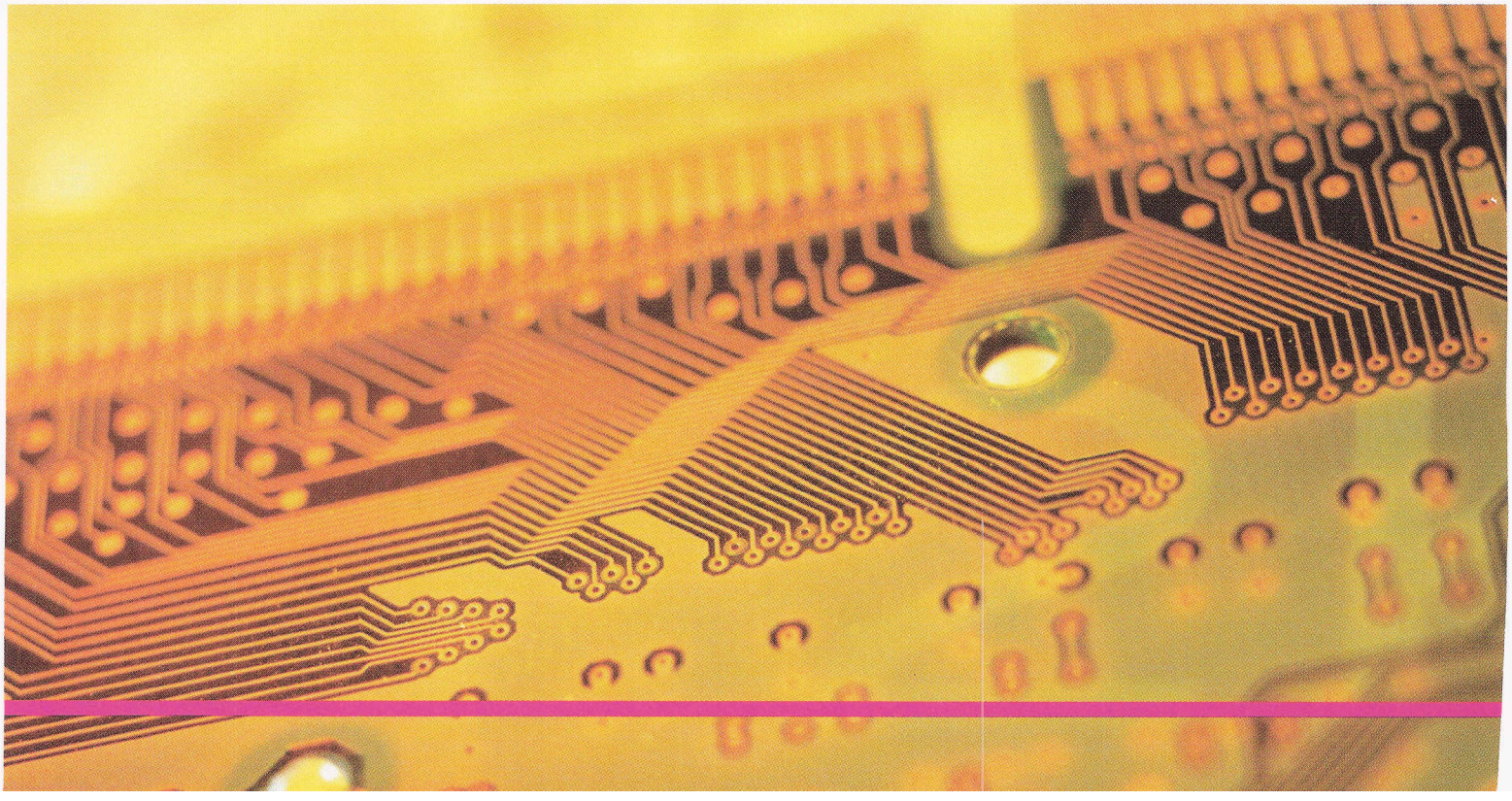
4

BOOLEAN ALGEBRA AND LOGIC SIMPLIFICATION

CHAPTER OUTLINE



CHAPTER OBJECTIVES

- Apply the basic laws and rules of Boolean algebra
- Apply DeMorgan's theorems to Boolean expressions
- Describe gate networks with Boolean expressions
- Evaluate Boolean expressions
- Simplify expressions by using the laws and rules of Boolean algebra
- Convert any Boolean expression into a sum-of-products (SOP) form
- Convert any Boolean expression into a product of-sums (POS) form
- Use a Karnaugh map to simplify Boolean expressions
- Use a Karnaugh map to simplify truth table functions
- Utilize "don't care" conditions to simplify logic functions
- Write a VHDL program for simple logic
- Apply Boolean algebra, the Karnaugh map method, and VHDL to a system application

KEY TERMS

- Variable
- Complement
- Sum term
- Product term
- Sum-of-products (SOP)
- Product-of-sums (POS)
- Karnaugh map
- Minimization
- "Don't care"
- VHDL

INTRODUCTION

In 1854, George Boole published a work titled *An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities.* It was in this publication that a "logical algebra," known today as Boolean algebra, was formulated. Boolean algebra is a convenient and systematic way of expressing and analyzing the operation of logic circuits. Claude Shannon was the first to apply Boole's work to the analysis and design of logic circuits. In 1938, Shannon wrote a thesis at MIT titled *A Symbolic Analysis of Relay and Switching Circuits.*

This chapter covers the laws, rules, and theorems of Boolean algebra and their application to digital circuits. You will learn how to define a given circuit with a Boolean expression and then evaluate its operation. You will also learn how to simplify logic circuits using the methods of Boolean algebra and Karnaugh maps.

The hardware description language VHDL for programming logic devices is introduced.

DIGITAL SYSTEM APPLICATION PREVIEW

The Digital System Application illustrates concepts taught in this chapter. The 7-segment display logic in the tablet counting and control system from Chapter 1 is a good way to illustrate the application of Boolean algebra and the Karnaugh map method to obtain the simplest possible implementation in the design of logic circuits. Therefore, in this digital system application, the focus is on the BCD to 7-segment logic that drives the two system displays shown in Figure 1–58.

WWW. VISIT THE COMPANION WEBSITE

Study aids for this chapter are available at

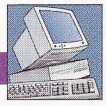
http://www.prenhall.com/floyd

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



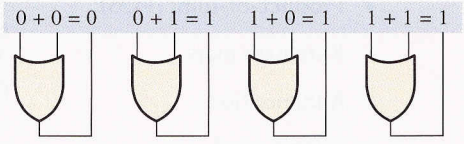
COMPUTER NOTE

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.

Variable, complement, and *literal* are terms used in Boolean algebra. A **variable** is a symbol (usually an italic uppercase letter) used to represent a logical quantity. Any single variable can have 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 $\overline{A}$. If $A = 1$, then $\overline{A} = 0$. If $A = 0$, then $\overline{A} = 1$. 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:



The OR gate is a Boolean adder.

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 $A + B$, $A + \overline{B}$, $A + B + \overline{C}$, and $\overline{A} + B + C + \overline{D}$.

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.

EXAMPLE 4–1

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

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

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

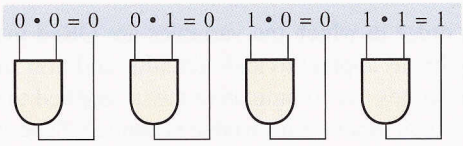
Related Problem* Determine the values of A and B that make the sum term $\overline{A} + B$ equal to 0.

*Answers are at the end of the chapter.

Boolean 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:

The AND gate is a Boolean multiplier.



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, $A\overline{B}$, ABC, and $A\overline{B}C\overline{D}$.

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 $A\overline{B}C\overline{D}$ equal to 1.

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

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

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

SECTION 4–1 REVIEW

Answers are at the end of the chapter.

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

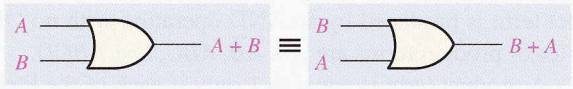
Equation 4–1

$$A + B = B + A$$

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 ≡ means "equivalent to.")

▶ **FIGURE 4–1**

Application of commutative law of addition.



The *commutative law of multiplication* for two variables is

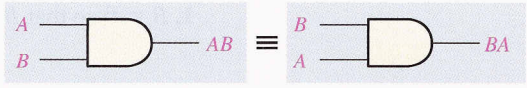
Equation 4–2

$$AB = BA$$

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:

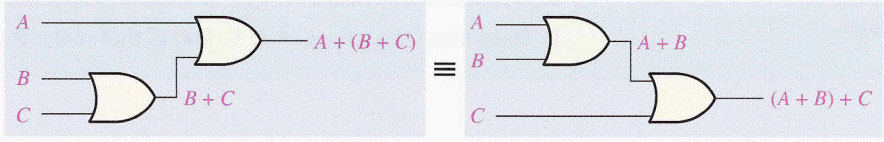
Equation 4–3

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

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:

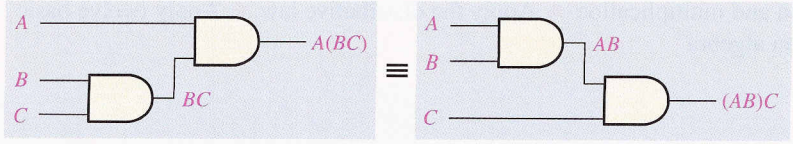
Equation 4–4

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

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.

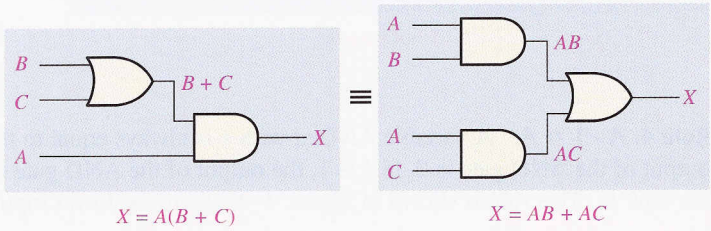


Distributive Law The distributive law is written for three variables as follows:

Equation 4–5

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

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.

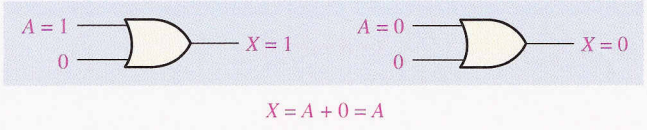
◀ **TABLE 4–1**

Basic rules of Boolean algebra.

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

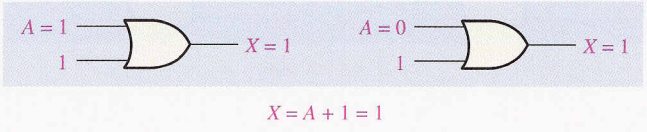
A, B, or *C* can represent a single variable or a combination of variables.

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**

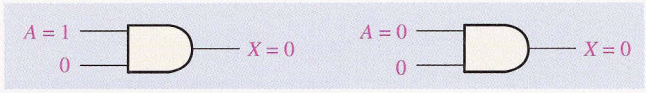
Rule 2. $A + 1 = 1$ A variable ORed with 1 is always equal to 1. A 1 on an input to an OR gate produces a 1 on the output, regardless of the value of the variable on the other input. This rule is illustrated in Figure 4–7, where the lower input is fixed at 1.



◀ **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.

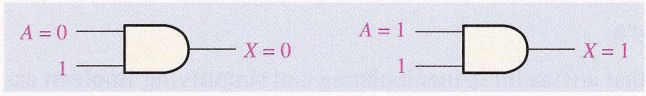
▶ **FIGURE 4–8**



$X = A \cdot 0 = 0$

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.

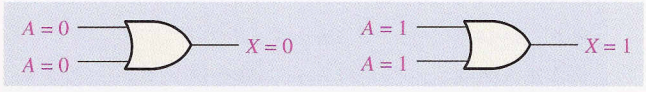
▶ **FIGURE 4–9**



$X = A \cdot 1 = A$

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.

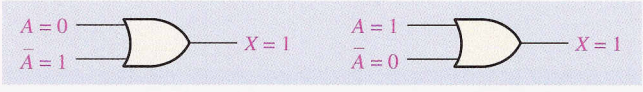
▶ **FIGURE 4–10**



$X = A + A = A$

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.

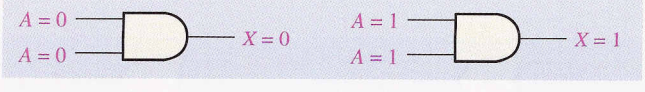
▶ **FIGURE 4–11**



$X = A + \bar{A} = 1$

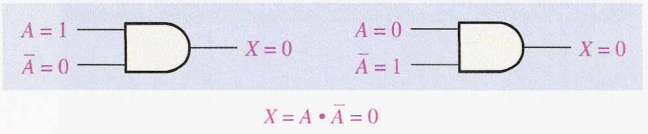
Rule 7. $A \cdot A = A$ A variable ANDed with itself is always equal to the variable. If $A = 0$, then $0 \cdot 0 = 0$; and if $A = 1$, then $1 \cdot 1 = 1$. Figure 4–12 illustrates this rule.

▶ **FIGURE 4–12**



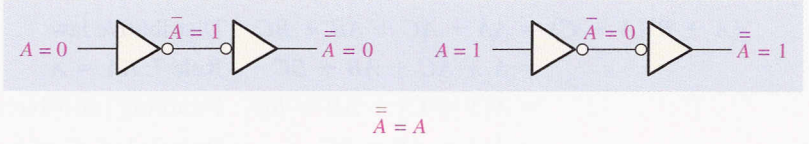
$X = A \cdot A = A$

Rule 8. $A \cdot \overline{A} = 0$ A variable ANDed with its complement is always equal to 0. Either A or $\overline{A}$ 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. $\overline{\overline{A}} = A$ 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 $\overline{A}$. If you then take $\overline{A}$ 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

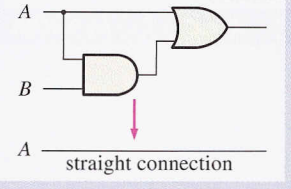
Rule 10. $A + AB = A$ This rule can be proved by applying the distributive law, rule 2, and rule 4 as follows:

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

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

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

equal



◀ TABLE 4–2

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

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

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

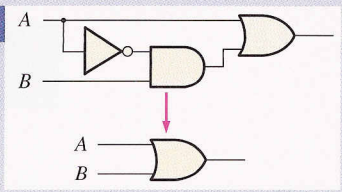
The proof is shown in Table 4–3, which shows the truth table and the resulting logic circuit simplification.

▶ **TABLE 4–3**

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

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



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

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

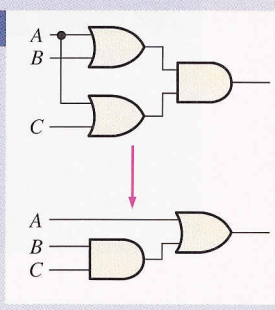
The proof is shown in Table 4–4, which shows the truth table and the resulting logic circuit simplification.

▼ **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 REVIEW

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

One of DeMorgan's theorems 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

$$\overline{XY} = \overline{X} + \overline{Y}$$ Equation 4–6

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.

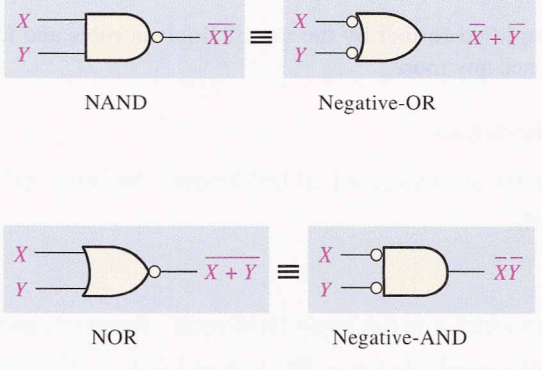
Stated another way,

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

$$\overline{X + Y} = \overline{X}\,\overline{Y}$$ Equation 4–7

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



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

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

◀ **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.

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

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

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

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

$$\overline{WXYZ} = \overline{W} + \overline{X} + \overline{Y} + \overline{Z}$$

$$\overline{W + X + Y + Z} = \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$$

Related Problem Apply DeMorgan's theorem to the expression $\overline{\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}}$.

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 $\overline{X + Y} = \overline{X}\,\overline{Y}$ individually, as follows:

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

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 $\overline{BC}$. A final application of DeMorgan's theorem gives the following result:

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

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

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

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

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

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

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).

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

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

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

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

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

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)** $\overline{A\overline{B} + \overline{C}D + EF}$

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

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

(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 $A\overline{B} = X$, $\overline{C}D = Y$, and $EF = Z$. The expression $\overline{A\overline{B} + \overline{C}D + EF}$ is of the form $\overline{X + Y + Z} = \overline{X}\,\overline{Y}\,\overline{Z}$ and can be rewritten as

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

Next, apply DeMorgan's theorem to each of the terms $\overline{A\overline{B}}$, $\overline{\overline{C}D}$, and $\overline{EF}$.

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

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

EXAMPLE 4–6

Apply DeMorgan's theorems to each expression:

(a) $\overline{(\overline{A + B}) + \overline{C}}$ **(b)** $\overline{(\overline{A} + B) + CD}$ **(c)** $\overline{(A + B)\overline{C}\,\overline{D} + E + \overline{F}}$

Solution

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

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

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

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

EXAMPLE 4–7

The Boolean expression for an exclusive-OR gate is $A\overline{B} + \overline{A}B$. 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:

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

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

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

The final expression for the XNOR is $\overline{A}\,\overline{B} + AB$. 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 REVIEW

1. Apply DeMorgan's theorems to the following expressions:
   (a) $\overline{ABC} + (\overline{D} + E)$ **(b)** $\overline{(A + B)C}$ **(c)** $\overline{A + B + C + \overline{D}E}$

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

A logic circuit can be described by a Boolean equation.

To derive the Boolean expression for a given 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 as follows:

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 $B + CD$.
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.

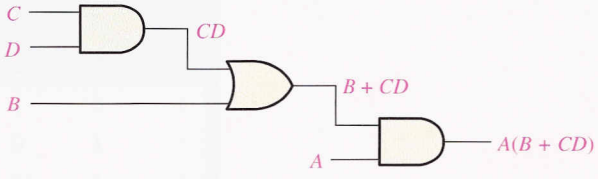


FIGURE 4–16

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

Constructing a Truth Table for a Logic Circuit

A logic circuit can be described by a truth table.

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

SECTION 4–4 REVIEW

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 4–8 through 4–11 illustrate Boolean simplification.

EXAMPLE 4–8

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.

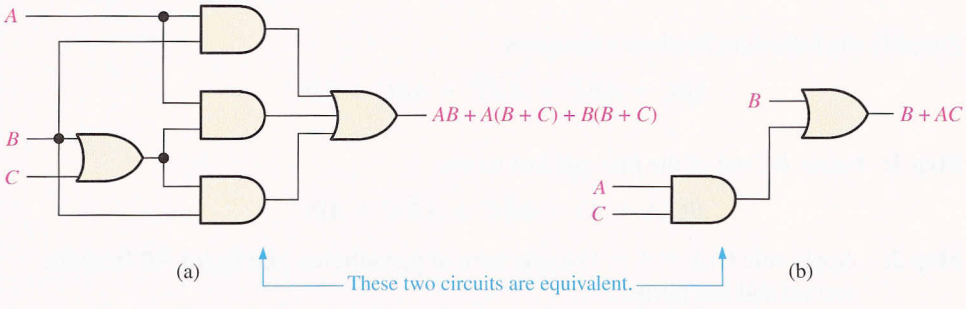
$$B + AC$$

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

Figure 4–17 shows that the simplification process in Example 4–8 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.

Simplification means fewer gates for the same function.



▲ **FIGURE 4–17**

Gate circuits for Example 4–8. Open file F04-17 to verify equivalency.

EXAMPLE 4–9

Simplify the following Boolean expression:

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

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.

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

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

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

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

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

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

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

Step 5: Apply the distributive law.

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

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

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

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 $[AB(C + \overline{BD}) + \overline{AB}]CD$.

EXAMPLE 4–10

Simplify the following Boolean expression:

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

Solution

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

EXAMPLE 4–11

Simplify the following Boolean expression:

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

Solution

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

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

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

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

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

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

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

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

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

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

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

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

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

SECTION 4–5 REVIEW

1. Simplify the following Boolean expressions if possible:
   (a) $A + AB + A\overline{B}C$ (b) $(\overline{A} + B)C + ABC$ (c) $A\overline{B}C(BD + CDE) + A\overline{C}$
2. 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

An SOP expression can be implemented with one OR and two or more ANDs.

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

$$AB + ABC$$
$$ABC + CDE + \overline{B}C\overline{D}$$
$$\overline{A}B + \overline{A}B\overline{C} + AC$$

Also, an SOP expression can contain a single-variable term, as in $A + \overline{A}\,\overline{B}C + BC\overline{D}$. 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 $\overline{A}\,\overline{B}\,\overline{C}$ 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 $\overline{A}B + A\overline{B}C$ is the set of variables A, B, C and the domain of the expression $AB\overline{C} + C\overline{D}E + \overline{B}C\overline{D}$ 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 4–18 for the expression $AB + BCD + AC$. The output X of the OR gate equals the SOP expression.

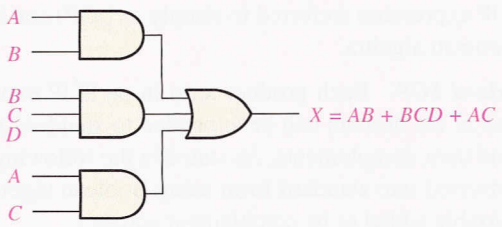


FIGURE 4–18

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. Using only NAND gates, an AND/OR function can be accomplished, as illustrated in Figure 4–19. 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.

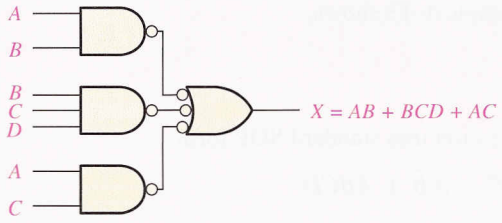


FIGURE 4–19

This NAND/NAND implementation is equivalent to the AND/OR in Figure 4–18.

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 4–12

Convert each of the following Boolean expressions to SOP form:

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

Solution

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

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

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

Related Problem Convert $\overline{A}B\overline{C} + (A + \overline{B})(B + \overline{C} + A\overline{B})$ 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 $\overline{A}B\overline{C} + A\overline{B}D + \overline{A}B\overline{C}D$ 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, $A\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + AB\overline{C}\,\overline{D}$ 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 4–13 shows.

EXAMPLE 4–13

Convert the following Boolean expression into standard SOP form:

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

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

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

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

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

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

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

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

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

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

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

Related Problem Convert the expression $W\overline{X}Y + \overline{X}Y\overline{Z} + WX\overline{Y}$ 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 $A\overline{B}C\overline{D}$ 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.

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

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 4–14

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

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

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

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

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

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

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

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 4–20 shows for the expression $(A + B)(B + C + D)(A + C)$. The output X of the AND gate equals the POS expression.

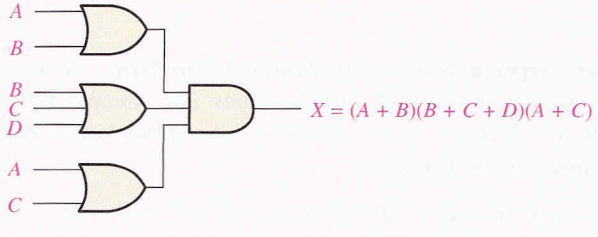


FIGURE 4–20

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,

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

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 ($A \cdot \overline{A} = 0$) 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 4–15

Convert the following Boolean expression into standard POS form:

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

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

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

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

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

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

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

Related Problem Convert the expression $(A + \overline{B})(B + C)$ 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 $A + \overline{B} + C + \overline{D}$ 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.

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

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 4–16

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

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

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

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

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

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

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:

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

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 4–17

Convert the following SOP expression to an equivalent POS expression:

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

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

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

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

SECTION 4–6 REVIEW

1. Identify each of the following expressions as SOP, standard SOP, POS, or standard POS:
   (a) $AB + \overline{A}BD + \overline{A}C\overline{D}$ (b) $(A + \overline{B} + C)(A + B + \overline{C})$
   (c) $\overline{A}BC + AB\overline{C}$ (d) $A(A + \overline{C})(A + B)$
2. Convert each SOP expression in Question 1 to standard form.
3. 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 4–18.

EXAMPLE 4–18

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

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 $\overline{A}\,\overline{B}C$: 001; $A\overline{B}\,\overline{C}$: 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	
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

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

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 4–19.

EXAMPLE 4–19

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 A	B	C	OUTPUT 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	

Notice that the truth table in this example is the same as the one in Example 4–18. 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:

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

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:

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

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

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

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:

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

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

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

EXAMPLE 4–20

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:

$$011 \longrightarrow \overline{A}BC$$
$$100 \longrightarrow A\overline{B}\,\overline{C}$$
$$110 \longrightarrow AB\overline{C}$$
$$111 \longrightarrow ABC$$

The resulting standard SOP expression for the output X is

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

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:

$$000 \longrightarrow A + B + C$$
$$001 \longrightarrow A + B + \overline{C}$$
$$010 \longrightarrow A + \overline{B} + C$$
$$101 \longrightarrow \overline{A} + B + \overline{C}$$

The resulting standard POS expression for the output X is

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

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 they should either both be 1 or both be 0, depending on the binary value.

SECTION 4–7 REVIEW

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.

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 4–11 deals with five variables using a 32-cell Karnaugh map. Another method, which is beyond the scope of this book, called the Quine-McClusky method can be used for higher numbers of variables.

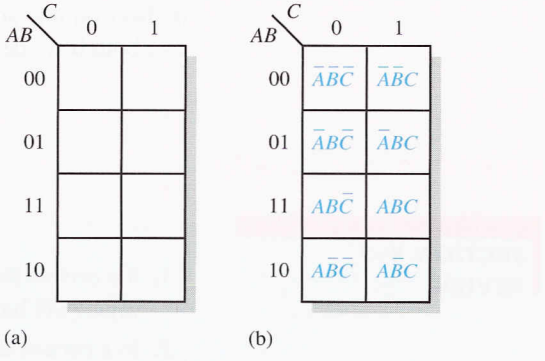
The number of cells in a Karnaugh map is equal to the total number of possible input variable combinations as is the number of rows in a truth table. For three variables, the number of cells is $2^3 = 8$. For four variables, the number of cells is $2^4 = 16$.

The 3-Variable Karnaugh Map

The 3-variable Karnaugh map is an array of eight cells, as shown in Figure 4–21(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 4–21(b) shows the standard product terms that are represented by each cell in the Karnaugh map.

▶ **FIGURE 4–21**

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 4–22(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 com-

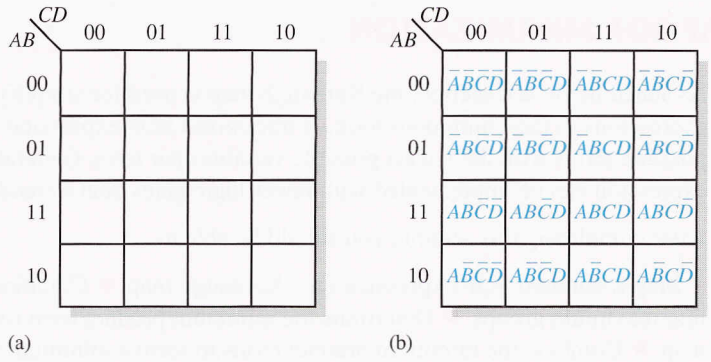


FIGURE 4–22

A 4-variable Karnaugh map.

bined 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 4–22(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.

Cells that differ by only one variable are adjacent.

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

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, 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 4–23 illustrates the cell adjacencies with a 4-variable map, although the same rules for adjacency apply to Karnaugh maps with any number of cells.

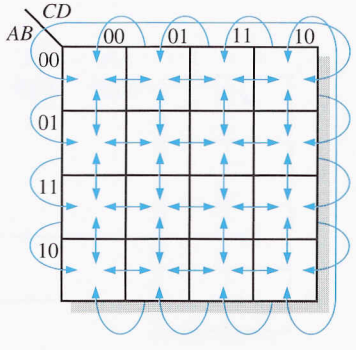


FIGURE 4–23

Adjacent cells on a Karnaugh map are those that differ by only one variable. Arrows point between adjacent cells.

SECTION 4–8 REVIEW

1. In a 3-variable Karnaugh map, what is the binary value for the cell in each of the following locations:
   (a) upper left corner (b) lower right corner
   (c) lower left corner (d) upper right corner
2. What is the standard product term for each cell in Question 1 for variables X, Y, and Z?
3. Repeat Question 1 for a 4-variable map.
4. 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 $A\overline{B}C$, a 1 goes in the 101 cell on a 3-variable map.

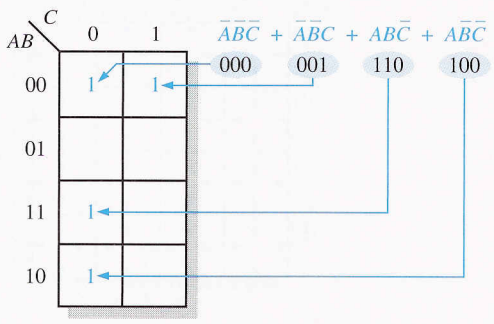
When an SOP expression is completely mapped, there will be a number of 1s on the Karnaugh map equal to the number of product terms in the standard SOP expression. The cells that do not have a 1 are the cells for which the expression is 0. Usually, when working with SOP expressions, the 0s are left off the map. The following steps and the illustration in Figure 4–24 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 4–24**

Example of mapping a standard SOP expression.



EXAMPLE 4–21

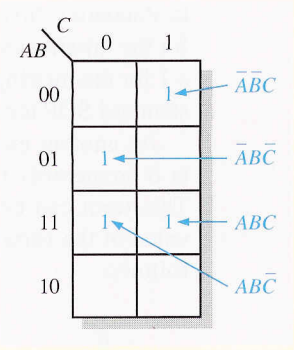
Map the following standard SOP expression on a Karnaugh map:

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

Solution Evaluate the expression as shown below. Place a 1 on the 3-variable Karnaugh map in Figure 4–25 for each standard product term in the expression.

$$\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$$
$$001 \quad 010 \quad 110 \quad 111$$

▶ FIGURE 4–25



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

EXAMPLE 4–22

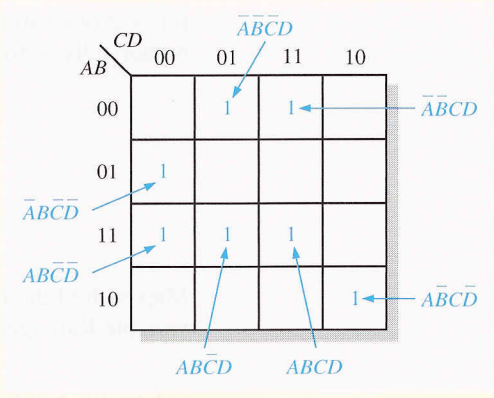
Map the following standard SOP expression on a Karnaugh map:

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

Solution Evaluate the expression as shown below. Place a 1 on the 4-variable Karnaugh map in Figure 4–26 for each standard product term in the expression.

$$\overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}D + ABCD + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}C\overline{D}$$
$$0011 \quad 0100 \quad 1101 \quad 1111 \quad 1100 \quad 0001 \quad 1010$$

▶ FIGURE 4–26



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

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

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 $A\overline{B}$. 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 $\overline{C}$: 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 $A\overline{B}\,\overline{C}$ and $A\overline{B}C$.

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 $\overline{A}\overline{B}\overline{C}$, $\overline{A}BC$, $AB\overline{C}$, and ABC.

EXAMPLE 4–23

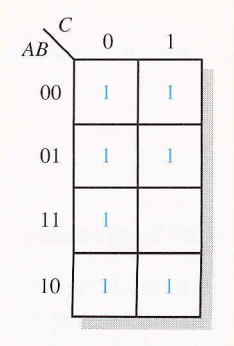
Map the following SOP expression on a Karnaugh map: $\overline{A} + A\overline{B} + AB\overline{C}$.

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:

$\overline{A}$	$+ A\overline{B}$	$+ AB\overline{C}$
000	100	110
001	101	
010		
011		

Map each of the resulting binary values by placing a 1 in the appropriate cell of the 3-variable Karnaugh map in Figure 4–27.

▶ **FIGURE 4–27**



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

EXAMPLE 4–24

Map the following SOP expression on a Karnaugh map:

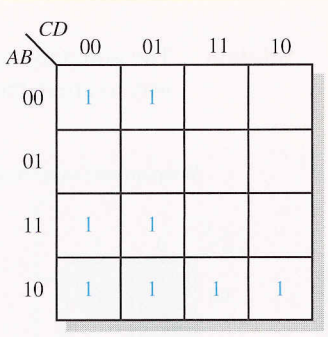
$$\overline{B}\,\overline{C} + A\overline{B} + AB\overline{C} + A\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}CD$$

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:

$\overline{B}\,\overline{C}$	$A\overline{B}$	+ $AB\overline{C}$	+ $A\overline{B}C\overline{D}$	+ $\overline{A}\,\overline{B}\,\overline{C}D$	+ $A\overline{B}CD$
0000	1000	1100	1010	0001	1011
0001	1001	1101			
1000	1010				
1001	1011				

Map each of the resulting binary values by placing a 1 in the appropriate cell of the 4-variable Karnaugh map in Figure 4–28. Notice that some of the values in the expanded expression are redundant.

► **FIGURE 4–28**



Related Problem Map the expression $A + \overline{C}D + AC\overline{D} + \overline{A}BC\overline{D}$ 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.

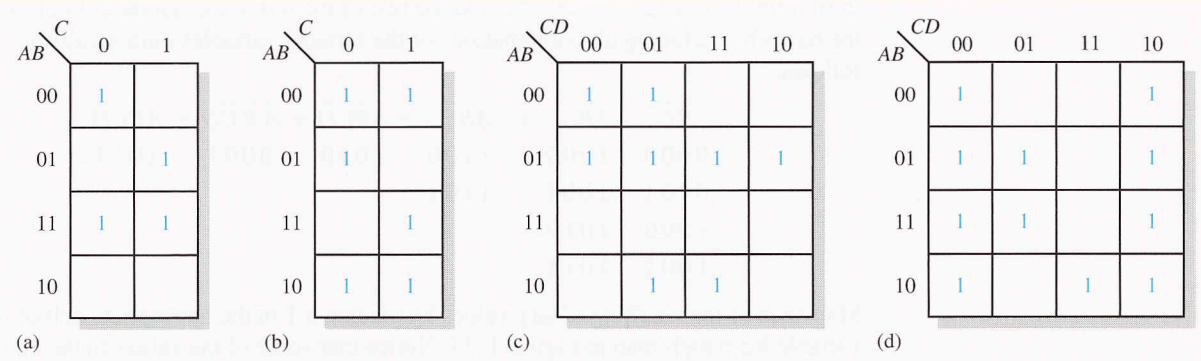
Grouping the 1s You can group 1s on the Karnaugh map according to the following rules by enclosing those adjacent cells containing 1s. The goal is to maximize the size of the groups and to minimize the number of groups.

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 noncommon 1s.

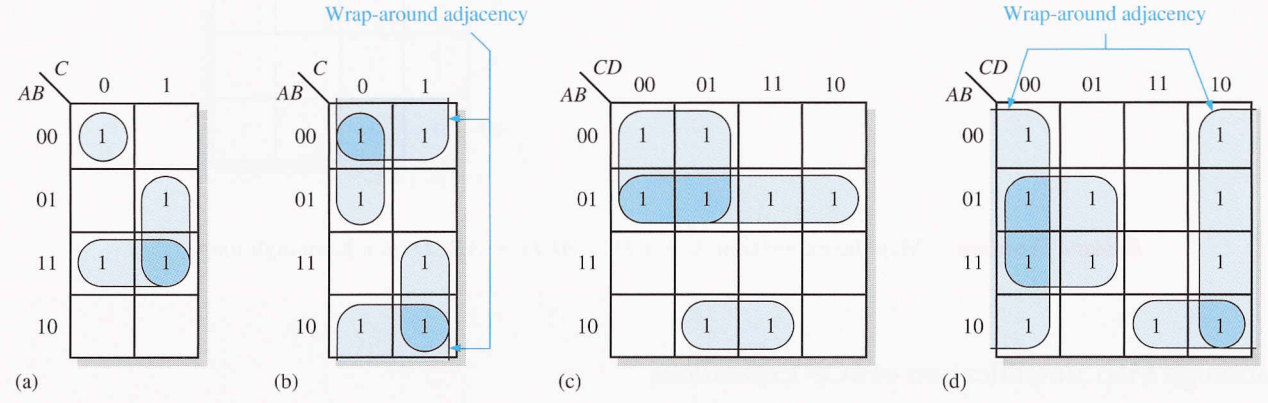
EXAMPLE 4–25

Group the 1s in each of the Karnaugh maps in Figure 4–29.



▲ **FIGURE 4–29**

Solution The groupings are shown in Figure 4–30. In some cases, there may be more than one way to group the 1s to form maximum groupings.



▲ **FIGURE 4–30**

Related Problem Determine if there are other ways to group the 1s in Figure 4–30 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

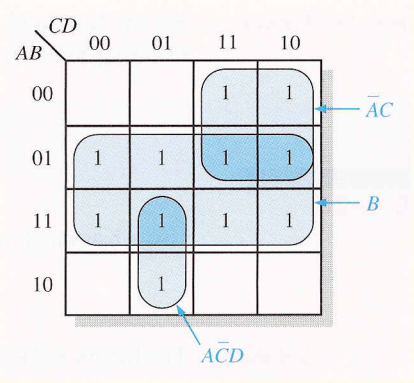
or complemented) within the group. Variables that occur both uncomplemented and complemented within the group are eliminated. These are called *contradictory variables.*

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 4–26

Determine the product terms for the Karnaugh map in Figure 4–31 and write the resulting minimum SOP expression.

▶ **FIGURE 4–31**



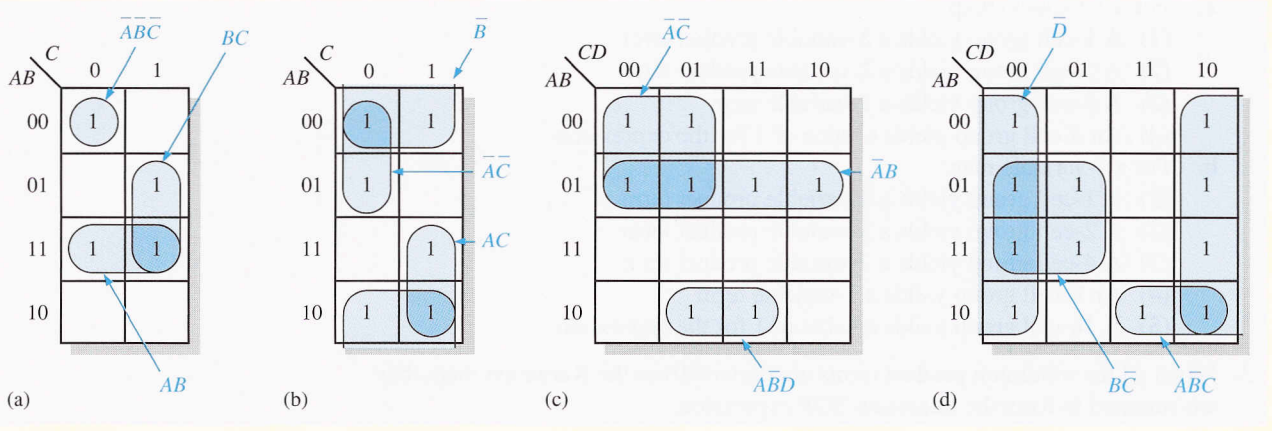
Solution Eliminate variables that are in a grouping in both complemented and uncomplemented forms. In Figure 4–31, the product term for the 8-cell group is B because the cells within that group contain both A and $\overline{A}$, C and $\overline{C}$, and D and $\overline{D}$, which are eliminated. The 4-cell group contains B, $\overline{B}$, D, and $\overline{D}$, leaving the variables $\overline{A}$ and C, which form the product term $\overline{A}C$. The 2-cell group contains B and $\overline{B}$, leaving variables A, $\overline{C}$, and D which form the product term $A\overline{C}D$. Notice how overlapping is used to maximize the size of the groups. The resulting minimum SOP expression is the sum of these product terms:

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

Related Problem For the Karnaugh map in Figure 4–31, add a 1 in the lower right cell (1010) and determine the resulting SOP expression.

EXAMPLE 4–27

Determine the product terms for each of the Karnaugh maps in Figure 4–32 and write the resulting minimum SOP expression.



▲ FIGURE 4–32

Solution The resulting minimum product term for each group is shown in Figure 4–32. The minimum SOP expressions for each of the Karnaugh maps in the figure are

(a) $AB + BC + \overline{A}\,\overline{B}\,\overline{C}$ **(b)** $\overline{B} + \overline{A}\,\overline{C} + AC$

(c) $\overline{A}B + \overline{A}\,\overline{C} + A\overline{B}D$ **(d)** $\overline{D} + A\overline{B}C + B\overline{C}$

Related Problem For the Karnaugh map in Figure 4–32(d), add a 1 in the 0111 cell and determine the resulting SOP expression.

EXAMPLE 4–28

Use a Karnaugh map to minimize the following standard SOP expression:

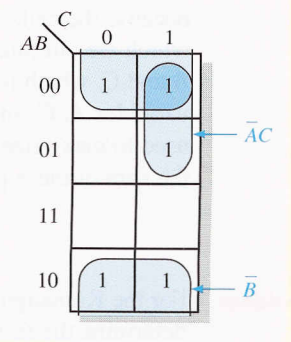
$$A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C}$$

Solution The binary values of the expression are

$$101 + 011 + 011 + 000 + 100$$

Map the standard SOP expression and group the cells as shown in Figure 4–33.

▶ FIGURE 4–33



Notice the "wrap around" 4-cell group that includes the top row and the bottom row of 1s. The remaining 1 is absorbed in an overlapping group of two cells. The group of four 1s produces a single variable term, $\overline{B}$. This is determined by observing that within the group, $\overline{B}$ is the only variable that does not change from cell to cell. The group of two 1s produces a 2-variable term $\overline{A}C$. This is determined by observing that within the group, $\overline{A}$ and C do not change from one cell to the next. The product term for each group is shown. The resulting minimum SOP expression is

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

Keep in mind that this minimum expression is equivalent to the original standard expression.

Related Problem Use a Karnaugh map to simplify the following standard SOP expression:

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

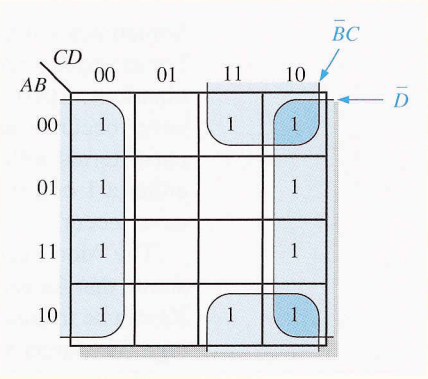
EXAMPLE 4–29

Use a Karnaugh map to minimize the following SOP expression:

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

Solution The first term $\overline{B}\,\overline{C}\,\overline{D}$ must be expanded into $A\overline{B}\,\overline{C}\,\overline{D}$ and $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ to get the standard SOP expression, which is then mapped; and the cells are grouped as shown in Figure 4–34.

▶ **FIGURE 4–34**



Notice that both groups exhibit "wrap around" adjacency. The group of eight is formed because the cells in the outer columns are adjacent. The group of four is formed to pick up the remaining two 1s because the top and bottom cells are adjacent. The product term for each group is shown. The resulting minimum SOP expression is

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

Keep in mind that this minimum expression is equivalent to the original standard expression.

Related Problem Use a Karnaugh map to simplify the following SOP expression:

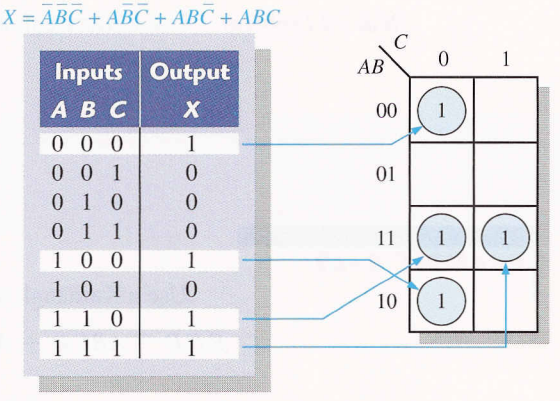
$$\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + W\overline{X}YZ + W\overline{X}\,\overline{Y}Z + \overline{W}YZ + W\overline{X}\,\overline{Y}\,\overline{Z}$$

Mapping Directly from a Truth Table

You have seen how to map a Boolean expression; now you will learn how to go directly from a truth table to a Karnaugh map. Recall that a truth table gives the output of a Boolean expression for all possible input variable combinations. An example of a Boolean expression and its truth table representation is shown in Figure 4–35. Notice in the truth table that the output X is 1 for four different input variable combinations. The 1s in the output column of the truth table are mapped directly onto a Karnaugh map into the cells corresponding to the values of the associated input variable combinations, as shown in Figure 4–35. In the figure you can see that the Boolean expression, the truth table, and the Karnaugh map are simply different ways to represent a logic function.

FIGURE 4–35

Example of mapping directly from a truth table to a Karnaugh map.



"Don't Care" Conditions

Sometimes a situation arises in which some input variable combinations are not allowed. For example, recall that in the BCD code covered in Chapter 2, there are six invalid combinations: 1010, 1011, 1100, 1101, 1110, and 1111. Since these unallowed states will never occur in an application involving the BCD code, they can be treated as **"don't care"** terms with respect to their effect on the output. That is, for these "don't care" terms either a 1 or a 0 may be assigned to the output; it really does not matter since they will never occur.

The "don't care" terms can be used to advantage on the Karnaugh map. Figure 4–36 shows that for each "don't care" term, an X is placed in the cell. When grouping the 1s, the Xs can be treated as 1s to make a larger grouping or as 0s if they cannot be used to advantage. The larger a group, the simpler the resulting term will be.

The truth table in Figure 4–36(a) describes a logic function that has a 1 output only when the BCD code for 7, 8, or 9 is present on the inputs. If the "don't cares" are used as 1s, the resulting expression for the function is $A + BCD$, as indicated in part (b). If the "don't cares" are not used as 1s, the resulting expression is $A\overline{B}\,\overline{C} + \overline{A}BCD$; so you can see the advantage of using "don't care" terms to get the simplest expression.

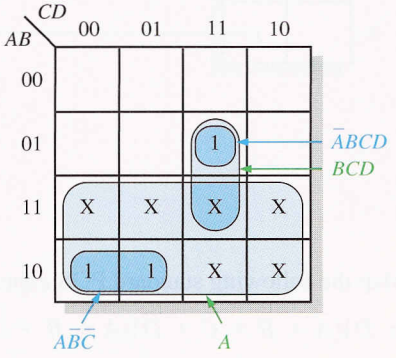
SECTION 4–9 REVIEW

1. Lay out Karnaugh maps for three and four variables.
2. Group the 1s and write the simplified SOP expression for the Karnaugh map in Figure 4–25.
3. Write the original standard SOP expressions for each of the Karnaugh maps in Figure 4–32.

Inputs A B C D	Output 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

(a) Truth table

Don't cares



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

FIGURE 4–36
Example of the use of "don't care" conditions to simplify an expression.

4–10 KARNAUGH MAP POS MINIMIZATION

In the last section, you studied the minimization of an SOP expression using a Karnaugh map. In this section, we will focus on POS expressions. The approaches are much the same except that with POS expressions, 0s representing the standard sum terms are placed on the Karnaugh map instead of 1s.

After completing this section, you should be able to

■ Map a standard POS expression on a Karnaugh map ■ Combine the 0s on the map into maximum groups ■ Determine the minimum sum term for each group on the map ■ Combine the minimum sum terms to form a minimum POS expression ■ Use the Karnaugh map to convert between POS and SOP

Mapping a Standard POS Expression

For a POS expression in standard form, a 0 is placed on the Karnaugh map for each sum term in the expression. Each 0 is placed in a cell corresponding to the value of a sum term. For example, for the sum term $A + \bar{B} + C$, a 0 goes in the 010 cell on a 3-variable map.

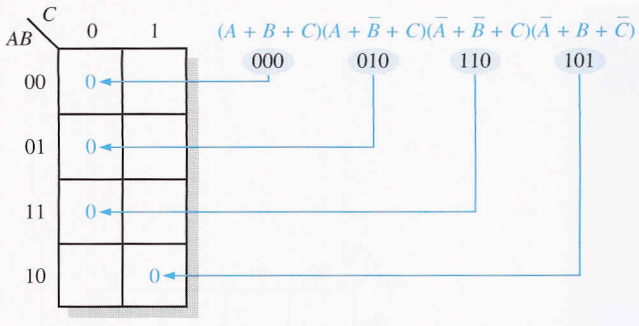
When a POS expression is completely mapped, there will be a number of 0s on the Karnaugh map equal to the number of sum terms in the standard POS expression. The cells that do not have a 0 are the cells for which the expression is 1. Usually, when working with POS expressions, the 1s are left off. The following steps and the illustration in Figure 4–37 show the mapping process.

Step 1. Determine the binary value of each sum term in the standard POS expression. This is the binary value that makes the term equal to 0.

Step 2. As each sum term is evaluated, place a 0 on the Karnaugh map in the corresponding cell.

▶ **FIGURE 4–37**

Example of mapping a standard POS expression.



EXAMPLE 4–30

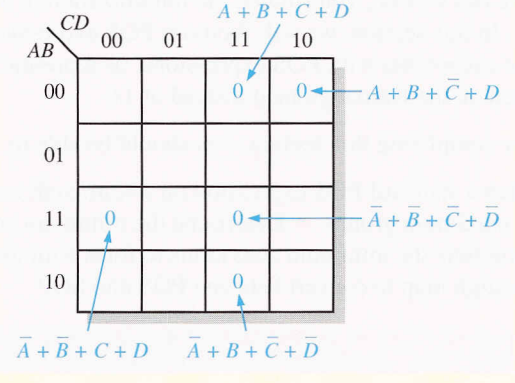
Map the following standard POS expression on a Karnaugh map:

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

Solution Evaluate the expression as shown below and place a 0 on the 4-variable Karnaugh map in Figure 4–38 for each standard sum term in the expression.

$$\underset{1100}{(\overline{A} + \overline{B} + C + D)}\underset{1011}{(\overline{A} + B + \overline{C} + \overline{D})}\underset{0010}{(A + B + \overline{C} + D)}\underset{1111}{(\overline{A} + \overline{B} + \overline{C} + \overline{D})}\underset{0011}{(A + B + \overline{C} + \overline{D})}$$

▶ **FIGURE 4–38**



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

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

Karnaugh Map Simplification of POS Expressions

The process for minimizing a POS expression is basically the same as for an SOP expression except that you group 0s to produce minimum sum terms instead of grouping 1s to produce minimum product terms. The rules for grouping the 0s are the same as those for grouping the 1s that you learned in Section 4–9.

EXAMPLE 4–31

Use a Karnaugh map to minimize the following standard POS expression:

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

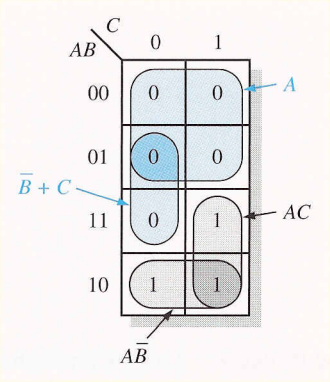
Also, derive the equivalent SOP expression.

Solution The combinations of binary values of the expression are

$$(0 + 0 + 0)\,(0 + 0 + 1)\,(0 + 1 + 0)\,(0 + 1 + 1)\,(1 + 1 + 0)$$

Map the standard POS expression and group the cells as shown in Figure 4–39.

▶ **FIGURE 4–39**



Notice how the 0 in the 110 cell is included into a 2-cell group by utilizing the 0 in the 4-cell group. The sum term for each blue group is shown in the figure and the resulting minimum POS expression is

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

Keep in mind that this minimum POS expression is equivalent to the original standard POS expression.

Grouping the 1s as shown by the gray areas yields an SOP expression that is equivalent to grouping the 0s.

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

Related Problem Use a Karnaugh map to simplify the following standard POS expression:

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

EXAMPLE 4–32

Use a Karnaugh map to minimize the following POS expression:

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

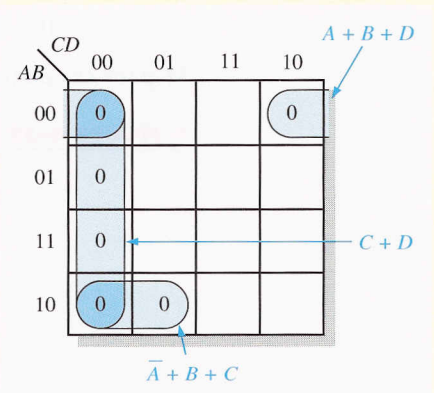
Solution The first term must be expanded into $\overline{A} + B + C + D$ and $A + B + C + D$ to get a standard POS expression, which is then mapped; and the cells are grouped as shown in

Figure 4–40. The sum term for each group is shown and the resulting minimum POS expression is

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

Keep in mind that this minimum POS expression is equivalent to the original standard POS expression.

FIGURE 4–40



Related Problem Use a Karnaugh map to simplify the following POS expression:

$$(W + \overline{X} + Y + \overline{Z})(W + X + Y + Z)(W + \overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + Z)$$

Converting Between POS and SOP Using the Karnaugh Map

When a POS expression is mapped, it can easily be converted to the equivalent SOP form directly from the Karnaugh map. Also, given a mapped SOP expression, an equivalent POS expression can be derived directly from the map. This provides a good way to compare both minimum forms of an expression to determine if one of them can be implemented with fewer gates than the other.

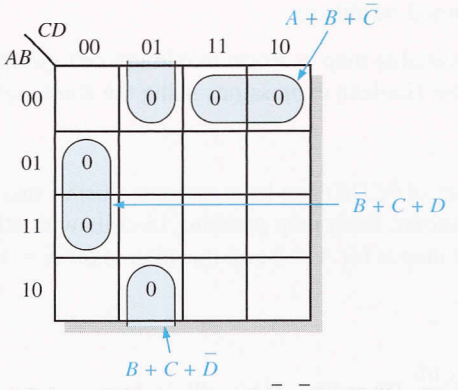
For a POS expression, all the cells that do not contain 0s contain 1s, from which the SOP expression is derived. Likewise, for an SOP expression, all the cells that do not contain 1s contain 0s, from which the POS expression is derived. Example 4–33 illustrates this conversion.

EXAMPLE 4–33

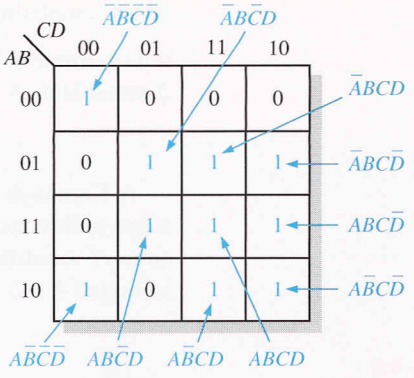
Using a Karnaugh map, convert the following standard POS expression into a minimum POS expression, a standard SOP expression, and a minimum SOP expression.

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

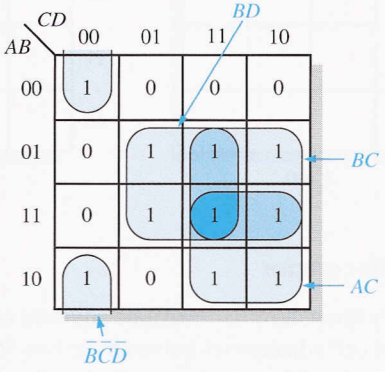
Solution The 0s for the standard POS expression are mapped and grouped to obtain the minimum POS expression in Figure 4–41(a). In Figure 4–41(b), 1s are added to the cells that do not contain 0s. From each cell containing a 1, a standard product term is obtained as indicated. These product terms form the standard SOP expression. In Figure 4–41(c), the 1s are grouped and a minimum SOP expression is obtained.



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



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



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

▲ FIGURE 4–41

Related Problem Use a Karnaugh map to convert the following expression to minimum SOP form:

$$(W + \overline{X} + Y + \overline{Z})(\overline{W} + X + \overline{Y} + \overline{Z})(\overline{W} + \overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + \overline{Z})$$

SECTION 4–10 REVIEW

1. What is the difference in mapping a POS expression and an SOP expression?
2. What is the standard sum term expressed with variables *A*, *B*, *C*, and *D* for a 0 in cell 1011 of the Karnaugh map?
3. What is the standard product term expressed with variables *A*, *B*, *C*, and *D* for a 1 in cell 0010 of the Karnaugh map?

4–11 FIVE-VARIABLE KARNAUGH MAPS

Boolean functions with five variables can be simplified using a 32-cell Karnaugh map. Actually, two 4-variable maps (16 cells each) are used to construct a 5-variable map. You already know the cell adjacencies within each of the 4-variable maps and how to form groups of cells containing 1s to simplify an SOP expression. All you need to learn for five variables is the cell adjacencies between the two 4-variable maps and how to group those adjacent 1s.

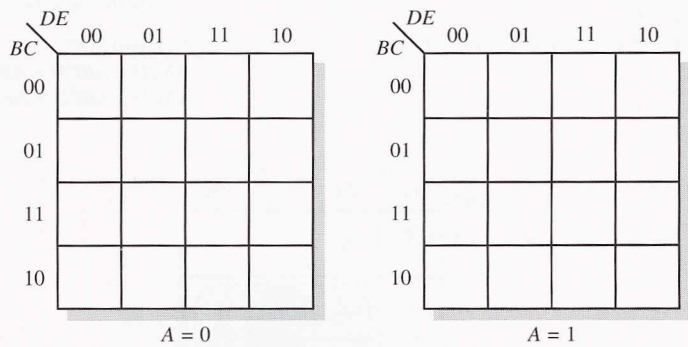
After completing this section, you should be able to

■ Determine cell adjacencies in a 5-variable map ■ Form maximum cell groupings in a 5-variable map ■ Minimize 5-variable Boolean expressions using the Karnaugh map

A Karnaugh map for five variables (*ABCDE*) can be constructed using two 4-variable maps with which you are already familiar. Each map contains 16 cells with all combinations of variables *B, C, D,* and *E*. One map is for $A = 0$ and the other is for $A = 1$, as shown in Figure 4–42.

▶ **FIGURE 4–42**

A 5-variable Karnaugh map.



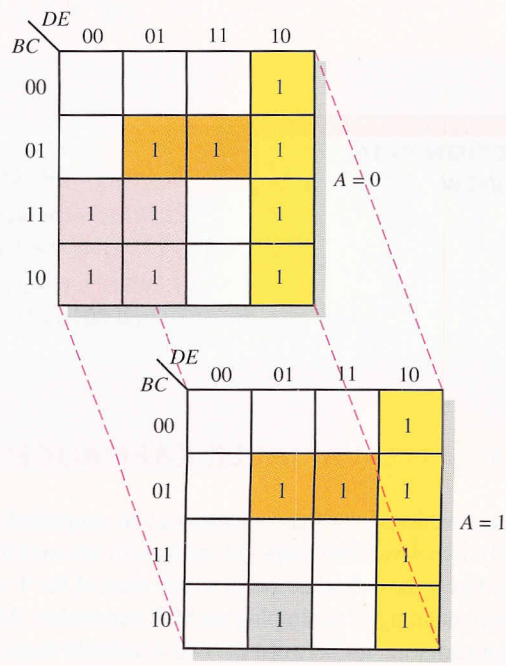
Cell Adjacencies

You already know how to determine adjacent cells within the 4-variable map. The best way to visualize cell adjacencies between the two 16-cell maps is to imagine that the $A = 0$ map is placed on top of the $A = 1$ map. Each cell in the $A = 0$ map is adjacent to the cell directly below it in the $A = 1$ map.

To illustrate, an example with four groups is shown in Figure 4–43 with the maps in a 3-dimensional arrangement. The 1s in the yellow cells form an 8-bit group (four in the $A = 0$

▶ **FIGURE 4–43**

Illustration of groupings of 1s in adjacent cells of a 5-variable map.



map combined with four in the $A = 1$ map). The 1s in the orange cells form a 4-bit group. The 1s in the light red cells form a 4-bit group only in the $A = 0$ map. The 1 in the gray cell in the $A = 1$ map is grouped with the 1 in the lower right light red cell in the $A = 0$ map to form a 2-bit group.

Determining the Boolean Expression The original SOP Boolean expression that is plotted on the Karnaugh map in Figure 4–43 contains seventeen 5-variable terms because there are seventeen 1s on the map. As you know, only the variables that do not change from uncomplemented to complemented or vice versa within a group remain in the expression for that group. The simplified expression taken from the map is developed as follows:

- The term for the yellow group is $D\overline{E}$.
- The term for the orange group is $\overline{B}CE$.
- The term for the light red group is $\overline{A}B\overline{D}$.
- The term for the gray cell grouped with the red cell is $B\overline{C}\,\overline{D}E$.

Combining these terms into the simplified SOP expression yields

$$X = D\overline{E} + \overline{B}CE + \overline{A}B\overline{D} + B\overline{C}\,\overline{D}E$$

EXAMPLE 4–34

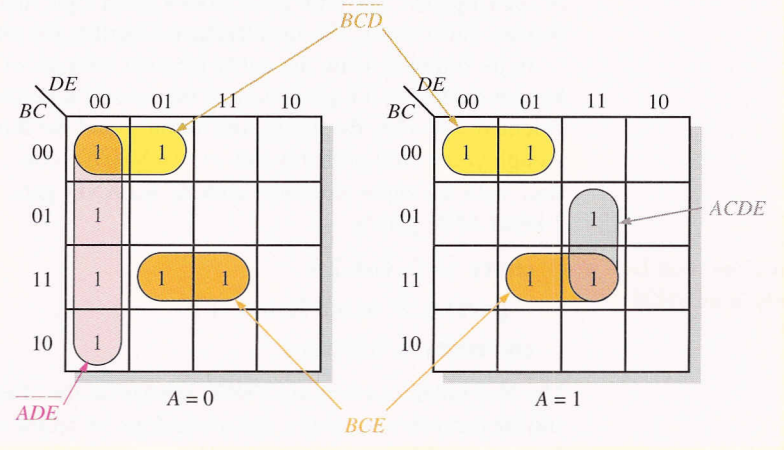
Use a Karnaugh map to minimize the following standard SOP 5-variable expression:

$$X = \overline{A}\,\overline{B}C\overline{D}\,\overline{E} + \overline{A}\,\overline{B}CD\overline{E} + \overline{A}B\overline{C}\,\overline{D}\,\overline{E} + \overline{A}B\overline{C}D\overline{E} + \overline{A}B\overline{C}\,\overline{D}E + \overline{A}B\overline{C}DE$$
$$+ \overline{A}BCDE + A\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + A\overline{B}\,\overline{C}D\overline{E} + A\overline{B}C\overline{D}\,\overline{E} + A\overline{B}CD\overline{E} + ABC\overline{D}\,\overline{E}$$

Solution Map the SOP expression. Figure 4–44 shows the groupings and their corresponding terms. Combining the terms yields the following minimized SOP expression:

$$X + \overline{A}D\overline{E} + \overline{B}\,\overline{C}D + BCE + ACDE$$

▶ **FIGURE 4–44**



Related Problem Minimize the following expression:

$$Y = \overline{A}\,\overline{B}C\overline{D}\,\overline{E} + \overline{A}\,\overline{B}CD\overline{E} + \overline{A}B\overline{C}D\overline{E} + \overline{A}B\overline{C}\,\overline{D}\,\overline{E} + A\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + A\overline{B}C\overline{D}\,\overline{E} + ABC\overline{D}\,\overline{E} + AB\overline{C}\,\overline{D}E$$
$$+ \overline{A}\,\overline{B}\,\overline{C}D\overline{E} + \overline{A}\,\overline{B}CDE + \overline{A}BCD\overline{E} + \overline{A}B\overline{C}D\overline{E} + A\overline{B}CD\overline{E} + A\overline{B}C\overline{D}\,\overline{E} + ABCD\overline{E} + AB\overline{C}D\overline{E}$$

SECTION 4–11 REVIEW

1. Why does a 5-variable Karnaugh map require 32 cells?
2. What is the expression represented by a 5-variable Karnaugh map in which each cell contains a 1?

4–12 VHDL (optional)

This optional section provides a brief introduction to VHDL and is not meant to teach the complete structure and syntax of the language. For more detailed information and instruction, refer to the footnote. Hardware description languages (HDLs) are tools for logic design entry, called text entry, that are used to implement logic designs in programmable logic devices. Although VHDL provides multiple ways to describe a logic circuit, only the simplest and most direct programming examples of text entry are discussed here.

After completing this section, you should be able to

- State the essential elements of VHDL
- Write a simple VHDL program

The V in VHDL* stands for VHSIC (*V*ery *H*igh *S*peed *I*ntegrated *C*ircuit) and the HDL, of course, stands for hardware description language. As mentioned, **VHDL** is a standard language adopted by the IEEE (Institute of Electrical and Electronics Engineers) and is designated IEEE Std. 1076-1993. VHDL is a complex and comprehensive language and using it to its full potential involves a lot of effort and experience.

VHDL provides three basic approaches to describing a digital circuit using software: *behavioral, data flow,* and *structural.* We will restrict this discussion to the data flow approach in which you write Boolean-type statements to describe a logic circuit. Keep in mind that VHDL, as well as the other HDLs, is a tool for implementing digital designs and is, therefore, a means to an end and not an end in itself.

It is relatively easy to write programs to describe simple logic circuits in VHDL. The logical operators are the following VHDL keywords: **and, or, not, nand, nor, xor,** and **xnor.** The two essential elements in any VHDL program are the entity and the architecture, and they must be used together. The **entity** describes a given logic function in terms of its external inputs and outputs, called ports. The **architecture** describes the internal operation of the logic function.

In its simplest form, the entity element consists of three statements: The first statement assigns a name to a logic function; the second statement, called the *port* statement which is indented, specifies the inputs and outputs; and the third statement is the *end* statement. Although you would probably not write a VHDL program for a single gate, it is instructive to start with a simple example such as an AND gate. The VHDL entity declaration for a 2-input AND gate is

Colons and semicolons must be used appropriately in all VHDL programs.

```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;
```

The blue boldface terms are VHDL keywords; the other terms are identifiers that you assign; and the parentheses, colons, and semicolons are required VHDL syntax. As you can see, A and B are specified as input bits and X is specified as an output bit. The port identifiers A, B, and X as well as the entity name AND_Gate2 are user-defined and can be renamed. As in all HDLs, the placement of colons and semicolons is crucial and must be strictly adhered to.

The VHDL architecture element of the program for the 2-input AND gate described by the entity is

*See Floyd, Thomas. 2003. *Digital Fundamentals with VHDL.* Prentice Hall; Pellerin, David and Taylor, Douglas. 1997. *VHDL Made Easy!* Prentice Hall; Bhasker, Jayaram. 1999. *A VHDL Primer,* 3 ed. Prentice Hall.

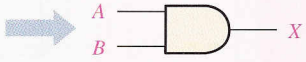
```
architecture LogicFunction of AND_Gate2 is
begin
  X ⇐ A and B;
end architecture LogicFunction;
```

Again, the VHDL keywords are blue boldface, and the semicolons and the assignment operator ⇐ are required syntax. The first statement of the architecture element must reference the entity name.

The entity and the architecture are combined into a single VHDL program to describe an AND gate, as illustrated in Figure 4–45.

```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;

architecture LogicFunction of AND_Gate2 is
begin
  X <= A and B;
end architecture LogicFunction;
```



◀ **FIGURE 4–45**

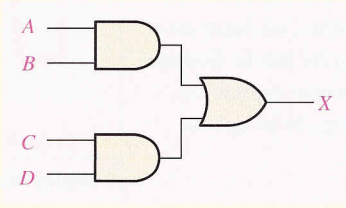
A VHDL program for a 2-input AND gate.

Writing Boolean Expressions in VHDL As you saw, the expression for a 2-input AND gate, $X = AB$, is written in VHDL as X ⇐ A **and** B;. Any Boolean expression can be written using VHDL keywords **not, and, or, nand, nor, xor,** and **xnor.** For example, the Boolean expression $X = A + B + C$ is written in VHDL as X ⇐ A **or** B **or** C;. The Boolean expression $X = A\overline{B} + \overline{C}D$ can be written as the VHDL statement X ⇐ (A **and not** B) **or** (**not** C **and** D);. As another example, the VHDL statement for a 2-input NAND gate can be written as X ⇐ **not**(A **and** B); or it can be written as X ⇐ A **nand** B;.

EXAMPLE 4–35

Write a VHDL program to describe the logic circuit in Figure 4–46.

▶ **FIGURE 4–46**



Solution This AND/OR logic circuit is described in Boolean algebra as

$$X = AB + CD$$

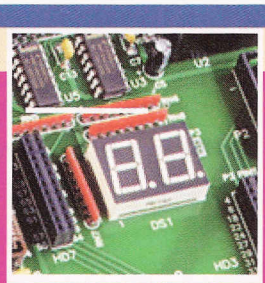
The VHDL program follows. The entity name is AND_OR.

```
entity AND_OR is
  port (A, B, C, D: in bit; X: out bit);
end entity AND_OR;
architecture LogicFunction of AND_OR is
begin
  X <= (A and B) or (C and D);
end architecture LogicFunction;
```

Related Problem Write the VHDL statement to describe the logic circuit if a NOR gate replaces the OR gate in Figure 4–46.

SECTION 4–12 REVIEW

1. What is an HDL?
2. Name the two essential design elements in a VHDL program.
3. What does the entity do?
4. What does the architecture do?



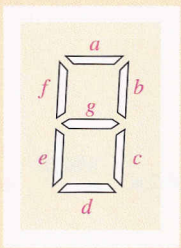
DIGITAL SYSTEM APPLICATION

Seven-segment displays are used in many types of products. The tablet-counting and control system that was described in Chapter 1 has two 7-segment displays. These displays are used with logic circuits that decode a binary coded decimal (BCD) number and activate the appropriate digits on the display. In this digital system application, we focus on a minimum-gate design for this to illustrate an application of Boolean expressions and the Karnaugh map. As an option, VHDL is also applied.

The 7-Segment Display

Figure 4–47 shows a common display format composed of seven elements or segments. Energizing certain combinations of these segments can cause each of the ten decimal digits to be displayed. Figure 4–48 illustrates this method of digital display for each of the ten digits by using a red segment to represent one that is energized. To produce a 1, segments *b* and *c* are energized; to produce a 2, segments *a*, *b*, *g*, *e*, and *d* are used; and so on.

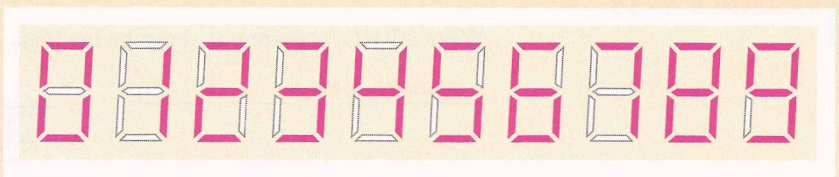
LED Displays One common type of 7-segment display consists of light-emitting diodes (**LEDs**) arranged as shown in Figure 4–49. Each segment is an LED that emits light when there is current through it. In Figure 4–49(a) the common-anode arrangement requires the driving circuit to provide a low-level voltage in order to activate a given segment. When a LOW is applied to a segment input, the LED is turned on, and there is current through it. In Figure 4–49(b) the common-cathode arrangement requires the driver to provide a high-level voltage to activate a segment. When a HIGH is applied to a segment input, the LED is turned on and there is current through it.



▲ **FIGURE 4–47**

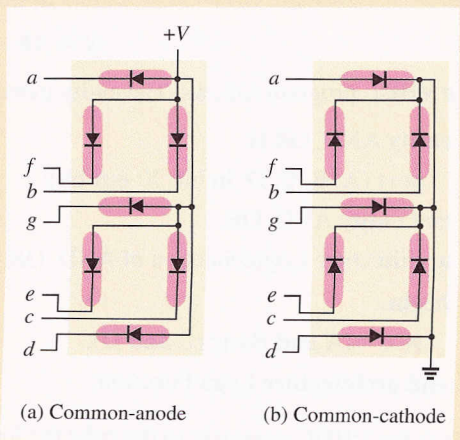
Seven-segment display format showing arrangement of segments.

LCD Displays Another common type of 7-segment display is the liquid crystal display (**LCD**). LCDs operate by polarizing light so that a nonactivated segment reflects incident light and thus appears invisible against its background. An activated segment does not reflect incident light and thus appears dark. LCDs consume much less power than LEDs but



▲ **FIGURE 4–48**

Display of decimal digits with a 7-segment device.



◀ **FIGURE 4–49**

Arrangements of 7-segment LED displays.

cannot be seen in the dark, while LEDs can.

Segment Logic

Each segment is used for various decimal digits, but no one segment is used for all ten digits. Therefore, each segment must be activated by its own decoding circuit that detects the occurrence of any of the numbers in which the segment is used. From Figures 4–47 and 4–48, the segments that are required to be activated for each displayed digit are determined and listed in Table 4–9.

Truth Table for the Segment Logic

The segment decoding logic requires four binary coded decimal (BCD) inputs and seven outputs, one for each segment in the display, as indicated in the block diagram of Figure 4–50. The multiple-output truth table, shown in Table 4–10, is actually seven truth tables in one and could be separated into a separate table for each segment. A 1 in the segment output columns of the table indicates an activated segment.

Since the BCD code does not include the binary values 1010, 1011, 1100, 1101, 1110, and 1111, these combinations will never appear on the inputs and can therefore be treated as "don't care" (X) conditions, as indicated in the truth table. To conform with the practice of most IC manufacturers, *A* represents the least significant bit and *D* represents the most significant bit in this particular application.

Boolean Expressions for the Segment Logic From the truth table, a standard SOP or POS expression can be written for each segment. For example, the standard SOP expression for segment *a* is

$$a = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}BA + \overline{D}C\overline{B}A + \overline{D}CBA + D\overline{C}\,\overline{B}\,\overline{A} + D\overline{C}B\overline{A} + \overline{D}CB\overline{A} + \overline{D}CBA + D\overline{C}\,\overline{B}\,\overline{A} + D\overline{C}\overline{B}A$$

and the standard SOP expression for segment *e* is

$$e = \overline{D}\,\overline{C}B\overline{A} + \overline{D}\,\overline{C}BA + \overline{D}CB\overline{A} + D\overline{C}\,\overline{B}\,\overline{A}$$

Expressions for the other segments can be similarly developed. As you can see, the expression for segment *a* has eight product terms and the expression for segment *e* has four product terms representing each of the BCD inputs that activate that segment. This means that the standard SOP implementation of segment-*a* logic requires an AND-OR circuit consisting of eight 4-input AND gates and one 8-input OR gate. The implementation of segment-*e* logic requires four 4-input AND gates and one 4-input OR gate. In both cases, four inverters are required to produce the complement of each variable.

Karnaugh Map Minimization of the Segment Logic Let's begin by obtaining a minimum SOP expression for segment *a*. A Karnaugh map for segment *a* is shown in Figure 4–51 and the following steps are carried out:

Step 1. The 1s are mapped directly from Table 4–10.

TABLE 4–9

Active segments for each decimal digit.

DIGIT	SEGMENTS ACTIVATED
0	*a, b, c, d, e, f*
1	*b, c*
2	*a, b, d, e, g*
3	*a, b, c, d, g*
4	*b, c, f, g*
5	*a, c, d, f, g*
6	*a, c, d, e, f, g*
7	*a, b, c*
8	*a, b, c, d, e, f, g*
9	*a, b, c, d, f, g*

FIGURE 4–50

Block diagram of 7-segment logic and display.

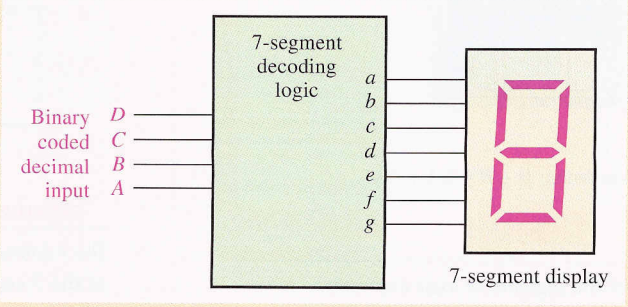


TABLE 4–10

Truth table for 7-segment logic.

DECIMAL DIGIT	INPUTS				SEGMENT OUTPUTS						
	D	*C*	*B*	*A*	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

Output = 1 means segment is activated (on)

Output = 0 means segment is not activated (off)

Output = X means "don't care"

Step 2. All of the "don't cares" (X) are placed on the map.

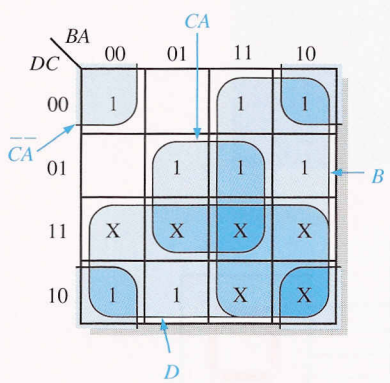
Step 3. The 1s are grouped as shown. "Don't cares" and overlapping of cells are utilized to form the largest groups possible.

Step 4. Write the minimum product term for each group and sum the terms to form the minimum SOP expression.

Keep in mind that "don't cares" do not have to be included in a group, but in this case all of them are used. Also, notice that the 1s in the corner cells are grouped with a "don't care" using the "wrap around" adjacency of the corner cells.

Standard SOP expression:

$\overline{D}\overline{C}\overline{B}\overline{A} + \overline{D}\overline{C}\overline{B}A + \overline{D}\overline{C}B\overline{A} + \overline{D}\overline{C}BA + \overline{D}C\overline{B}\overline{A} + \overline{D}CB\overline{A} + D\overline{C}\overline{B}\overline{A} + D\overline{C}\overline{B}A$



Minimum SOP expression: $D + B + CA + \overline{C}\overline{A}$

FIGURE 4–51

Karnaugh map minimization of the segment-*a* logic expression.

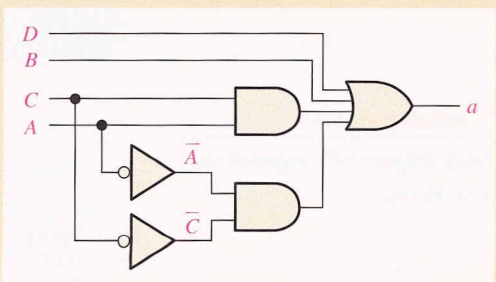


FIGURE 4–52

The minimum logic implementation for segment *a* of the 7-segment display.

Minimum Implementation of Segment-*a* Logic The minimum SOP expression taken from the Karnaugh map in Figure 4–52 for the segment-*a* logic is

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

This expression can be implemented with two 2-input AND gates, one 4-input OR gate and two inverters as shown in Figure 4–52. Compare this to the standard SOP implementation for segment-*a* logic discussed earlier; you'll see that the number of gates and inverters has been reduced from thirteen to five and, as a result, the number of interconnections has been significantly reduced.

The minimum logic for each of the remaining six segments (*b*, *c*, *d*, *e*, *f*, and *g*) can be obtained with a similar approach.

VHDL Implementation (optional)

All of the segment logic can be described by VHDL for implementation in a programmable logic device. Segment-*a* logic can be described by the following VHDL program:

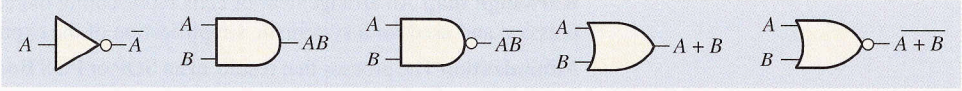
```
entity SEGLOGIC is
  port (A, B, C, D: in bit; SEGa: out bit);
end entity SEGLOGIC;
architecture LogicFunction of
  SEGLOGIC is
begin
  SEGa <= (A and C) or (not A
    and not C) or B or D;
end architecture LogicFunction;
```

System Assignment

- *Activity 1:* Determine the minimum logic for segment *b*.
- *Activity 2:* Determine the minimum logic for segment *c*.
- *Activity 3:* Determine the minimum logic for segment *d*.
- *Activity 4:* Determine the minimum logic for segment *e*.
- *Activity 5:* Determine the minimum logic for segment *f*.
- *Activity 6:* Determine the minimum logic for segment *g*.
- *Optional Activity:* Complete the VHDL program for all seven segments by including each segment logic description in the architecture.

SUMMARY

- Gate symbols and Boolean expressions for the outputs of an inverter and 2-input gates are shown in Figure 4–53.



▲ **FIGURE 4–53**

- Commutative laws: $A + B = B + A$
  $AB = BA$
- Associative laws: $A + (B + C) = (A + B) + C$
  $A(BC) = (AB)C$
- Distributive law: $A(B + C) = AB + AC$
- Boolean rules:

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

- DeMorgan's theorems:
  1. The complement of a product is equal to the sum of the complements of the terms in the product.

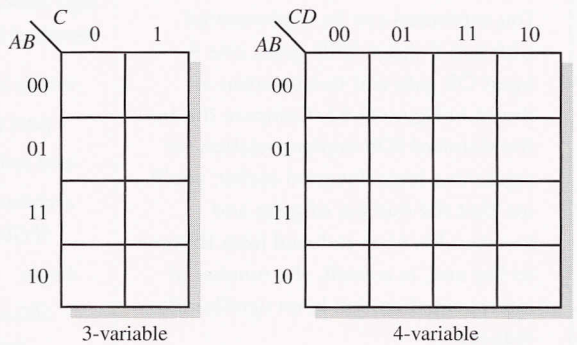
  $$\overline{XY} = \overline{X} + \overline{Y}$$

2 The complement of a sum is equal to the product of the complements of the terms in the sum.

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

- Karnaugh maps for 3 and 4 variables are shown in Figure 4–54. A 5-variable map is formed from two 4-variable maps.

▶ **FIGURE 4–54**



- The basic design element in VHDL is an entity/architecture pair.

KEY TERMS

Key terms and other bold terms in the chapter are defined in the end-of-book glossary.

Complement The inverse or opposite of a number. In Boolean algebra, the inverse function, expressed with a bar over a variable. The complement of a 1 is 0, and vice versa.

"Don't care" A combination of input literals that cannot occur and can be used as a 1 or a 0 on a Karnaugh map for simplification.

Karnaugh map An arrangement of cells representing the combinations of literals in a Boolean expression and used for a systematic simplification of the expression.

Minimization The process that results in an SOP or POS Boolean expression that contains the fewest possible literals per term.

Product-of-sums (POS) A form of Boolean expression that is basically the ANDing of ORed terms.

Product term The Boolean product of two or more literals equivalent to an AND operation.

Sum-of-products (SOP) A form of Boolean expression that is basically the ORing of ANDed terms.

Sum term The Boolean sum of two or more literals equivalent to an OR operation.

Variable A symbol used to represent a logical quantity that can have a value of 1 or 0, usually designated by an italic letter.

VHDL A standard hardware description language. IEEE Std. 1076-1993.

SELF-TEST

Answers are at the end of the chapter.

1. The complement of a variable is always
   (a) 0 **(b)** 1 **(c)** equal to the variable **(d)** the inverse of the variable
2. The Boolean expression $A + \overline{B} + C$ is
   (a) a sum term **(b)** a literal term **(c)** a product term **(d)** a complemented term
3. The Boolean expression $A\overline{B}C\overline{D}$ is
   (a) a sum term **(b)** a product term **(c)** a literal term **(d)** always 1

4. The domain of the expression $A\overline{B}CD + A\overline{B} + \overline{C}D + B$ is
   (a) A and D **(b)** B only **(c)** A, B, C, and D **(d)** none of these
5. According to the commutative law of addition,
   (a) $AB = BA$ **(b)** $A = A + A$
   (c) $A + (B + C) = (A + B) + C$ **(d)** $A + B = B + A$
6. According to the associative law of multiplication,
   (a) $B = BB$ **(b)** $A(BC) = (AB)C$ **(c)** $A + B = B + A$ **(d)** $B + B(B + 0)$
7. According to the distributive law,
   (a) $A(B + C) = AB + AC$ **(b)** $A(BC) = ABC$ **(c)** $A(A + 1) = A$ **(d)** $A + AB = A$
8. Which one of the following is *not* a valid rule of Boolean algebra?
   (a) $A + 1 = 1$ **(b)** $A = \overline{A}$ **(c)** $AA = A$ **(d)** $A + 0 = A$
9. Which of the following rules states that if one input of an AND gate is always 1, the output is equal to the other input?
   (a) $A + 1 = 1$ **(b)** $A + A = A$ **(c)** $A \cdot A = A$ **(d)** $A \cdot 1 = A$
10. According to DeMorgan's theorems, the following equality(s) is (are) correct:
    (a) $\overline{AB} = \overline{A} + \overline{B}$ **(b)** $\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$
    (c) $\overline{A + B + C} = \overline{A}\,\overline{B}\,\overline{C}$ **(d)** all of these
11. The Boolean expression $X = AB + CD$ represents
    (a) two ORs ANDed together **(b)** a 4-input AND gate
    (c) two ANDs ORed together **(d)** an exclusive-OR
12. An example of a sum-of-products expression is
    (a) $A + B(C + D)$ **(b)** $\overline{A}B + A\overline{C} + A\overline{B}C$
    (c) $(\overline{A} + B + C)(A + \overline{B} + C)$ **(d)** both answers (a) and (b)
13. An example of a product-of-sums expression is
    (a) $A(B + C) + A\overline{C}$ **(b)** $(A + B)(\overline{A} + B + \overline{C})$
    (c) $\overline{A} + \overline{B} + BC$ **(d)** both answers (a) and (b)
14. An example of a standard SOP expression is
    (a) $\overline{A}B + A\overline{B}C + AB\overline{D}$ **(b)** $A\overline{B}C + A\overline{C}D$
    (c) $A\overline{B} + \overline{A}B + AB$ **(d)** $A\overline{B}C\overline{D} + \overline{A}B + \overline{A}$
15. A 3-variable Karnaugh map has
    (a) eight cells **(b)** three cells **(c)** sixteen cells **(d)** four cells
16. In a 4-variable Karnaugh map, a 2-variable product term is produced by
    (a) a 2-cell group of 1s **(b)** an 8-cell group of 1s
    (c) a 4-cell group of 1s **(d)** a 4-cell group of 0s
17. On a Karnaugh map, grouping the 0s produces
    (a) a product-of-sums expression **(b)** a sum-of-products expression
    (c) a "don't care" condition **(d)** AND-OR logic
18. A 5-variable Karnaugh map has
    (a) sixteen cells **(b)** thirty-two cells **(c)** sixty-four cells
19. An SPLD that has a programmable AND array and a fixed OR array is a
    (a) PROM **(b)** PLA **(c)** PAL **(d)** GAL
20. VHDL is a type of
    (a) programmable logic **(b)** hardware description language
    (c) programmable array **(d)** logical mathematics
21. In VHDL, a port is
    (a) a type of entity **(b)** a type of architecture
    (c) an input or output **(d)** a type of variable

PROBLEMS

Answers to odd-numbered problems are at the end of the book.

SECTION 4–1 Boolean Operations and Expressions

1. Using Boolean notation, write an expression that is a 1 whenever one or more of its variables (A, B, C, and D) are 1s.
2. Write an expression that is a 1 only if all of its variables (A, B, C, D, and E) are 1s.
3. Write an expression that is a 1 when one or more of its variables (A, B, and C) are 0s.
4. Evaluate the following operations:
   (a) $0 + 0 + 1$ **(b)** $1 + 1 + 1$ **(c)** $1 \cdot 0 \cdot 0$
   (d) $1 \cdot 1 \cdot 1$ **(e)** $1 \cdot 0 \cdot 1$ **(f)** $1 \cdot 1 + 0 \cdot 1 \cdot 1$
5. Find the values of the variables that make each product term 1 and each sum term 0.
   (a) AB **(b)** $A\overline{B}C$ **(c)** $A + B$ **(d)** $\overline{A} + B + \overline{C}$
   (e) $\overline{A} + \overline{B} + C$ **(f)** $\overline{A} + B$ **(g)** $A\overline{B}\,\overline{C}$
6. Find the value of X for all possible values of the variables.
   (a) $X = (A + B)C + B$ **(b)** $X = (\overline{A + B})C$ **(c)** $X = A\overline{B}C + AB$
   (d) $X = (A + B)(\overline{A} + B)$ **(e)** $X = (A + BC)(\overline{B} + \overline{C})$

SECTION 4–2 Laws and Rules of Boolean Algebra

7. Identify the law of Boolean algebra upon which each of the following equalities is based:
   (a) $A\overline{B} + CD + A\overline{C}D + B = B + A\overline{B} + A\overline{C}D + CD$
   (b) $AB\overline{C}D + \overline{ABC} = D\overline{C}BA + \overline{CBA}$
   (c) $AB(CD + E\overline{F} + GH) = ABCD + ABE\overline{F} + ABGH$
8. Identify the Boolean rule(s) on which each of the following equalities is based:
   (a) $\overline{\overline{AB} + \overline{CD}} + \overline{EF} = AB + CD + \overline{EF}$ **(b)** $A\overline{A}B + AB\overline{C} + AB\overline{B} = AB\overline{C}$
   (c) $A(BC + BC) + AC = A(BC) + AC$ **(d)** $AB(C + \overline{C}) + AC = AB + AC$
   (e) $A\overline{B} + A\overline{B}C = A\overline{B}$ **(f)** $ABC + \overline{AB} + \overline{AB}CD = ABC + \overline{AB} + D$

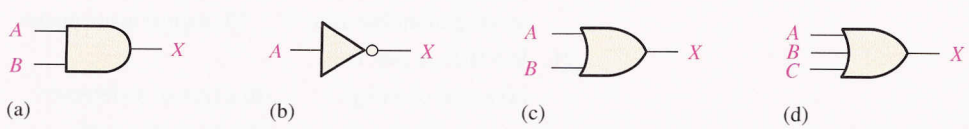
SECTION 4–3 DeMorgan's Theorems

9. Apply DeMorgan's theorems to each expression:
   (a) $\overline{A + \overline{B}}$ **(b)** $\overline{\overline{A}B}$ **(c)** $\overline{A + B + C}$ **(d)** $\overline{ABC}$
   (e) $\overline{A(B + C)}$ **(f)** $\overline{A\overline{B}} + \overline{CD}$ **(g)** $\overline{AB + CD}$ **(h)** $\overline{(A + \overline{B})(\overline{C} + D)}$
10. Apply DeMorgan's theorems to each expression:
    (a) $\overline{A\overline{B}(C + \overline{D})}$ **(b)** $\overline{AB(CD + EF)}$
    (c) $\overline{(A + \overline{B} + C + \overline{D})} + \overline{ABC\overline{D}}$ **(d)** $\overline{(\overline{A} + B + C + D)(A\overline{B}\,\overline{C}D)}$
    (e) $\overline{\overline{AB}(CD + \overline{E}F)(\overline{AB} + \overline{CD})}$
11. Apply DeMorgan's theorems to the following:
    (a) $\overline{(\overline{ABC})(\overline{EFG})} + \overline{(\overline{HIJ})(\overline{KLM})}$ **(b)** $\overline{(A + \overline{B\overline{C}} + CD)} + \overline{\overline{BC}}$
    (c) $\overline{(\overline{A + B})(\overline{C + D})(\overline{E + F})(\overline{G + H})}$

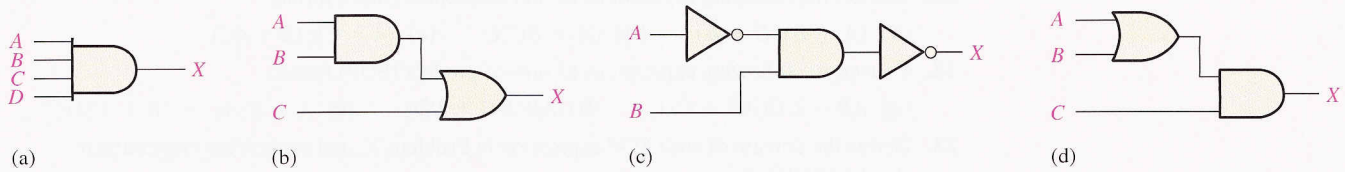
SECTION 4–4 Boolean Analysis of Logic Circuits

12. Write the Boolean expression for each of the logic gates in Figure 4–55.

▶ FIGURE 4–55



13. Write the Boolean expression for each of the logic circuits in Figure 4–56.



▲ **FIGURE 4–56**

14. Draw the logic circuit represented by each of the following expressions:
(a) $A + B + C$ **(b)** ABC **(c)** $AB + C$ **(d)** $AB + CD$

15. Draw the logic circuit represented by each expression:
(a) $A\overline{B} + \overline{A}B$ **(b)** $AB + \overline{A}\,\overline{B} + \overline{A}BC$
(c) $\overline{A}B(C + \overline{D})$ **(d)** $A + B[C + D(B + \overline{C})]$

16. Construct a truth table for each of the following Boolean expressions:
(a) $A + B$ **(b)** AB **(c)** $AB + BC$
(d) $(A + B)C$ **(e)** $(A + B)(\overline{B} + C)$

SECTION 4–5 Simplification Using Boolean Algebra

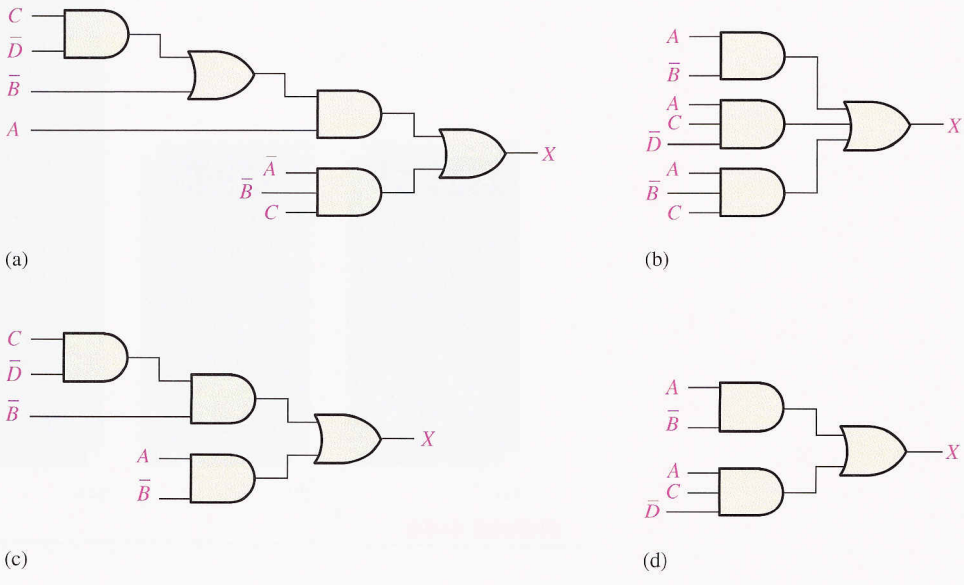
17. Using Boolean algebra techniques, simplify the following expressions as much as possible:
(a) $A(A + B)$ **(b)** $A(\overline{A} + AB)$ **(c)** $BC + \overline{B}C$
(d) $A(A + \overline{A}B)$ **(e)** $A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}C$

18. Using Boolean algebra, simplify the following expressions:
(a) $(A + \overline{B})(A + C)$ **(b)** $\overline{A}B + \overline{A}B\overline{C} + \overline{A}BCD + \overline{A}B\overline{C}\,\overline{D}E$
(c) $AB + \overline{A}B\overline{C} + A$ **(d)** $(A + \overline{A})(AB + AB\overline{C})$
(e) $AB + (\overline{A} + \overline{B})C + AB$

19. Using Boolean algebra, simplify each expression:
(a) $BD + B(D + E) + \overline{D}(D + F)$ **(b)** $\overline{A}\,\overline{B}C + \overline{(A + B + \overline{C})} + \overline{A}\,\overline{B}\,\overline{C}D$
(c) $(B + BC)(B + \overline{B}C)(B + D)$ **(d)** $ABCD + AB(\overline{CD}) + (\overline{AB})CD$
(e) $ABC[AB + \overline{C}(BC + AC)]$

20. Determine which of the logic circuits in Figure 4–57 are equivalent.

▶ **FIGURE 4–57**



SECTION 4–6 Standard Forms of Boolean Expressions

21. Convert the following expressions to sum-of-product (SOP) forms:

(a) $(A + B)(C + \overline{B})$ **(b)** $(A + \overline{B}C)C$ **(c)** $(A + C)(AB + AC)$

22. Convert the following expressions to sum-of-product (SOP) forms:

(a) $AB + CD(A\overline{B} + CD)$ **(b)** $AB(\overline{B}\,\overline{C} + BD)$ **(c)** $A + B[AC + (B + \overline{C})D]$

23. Define the domain of each SOP expression in Problem 21 and convert the expression to standard SOP form.

24. Convert each SOP expression in Problem 22 to standard SOP form.

25. Determine the binary value of each term in the standard SOP expressions from Problem 23.

26. Determine the binary value of each term in the standard SOP expressions from Problem 24.

27. Convert each standard SOP expression in Problem 23 to standard POS form.

28. Convert each standard SOP expression in Problem 24 to standard POS form.

SECTION 4–7 Boolean Expressions and Truth Tables

29. Develop a truth table for each of the following standard SOP expressions:

(a) $A\overline{B}C + \overline{A}B\overline{C} + ABC$ **(b)** $\overline{X}\overline{Y}Z + \overline{X}Y\overline{Z} + XY\overline{Z} + X\overline{Y}Z + \overline{X}YZ$

30. Develop a truth table for each of the following standard SOP expressions:

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

(b) $WXYZ + WXY\overline{Z} + \overline{W}XYZ + W\overline{X}YZ + WX\overline{Y}Z$

31. Develop a truth table for each of the SOP expressions:

(a) $\overline{A}B + AB\overline{C} + \overline{A}\,\overline{C} + A\overline{B}C$ **(b)** $\overline{X} + Y\overline{Z} + WZ + X\overline{Y}Z$

32. Develop a truth table for each of the standard POS expressions:

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

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

33. Develop a truth table for each of the standard POS expressions:

(a) $(A + B)(A + C)(A + B + C)$

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

34. For each truth table in Figure 4–58, derive a standard SOP and a standard POS expression.

ABC	X
000	0
001	1
010	0
011	0
100	1
101	1
110	0
111	1

(a)

ABC	X
000	0
001	0
010	0
011	0
100	0
101	1
110	1
111	1

(b)

ABCD	X
0000	1
0001	1
0010	0
0011	1
0100	0
0101	1
0110	1
0111	0
1000	0
1001	1
1010	0
1011	0
1100	1
1101	0
1110	0
1111	0

(c)

ABCD	X
0000	0
0001	0
0010	1
0011	0
0100	1
0101	1
0110	0
0111	1
1000	0
1001	0
1010	0
1011	1
1100	1
1101	0
1110	0
1111	1

(d)

▲ FIGURE 4–58

SECTION 4–8 The Karnaugh Map

35. Draw a 3-variable Karnaugh map and label each cell according to its binary value.

36. Draw a 4-variable Karnaugh map and label each cell according to its binary value.

37. Write the standard product term for each cell in a 3-variable Karnaugh map.

SECTION 4–9 Karnaugh MAP SOP Minimization

38. Use a Karnaugh map to find the minimum SOP form for each expression:

(a) $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}C$ **(b)** $AC(\overline{B} + C)$

(c) $\overline{A}(BC + B\overline{C}) + A(BC + B\overline{C})$ **(d)** $\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + AB\overline{C}$

39. Use a Karnaugh map to simplify each expression to a minimum SOP form:

(a) $\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C + \overline{A}BC + AB\overline{C}$ **(b)** $AC[\overline{B} + B(B + \overline{C})]$

(c) $DE\overline{F} + \overline{D}E\overline{F} + \overline{D}\,\overline{E}\,\overline{F}$

40. Expand each expression to a standard SOP form:

(a) $AB + A\overline{B}C + ABC$ **(b)** $A + BC$

(c) $A\overline{B}\,\overline{C}D + AC\overline{D} + B\overline{C}D + \overline{A}BC\overline{D}$ **(d)** $A\overline{B} + A\overline{B}\,\overline{C}D + CD + B\overline{C}D + ABCD$

41. Minimize each expression in Problem 40 with a Karnaugh map.

42. Use a Karnaugh map to reduce each expression to a minimum SOP form:

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

(b) $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + ABCD + ABC\overline{D}$

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

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

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

43. Reduce the function specified in the truth table in Figure 4–59 to its minimum SOP form by using a Karnaugh map.

44. Use the Karnaugh map method to implement the minimum SOP expression for the logic function specified in the truth table in Figure 4–60.

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

▲ FIGURE 4–59

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

▲ FIGURE 4–60

45. Solve Problem 44 for a situation in which the last six binary combinations are not allowed.

SECTION 4–10 Karnaugh Map POS Minimization

46. Use a Karnaugh map to find the minimum POS for each expression:

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

(b) $(X + \overline{Y})(\overline{X} + Z)(X + \overline{Y} + \overline{Z})(\overline{X} + \overline{Y} + Z)$

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

47. Use a Karnaugh map to simplify each expression to minimum POS form:

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

(b) $(X + \overline{Y})(W + \overline{Z})(\overline{X} + \overline{Y} + \overline{Z})(W + X + Y + Z)$

48. For the function specified in the truth table of Figure 4–59, determine the minimum POS expression using a Karnaugh map.

49. Determine the minimum POS expression for the function in the truth table of Figure 4–60.

50. Convert each of the following POS expressions to minimum SOP expressions using a Karnaugh map:

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

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

SECTION 4–11 Five-Variable Karnaugh Maps

51. Minimize the following SOP expression using a Karnaugh map:

$$X = \overline{A}B\overline{C}D\overline{E} + \overline{A}\,B\,\overline{C}DE + A\overline{B}\,\overline{C}DE + AB\overline{C}\,D\,\overline{E} + \overline{A}BCD\overline{E} + \overline{A}BC\overline{D}E + \overline{A}\,B\,\overline{C}\,D\,\overline{E} + \overline{A}\,\overline{B}CDE + AB\overline{C}D\overline{E} + AB\overline{C}DE$$

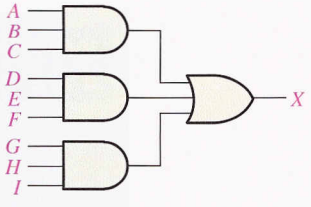
52. Apply the Karnaugh map method to minimize the following SOP expression:

$$A = \overline{V}WXYZ + V\overline{W}XYZ + VW\overline{X}YZ + VWX\overline{Y}Z + VWXY\overline{Z} + \overline{V}\,\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + \overline{V}\,\overline{W}X\overline{Y}\overline{Z} + \overline{V}W\overline{X}\overline{Y}\overline{Z} + \overline{V}W\overline{X}Y\overline{Z}$$

SECTION 4–12 VHDL (optional)

53. Write a VHDL program for the logic circuit in Figure 4–61.

▶ **FIGURE 4–61**



54. Write a program in VHDL for the expression

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

Digital System Application

55. If you are required to choose a type of digital display for low light conditions, will you select LED or LCD 7-segment displays? Why?

56. Explain why the codes 1010, 1011, 1100, 1101, 1110, and 1111 fall into the "don't care" category in 7-segment display applications.

57. For segment *b*, how many fewer gates and inverters does it take to implement the minimum SOP expression than the standard SOP expression?

58. Repeat Problem 57 for the logic for segments *c* through *g*.

Special Design Problems

59. The logic for segment *a* in Figure 4–52 produces a HIGH output to activate the segment and so do the circuits for each of the other segments. If a type of 7-segment display is used that requires a LOW to activate each segment, modify the segment logic accordingly.

60. Redesign the logic for segment *a* using a minimum POS approach. Which is simpler, minimum POS or the minimum SOP?

61. Repeat Problem 60 for segments *b* through *g*.

62. Summarize the results of your redesign effort in Problems 60 and 61 and recommend the best design based on fewer ICs. Specify the types of ICs.

Multisim Troubleshooting Practice

63. Open file P04-63, apply input signals, and observe the operation of the logic circuit. Determine whether or not a fault exists.

64. Open file P04-64, apply input signals, and observe the operation of the logic circuit. Determine whether or not a fault exists.

65. Open file P04-65, apply input signals, and observe the operation of the logic circuit. Determine whether or not a fault exists.

ANSWERS

SECTION REVIEWS

SECTION 4–1 Boolean Operations and Expressions

1. $\overline{A} = \overline{0} = 1$ **2.** $A = 1, B = 1, C = 0; \overline{A} + \overline{B} + C = \overline{1} + \overline{1} + 0 = 0 + 0 + 0 = 0$

3. $A = 1, B = 0, C = 1; A\overline{B}C = 1 \cdot \overline{0} \cdot 1 = 1 \cdot 1 \cdot 1 = 1$

SECTION 4–2 Laws and Rules of Boolean Algebra

1. $A + (B + C + D) = (A + B + C) + D$ **2.** $A(B + C + D) = AB + AC + AD$

SECTION 4–3 DeMorgan's Theorems

1. **(a)** $\overline{ABC} + \overline{(\overline{D} + E)} = \overline{A} + \overline{B} + \overline{C} + D\overline{E}$ **(b)** $\overline{(A + B)C} = \overline{A}\,\overline{B} + \overline{C}$

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

SECTION 4–4 Boolean Analysis of Logic Circuits

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

2. Abbreviated truth table: The expression is a 1 when A is 1 or when B and C are 1s or when B and D are 1s. The expression is 0 for all other variable combinations.

SECTION 4–5 Simplification Using Boolean Algebra

1. **(a)** $A + AB + A\overline{B}C = A$ **(b)** $(\overline{A} + B)C + ABC = C(\overline{A} + B)$

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

2. **(a)** *Original:* 2 AND gates, 1 OR gate, 1 inverter; *Simplified:* No gates (straight connection)

(b) *Original:* 2 OR gates, 2 AND gates, 1 inverter; *Simplified:* 1 OR gate, 1 AND gate, 1 inverter

(c) *Original:* 5 AND gates, 2 OR gates, 2 inverters; *Simplified:* 2 AND gates, 1 OR gate, 2 inverters

SECTION 4–6 Standard Forms of Boolean Expressions

1. **(a)** SOP **(b)** standard POS **(c)** standard SOP **(d)** POS

2. **(a)** $A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}CD + ABC\overline{D} + ABCD + \overline{A}\overline{B}CD + \overline{A}BCD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}BC\overline{D}$

(c) Already standard

3. **(b)** Already standard

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

SECTION 4–7 Boolean Expressions and Truth Tables

1. $2^5 = 32$ **2.** $0110 \longrightarrow \overline{W}XY\overline{Z}$ **3.** $1100 \longrightarrow \overline{W} + \overline{X} + Y + Z$

SECTION 4–8 The Karnaugh Map

1. **(a)** upper left cell: 000 **(b)** lower right cell: 101 **(c)** lower left cell: 100
(d) upper right cell: 001

2. **(a)** upper left cell: $\overline{X}\,\overline{Y}\,\overline{Z}$ **(b)** lower right cell: $X\overline{Y}Z$ **(c)** lower left cell: $X\overline{Y}\,\overline{Z}$
(d) upper right cell: $\overline{X}\,\overline{Y}Z$

3. **(a)** upper left cell: 0000 **(b)** lower right cell: 1010 **(c)** lower left cell: 1000
(d) upper right cell: 0010

4. **(a)** upper left cell: $\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$ **(b)** lower right cell: $W\overline{X}Y\overline{Z}$ **(c)** lower left cell: $W\overline{X}\,\overline{Y}\,\overline{Z}$
(d) upper right cell: $\overline{W}\overline{X}YZ$

SECTION 4–9 Karnaugh Map SOP Minimization

1. 8-cell map for 3 variables; 16-cell map for 4 variables

2. $AB + B\overline{C} + \overline{A}\,\overline{B}C$

3. **(a)** $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + AB\overline{C} + AB\overline{C}$
(b) $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + ABC + A\overline{B}\,\overline{C} + A\overline{B}C$
(c) $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}B\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}\,\overline{C}D + A\overline{B}C\overline{D}$
(d) $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + AB\overline{C}D + A\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$

SECTION 4–10 Karnaugh Map POS Minimization

1. In mapping a POS expression, 0s are placed in cells whose value makes the standard sum term zero; and in mapping an SOP expression 1s are placed in cells having the same values as the product terms.

2. 0 in the 1011 cell: $\overline{A} + B + \overline{C} + \overline{D}$ **3.** 1 in the 0010 cell: $\overline{A}\,\overline{B}C\overline{D}$

SECTION 4–11 Five-Variable Karnaugh Maps

1. There are 32 combinations of 5 variables ($2^5 = 32$).

2. $X = 1$ because the function is 1 for all possible combinations of 5 variables.

SECTION 4–12 VHDL (optional)

1. An HDL is a hardware description language for programmable logic.

2. Entity and architecture

3. The entity specifies the inputs and outputs of a logic function.

4. The architecture specifies operation of a logic function.

RELATED PROBLEMS FOR EXAMPLES

4–1 $\overline{A} + B = 0$ when $A = 1$ and $B = 0$.

4–2 $\overline{A}\,\overline{B} = 1$ when $A = 0$ and $B = 0$. **4–3** XYZ

4–4 $W + X + Y + Z$ **4–5** $ABC\overline{D}\,\overline{E}$ **4–6** $(A + \overline{B} + \overline{C}D)\overline{E}$

4–7 $\overline{ABCD} = \overline{A} + \overline{B} + \overline{C} + \overline{D}$ **4–8** $A\overline{B}$ **4–9** CD

4–10 $AB\overline{C} + \overline{A}C + \overline{A}\,\overline{B}$

4–11 $\overline{A} + \overline{B} + \overline{C}$ **4–12** $\overline{A}B\overline{C} + AB + A\overline{C} + A\overline{B} + B\overline{C}$

4–13 $W\overline{X}YZ + W\overline{X}Y\overline{Z} + W\overline{X}\overline{Y}Z + \overline{W}\overline{X}YZ + WX\overline{Y}Z + WX\overline{Y}\overline{Z}$

4–14 011, 101, 110, 010, 111. Yes

4–15 $(A + \overline{B} + C)(A + \overline{B} + \overline{C})(A + B + C)(\overline{A} + B + C)$

4–16 010, 100, 001, 111, 011. Yes **4–17** SOP and POS expressions are equivalent.

4–18 See Table 4–11. **4–19** See Table 4–12.

▼ **TABLE 4–11**

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

▼ **TABLE 4–12**

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

4–20 The SOP and POS expressions are equivalent. **4–21** See Figure 4–62.

4–22 See Figure 4–63. **4–23** See Figure 4–64. **4–24** See Figure 4–65.

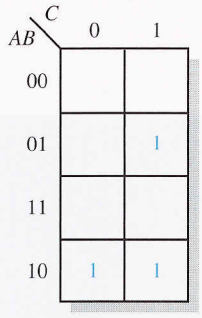
4–25 No other ways **4–26** $X = B + \overline{A}C + A\overline{C}D + C\overline{D}$

4–27 $X = \overline{D} + A\overline{B}C + B\overline{C} + \overline{A}B$

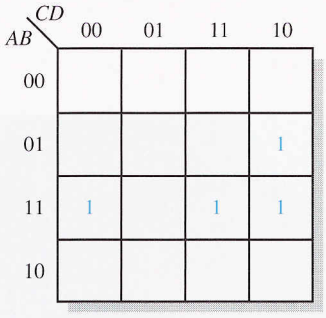
4–28 $Q = X + Y$

4–29 $Q = \overline{X}\,\overline{Y}\,\overline{Z} + W\overline{X}Z + \overline{W}YZ$

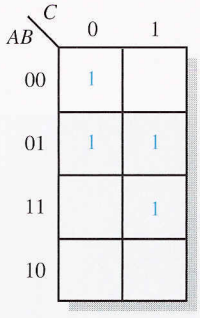
4–30 See Figure 4–66.



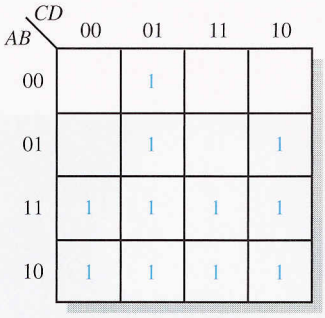
▲ **FIGURE 4–62**



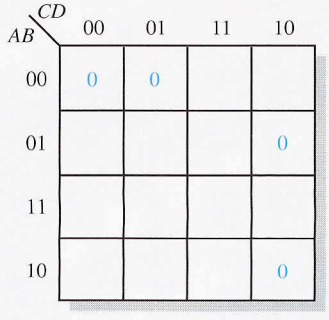
▲ **FIGURE 4–63**



▲ **FIGURE 4–64**



▲ **FIGURE 4–65**



▲ **FIGURE 4–66**

4–31 $Q = (X + \overline{Y})(X + \overline{Z})(\overline{X} + Y + Z)$

4–32 $Q = (\overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + Z)(W + X + Y + Z)(W + \overline{X} + Y + \overline{Z})$

4–33 $Q = \overline{Y}\,\overline{Z} + \overline{X}\,\overline{Z} + \overline{W}Y + \overline{X}\,\overline{Y}Z$

4–34 $Y = \overline{D}\,\overline{E} + \overline{A}\,\overline{E} + \overline{B}\,\overline{C}\,\overline{E}$

4–35 X ⇐ (A **and** B) **nor** (C **and** D);

SELF-TEST

1. (d) **2.** (a) **3.** (b) **4.** (c) **5.** (d) **6.** (b) **7.** (a) **8.** (b)
9. (d) **10.** (d) **11.** (c) **12.** (b) **13.** (b) **14.** (c) **15.** (a) **16.** (c)
17. (a) **18.** (b) **19.** (c) **20.** (b) **21.** (c)