circuits, a sum term is produced by an OR operation with no AND operations involved. Some examples of sum terms are $A + B$, $A + \bar{B}$, $A + B + \bar{C}$, and $\bar{A} + B + C + \bar{D}$.

A sum term is equal to 1 when one or more of the literals in the term are 1. A sum term is equal to 0 only if each of the literals is 0.

COMPUTER NOTE

In a microprocessor, the arithmetic logic unit (ALU) performs arithmetic and Boolean logic operations on digital data as directed by program instructions. Logical operations are equivalent to the basic gate operations that you are familiar with but deal with a minimum of 8 bits at a time. Examples of Boolean logic instructions are AND, OR, NOT, and XOR, which are called *mnemonics*. An assembly language program uses the mnemonics to specify an operation. Another program called an *assembler* translates the mnemonics into a binary code that can be understood by the microprocessor.

The OR operation is the Boolean form of addition.

EXAMPLE 4-1

Determine the values of A , B , C , and D that make the sum term $A + \bar{B} + C + \bar{D}$ equal to 0.

Solution For the sum term to be 0, each of the literals in the term must be 0. Therefore, $A = 0$, $B = 1$ so that $\bar{B} = 0$, $C = 0$, and $D = 1$ so that $\bar{D} = 0$.

$$A + \bar{B} + C + \bar{D} = 0 + \bar{1} + 0 + \bar{1} = 0 + 0 + 0 + 0 = 0$$

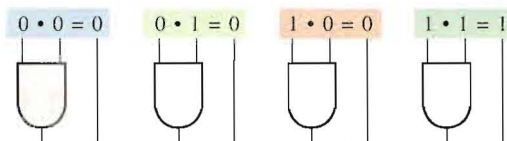
Related Problem* Determine the values of A and B that make the sum term $\bar{A} + B$ equal to 0.

*Answers are at the end of the chapter.

Boolean Multiplication

The AND operation is the Boolean form of multiplication.

Also recall from Chapter 3 that **Boolean multiplication** is equivalent to the AND operation and the basic rules are illustrated with their relation to the AND gate as follows:



In Boolean algebra, a **product term** is the product of literals. In logic circuits, a product term is produced by an AND operation with no OR operations involved. Some examples of product terms are AB , $A\bar{B}$, ABC , and $A\bar{B}\bar{C}\bar{D}$.

A product term is equal to 1 only if each of the literals in the term is 1. A product term is equal to 0 when one or more of the literals are 0.

EXAMPLE 4-2

Determine the values of A , B , C , and D that make the product term $A\bar{B}\bar{C}\bar{D}$ equal to 1.

Solution For the product term to be 1, each of the literals in the term must be 1. Therefore, $A = 1$, $B = 0$ so that $\bar{B} = 1$, $C = 1$, and $D = 0$ so that $\bar{D} = 1$.

$$A\bar{B}\bar{C}\bar{D} = 1 \cdot \bar{0} \cdot 1 \cdot \bar{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Related Problem Determine the values of A and B that make the product term $\bar{A}\bar{B}$ equal to 1.

SECTION 4-1 CHECKUP

Answers are at the end of the chapter.

1. If $A = 0$, what does \bar{A} equal?
2. Determine the values of A , B , and C that make the sum term $\bar{A} + \bar{B} + C$ equal to 0.
3. Determine the values of A , B , and C that make the product term $A\bar{B}\bar{C}$ equal to 1.

4-2 LAWS AND RULES OF BOOLEAN ALGEBRA

As in other areas of mathematics, there are certain well-developed rules and laws that must be followed in order to properly apply Boolean algebra. The most important of these are presented in this section.

After completing this section, you should be able to

- ♦ Apply the commutative laws of addition and multiplication
- ♦ Apply the associative laws of addition and multiplication
- ♦ Apply the distributive law
- ♦ Apply twelve basic rules of Boolean algebra

Laws of Boolean Algebra

The basic laws of Boolean algebra—the **commutative laws** for addition and multiplication, the **associative laws** for addition and multiplication, and the **distributive law**—are the same as in ordinary algebra. Each of the laws is illustrated with two or three variables, but the number of variables is not limited to this.

Commutative Laws The *commutative law of addition* for two variables is written as

$$A + B = B + A$$

Equation 4-1

This law states that the order in which the variables are ORed makes no difference. Remember, in Boolean algebra as applied to logic circuits, addition and the OR operation are the same. Figure 4-1 illustrates the commutative law as applied to the OR gate and shows that it doesn't matter to which input each variable is applied. (The symbol \equiv means "equivalent to.")

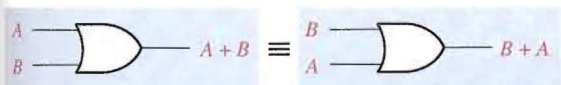


FIGURE 4-1

Application of commutative law of addition.

The *commutative law of multiplication* for two variables is

$$AB = BA$$

Equation 4-2

This law states that the order in which the variables are ANDed makes no difference. Figure 4-2 illustrates this law as applied to the AND gate.

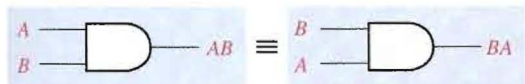


FIGURE 4-2

Application of commutative law of multiplication.

Associative Laws The *associative law of addition* is written as follows for three variables:

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

Equation 4-3

This law states that when ORing more than two variables, the result is the same regardless of the grouping of the variables. Figure 4-3 illustrates this law as applied to 2-input OR gates.

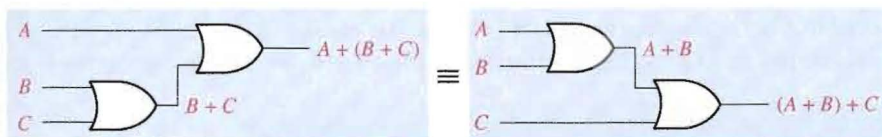


FIGURE 4-3

Application of associative law of addition. Open file F04-03 to verify.



The *associative law of multiplication* is written as follows for three variables:

$$A(BC) = (AB)C$$

Equation 4-4

This law states that it makes no difference in what order the variables are grouped when ANDing more than two variables. Figure 4-4 illustrates this law as applied to 2-input AND gates.

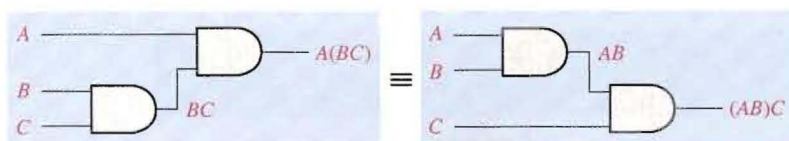


FIGURE 4-4

Application of associative law of multiplication. Open file F04-04 to verify.



Equation 4-5

Distributive Law The distributive law is written for three variables as follows:

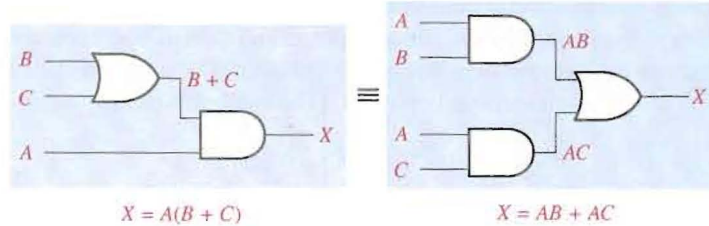
$$A(B + C) = AB + AC$$

This law states that ORing two or more variables and then ANDing the result with a single variable is equivalent to ANDing the single variable with each of the two or more variables and then ORing the products. The distributive law also expresses the process of *factoring* in which the common variable *A* is factored out of the product terms, for example, $AB + AC = A(B + C)$. Figure 4-5 illustrates the distributive law in terms of gate implementation.



► FIGURE 4-5

Application of distributive law. Open file F04-05 to verify.



Rules of Boolean Algebra

Table 4-1 lists 12 basic rules that are useful in manipulating and simplifying **Boolean expressions**. Rules 1 through 9 will be viewed in terms of their application to logic gates. Rules 10 through 12 will be derived in terms of the simpler rules and the laws previously discussed.

► TABLE 4-1

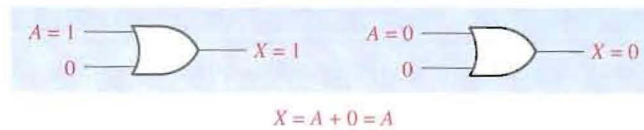
Basic rules of Boolean algebra.

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

A, B, or C can represent a single variable or a combination of variables.

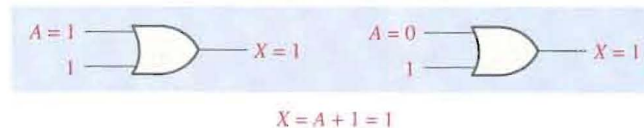
Rule 1: $A + 0 = A$ A variable ORed with 0 is always equal to the variable. If the input variable *A* is 1, the output variable *X* is 1, which is equal to *A*. If *A* is 0, the output is 0, which is also equal to *A*. This rule is illustrated in Figure 4-6, where the lower input is fixed at 0.

► FIGURE 4-6

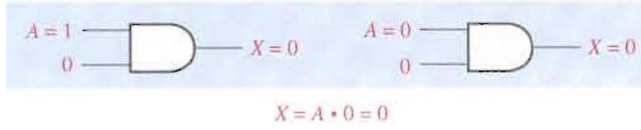


Rule 2: $A + 1 = 1$ A variable ORed with 1 is always equal to 1. A 1 on an input to an OR gate produces a 1 on the output, regardless of the value of the variable on the other input. This rule is illustrated in Figure 4-7, where the lower input is fixed at 1.

► FIGURE 4-7

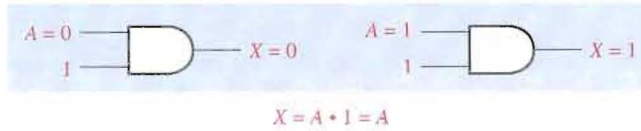


Rule 3: $A \cdot 0 = 0$ A variable ANDed with 0 is always equal to 0. Any time one input to an AND gate is 0, the output is 0, regardless of the value of the variable on the other input. This rule is illustrated in Figure 4–8, where the lower input is fixed at 0.



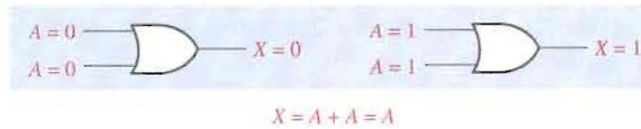
◀ FIGURE 4–8

Rule 4: $A \cdot 1 = A$ A variable ANDed with 1 is always equal to the variable. If A is 0, the output of the AND gate is 0. If A is 1, the output of the AND gate is 1 because both inputs are now 1s. This rule is shown in Figure 4–9, where the lower input is fixed at 1.



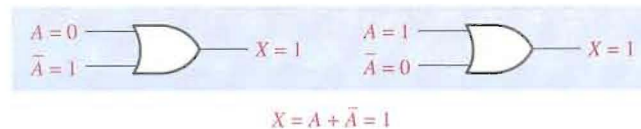
◀ FIGURE 4–9

Rule 5: $A + A = A$ A variable ORed with itself is always equal to the variable. If A is 0, then $0 + 0 = 0$; and if A is 1, then $1 + 1 = 1$. This is shown in Figure 4–10, where both inputs are the same variable.



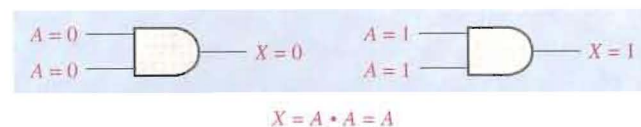
◀ FIGURE 4–10

Rule 6: $A + \bar{A} = 1$ A variable ORed with its complement is always equal to 1. If A is 0, then $0 + \bar{0} = 0 + 1 = 1$. If A is 1, then $1 + \bar{1} = 1 + 0 = 1$. See Figure 4–11, where one input is the complement of the other.



◀ FIGURE 4–11

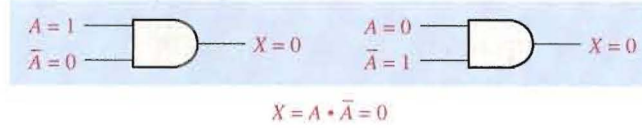
Rule 7: $A \cdot A = A$ A variable ANDed with itself is always equal to the variable. If $A = 0$, then $0 \cdot 0 = 0$; and if $A = 1$, then $1 \cdot 1 = 1$. Figure 4–12 illustrates this rule.



◀ FIGURE 4–12

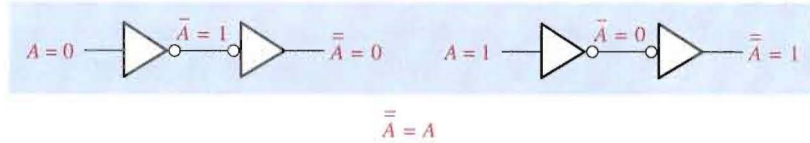
Rule 8: $A \cdot \bar{A} = 0$ A variable ANDed with its complement is always equal to 0. Either A or \bar{A} will always be 0; and when a 0 is applied to the input of an AND gate, the output will be 0 also. Figure 4–13 illustrates this rule.

▶ FIGURE 4–13



Rule 9: $\bar{\bar{A}} = A$ The double complement of a variable is always equal to the variable. If you start with the variable A and complement (invert) it once, you get \bar{A} . If you then take \bar{A} and complement (invert) it, you get A , which is the original variable. This rule is shown in Figure 4–14 using inverters.

▶ FIGURE 4–14



Rule 10: $A + AB = A$ This rule can be proved by applying the distributive law, rule 2, and rule 4 as follows:

$$\begin{aligned}
 A + AB &= A \cdot 1 + AB = A(1 + B) && \text{Factoring (distributive law)} \\
 &= A \cdot 1 && \text{Rule 2: } (1 + B) = 1 \\
 &= A && \text{Rule 4: } A \cdot 1 = A
 \end{aligned}$$

The proof is shown in Table 4–2, which shows the truth table and the resulting logic circuit simplification.

▶ TABLE 4–2



Rule 10: $A + AB = A$. Open file T04-02 to verify.

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

↑ equal ↑

Rule 11: $A + \bar{A}B = A + B$ This rule can be proved as follows:

$$\begin{aligned}
 A + \bar{A}B &= (A + AB) + \bar{A}B && \text{Rule 10: } A = A + AB \\
 &= (AA + AB) + \bar{A}B && \text{Rule 7: } A = AA \\
 &= AA + AB + \bar{A}\bar{A} + \bar{A}B && \text{Rule 8: adding } \bar{A}\bar{A} = 0 \\
 &= (A + \bar{A})(A + B) && \text{Factoring} \\
 &= 1 \cdot (A + B) && \text{Rule 6: } A + \bar{A} = 1 \\
 &= A + B && \text{Rule 4: drop the 1}
 \end{aligned}$$

The proof is shown in Table 4–3, which shows the truth table and the resulting logic circuit simplification.

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

↑ equal ↑

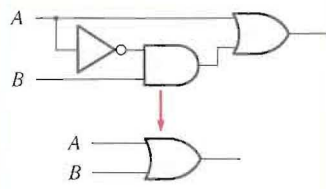


TABLE 4-3

Rule 11: $A + \bar{A}B = A + B$. Open file T04-03 to verify.



Rule 12: $(A + B)(A + C) = A + BC$ This rule can be proved as follows:

$$\begin{aligned}
 (A + B)(A + C) &= AA + AC + AB + BC && \text{Distributive law} \\
 &= A + AC + AB + BC && \text{Rule 7: } AA = A \\
 &= A(1 + C) + AB + BC && \text{Factoring (distributive law)} \\
 &= A \cdot 1 + AB + BC && \text{Rule 2: } 1 + C = 1 \\
 &= A(1 + B) + BC && \text{Factoring (distributive law)} \\
 &= A \cdot 1 + BC && \text{Rule 2: } 1 + B = 1 \\
 &= A + BC && \text{Rule 4: } A \cdot 1 = A
 \end{aligned}$$

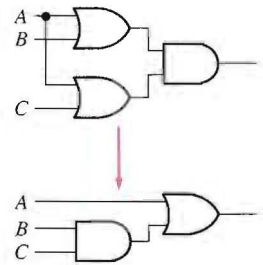
The proof is shown in Table 4-4, which shows the truth table and the resulting logic circuit simplification.

TABLE 4-4

Rule 12: $(A + B)(A + C) = A + BC$. Open file T04-04 to verify.

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 ↑



**SECTION 4-2
CHECKUP**

1. Apply the associative law of addition to the expression $A + (B + C + D)$.
2. Apply the distributive law to the expression $A(B + C + D)$.

4-3 DEMORGAN'S THEOREMS

DeMorgan, a mathematician who knew Boole, proposed two theorems that are an important part of Boolean algebra. In practical terms, DeMorgan's theorems provide mathematical verification of the equivalency of the NAND and negative-OR gates and the equivalency of the NOR and negative-AND gates, which were discussed in Chapter 3.

After completing this section, you should be able to

- ♦ State DeMorgan's theorems
- ♦ Relate DeMorgan's theorems to the equivalency of the NAND and negative-OR gates and to the equivalency of the NOR and negative-AND gates
- ♦ Apply DeMorgan's theorems to the simplification of Boolean expressions

To apply DeMorgan's theorem, break the bar over the product of variables and change the sign from AND to OR.

DeMorgan's first theorem is stated as follows:

The complement of a product of variables is equal to the sum of the complements of the variables.

Stated another way,

The complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables.

The formula for expressing this theorem for two variables is

Equation 4-6

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

DeMorgan's second theorem is stated as follows:

The complement of a sum of variables is equal to the product of the complements of the variables.

Stated another way,

The complement of two or more ORed variables is equivalent to the AND of the complements of the individual variables.

The formula for expressing this theorem for two variables is

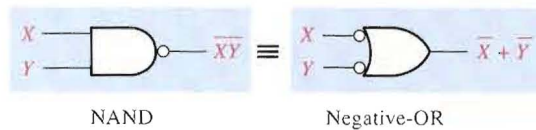
Equation 4-7

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

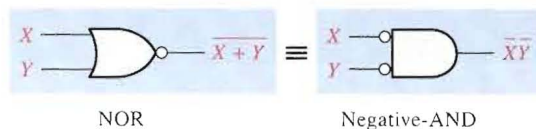
Figure 4-15 shows the gate equivalencies and truth tables for Equations 4-6 and 4-7.

► FIGURE 4-15

Gate equivalencies and the corresponding truth tables that illustrate DeMorgan's theorems. Notice the equality of the two output columns in each table. This shows that the equivalent gates perform the same logic function.



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



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

As stated, DeMorgan's theorems also apply to expressions in which there are more than two variables. The following examples illustrate the application of DeMorgan's theorems to 3-variable and 4-variable expressions.

EXAMPLE 4-3

 Apply DeMorgan's theorems to the expressions \overline{XYZ} and $\overline{X + Y + Z}$.

Solution

$$\begin{aligned}\overline{XYZ} &= \overline{X} + \overline{Y} + \overline{Z} \\ \overline{X + Y + Z} &= \overline{XYZ}\end{aligned}$$

Related Problem

 Apply DeMorgan's theorem to the expression $\overline{\overline{X} + \overline{Y} + \overline{Z}}$.

EXAMPLE 4-4

 Apply DeMorgan's theorems to the expressions \overline{WXYZ} and $\overline{W + X + Y + Z}$.

Solution

$$\begin{aligned}\overline{WXYZ} &= \overline{W} + \overline{X} + \overline{Y} + \overline{Z} \\ \overline{W + X + Y + Z} &= \overline{WXYZ}\end{aligned}$$

Related Problem

 Apply DeMorgan's theorem to the expression $\overline{\overline{WXYZ}}$.

Each variable in DeMorgan's theorems as stated in Equations 4-6 and 4-7 can also represent a combination of other variables. For example, X can be equal to the term $AB + C$, and Y can be equal to the term $A + BC$. So if you can apply DeMorgan's theorem for two variables as stated by $\overline{XY} = \overline{X} + \overline{Y}$ to the expression $\overline{(AB + C)(A + BC)}$, you get the following result:

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

Notice that in the preceding result you have two terms, $\overline{AB + C}$ and $\overline{A + BC}$, to each of which you can again apply DeMorgan's theorem $\overline{X + Y} = \overline{X}Y$ individually, as follows:

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

Notice that you still have two terms in the expression to which DeMorgan's theorem can again be applied. These terms are \overline{AB} and BC . A final application of DeMorgan's theorem gives the following result:

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

Although this result can be simplified further by the use of Boolean rules and laws, DeMorgan's theorems cannot be used any more.

Applying DeMorgan's Theorems

The following procedure illustrates the application of DeMorgan's theorems and Boolean algebra to the specific expression

$$\overline{\overline{A + BC} + D(E + F)}$$

Step 1: Identify the terms to which you can apply DeMorgan's theorems, and think of each term as a single variable. Let $\overline{A + BC} = X$ and $D(E + F) = Y$.

Step 2: Since $\overline{X + Y} = \overline{X}Y$,

$$\overline{\overline{A + BC} + D(E + F)} = \overline{\overline{A + BC}}\overline{D(E + F)}$$

Step 3: Use rule 9 ($\overline{\overline{A}} = A$) to cancel the double bars over the left term (this is not part of DeMorgan's theorem).

$$\overline{\overline{A + BC}}\overline{D(E + F)} = (A + BC)\overline{D(E + F)}$$

Step 4: Applying DeMorgan's theorem to the second term,

$$(A + \overline{BC})(\overline{D(E + \overline{F})}) = (A + \overline{BC})(\overline{D} + \overline{(\overline{E + \overline{F}})})$$

Step 5: Use rule 9 ($\overline{\overline{A}} = A$) to cancel the double bars over the $E + \overline{F}$ part of the term.

$$(A + \overline{BC})(\overline{D} + \overline{E + \overline{F}}) = (A + \overline{BC})(\overline{D} + E + \overline{F})$$

The following three examples will further illustrate how to use DeMorgan's theorems.

EXAMPLE 4-5

Apply DeMorgan's theorems to each of the following expressions:

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

(b) $\overline{ABC + DEF}$

(c) $\overline{\overline{AB} + \overline{CD} + EF}$

Solution: (a) Let $A + B + C = X$ and $D = Y$. The expression $\overline{(A + B + C)D}$ is of the form $\overline{XY} = \overline{X} + \overline{Y}$ and can be rewritten as

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

Next, apply DeMorgan's theorem to the term $\overline{A + B + C}$.

$$\overline{A + B + C} + \overline{D} = \overline{ABC} + \overline{D}$$

(b) Let $ABC = X$ and $DEF = Y$. The expression $\overline{ABC + DEF}$ is of the form $\overline{X + Y} = \overline{X} \overline{Y}$ and can be rewritten as

$$\overline{ABC + DEF} = \overline{ABC} \overline{DEF}$$

Next, apply DeMorgan's theorem to each of the terms \overline{ABC} and \overline{DEF} .

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

(c) Let $\overline{AB} = X$, $\overline{CD} = Y$, and $EF = Z$. The expression $\overline{\overline{AB} + \overline{CD} + EF}$ is of the form $\overline{X + Y + Z} = \overline{X} \overline{Y} \overline{Z}$ and can be rewritten as

$$\overline{\overline{AB} + \overline{CD} + EF} = \overline{\overline{AB}} \overline{\overline{CD}} \overline{EF}$$

Next, apply DeMorgan's theorem to each of the terms $\overline{\overline{AB}}$, $\overline{\overline{CD}}$, and \overline{EF} .

$$\overline{\overline{AB}} \overline{\overline{CD}} \overline{EF} = (\overline{A} + \overline{B})(\overline{C} + \overline{D})(\overline{E} + \overline{F})$$

Related Problem Apply DeMorgan's theorems to the expression $\overline{\overline{ABC} + D + E}$.

EXAMPLE 4-6

Apply DeMorgan's theorems to each expression:

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

(b) $\overline{(\overline{A} + B) + CD}$

(c) $\overline{(A + B)\overline{CD} + E + \overline{F}}$

Solution: (a) $\overline{(A + B) + \overline{C}} = \overline{(\overline{A + B})\overline{\overline{C}}} = (A + B)C$

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

(c) $\overline{(A + B)\overline{CD} + E + \overline{F}} = \overline{((A + B)\overline{CD})(E + \overline{F})} = \overline{\overline{AB} + C + D}\overline{EF}$

Related Problem Apply DeMorgan's theorems to the expression $\overline{\overline{AB}(C + \overline{D}) + E}$.

EXAMPLE 4-7

The Boolean expression for an exclusive-OR gate is $A\bar{B} + \bar{A}B$. With this as a starting point, use DeMorgan's theorems and any other rules or laws that are applicable to develop an expression for the exclusive-NOR gate.

Solution Start by complementing the exclusive-OR expression and then applying DeMorgan's theorems as follows:

$$\overline{A\bar{B} + \bar{A}B} = (\overline{A\bar{B}})(\overline{\bar{A}B}) = (\bar{A} + \bar{\bar{B}})(\bar{\bar{A}} + \bar{B}) = (\bar{A} + B)(A + \bar{B})$$

Next, apply the distributive law and rule 8 ($A \cdot \bar{A} = 0$).

$$(\bar{A} + B)(A + \bar{B}) = \bar{A}A + \bar{A}\bar{B} + AB + B\bar{B} = \bar{A}\bar{B} + AB$$

The final expression for the XNOR is $\bar{A}\bar{B} + AB$. Note that this expression equals 1 any time both variables are 0s or both variables are 1s.

Related Problem Starting with the expression for a 4-input NAND gate, use DeMorgan's theorems to develop an expression for a 4-input negative-OR gate.

**SECTION 4-3
CHECKUP**

1. Apply DeMorgan's theorems to the following expressions:

(a) $\overline{ABC} + \overline{(D + E)}$ (b) $\overline{(A + B)C}$ (c) $\overline{A + B + C} + \overline{DE}$

4-4 BOOLEAN ANALYSIS OF LOGIC CIRCUITS

Boolean algebra provides a concise way to express the operation of a logic circuit formed by a combination of logic gates so that the output can be determined for various combinations of input values.

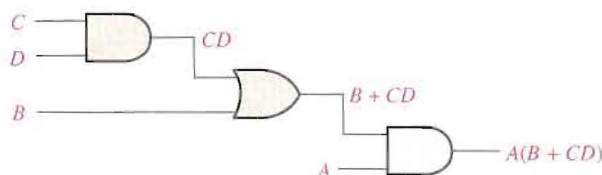
After completing this section, you should be able to

- ♦ Determine the Boolean expression for a combination of gates
- ♦ Evaluate the logic operation of a circuit from the Boolean expression
- ♦ Construct a truth table

Boolean Expression for a Logic Circuit

To derive the Boolean expression for a given combinational logic circuit, begin at the left-most inputs and work toward the final output, writing the expression for each gate. For the example circuit in Figure 4-16, the Boolean expression is determined in the following three steps:

1. The expression for the left-most AND gate with inputs C and D is CD .
2. The output of the left-most AND gate is one of the inputs to the OR gate and B is the other input. Therefore, the expression for the OR gate is $B + CD$.



A combinational logic circuit can be described by a Boolean equation.

◀ **FIGURE 4-16**

A combinational logic circuit showing the development of the Boolean expression for the output.

3. The output of the OR gate is one of the inputs to the right-most AND gate and A is the other input. Therefore, the expression for this AND gate is $A(B + CD)$, which is the final output expression for the entire circuit.

Constructing a Truth Table for a Logic Circuit

Once the Boolean expression for a given logic circuit has been determined, a truth table that shows the output for all possible values of the input variables can be developed. The procedure requires that you evaluate the Boolean expression for all possible combinations of values for the input variables. In the case of the circuit in Figure 4–16, there are four input variables (A , B , C , and D) and therefore sixteen ($2^4 = 16$) combinations of values are possible.

Evaluating the Expression To evaluate the expression $A(B + CD)$, first find the values of the variables that make the expression equal to 1, using the rules for Boolean addition and multiplication. In this case, the expression equals 1 only if $A = 1$ and $B + CD = 1$ because

$$A(B + CD) = 1 \cdot 1 = 1$$

Now determine when the $B + CD$ term equals 1. The term $B + CD = 1$ if either $B = 1$ or $CD = 1$ or if both B and CD equal 1 because

$$B + CD = 1 + 0 = 1$$

$$B + CD = 0 + 1 = 1$$

$$B + CD = 1 + 1 = 1$$

The term $CD = 1$ only if $C = 1$ and $D = 1$.

To summarize, the expression $A(B + CD) = 1$ when $A = 1$ and $B = 1$ regardless of the values of C and D or when $A = 1$ and $C = 1$ and $D = 1$ regardless of the value of B . The expression $A(B + CD) = 0$ for all other value combinations of the variables.

Putting the Results in Truth Table Format The first step is to list the sixteen input variable combinations of 1s and 0s in a binary sequence as shown in Table 4–5. Next, place a 1 in the output column for each combination of input variables that was determined in the evaluation. Finally, place a 0 in the output column for all other combinations of input variables. These results are shown in the truth table in Table 4–5.


▶ TABLE 4–5

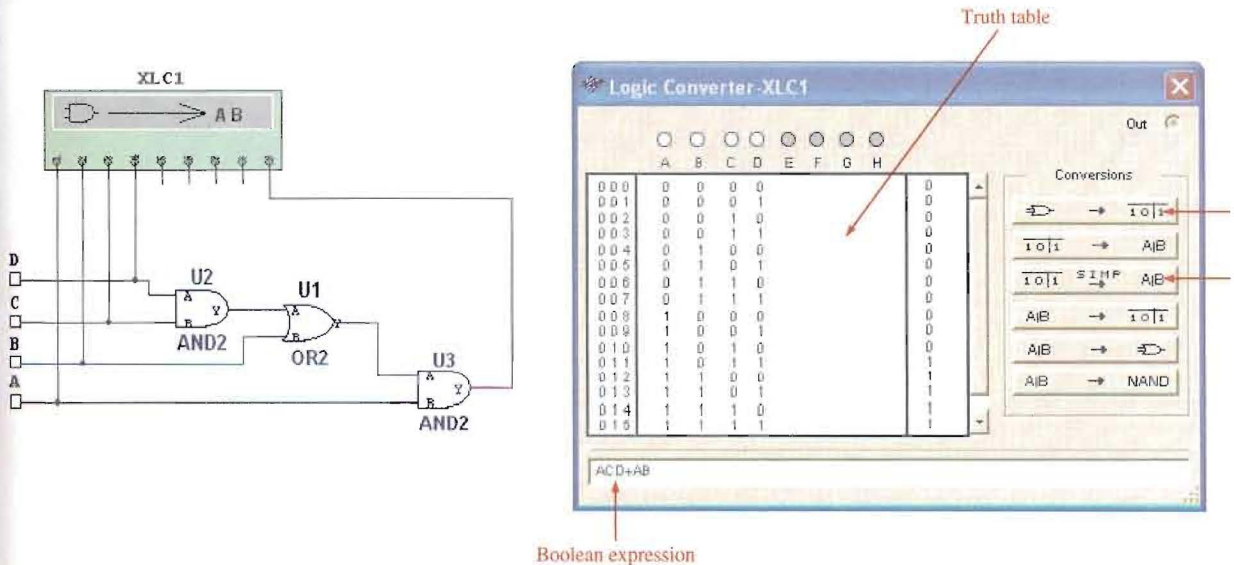
Truth table for the logic circuit in Figure 4–16.

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

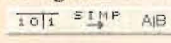
EXAMPLE 4-8

Use Multisim to generate the truth table for the logic circuit in Figure 4-16.

Solution Construct the circuit in Multisim and connect the Multisim Logic Converter to the inputs and output, as shown in Figure 4-17. Click on the  conversion bar, and the truth table appears in the display as shown.



▲ **FIGURE 4-17**

You can also generate the simplified Boolean expression from the truth table by clicking on .

Related Problem



Open Multisim to create the setup and do the conversions shown in this example.

**SECTION 4-4
CHECKUP**

1. Replace the AND gates with OR gates and the OR gate with an AND gate in Figure 4-16 and determine the Boolean expression for the output.
2. Construct a truth table for the circuit in Question 1.

4-5 SIMPLIFICATION USING BOOLEAN ALGEBRA

Many times in the application of Boolean algebra, you have to reduce a particular expression to its simplest form or change its form to a more convenient one to implement the expression most efficiently. The approach taken in this section is to use the basic laws, rules, and theorems of Boolean algebra to manipulate and simplify an expression. This method depends on a thorough knowledge of Boolean algebra and considerable practice in its application, not to mention a little ingenuity and cleverness.

After completing this section, you should be able to

- ♦ Apply the laws, rules, and theorems of Boolean algebra to simplify general expressions

A simplified Boolean expression uses the fewest gates possible to implement a given expression. Examples 4–9 through 4–12 illustrate Boolean simplification.

EXAMPLE 4–9

Using Boolean algebra techniques, simplify this expression:

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

Solution The following is not necessarily the only approach.

Step 1: Apply the distributive law to the second and third terms in the expression, as follows:

$$AB + AB + AC + BB + BC$$

Step 2: Apply rule 7 ($BB = B$) to the fourth term.

$$AB + AB + AC + B + BC$$

Step 3: Apply rule 5 ($AB + AB = AB$) to the first two terms.

$$AB + AC + B + BC$$

Step 4: Apply rule 10 ($B + BC = B$) to the last two terms.

$$AB + AC + B$$

Step 5: Apply rule 10 ($AB + B = B$) to the first and third terms.

$$B + AC$$

At this point the expression is simplified as much as possible. Once you gain experience in applying Boolean algebra, you can often combine many individual steps.

Related Problem Simplify the Boolean expression $\overline{AB} + A(\overline{B + C}) + B(\overline{B + C})$.

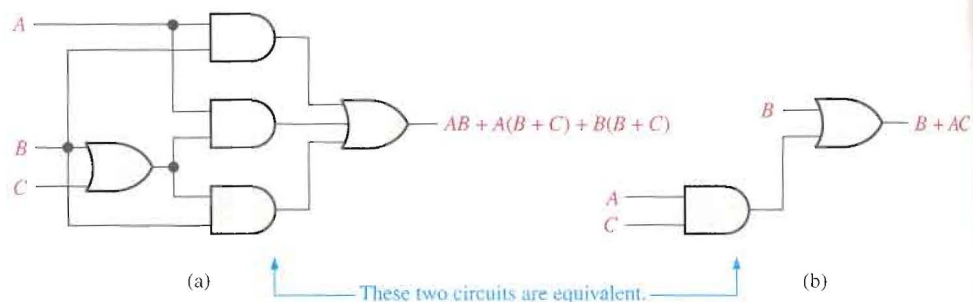
Simplification means fewer gates for the same function.



► FIGURE 4–18

Gate circuits for Example 4–9. Open file F04-18 to verify equivalency.

Figure 4–18 shows that the simplification process in Example 4–9 has significantly reduced the number of logic gates required to implement the expression. Part (a) shows that five gates are required to implement the expression in its original form; however, only two gates are needed for the simplified expression, shown in part (b). It is important to realize that these two gate circuits are equivalent. That is, for any combination of levels on the A, B, and C inputs, you get the same output from either circuit.



EXAMPLE 4-10

Simplify the following Boolean expression:

$$[\overline{A}B(C + BD) + \overline{A}\overline{B}]C$$

Note that brackets and parentheses mean the same thing: the term inside is multiplied (ANDed) with the term outside.

Solution **Step 1:** Apply the distributive law to the terms within the brackets.

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

Step 2: Apply rule 8 ($\overline{B}B = 0$) to the second term within the parentheses.

$$(\overline{A}BC + A \cdot 0 \cdot D + \overline{A}\overline{B})C$$

Step 3: Apply rule 3 ($A \cdot 0 \cdot D = 0$) to the second term within the parentheses.

$$(\overline{A}BC + 0 + \overline{A}\overline{B})C$$

Step 4: Apply rule 1 (drop the 0) within the parentheses.

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

Step 5: Apply the distributive law.

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

Step 6: Apply rule 7 ($CC = C$) to the first term.

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

Step 7: Factor out $\overline{B}C$.

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

Step 8: Apply rule 6 ($A + \overline{A} = 1$).

$$\overline{B}C \cdot 1$$

Step 9: Apply rule 4 (drop the 1).

$$\overline{B}C$$

Related Problem Simplify the Boolean expression $[AB(C + \overline{B}D) + \overline{A}\overline{B}]CD$.

EXAMPLE 4-11

Simplify the following Boolean expression:

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

Solution **Step 1:** Factor BC out of the first and last terms.

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

Step 2: Apply rule 6 ($\overline{A} + A = 1$) to the term in parentheses, and factor $\overline{A}\overline{B}$ from the second and last terms.

$$BC \cdot 1 + \overline{A}\overline{B}(\overline{C} + C) + \overline{A}\overline{B}C$$

Step 3: Apply rule 4 (drop the 1) to the first term and rule 6 ($\overline{C} + C = 1$) to the term in parentheses.

$$BC + \overline{A}\overline{B} \cdot 1 + \overline{A}\overline{B}C$$

Step 4: Apply rule 4 (drop the 1) to the second term.

$$BC + \overline{AB} + \overline{ABC}$$

Step 5: Factor \overline{B} from the second and third terms.

$$BC + \overline{B}(A + \overline{AC})$$

Step 6: Apply rule 11 ($A + \overline{AC} = A + \overline{C}$) to the term in parentheses.

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

Step 7: Use the distributive and commutative laws to get the following expression:

$$BC + \overline{AB} + \overline{BC}$$

Related Problem Simplify the Boolean expression $\overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC}$.

EXAMPLE 4-12

Simplify the following Boolean expression:

$$\overline{AB} + \overline{AC} + \overline{ABC}$$

Solution **Step 1:** Apply DeMorgan's theorem to the first term.

$$(\overline{AB})(\overline{AC}) + \overline{ABC}$$

Step 2: Apply DeMorgan's theorem to each term in parentheses.

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

Step 3: Apply the distributive law to the two terms in parentheses.

$$\overline{AA} + \overline{AC} + \overline{AB} + \overline{BC} + \overline{ABC}$$

Step 4: Apply rule 7 ($\overline{AA} = \overline{A}$) to the first term, and apply rule 10 [$\overline{AB} + \overline{ABC} = \overline{AB}(1 + C) = \overline{AB}$] to the third and last terms.

$$\overline{A} + \overline{AC} + \overline{AB} + \overline{BC}$$

Step 5: Apply rule 10 [$\overline{A} + \overline{AC} = \overline{A}(1 + C) = \overline{A}$] to the first and second terms.

$$\overline{A} + \overline{AB} + \overline{BC}$$

Step 6: Apply rule 10 [$\overline{A} + \overline{AB} = \overline{A}(1 + B) = \overline{A}$] to the first and second terms.


$$\overline{A} + \overline{BC}$$

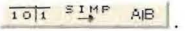
Related Problem Simplify the Boolean expression $\overline{AB} + \overline{AC} + \overline{ABC}$.


EXAMPLE 4-13

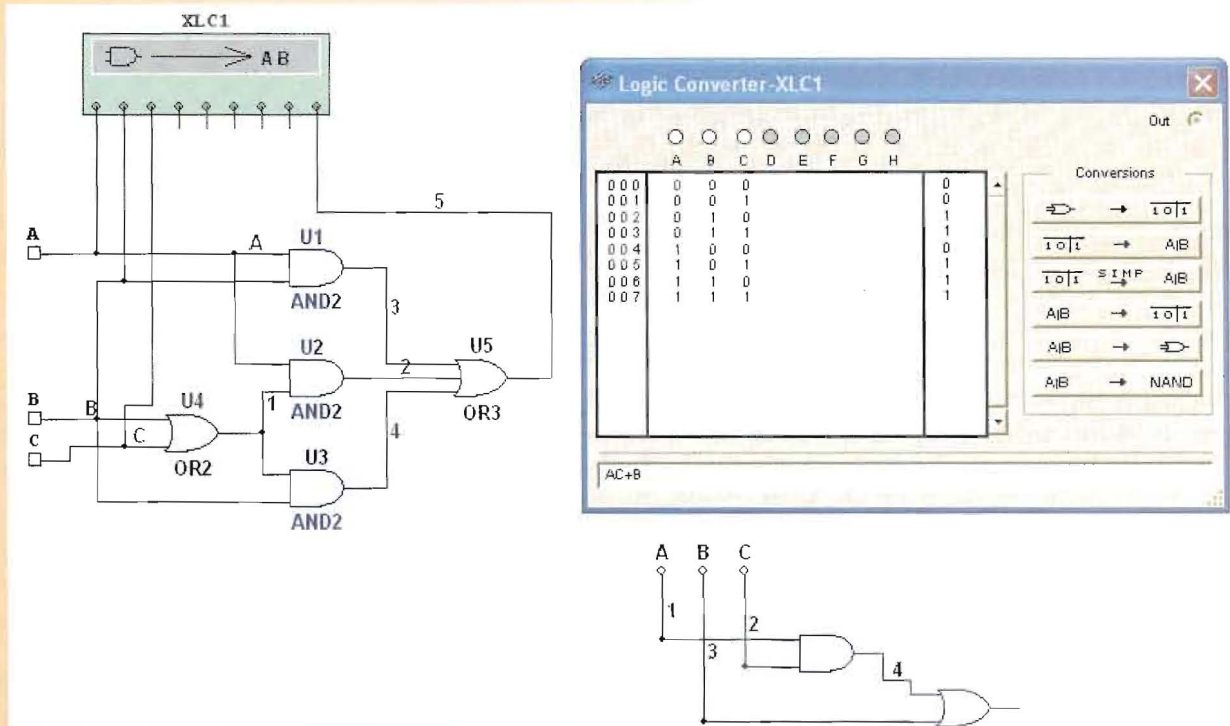
Use Multisim to perform the logic simplification shown in Figure 4-18.

Solution **Step 1:** Connect the Multisim Logic Converter to the circuit as shown in Figure 4-19.

Step 2: Generate the truth table by clicking on .

Step 3: Generate the simplified Boolean expression by clicking on .

Step 4: Generate the simplified logic circuit by clicking on .



▲ FIGURE 4-19

Related Problem

Use Multisim to create the setup and perform the logic simplification illustrated in this example.

**SECTION 4-5
CHECKUP**

- Simplify the following Boolean expressions if possible:
 - $A + AB + \bar{A}BC$
 - $(\bar{A} + B)C + ABC$
 - $\bar{A}C(BD + CDE) + \bar{A}C$
- Implement each expression in Question 1 as originally stated with the appropriate logic gates. Then implement the simplified expression, and compare the number of gates.

4-6 STANDARD FORMS OF BOOLEAN EXPRESSIONS

All Boolean expressions, regardless of their form, can be converted into either of two standard forms: the sum-of-products form or the product-of-sums form. Standardization makes the evaluation, simplification, and implementation of Boolean expressions much more systematic and easier.

After completing this section, you should be able to

- ♦ Identify a sum-of-products expression
- ♦ Determine the domain of a Boolean expression
- ♦ Convert any sum-of-products expression to a standard form

- ♦ Evaluate a standard sum-of-products expression in terms of binary values
- ♦ Identify a product-of-sums expression
- ♦ Convert any product-of-sums expression to a standard form
- ♦ Evaluate a standard product-of-sums expression in terms of binary values
- ♦ Convert from one standard form to the other

The Sum-of-Products (SOP) Form

A product term was defined in Section 4-1 as a term consisting of the product (Boolean multiplication) of literals (variables or their complements). When two or more product terms are summed by Boolean addition, the resulting expression is a **sum-of-products (SOP)**. Some examples are

$$\begin{aligned}
 &AB + ABC \\
 &ABC + CDE + \overline{BCD} \\
 &\overline{AB} + \overline{ABC} + AC
 \end{aligned}$$

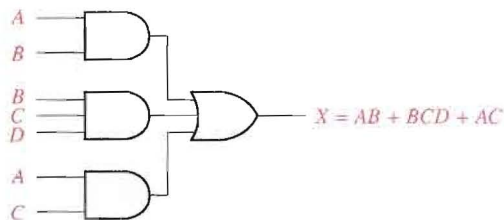
Also, an SOP expression can contain a single-variable term, as in $A + \overline{ABC} + \overline{BCD}$. Refer to the simplification examples in the last section, and you will see that each of the final expressions was either a single product term or in SOP form. In an SOP expression, a single overbar cannot extend over more than one variable; however, more than one variable in a term can have an overbar. For example, an SOP expression can have the term \overline{ABC} but not \overline{ABC} .

Domain of a Boolean Expression The **domain** of a general Boolean expression is the set of variables contained in the expression in either complemented or uncomplemented form. For example, the domain of the expression $\overline{AB} + \overline{ABC}$ is the set of variables A, B, C and the domain of the expression $\overline{ABC} + CDE + \overline{BCD}$ is the set of variables A, B, C, D, E .

AND/OR Implementation of an SOP Expression Implementing an SOP expression simply requires ORing the outputs of two or more AND gates. A product term is produced by an AND operation, and the sum (addition) of two or more product terms is produced by an OR operation. Therefore, an SOP expression can be implemented by AND-OR logic in which the outputs of a number (equal to the number of product terms in the expression) of AND gates connect to the inputs of an OR gate, as shown in Figure 4-20 for the expression $AB + BCD + AC$. The output X of the OR gate equals the SOP expression.

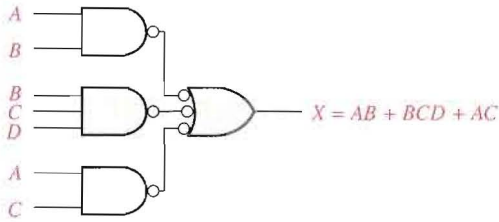
► **FIGURE 4-20**

Implementation of the SOP expression $AB + BCD + AC$.



NAND/NAND Implementation of an SOP Expression NAND gates can be used to implement an SOP expression. By using only NAND gates, an AND/OR function can be accomplished, as illustrated in Figure 4-21. The first level of NAND gates feed into a NAND gate that acts as a negative-OR gate. The NAND and negative-OR inversions cancel and the result is effectively an AND/OR circuit.

An SOP expression can be implemented with one OR gate and two or more AND gates.



◀ FIGURE 4-21

This NAND/NAND implementation is equivalent to the AND/OR in Figure 4-20.

Conversion of a General Expression to SOP Form

Any logic expression can be changed into SOP form by applying Boolean algebra techniques. For example, the expression $A(B + CD)$ can be converted to SOP form by applying the distributive law:

$$A(B + CD) = AB + ACD$$

EXAMPLE 4-14

Convert each of the following Boolean expressions to SOP form:

(a) $AB + B(CD + EF)$ (b) $(A + B)(B + C + D)$ (c) $\overline{(A + B)} + C$

Solution

(a) $AB + B(CD + EF) = AB + BCD + BEF$

(b) $(A + B)(B + C + D) = AB + AC + AD + BB + BC + BD$

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

Related Problem

Convert $\overline{ABC} + (A + \overline{B})(B + \overline{C} + \overline{AB})$ to SOP form.

The Standard SOP Form

So far, you have seen SOP expressions in which some of the product terms do not contain all of the variables in the domain of the expression. For example, the expression $\overline{ABC} + \overline{ABD} + \overline{ABC}D$ has a domain made up of the variables A , B , C , and D . However, notice that the complete set of variables in the domain is not represented in the first two terms of the expression; that is, D or \overline{D} is missing from the first term and C or \overline{C} is missing from the second term.

A *standard SOP expression* is one in which *all* the variables in the domain appear in each product term in the expression. For example, $\overline{ABCD} + \overline{ABCD} + \overline{ABC}D$ is a standard SOP expression. Standard SOP expressions are important in constructing truth tables, covered in Section 4-7, and in the Karnaugh map simplification method, which is covered in Section 4-8. Any nonstandard SOP expression (referred to simply as SOP) can be converted to the standard form using Boolean algebra.

Converting Product Terms to Standard SOP Each product term in an SOP expression that does not contain all the variables in the domain can be expanded to standard form to include all variables in the domain and their complements. As stated in the following steps, a nonstandard SOP expression is converted into standard form using Boolean algebra rule 6 ($A + \overline{A} = 1$) from Table 4-1: A variable added to its complement equals 1.

Step 1: Multiply each nonstandard product term by a term made up of the sum of a missing variable and its complement. This results in two product terms. As you know, you can multiply anything by 1 without changing its value.

Step 2: Repeat Step 1 until all resulting product terms contain all variables in the domain in either complemented or uncomplemented form. In converting a product term to standard form, the number of product terms is doubled for each missing variable, as Example 4-15 shows.

EXAMPLE 4-15

Convert the following Boolean expression into standard SOP form:

$$\overline{ABC} + \overline{AB} + ABC\overline{D}$$

Solution The domain of this SOP expression is A, B, C, D . Take one term at a time. The first term, \overline{ABC} , is missing variable D or \overline{D} , so multiply the first term by $D + \overline{D}$ as follows:

$$\overline{ABC} = \overline{ABC}(D + \overline{D}) = \overline{ABC}D + \overline{ABC}\overline{D}$$

In this case, two standard product terms are the result.

The second term, \overline{AB} , is missing variables C or \overline{C} and D or \overline{D} , so first multiply the second term by $C + \overline{C}$ as follows:

$$\overline{AB} = \overline{AB}(C + \overline{C}) = \overline{ABC} + \overline{ABC}\overline{C}$$

The two resulting terms are missing variable D or \overline{D} , so multiply both terms by $D + \overline{D}$ as follows:

$$\begin{aligned}\overline{AB} &= \overline{ABC} + \overline{ABC}\overline{C} = \overline{ABC}(D + \overline{D}) + \overline{ABC}\overline{C}(D + \overline{D}) \\ &= \overline{ABC}D + \overline{ABC}\overline{C}D + \overline{ABC}\overline{C}\overline{D} + \overline{ABC}\overline{C}\overline{D}\end{aligned}$$

In this case, four standard product terms are the result.

The third term, $ABC\overline{D}$, is already in standard form. The complete standard SOP form of the original expression is as follows:

$$\overline{ABC} + \overline{AB} + ABC\overline{D} = \overline{ABC}D + \overline{ABC}\overline{C}D + \overline{ABC}\overline{C}\overline{D} + \overline{ABC}\overline{C}\overline{D} + \overline{ABC}D + \overline{ABC}\overline{C}\overline{D} + ABC\overline{D}$$

Related Problem Convert the expression $\overline{W}XY + \overline{X}YZ + W\overline{X}Y$ to standard SOP form.

Binary Representation of a Standard Product Term A standard product term is equal to 1 for only one combination of variable values. For example, the product term $\overline{ABC}D$ is equal to 1 when $A = 1, B = 0, C = 1, D = 0$, as shown below, and is 0 for all other combinations of values for the variables.

$$\overline{ABC}D = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

In this case, the product term has a binary value of 1010 (decimal ten).

Remember, a product term is implemented with an AND gate whose output is 1 only if each of its inputs is 1. Inverters are used to produce the complements of the variables as required.

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

EXAMPLE 4-16

Determine the binary values for which the following standard SOP expression is equal to 1:

$$ABCD + \overline{AB}\overline{C}D + \overline{ABC}\overline{D}$$

Solution The term $ABCD$ is equal to 1 when $A = 1, B = 1, C = 1$, and $D = 1$.

$$ABCD = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

The term $\overline{AB}\overline{C}D$ is equal to 1 when $A = 1, B = 0, C = 0$, and $D = 1$.

$$\overline{AB}\overline{C}D = 1 \cdot \overline{0} \cdot \overline{0} \cdot 1 = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

The term $\overline{A}\overline{B}\overline{C}\overline{D}$ is equal to 1 when $A = 0$, $B = 0$, $C = 0$, and $D = 0$.

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

The SOP expression equals 1 when any or all of the three product terms is 1.

Related Problem Determine the binary values for which the following SOP expression is equal to 1:

$$\overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + \overline{X}Y\overline{Z} + XYZ$$

Is this a standard SOP expression?

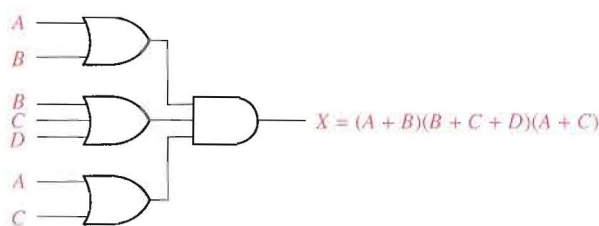
The Product-of-Sums (POS) Form

A sum term was defined in Section 4–1 as a term consisting of the sum (Boolean addition) of literals (variables or their complements). When two or more sum terms are multiplied, the resulting expression is a **product-of-sums (POS)**. Some examples are

$$\begin{aligned} &(\overline{A} + B)(A + \overline{B} + C) \\ &(\overline{A} + \overline{B} + \overline{C})(C + \overline{D} + E)(\overline{B} + C + D) \\ &(A + B)(A + \overline{B} + C)(\overline{A} + C) \end{aligned}$$

A POS expression can contain a single-variable term, as in $\overline{A}(A + \overline{B} + C)(\overline{B} + \overline{C} + D)$. In a POS expression, a single overbar cannot extend over more than one variable; however, more than one variable in a term can have an overbar. For example, a POS expression can have the term $\overline{A} + \overline{B} + \overline{C}$ but not $A + \overline{B} + \overline{C}$.

Implementation of a POS Expression Implementing a POS expression simply requires ANDing the outputs of two or more OR gates. A sum term is produced by an OR operation, and the product of two or more sum terms is produced by an AND operation. Therefore, a POS expression can be implemented by logic in which the outputs of a number (equal to the number of sum terms in the expression) of OR gates connect to the inputs of an AND gate, as Figure 4–22 shows for the expression $(A + B)(B + C + D)(A + C)$. The output X of the AND gate equals the POS expression.



◀ FIGURE 4–22

Implementation of the POS expression $(A + B)(B + C + D)(A + C)$.

The Standard POS Form

So far, you have seen POS expressions in which some of the sum terms do not contain all of the variables in the domain of the expression. For example, the expression

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

has a domain made up of the variables A , B , C , and D . Notice that the complete set of variables in the domain is not represented in the first two terms of the expression; that is, D or \overline{D} is missing from the first term and C or \overline{C} is missing from the second term.

A *standard POS expression* is one in which *all* the variables in the domain appear in each sum term in the expression. For example,

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

is a standard POS expression. Any nonstandard POS expression (referred to simply as POS) can be converted to the standard form using Boolean algebra.

Converting a Sum Term to Standard POS Each sum term in a POS expression that does not contain all the variables in the domain can be expanded to standard form to include all variables in the domain and their complements. As stated in the following steps, a nonstandard POS expression is converted into standard form using Boolean algebra rule 8 ($A \cdot \bar{A} = 0$) from Table 4-1: A variable multiplied by its complement equals 0.

- Step 1:** Add to each nonstandard product term a term made up of the product of the missing variable and its complement. This results in two sum terms. As you know, you can add 0 to anything without changing its value.
- Step 2:** Apply rule 12 from Table 4-1: $A + BC = (A + B)(A + C)$
- Step 3:** Repeat Step 1 until all resulting sum terms contain all variables in the domain in either complemented or uncomplemented form.

EXAMPLE 4-17

Convert the following Boolean expression into standard POS form:

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

Solution The domain of this POS expression is A, B, C, D . Take one term at a time. The first term, $A + \bar{B} + C$, is missing variable D or \bar{D} , so add $D\bar{D}$ and apply rule 12 as follows:

$$A + \bar{B} + C = A + \bar{B} + C + D\bar{D} = (A + \bar{B} + C + D)(A + \bar{B} + C + \bar{D})$$

The second term, $\bar{B} + C + \bar{D}$, is missing variable A or \bar{A} , so add $A\bar{A}$ and apply rule 12 as follows:

$$\bar{B} + C + \bar{D} = \bar{B} + C + \bar{D} + A\bar{A} = (A + \bar{B} + C + \bar{D})(\bar{A} + \bar{B} + C + \bar{D})$$

The third term, $A + \bar{B} + \bar{C} + D$, is already in standard form. The standard POS form of the original expression is as follows:

$$(A + \bar{B} + C)(\bar{B} + C + \bar{D})(A + \bar{B} + \bar{C} + D) = \\ (A + \bar{B} + C + D)(A + \bar{B} + C + \bar{D})(A + \bar{B} + C + \bar{D})(\bar{A} + \bar{B} + C + \bar{D})(A + \bar{B} + \bar{C} + D)$$

Related Problem Convert the expression $(A + \bar{B})(B + C)$ to standard POS form.

Binary Representation of a Standard Sum Term A standard sum term is equal to 0 for only one combination of variable values. For example, the sum term $A + \bar{B} + C + \bar{D}$ is 0 when $A = 0, B = 1, C = 0,$ and $D = 1$, as shown below, and is 1 for all other combinations of values for the variables.

$$A + \bar{B} + C + \bar{D} = 0 + \bar{1} + 0 + \bar{1} = 0 + 0 + 0 + 0 = 0$$

In this case, the sum term has a binary value of 0101 (decimal 5). Remember, a sum term is implemented with an OR gate whose output is 0 only if each of its inputs is 0. Inverters are used to produce the complements of the variables as required.

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

EXAMPLE 4–18

Determine the binary values of the variables for which the following standard POS expression is equal to 0:

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

Solution The term $A + B + C + D$ is equal to 0 when $A = 0$, $B = 0$, $C = 0$, and $D = 0$.

$$A + B + C + D = 0 + 0 + 0 + 0 = 0$$

The term $A + \bar{B} + \bar{C} + D$ is equal to 0 when $A = 0$, $B = 1$, $C = 1$, and $D = 0$.

$$A + \bar{B} + \bar{C} + D = 0 + \bar{1} + \bar{1} + 0 = 0 + 0 + 0 + 0 = 0$$

The term $\bar{A} + \bar{B} + \bar{C} + \bar{D}$ is equal to 0 when $A = 1$, $B = 1$, $C = 1$, and $D = 1$.

$$\bar{A} + \bar{B} + \bar{C} + \bar{D} = \bar{1} + \bar{1} + \bar{1} + \bar{1} = 0 + 0 + 0 + 0 = 0$$

The POS expression equals 0 when any of the three sum terms equals 0.

Related Problem Determine the binary values for which the following POS expression is equal to 0:

$$(X + \bar{Y} + Z)(\bar{X} + Y + Z)(X + Y + \bar{Z})(\bar{X} + \bar{Y} + \bar{Z})(X + \bar{Y} + \bar{Z})$$

Is this a standard POS expression?

Converting Standard SOP to Standard POS

The binary values of the product terms in a given standard SOP expression are not present in the equivalent standard POS expression. Also, the binary values that are not represented in the SOP expression are present in the equivalent POS expression. Therefore, to convert from standard SOP to standard POS, the following steps are taken:

- Step 1:** Evaluate each product term in the SOP expression. That is, determine the binary numbers that represent the product terms.
- Step 2:** Determine all of the binary numbers not included in the evaluation in Step 1.
- Step 3:** Write the equivalent sum term for each binary number from Step 2 and express in POS form.

Using a similar procedure, you can go from POS to SOP.

EXAMPLE 4–19

Convert the following SOP expression to an equivalent POS expression:

$$\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

Solution The evaluation is as follows:

$$000 + 010 + 011 + 101 + 111$$

Since there are three variables in the domain of this expression, there are a total of eight (2^3) possible combinations. The SOP expression contains five of these combinations, so the POS must contain the other three which are 001, 100, and 110. Remember, these are the binary values that make the sum term 0. The equivalent POS expression is

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

Related Problem Verify that the SOP and POS expressions in this example are equivalent by substituting binary values into each.

SECTION 4-6
CHECKUP

- Identify each of the following expressions as SOP, standard SOP, POS, or standard POS:
 - $AB + \overline{A}BD + \overline{A}C\overline{D}$
 - $(A + \overline{B} + C)(A + B + \overline{C})$
 - $\overline{A}BC + A\overline{B}C$
 - $(A + \overline{C})(A + B)$
- Convert each SOP expression in Question 1 to standard form.
- Convert each POS expression in Question 1 to standard form.

4-7 BOOLEAN EXPRESSIONS AND TRUTH TABLES

All standard Boolean expressions can be easily converted into truth table format using binary values for each term in the expression. The truth table is a common way of presenting, in a concise format, the logical operation of a circuit. Also, standard SOP or POS expressions can be determined from a truth table. You will find truth tables in data sheets and other literature related to the operation of digital circuits.

After completing this section, you should be able to

- Convert a standard SOP expression into truth table format
- Convert a standard POS expression into truth table format
- Derive a standard expression from a truth table
- Properly interpret truth table data

Converting SOP Expressions to Truth Table Format

Recall from Section 4-6 that an SOP expression is equal to 1 only if at least one of the product terms is equal to 1. A truth table is simply a list of the possible combinations of input variable values and the corresponding output values (1 or 0). For an expression with a domain of two variables, there are four different combinations of those variables ($2^2 = 4$). For an expression with a domain of three variables, there are eight different combinations of those variables ($2^3 = 8$). For an expression with a domain of four variables, there are sixteen different combinations of those variables ($2^4 = 16$), and so on.

The first step in constructing a truth table is to list all possible combinations of binary values of the variables in the expression. Next, convert the SOP expression to standard form if it is not already. Finally, place a 1 in the output column (X) for each binary value that makes the standard SOP expression a 1 and place a 0 for all the remaining binary values. This procedure is illustrated in Example 4-20.

EXAMPLE 4-20

Develop a truth table for the standard SOP expression $\overline{\overline{A}BC} + \overline{A}\overline{\overline{B}C} + ABC$.

Solution

There are three variables in the domain, so there are eight possible combinations of binary values of the variables as listed in the left three columns of Table 4-6. The binary values that make the product terms in the expressions equal to 1 are $\overline{\overline{A}BC}$: 001; $\overline{A}\overline{\overline{B}C}$: 100; and ABC : 111. For each of these binary values, place a 1 in the output column as shown in the table. For each of the remaining binary combinations, place a 0 in the output column.

TABLE 4-6

INPUTS			OUTPUT	PRODUCT TERM
A	B	C	X	
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

Related Problem Create a truth table for the standard SOP expression $\overline{A}\overline{B}C + A\overline{B}\overline{C}$.

Converting POS Expressions to Truth Table Format

Recall that a POS expression is equal to 0 only if at least one of the sum terms is equal to 0. To construct a truth table from a POS expression, list all the possible combinations of binary values of the variables just as was done for the SOP expression. Next, convert the POS expression to standard form if it is not already. Finally, place a 0 in the output column (X) for each binary value that makes the expression a 0 and place a 1 for all the remaining binary values. This procedure is illustrated in Example 4-21.

EXAMPLE 4-21

Determine the truth table for the following standard POS expression:

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

Solution There are three variables in the domain and the eight possible binary values are listed in the left three columns of Table 4-7. The binary values that make the sum terms in the expression equal to 0 are $A + B + C$: 000; $A + \overline{B} + C$: 010; $A + \overline{B} + \overline{C}$: 011; $\overline{A} + B + \overline{C}$: 101; and $\overline{A} + \overline{B} + C$: 110. For each of these binary values, place a 0 in the output column as shown in the table. For each of the remaining binary combinations, place a 1 in the output column.

TABLE 4-7

INPUTS			OUTPUT	SUM TERM
A	B	C	X	
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	

Notice that the truth table in this example is the same as the one in Example 4–20. This means that the SOP expression in the previous example and the POS expression in this example are equivalent.

Related Problem Develop a truth table for the following standard POS expression:

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

Determining Standard Expressions from a Truth Table

To determine the standard SOP expression represented by a truth table, list the binary values of the input variables for which the output is 1. Convert each binary value to the corresponding product term by replacing each 1 with the corresponding variable and each 0 with the corresponding variable complement. For example, the binary value 1010 is converted to a product term as follows:

$$1010 \longrightarrow A\bar{B}C\bar{D}$$

If you substitute, you can see that the product term is 1:

$$A\bar{B}C\bar{D} = 1 \cdot \bar{0} \cdot 1 \cdot \bar{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

To determine the standard POS expression represented by a truth table, list the binary values for which the output is 0. Convert each binary value to the corresponding sum term by replacing each 1 with the corresponding variable complement and each 0 with the corresponding variable. For example, the binary value 1001 is converted to a sum term as follows:

$$1001 \longrightarrow \bar{A} + B + C + \bar{D}$$

If you substitute, you can see that the sum term is 0:

$$\bar{A} + B + C + \bar{D} = \bar{1} + 0 + 0 + \bar{1} = 0 + 0 + 0 + 0 = 0$$

EXAMPLE 4–22

From the truth table in Table 4–8, determine the standard SOP expression and the equivalent standard POS expression.

TABLE 4–8

INPUTS			OUTPUT
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Solution There are four 1s in the output column and the corresponding binary values are 011, 100, 110, and 111. Convert these binary values to product terms as follows:

$$011 \longrightarrow \bar{A}BC$$

$$100 \longrightarrow A\bar{B}\bar{C}$$

$$110 \longrightarrow A\bar{B}C$$

$$111 \longrightarrow ABC$$

The resulting standard SOP expression for the output X is

$$X = \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C + ABC$$

For the POS expression, the output is 0 for binary values 000, 001, 010, and 101. Convert these binary values to sum terms as follows:

$$000 \longrightarrow A + B + C$$

$$001 \longrightarrow A + B + \bar{C}$$

$$010 \longrightarrow A + \bar{B} + C$$

$$101 \longrightarrow \bar{A} + B + \bar{C}$$

The resulting standard POS expression for the output X is

$$X = (A + B + C)(A + B + \bar{C})(A + \bar{B} + C)(\bar{A} + B + \bar{C})$$

Related Problem By substitution of binary values, show that the SOP and the POS expressions derived in this example are equivalent; that is, for any binary value each SOP and POS term should either both be 1 or both be 0, depending on the binary value.

SECTION 4-7 CHECKUP

1. If a certain Boolean expression has a domain of five variables, how many binary values will be in its truth table?
2. In a certain truth table, the output is a 1 for the binary value 0110. Convert this binary value to the corresponding product term using variables W , X , Y , and Z .
3. In a certain truth table, the output is a 0 for the binary value 1100. Convert this binary value to the corresponding sum term using variables W , X , Y , and Z .

4-8 THE KARNAUGH MAP

A Karnaugh map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest SOP or POS expression possible, known as the minimum expression. As you have seen, the effectiveness of algebraic simplification depends on your familiarity with all the laws, rules, and theorems of Boolean algebra and on your ability to apply them. The Karnaugh map, on the other hand, provides a “cookbook” method for simplification. Other simplification techniques include the Quine-McClusky method and the Espresso algorithm.

After completing this section, you should be able to

- ♦ Construct a Karnaugh map for three or four variables
- ♦ Determine the binary value of each cell in a Karnaugh map
- ♦ Determine the standard product term represented by each cell in a Karnaugh map
- ♦ Explain cell adjacency and identify adjacent cells

The purpose of a Karnaugh map is to simplify a Boolean expression.

A **Karnaugh map** is similar to a truth table because it presents all of the possible values of input variables and the resulting output for each value. Instead of being organized into columns and rows like a truth table, the Karnaugh map is an array of **cells** in which each cell represents a binary value of the input variables. The cells are arranged in a way so that simplification of a given expression is simply a matter of properly grouping the cells. Karnaugh maps can be used for expressions with two, three, four, and five variables, but we will discuss only 3-variable and 4-variable situations to illustrate the principles. Section 4–10 deals with five variables using a 32-cell Karnaugh map.

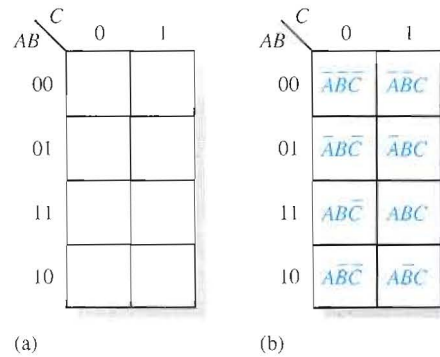
The number of cells in a Karnaugh map, as well as the number of rows in a truth table, is equal to the total number of possible input variable combinations. For three variables, the number of cells is $2^3 = 8$. For four variables, the number of cells is $2^4 = 16$.

The 3-Variable Karnaugh Map

The 3-variable Karnaugh map is an array of eight cells, as shown in Figure 4–23(a). In this case, A , B , and C are used for the variables although other letters could be used. Binary values of A and B are along the left side (notice the sequence) and the values of C are across the top. The value of a given cell is the binary values of A and B at the left in the same row combined with the value of C at the top in the same column. For example, the cell in the upper left corner has a binary value of 000 and the cell in the lower right corner has a binary value of 101. Figure 4–23(b) shows the standard product terms that are represented by each cell in the Karnaugh map.

► FIGURE 4–23

A 3-variable Karnaugh map showing product terms.



The 4-Variable Karnaugh Map

The 4-variable Karnaugh map is an array of sixteen cells, as shown in Figure 4–24(a). Binary values of A and B are along the left side and the values of C and D are across the top. The value of a given cell is the binary values of A and B at the left in the same row combined with the binary values of C and D at the top in the same column. For example, the cell in the upper right corner has a binary value of 0010 and the cell in the lower right corner has a binary value of 1010. Figure 4–24(b) shows the standard product terms that are represented by each cell in the 4-variable Karnaugh map.

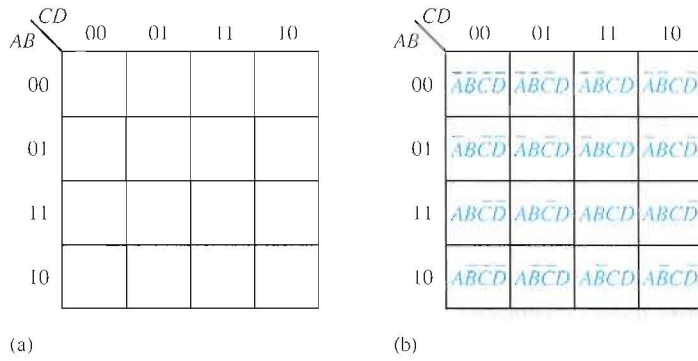
Cell Adjacency

The cells in a Karnaugh map are arranged so that there is only a single-variable change between adjacent cells. **Adjacency** is defined by a single-variable change. In the 3-variable map the 010 cell is adjacent to the 000 cell, the 011 cell, and the 110 cell. The 010 cell is not adjacent to the 001 cell, the 111 cell, the 100 cell, or the 101 cell.

Physically, each cell is adjacent to the cells that are immediately next to it on any of its four sides. A cell is not adjacent to the cells that diagonally touch any of its corners. Also,

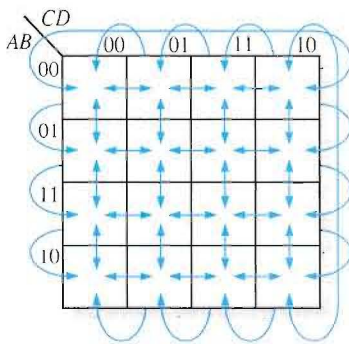
Cells that differ by only one variable are adjacent.

Cells with values that differ by more than one variable are not adjacent.



◀ **FIGURE 4-24**
A 4-variable Karnaugh map.

the cells in the top row are adjacent to the corresponding cells in the bottom row and the cells in the outer left column are adjacent to the corresponding cells in the outer right column. This is called “wrap-around” adjacency because you can think of the map as wrapping around from top to bottom to form a cylinder or from left to right to form a cylinder. Figure 4-25 illustrates the cell adjacencies with a 4-variable map, although the same rules for adjacency apply to Karnaugh maps with any number of cells.



◀ **FIGURE 4-25**
Adjacent cells on a Karnaugh map are those that differ by only one variable. Arrows point between adjacent cells.

The Quine-McClusky Method

Minimizing Boolean functions using Karnaugh maps is not applicable for more than five variables and practical only for up to four variables. Also, this method does not lend itself to be automated in the form of a computer program.

The Quine-McClusky method is more practical for logic simplification of functions with more than four or five variables. It also has the advantage of being easily implemented with a computer or programmable calculator.

The Quine-McClusky method is functionally similar to Karnaugh mapping, but the tabular form makes it more efficient for use in computer algorithms, and it also gives a way to check that the minimal form of a Boolean function has been reached. This method is sometimes referred to as the *tabulation method*. An introduction to the Quine-McClusky method is provided in Appendix C.

Espresso Algorithm

Although the Quine-McClusky method is well suited to be implemented in a computer program and can handle more variables than the Karnaugh map method, the result is still far from efficient in terms of processing time and memory usage. Adding a variable to the function will roughly double both of these parameters because the truth table length increases exponentially with the number of variables. Functions with a large number of variables have to be minimized with other methods such as the Espresso logic minimizer, which has become the de facto world standard.

Compared to the other methods, Espresso is essentially more efficient in terms of reducing memory usage and computation time by several orders of magnitude. There is essentially no restrictions to the number of variables, output functions, and product terms of a combinational logic function. In general, tens of variables with tens of output functions can be handled by Espresso.

The Espresso algorithm has been incorporated as a standard logic function minimization step in most logic synthesis tools for programmable logic devices. For implementing a function in multilevel logic, the minimization result is optimized by factorization and mapped onto the available basic logic cells in the target device, such as an FPGA (Field-Programmable Gate Array).

SECTION 4–8 CHECKUP

- In a 3-variable Karnaugh map, what is the binary value for the cell in each of the following locations:
 - upper left corner
 - lower right corner
 - lower left corner
 - upper right corner
- What is the standard product term for each cell in Question 1 for variables X , Y , and Z ?
- Repeat Question 1 for a 4-variable map.
- Repeat Question 2 for a 4-variable map using variables W , X , Y , and Z .

4–9 KARNAUGH MAP SOP MINIMIZATION

As stated in the last section, the Karnaugh map is used for simplifying Boolean expressions to their minimum form. A minimized SOP expression contains the fewest possible terms with the fewest possible variables per term. Generally, a minimum SOP expression can be implemented with fewer logic gates than a standard expression.

After completing this section, you should be able to

- ♦ Map a standard SOP expression on a Karnaugh map
- ♦ Combine the 1s on the map into maximum groups
- ♦ Determine the minimum product term for each group on the map
- ♦ Combine the minimum product terms to form a minimum SOP expression
- ♦ Convert a truth table into a Karnaugh map for simplification of the represented expression
- ♦ Use “don’t care” conditions on a Karnaugh map

Mapping a Standard SOP Expression

For an SOP expression in standard form, a 1 is placed on the Karnaugh map for each product term in the expression. Each 1 is placed in a cell corresponding to the value of a product term. For example, for the product term $A\bar{B}C$, a 1 goes in the 101 cell on a 3-variable map.

When an SOP expression is completely mapped, there will be a number of 1s on the Karnaugh map equal to the number of product terms in the standard SOP expression. The cells that do not have a 1 are the cells for which the expression is 0. Usually, when work-

ing with SOP expressions, the 0s are left off the map. The following steps and the illustration in Figure 4–26 show the mapping process.

- Step 1:** Determine the binary value of each product term in the standard SOP expression. After some practice, you can usually do the evaluation of terms mentally.
- Step 2:** As each product term is evaluated, place a 1 on the Karnaugh map in the cell having the same value as the product term.

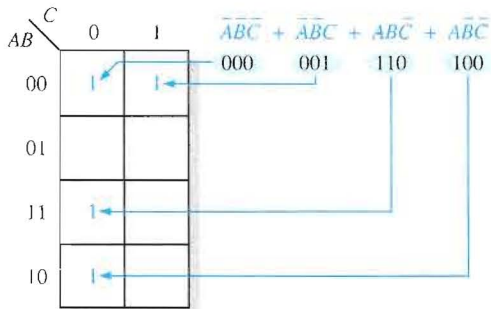


FIGURE 4–26
Example of mapping a standard SOP expression.

EXAMPLE 4–23

Map the following standard SOP expression on a Karnaugh map:

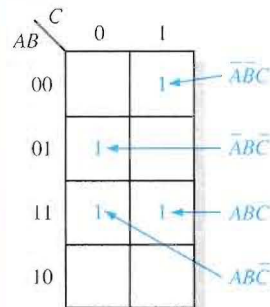
$$\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

Solution Evaluate the expression as shown below. Place a 1 on the 3-variable Karnaugh map in Figure 4–27 for each standard product term in the expression.

$$\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

001 010 110 111

FIGURE 4–27



Related Problem Map the standard SOP expression $\bar{A}BC + \bar{A}B\bar{C} + \bar{A}B\bar{C}$ on a Karnaugh map.

EXAMPLE 4–24

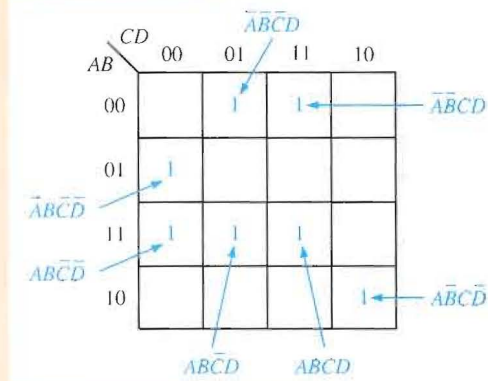
Map the following standard SOP expression on a Karnaugh map:

$$\bar{A}\bar{B}CD + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + ABCD + A\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D}$$

Solution Evaluate the expression as shown below. Place a 1 on the 4-variable Karnaugh map in Figure 4–28 for each standard product term in the expression.

$$\bar{A}\bar{B}CD + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + ABCD + A\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D}$$

0011 0100 1101 1111 1100 0001 1010



▲ FIGURE 4-28

Related Problem Map the following standard SOP expression on a Karnaugh map:

$$\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D}$$

Mapping a Nonstandard SOP Expression

A Boolean expression must first be in standard form before you use a Karnaugh map. If an expression is not in standard form, then it must be converted to standard form by the procedure covered in Section 4-6 or by numerical expansion. Since an expression should be evaluated before mapping anyway, numerical expansion is probably the most efficient approach.

Numerical Expansion of a Nonstandard Product Term Recall that a nonstandard product term has one or more missing variables. For example, assume that one of the product terms in a certain 3-variable SOP expression is $\bar{A}\bar{B}$. This term can be expanded numerically to standard form as follows. First, write the binary value of the two variables and attach a 0 for the missing variable \bar{C} : 100. Next, write the binary value of the two variables and attach a 1 for the missing variable C : 101. The two resulting binary numbers are the values of the standard SOP terms $\bar{A}\bar{B}\bar{C}$ and $\bar{A}\bar{B}C$.

As another example, assume that one of the product terms in a 3-variable expression is B (remember that a single variable counts as a product term in an SOP expression). This term can be expanded numerically to standard form as follows. Write the binary value of the variable; then attach all possible values for the missing variables A and C as follows:

- B
- 010
- 011
- 110
- 111

The four resulting binary numbers are the values of the standard SOP terms $\bar{A}\bar{B}\bar{C}$, $\bar{A}\bar{B}C$, $A\bar{B}\bar{C}$, and $A\bar{B}C$.

EXAMPLE 4–25

Map the following SOP expression on a Karnaugh map: $\bar{A} + \bar{A}\bar{B} + ABC\bar{C}$.

Solution The SOP expression is obviously not in standard form because each product term does not have three variables. The first term is missing two variables, the second term is missing one variable, and the third term is standard. First expand the terms numerically as follows:

$$\begin{array}{l} \bar{A} + \bar{A}\bar{B} + ABC\bar{C} \\ 000 \quad 100 \quad 110 \\ 001 \quad 101 \\ 010 \\ 011 \end{array}$$

Map each of the resulting binary values by placing a 1 in the appropriate cell of the 3-variable Karnaugh map in Figure 4–29.

▶ **FIGURE 4–29**

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

Related Problem Map the SOP expression $BC + \bar{A}\bar{C}$ on a Karnaugh map.

EXAMPLE 4–26

Map the following SOP expression on a Karnaugh map:

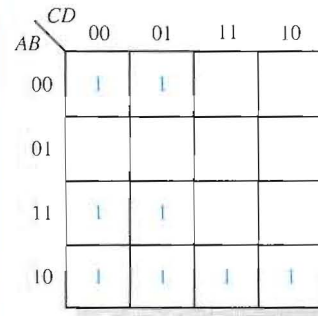
$$\bar{B}\bar{C} + \bar{A}\bar{B} + \bar{A}BC\bar{C} + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}C\bar{D}$$

Solution The SOP expression is obviously not in standard form because each product term does not have four variables. The first and second terms are both missing two variables, the third term is missing one variable, and the rest of the terms are standard. First expand the terms by including all combinations of the missing variables numerically as follows:

$$\begin{array}{l} \bar{B}\bar{C} + \bar{A}\bar{B} + \bar{A}BC\bar{C} + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}C\bar{D} \\ 0000 \quad 1000 \quad 1100 \quad 1010 \quad 0001 \quad 1011 \\ 0001 \quad 1001 \quad 1101 \\ 1000 \quad 1010 \\ 1001 \quad 1011 \end{array}$$

Map each of the resulting binary values by placing a 1 in the appropriate cell of the 4-variable Karnaugh map in Figure 4–30. Notice that some of the values in the expanded expression are redundant.

▶ FIGURE 4-30



Related Problem Map the expression $A + \overline{CD} + A\overline{C}\overline{D} + \overline{A}BC\overline{D}$ on a Karnaugh map.

Karnaugh Map Simplification of SOP Expressions

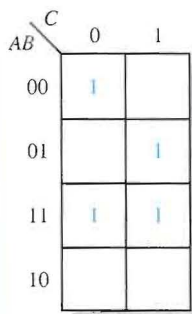
The process that results in an expression containing the fewest possible terms with the fewest possible variables is called **minimization**. After an SOP expression has been mapped, a minimum SOP expression is obtained by grouping the 1s and determining the minimum SOP expression from the map. (For simplification of POS expressions, see Appendix B.)

Grouping the 1s You can group 1s on the Karnaugh map according to the following rules by enclosing those adjacent cells containing 1s. The goal is to maximize the size of the groups and to minimize the number of groups.

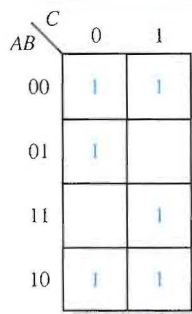
1. A group must contain either 1, 2, 4, 8, or 16 cells, which are all powers of two. In the case of a 3-variable map, $2^3 = 8$ cells is the maximum group.
2. Each cell in a group must be adjacent to one or more cells in that same group, but all cells in the group do not have to be adjacent to each other.
3. Always include the largest possible number of 1s in a group in accordance with rule 1.
4. Each 1 on the map must be included in at least one group. The 1s already in a group can be included in another group as long as the overlapping groups include non-common 1s.

EXAMPLE 4-27

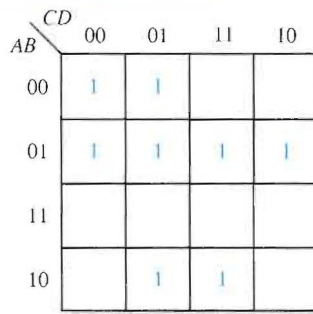
Group the 1s in each of the Karnaugh maps in Figure 4-31.



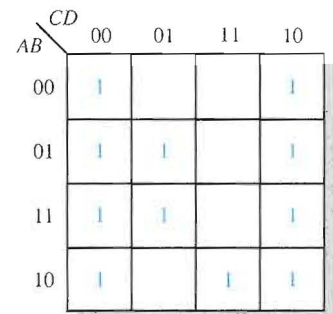
(a)



(b)



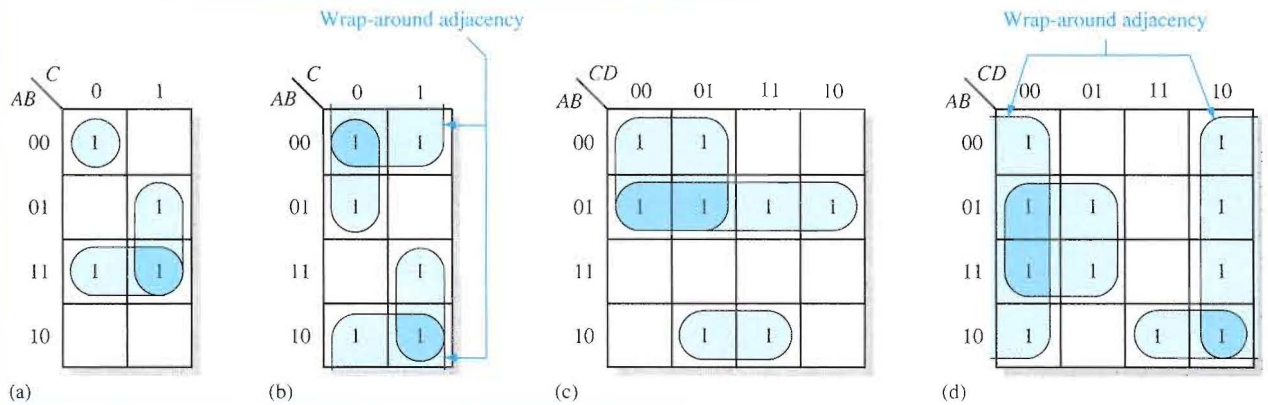
(c)



(d)

▶ FIGURE 4-31

Solution The groupings are shown in Figure 4–32. In some cases, there may be more than one way to group the 1s to form maximum groupings.



▲ FIGURE 4–32

Related Problem Determine if there are other ways to group the 1s in Figure 4–32 to obtain a minimum number of maximum groupings.

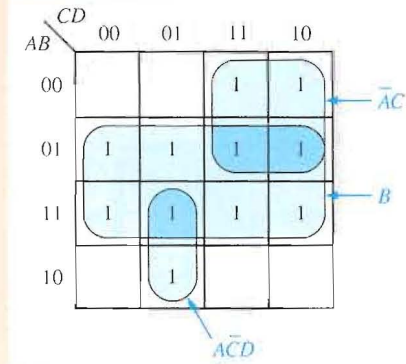
Determining the Minimum SOP Expression from the Map When all the 1s representing the standard product terms in an expression are properly mapped and grouped, the process of determining the resulting minimum SOP expression begins. The following rules are applied to find the minimum product terms and the minimum SOP expression:

1. Group the cells that have 1s. Each group of cells containing 1s creates one product term composed of all variables that occur in only one form (either uncomplemented or complemented) within the group. Variables that occur both uncomplemented and complemented within the group are eliminated. These are called *contradictory variables*.
2. Determine the minimum product term for each group.
 - a. For a 3-variable map:
 - (1) A 1-cell group yields a 3-variable product term
 - (2) A 2-cell group yields a 2-variable product term
 - (3) A 4-cell group yields a 1-variable term
 - (4) An 8-cell group yields a value of 1 for the expression
 - b. For a 4-variable map:
 - (1) A 1-cell group yields a 4-variable product term
 - (2) A 2-cell group yields a 3-variable product term
 - (3) A 4-cell group yields a 2-variable product term
 - (4) An 8-cell group yields a 1-variable term
 - (5) A 16-cell group yields a value of 1 for the expression
3. When all the minimum product terms are derived from the Karnaugh map, they are summed to form the minimum SOP expression.

EXAMPLE 4–28

Determine the product terms for the Karnaugh map in Figure 4–33 and write the resulting minimum SOP expression.

▶ **FIGURE 4–33**



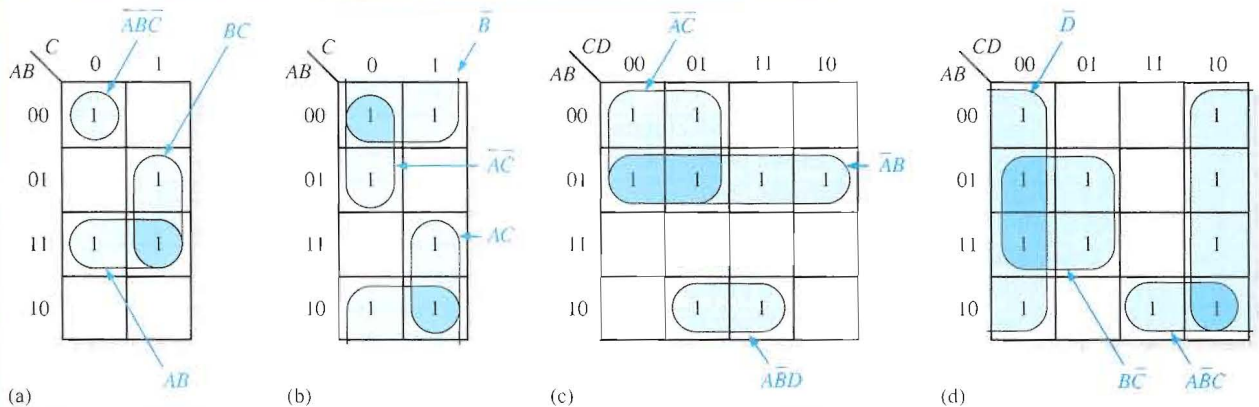
Solution Eliminate variables that are in a grouping in both complemented and uncomplemented forms. In Figure 4–33, the product term for the 8-cell group is B because the cells within that group contain both A and \overline{A} , C and \overline{C} , and D and \overline{D} , which are eliminated. The 4-cell group contains B , \overline{B} , D , and \overline{D} , leaving the variables \overline{A} and C , which form the product term \overline{AC} . The 2-cell group contains B and \overline{B} , leaving variables A , \overline{C} , and D which form the product term \overline{ACD} . Notice how overlapping is used to maximize the size of the groups. The resulting minimum SOP expression is the sum of these product terms:

$$B + \overline{AC} + \overline{ACD}$$

Related Problem For the Karnaugh map in Figure 4–33, add a 1 in the lower right cell (1010) and determine the resulting SOP expression.

EXAMPLE 4–29

Determine the product terms for each of the Karnaugh maps in Figure 4–34 and write the resulting minimum SOP expression.



▶ **FIGURE 4–34**