CHAPTER

# 12 Boolean Algebra

Rosen



The circuits in computers and other electronic devices have inputs, each of which is either a 0 or a 1, and produce outputs that are also 0s and 1s. Circuits can be constructed using any basic element that has two different states. Such elements include switches that can be in either the on or the off position and optical devices that can be either lit or unlit. In 1938 Claude Shannon showed how the basic rules of logic, first given by George Boole in 1854 in his *The Laws of Thought*, could be used to design circuits. These rules form the basis for Boolean algebra. In this chapter we develop the basic properties of Boolean algebra. The operation of a circuit is defined by a Boolean function that specifies the value of an output for each set of inputs. The first step in constructing a circuit is to represent its Boolean function by an expression built up using the basic operations of Boolean algebra. We will provide an algorithm for producing such expressions. The expression that we obtain may contain many more operations than are necessary to represent the function. Later in the chapter we will describe methods for finding an expression with the minimum number of sums and products that represents a Boolean function. The procedures that we will develop, Karnaugh maps and the Quine–McCluskey method, are important in the design of efficient circuits.

## 12.1 Boolean Functions

### Introduction

Boolean algebra provides the operations and the rules for working with the set $\{0, 1\}$. Electronic and optical switches can be studied using this set and the rules of Boolean algebra. The three operations in Boolean algebra that we will use most are complementation, the Boolean sum, and the Boolean product. The **complement** of an element, denoted with a bar, is defined by $\overline{0} = 1$ and $\overline{1} = 0$. The Boolean sum, denoted by $+$ or by *OR*, has the following values:

$$1 + 1 = 1, \qquad 1 + 0 = 1, \qquad 0 + 1 = 1, \qquad 0 + 0 = 0.$$

The Boolean product, denoted by $\cdot$ or by *AND*, has the following values:

$$1 \cdot 1 = 1, \qquad 1 \cdot 0 = 0, \qquad 0 \cdot 1 = 0, \qquad 0 \cdot 0 = 0.$$

When there is no danger of confusion, the symbol $\cdot$ can be deleted, just as in writing algebraic products. Unless parentheses are used, the rules of precedence for Boolean operators are: first, all complements are computed, followed by all Boolean products, followed by all Boolean sums. This is illustrated in Example 1.

**EXAMPLE 1** Find the value of $1 \cdot 0 + \overline{(0 + 1)}$.

*Solution:* Using the definitions of complementation, the Boolean sum, and the Boolean product, it follows that

$$\begin{aligned} 1 \cdot 0 + \overline{(0 + 1)} &= 0 + \overline{1} \\ &= 0 + 0 \\ &= 0. \end{aligned}$$

◀

The complement, Boolean sum, and Boolean product correspond to the logical operators, $\neg$, $\vee$, and $\wedge$, respectively, where 0 corresponds to **F** (false) and 1 corresponds to **T** (true). Equalities in Boolean algebra can be directly translated into equivalences of compound propositions. Conversely, equivalences of compound propositions can be translated into equalities in Boolean algebra. We will see later in this section why these translations yield valid logical equivalences and identities in Boolean algebra. Example 2 illustrates the translation from Boolean algebra to propositional logic.

**EXAMPLE 2** Translate $1 \cdot 0 + \overline{(0 + 1)} = 0$, the equality found in Example 1, into a logical equivalence.

*Solution:* We obtain a logical equivalence when we translate each 1 into a **T**, each 0 into an **F**, each Boolean sum into a disjunction, each Boolean product into a conjunction, and each complementation into a negation. We obtain

$$(\mathbf{T} \wedge \mathbf{F}) \vee \neg(\mathbf{T} \vee \mathbf{F}) \equiv \mathbf{F}.$$ ◀

Example 3 illustrates the translation from propositional logic to Boolean algebra.

**EXAMPLE 3** Translate the logical equivalence $(\mathbf{T} \wedge \mathbf{T}) \vee \neg\mathbf{F} \equiv \mathbf{T}$ into an identity in Boolean algebra.

*Solution:* We obtain an identity in Boolean algebra when we translate each **T** into a 1, each **F** into a 0, each disjunction into a Boolean sum, each conjunction into a Boolean product, and each negation into a complementation. We obtain

$$(1 \cdot 1) + \overline{0} = 1.$$ ◀

## Boolean Expressions and Boolean Functions

Let $B = \{0, 1\}$. Then $B^n = \{(x_1, x_2, \ldots, x_n) \mid x_i \in B \text{ for } 1 \leq i \leq n\}$ is the set of all possible $n$-tuples of 0s and 1s. The variable $x$ is called a **Boolean variable** if it assumes values only from $B$, that is, if its only possible values are 0 and 1. A function from $B^n$ to $B$ is called a **Boolean function of degree $n$**.

CLAUDE ELWOOD SHANNON (1916–2001) Claude Shannon was born in Petoskey, Michigan, and grew up in Gaylord, Michigan. His father was a businessman and a probate judge, and his mother was a language teacher and a high school principal. Shannon attended the University of Michigan, graduating in 1936. He continued his studies at M.I.T., where he took the job of maintaining the differential analyzer, a mechanical computing device consisting of shafts and gears built by his professor, Vannevar Bush. Shannon's master's thesis, written in 1936, studied the logical aspects of the differential analyzer. This master's thesis presents the first application of Boolean algebra to the design of switching circuits; it is perhaps the most famous master's thesis of the twentieth century. He received his Ph.D. from M.I.T. in 1940. Shannon joined Bell Laboratories in 1940, where he worked on transmitting data efficiently. He was one of the first people to use bits to represent information. At Bell Laboratories he worked on determining the amount of traffic that telephone lines can carry. Shannon made many fundamental contributions to information theory. In the early 1950s he was one of the founders of the study of artificial intelligence. He joined the M.I.T. faculty in 1956, where he continued his study of information theory.

Shannon had an unconventional side. He is credited with inventing the rocket-powered Frisbee. He is also famous for riding a unicycle down the hallways of Bell Laboratories while juggling four balls. Shannon retired when he was 50 years old, publishing papers sporadically over the following 10 years. In his later years he concentrated on some pet projects, such as building a motorized pogo stick. One interesting quote from Shannon, published in *Omni Magazine* in 1987, is "I visualize a time when we will be to robots what dogs are to humans. And I am rooting for the machines."

**EXAMPLE 4** The function $F(x, y) = x\overline{y}$ from the set of ordered pairs of Boolean variables to the set $\{0, 1\}$ is a Boolean function of degree 2 with $F(1, 1) = 0$, $F(1, 0) = 1$, $F(0, 1) = 0$, and $F(0, 0) = 0$. We display these values of $F$ in Table 1. ◀

**TABLE 1**

| $x$ | $y$ | $F(x,y)$ |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Boolean functions can be represented using expressions made up from variables and Boolean operations. The **Boolean expressions** in the variables $x_1, x_2, \ldots, x_n$ are defined recursively as

$0, 1, x_1, x_2, \ldots, x_n$ are Boolean expressions;
if $E_1$ and $E_2$ are Boolean expressions, then $\overline{E}_1$, $(E_1E_2)$, and $(E_1 + E_2)$ are Boolean expressions.

Each Boolean expression represents a Boolean function. The values of this function are obtained by substituting 0 and 1 for the variables in the expression. In Section 12.2 we will show that every Boolean function can be represented by a Boolean expression.

**EXAMPLE 5** Find the values of the Boolean function represented by $F(x, y, z) = xy + \overline{z}$.

*Solution:* The values of this function are displayed in Table 2. ◀

**TABLE 2**

| $x$ | $y$ | $z$ | $xy$ | $\overline{z}$ | $F(x, y, z) = xy + \overline{z}$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 |

Note that we can represent a Boolean function graphically by distinguishing the vertices of the $n$-cube that correspond to the $n$-tuples of bits where the function has value 1.

**EXAMPLE 6** The function $F(x, y, z) = xy + \overline{z}$ from $B^3$ to $B$ from Example 5 can be represented by distinguishing the vertices that correspond to the five 3-tuples $(1, 1, 1)$, $(1, 1, 0)$, $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 0)$, where $F(x, y, z) = 1$, as shown in Figure 1. These vertices are displayed using solid black circles. ◀

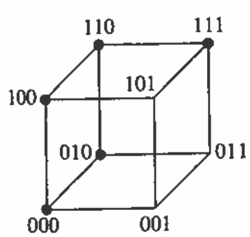



**FIGURE 1**

Boolean functions $F$ and $G$ of $n$ variables are equal if and only if $F(b_1, b_2, \ldots, b_n) = G(b_1, b_2, \ldots, b_n)$ whenever $b_1, b_2, \ldots, b_n$ belong to $B$. Two different Boolean expressions that represent the same function are called **equivalent**. For instance, the Boolean expressions $xy$, $xy + 0$, and $xy \cdot 1$ are equivalent. The **complement** of the Boolean function $F$ is the function $\overline{F}$, where $\overline{F}(x_1, \ldots, x_n) = \overline{F(x_1, \ldots, x_n)}$. Let $F$ and $G$ be Boolean functions of degree $n$. The **Boolean sum** $F + G$ and the **Boolean product** $FG$ are defined by

$$(F + G)(x_1, \ldots, x_n) = F(x_1, \ldots, x_n) + G(x_1, \ldots, x_n),$$
$$(FG)(x_1, \ldots, x_n) = F(x_1, \ldots, x_n)G(x_1, \ldots, x_n).$$

A Boolean function of degree two is a function from a set with four elements, namely, pairs of elements from $B = \{0, 1\}$, to $B$, a set with two elements. Hence, there are 16 different Boolean functions of degree two. In Table 3 we display the values of the 16 different Boolean functions of degree two, labeled $F_1, F_2, \ldots, F_{16}$.

complementation also does not appear.) This transforms the Boolean identity into the logical equivalence

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r).$$

This logical equivalence is one of the distributive laws for propositional logic in Table 6 in Section 1.3. ◀

Identities in Boolean algebra can be used to prove further identities. We demonstrate this in Example 10.

**EXAMPLE 10** Prove the **absorption law** $x(x + y) = x$ using the other identities of Boolean algebra shown in Table 5. (This is called an absorption law because absorbing $x + y$ into $x$ leaves $x$ unchanged.)

Extra Examples

*Solution:* We display steps used to derive this identity and the law used in each step:

| | |
|---|---|
| $x(x + y) = (x + 0)(x + y)$ | Identity law for the Boolean sum |
| $= x + 0 \cdot y$ | Distributive law of the Boolean sum over the Boolean product |
| $= x + y \cdot 0$ | Commutative law for the Boolean product |
| $= x + 0$ | Domination law for the Boolean product |
| $= x$ | Identity law for the Boolean sum. |

◀

## Duality

The identities in Table 5 come in pairs (except for the law of the double complement and the unit and zero properties). To explain the relationship between the two identities in each pair we use the concept of a dual. The **dual** of a Boolean expression is obtained by interchanging Boolean sums and Boolean products and interchanging 0s and 1s.

**EXAMPLE 11** Find the duals of $x(y + 0)$ and $\overline{x} \cdot 1 + (\overline{y} + z)$.

*Solution:* Interchanging $\cdot$ signs and $+$ signs and interchanging 0s and 1s in these expressions produces their duals. The duals are $x + (y \cdot 1)$ and $(\overline{x} + 0)(\overline{y}z)$, respectively. ◀

The dual of a Boolean function $F$ represented by a Boolean expression is the function represented by the dual of this expression. This dual function, denoted by $F^d$, does not depend on the particular Boolean expression used to represent $F$. An identity between functions represented by Boolean expressions remains valid when the duals of both sides of the identity are taken. (See Exercise 30 for the reason why this is true.) This result, called the **duality principle**, is useful for obtaining new identities.

**EXAMPLE 12** Construct an identity from the absorption law $x(x + y) = x$ by taking duals.

*Solution:* Taking the duals of both sides of this identity produces the identity $x + xy = x$, which is also called an absorption law and is shown in Table 5. ◀

## The Abstract Definition of a Boolean Algebra

In this section we have focused on Boolean functions and expressions. However, the results we have established can be translated into results about propositions or results about sets. Because of this, it is useful to define Boolean algebras abstractly. Once it is shown that a particular structure is a Boolean algebra, then all results established about Boolean algebras in general apply to this particular structure.

Boolean algebras can be defined in several ways. The most common way is to specify the properties that operations must satisfy, as is done in Definition 1.

**DEFINITION 1** A *Boolean algebra* is a set $B$ with two binary operations $\vee$ and $\wedge$, elements 0 and 1, and a unary operation $^{-}$ such that these properties hold for all $x$, $y$, and $z$ in $B$:

$$\left.\begin{aligned} x \vee 0 &= x \\ x \wedge 1 &= x \end{aligned}\right\} \quad \text{Identity laws}$$

$$\left.\begin{aligned} x \vee \overline{x} &= 1 \\ x \wedge \overline{x} &= 0 \end{aligned}\right\} \quad \text{Complement laws}$$

$$\left.\begin{aligned} (x \vee y) \vee z &= x \vee (y \vee z) \\ (x \wedge y) \wedge z &= x \wedge (y \wedge z) \end{aligned}\right\} \quad \text{Associative laws}$$

$$\left.\begin{aligned} x \vee y &= y \vee x \\ x \wedge y &= y \wedge x \end{aligned}\right\} \quad \text{Commutative laws}$$

$$\left.\begin{aligned} x \vee (y \wedge z) &= (x \vee y) \wedge (x \vee z) \\ x \wedge (y \vee z) &= (x \wedge y) \vee (x \wedge z) \end{aligned}\right\} \quad \text{Distributive laws}$$

Using the laws given in Definition 1, it is possible to prove many other laws that hold for every Boolean algebra, such as idempotent and domination laws. (See Exercises 35–42.)

From our previous discussion, $B = \{0, 1\}$ with the *OR* and *AND* operations and the complement operator, satisfies all these properties. The set of propositions in $n$ variables, with the $\vee$ and $\wedge$ operators, **F** and **T**, and the negation operator, also satisfies all the properties of a Boolean algebra, as can be seen from Table 6 in Section 1.3. Similarly, the set of subsets of a universal set $U$ with the union and intersection operations, the empty set and the universal set, and the set complementation operator, is a Boolean algebra as can be seen by consulting Table 1 in Section 2.2. So, to establish results about each of Boolean expressions, propositions, and sets, we need only prove results about abstract Boolean algebras.

Boolean algebras may also be defined using the notion of a lattice, discussed in Chapter 9. Recall that a lattice $L$ is a partially ordered set in which every pair of elements $x$, $y$ has a least upper bound, denoted by $\text{lub}(x, y)$ and a greatest lower bound denoted by $\text{glb}(x, y)$. Given two elements $x$ and $y$ of $L$, we can define two operations $\vee$ and $\wedge$ on pairs of elements of $L$ by $x \vee y = \text{lub}(x, y)$ and $x \wedge y = \text{glb}(x, y)$.

For a lattice $L$ to be a Boolean algebra as specified in Definition 1, it must have two properties. First, it must be **complemented**. For a lattice to be complemented it must have a least element 0 and a greatest element 1 and for every element $x$ of the lattice there must exist an element $\overline{x}$ such that $x \vee \overline{x} = 1$ and $x \wedge \overline{x} = 0$. Second, it must be **distributive**. This means that for every $x$, $y$, and $z$ in $L$, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ and $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$. Showing that a complemented, distributive lattice is a Boolean algebra has been left as Supplementary Exercise 39 in Chapter 9.

## Exercises

1. Find the values of these expressions.
   a) $1 \cdot \overline{0}$ b) $1 + \overline{1}$ c) $\overline{0} \cdot 0$ d) $\overline{(1 + 0)}$
2. Find the values, if any, of the Boolean variable $x$ that satisfy these equations.
   a) $x \cdot 1 = 0$ b) $x + x = 0$
   c) $x \cdot 1 = x$ d) $x \cdot \overline{x} = 1$
3. a) Show that $(1 \cdot 1) + (\overline{0 \cdot 1} + 0) = 1$.
   b) Translate the equation in part (a) into a propositional equivalence by changing each 0 into an **F**, each 1 into a **T**, each Boolean sum into a disjunction, each Boolean product into a conjunction, each complementation into a negation, and the equals sign into a propositional equivalence sign.
4. a) Show that $(\overline{1} \cdot \overline{0}) + (1 \cdot \overline{0}) = 1$.
   b) Translate the equation in part (a) into a propositional equivalence by changing each 0 into an **F**, each 1 into a **T**, each Boolean sum into a disjunction, each Boolean product into a conjunction, each complementation into a negation, and the equals sign into a propositional equivalence sign.
5. Use a table to express the values of each of these Boolean functions.
   a) $F(x, y, z) = \overline{x}y$
   b) $F(x, y, z) = x + yz$
   c) $F(x, y, z) = x\overline{y} + \overline{(xyz)}$
   d) $F(x, y, z) = x(yz + \overline{y}\,\overline{z})$
6. Use a table to express the values of each of these Boolean functions.
   a) $F(x, y, z) = \overline{z}$
   b) $F(x, y, z) = \overline{x}y + \overline{y}z$
   c) $F(x, y, z) = x\overline{y}z + \overline{(xyz)}$
   d) $F(x, y, z) = \overline{y}(xz + \overline{x}\,\overline{z})$
7. Use a 3-cube $Q_3$ to represent each of the Boolean functions in Exercise 5 by displaying a black circle at each vertex that corresponds to a 3-tuple where this function has the value 1.
8. Use a 3-cube $Q_3$ to represent each of the Boolean functions in Exercise 6 by displaying a black circle at each vertex that corresponds to a 3-tuple where this function has the value 1.
9. What values of the Boolean variables $x$ and $y$ satisfy $xy = x + y$?
10. How many different Boolean functions are there of degree 7?
11. Prove the absorption law $x + xy = x$ using the other laws in Table 5.
12. Show that $F(x, y, z) = xy + xz + yz$ has the value 1 if and only if at least two of the variables $x$, $y$, and $z$ have the value 1.
13. Show that $x\overline{y} + y\overline{z} + \overline{x}z = \overline{x}y + \overline{y}z + x\overline{z}$.

Exercises 14–23 deal with the Boolean algebra {0, 1} with addition, multiplication, and complement defined at the beginning of this section. In each case, use a table as in Example 8.

14. Verify the law of the double complement.
15. Verify the idempotent laws.
16. Verify the identity laws.
17. Verify the domination laws.
18. Verify the commutative laws.
19. Verify the associative laws.
20. Verify the first distributive law in Table 5.
21. Verify De Morgan's laws.
22. Verify the unit property.
23. Verify the zero property.

The Boolean operator $\oplus$, called the *XOR* operator, is defined by $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 1 = 1$, and $0 \oplus 0 = 0$.

24. Simplify these expressions.
    a) $x \oplus 0$ b) $x \oplus 1$
    c) $x \oplus x$ d) $x \oplus \overline{x}$
25. Show that these identities hold.
    a) $x \oplus y = (x + y)\overline{(xy)}$
    b) $x \oplus y = (x\overline{y}) + (\overline{x}y)$
26. Show that $x \oplus y = y \oplus x$.
27. Prove or disprove these equalities.
    a) $x \oplus (y \oplus z) = (x \oplus y) \oplus z$
    b) $x + (y \oplus z) = (x + y) \oplus (x + z)$
    c) $x \oplus (y + z) = (x \oplus y) + (x \oplus z)$
28. Find the duals of these Boolean expressions.
    a) $x + y$ b) $\overline{x}\,\overline{y}$
    c) $xyz + \overline{x}\,\overline{y}\,\overline{z}$ d) $x\overline{z} + x \cdot 0 + \overline{x} \cdot 1$
*29. Suppose that $F$ is a Boolean function represented by a Boolean expression in the variables $x_1, \ldots, x_n$. Show that $F^d(x_1, \ldots, x_n) = \overline{F(\overline{x}_1, \ldots, \overline{x}_n)}$.
*30. Show that if $F$ and $G$ are Boolean functions represented by Boolean expressions in $n$ variables and $F = G$, then $F^d = G^d$, where $F^d$ and $G^d$ are the Boolean functions represented by the duals of the Boolean expressions representing $F$ and $G$, respectively. [*Hint:* Use the result of Exercise 29.]
*31. How many different Boolean functions $F(x, y, z)$ are there such that $F(\overline{x}, \overline{y}, \overline{z}) = F(x, y, z)$ for all values of the Boolean variables $x$, $y$, and $z$?
*32. How many different Boolean functions $F(x, y, z)$ are there such that $F(\overline{x}, y, z) = F(x, \overline{y}, z) = F(x, y, \overline{z})$ for all values of the Boolean variables $x$, $y$, and $z$?
33. Show that you obtain De Morgan's laws for propositions (in Table 6 in Section 1.3) when you transform De Morgan's laws for Boolean algebra in Table 6 into logical equivalences.
34. Show that you obtain the absorption laws for propositions (in Table 6 in Section 1.3) when you transform the absorption laws for Boolean algebra in Table 6 into logical equivalences.

In Exercises 35–42, use the laws in Definition 1 to show that the stated properties hold in every Boolean algebra.

35. Show that in a Boolean algebra, the **idempotent laws** $x \vee x = x$ and $x \wedge x = x$ hold for every element $x$.

36. Show that in a Boolean algebra, every element $x$ has a unique complement $\overline{x}$ such that $x \vee \overline{x} = 1$ and $x \wedge \overline{x} = 0$.

37. Show that in a Boolean algebra, the complement of the element 0 is the element 1 and vice versa.

38. Prove that in a Boolean algebra, the **law of the double complement** holds; that is, $\overline{\overline{x}} = x$ for every element $x$.

39. Show that **De Morgan's laws** hold in a Boolean algebra. That is, show that for all $x$ and $y$, $\overline{(x \vee y)} = \overline{x} \wedge \overline{y}$ and $\overline{(x \wedge y)} = \overline{x} \vee \overline{y}$.

40. Show that in a Boolean algebra, the **modular properties** hold. That is, show that $x \wedge (y \vee (x \wedge z)) = (x \wedge y) \vee (x \wedge z)$ and $x \vee (y \wedge (x \vee z)) = (x \vee y) \wedge (x \vee z)$.

41. Show that in a Boolean algebra, if $x \vee y = 0$, then $x = 0$ and $y = 0$, and that if $x \wedge y = 1$, then $x = 1$ and $y = 1$.

42. Show that in a Boolean algebra, the **dual** of an identity, obtained by interchanging the $\vee$ and $\wedge$ operators and interchanging the elements 0 and 1, is also a valid identity.

43. Show that a complemented, distributive lattice is a Boolean algebra.

# 12.2 Representing Boolean Functions

Two important problems of Boolean algebra will be studied in this section. The first problem is: Given the values of a Boolean function, how can a Boolean expression that represents this function be found? This problem will be solved by showing that any Boolean function can be represented by a Boolean sum of Boolean products of the variables and their complements. The solution of this problem shows that every Boolean function can be represented using the three Boolean operators $\cdot$, $+$, and $^{-}$. The second problem is: Is there a smaller set of operators that can be used to represent all Boolean functions? We will answer this question by showing that all Boolean functions can be represented using only one operator. Both of these problems have practical importance in circuit design.

## Sum-of-Products Expansions

We will use examples to illustrate one important way to find a Boolean expression that represents a Boolean function.

**EXAMPLE 1** Find Boolean expressions that represent the functions $F(x, y, z)$ and $G(x, y, z)$, which are given in Table 1.

**TABLE 1**

| $x$ | $y$ | $z$ | $F$ | $G$ |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

*Solution:* An expression that has the value 1 when $x = z = 1$ and $y = 0$, and the value 0 otherwise, is needed to represent $F$. Such an expression can be formed by taking the Boolean product of $x$, $\overline{y}$, and $z$. This product, $x\overline{y}z$, has the value 1 if and only if $x = \overline{y} = z = 1$, which holds if and only if $x = z = 1$ and $y = 0$.

To represent $G$, we need an expression that equals 1 when $x = y = 1$ and $z = 0$, or $x = z = 0$ and $y = 1$. We can form an expression with these values by taking the Boolean sum of two different Boolean products. The Boolean product $xy\overline{z}$ has the value 1 if and only if $x = y = 1$ and $z = 0$. Similarly, the product $\overline{x}y\overline{z}$ has the value 1 if and only if $x = z = 0$ and $y = 1$. The Boolean sum of these two products, $xy\overline{z} + \overline{x}y\overline{z}$, represents $G$, because it has the value 1 if and only if $x = y = 1$ and $z = 0$, or $x = z = 0$ and $y = 1$. ◀

Example 1 illustrates a procedure for constructing a Boolean expression representing a function with given values. Each combination of values of the variables for which the function has the value 1 leads to a Boolean product of the variables or their complements.

**DEFINITION 1** A *literal* is a Boolean variable or its complement. A *minterm* of the Boolean variables $x_1, x_2, \ldots, x_n$ is a Boolean product $y_1 y_2 \cdots y_n$, where $y_i = x_i$ or $y_i = \overline{x}_i$. Hence, a minterm is a product of $n$ literals, with one literal for each variable.

A minterm has the value 1 for one and only one combination of values of its variables. More precisely, the minterm $y_1 y_2 \ldots y_n$ is 1 if and only if each $y_i$ is 1, and this occurs if and only if $x_i = 1$ when $y_i = x_i$ and $x_i = 0$ when $y_i = \overline{x}_i$.

**EXAMPLE 2** Find a minterm that equals 1 if $x_1 = x_3 = 0$ and $x_2 = x_4 = x_5 = 1$, and equals 0 otherwise.

*Solution:* The minterm $\overline{x}_1 x_2 \overline{x}_3 x_4 x_5$ has the correct set of values. ◀

By taking Boolean sums of distinct minterms we can build up a Boolean expression with a specified set of values. In particular, a Boolean sum of minterms has the value 1 when exactly one of the minterms in the sum has the value 1. It has the value 0 for all other combinations of values of the variables. Consequently, given a Boolean function, a Boolean sum of minterms can be formed that has the value 1 when this Boolean function has the value 1, and has the value 0 when the function has the value 0. The minterms in this Boolean sum correspond to those combinations of values for which the function has the value 1. The sum of minterms that represents the function is called the **sum-of-products expansion** or the **disjunctive normal form** of the Boolean function.

Links

(See Exercise 42 in Section 1.3 for the development of disjunctive normal form in propositional calculus.)

**EXAMPLE 3** Find the sum-of-products expansion for the function $F(x, y, z) = (x + y)\overline{z}$.

Extra Examples

*Solution:* We will find the sum-of-products expansion of $F(x, y, z)$ in two ways. First, we will use Boolean identities to expand the product and simplify. We find that

$$\begin{aligned} F(x, y, z) &= (x + y)\overline{z} & \\ &= x\overline{z} + y\overline{z} & \text{Distributive law} \\ &= x1\overline{z} + 1y\overline{z} & \text{Identity law} \\ &= x(y + \overline{y})\overline{z} + (x + \overline{x})y\overline{z} & \text{Unit property} \\ &= xy\overline{z} + x\overline{y}\,\overline{z} + xy\overline{z} + \overline{x}y\overline{z} & \text{Distributive law} \\ &= xy\overline{z} + x\overline{y}\,\overline{z} + \overline{x}y\,\overline{z}. & \text{Idempotent law} \end{aligned}$$

Second, we can construct the sum-of-products expansion by determining the values of $F$ for all possible values of the variables $x$, $y$, and $z$. These values are found in Table 2. The sum-of-products expansion of $F$ is the Boolean sum of three minterms corresponding to the three rows of this table that give the value 1 for the function. This gives

$$F(x, y, z) = xy\overline{z} + x\overline{y}\,\overline{z} + \overline{x}y\overline{z}.$$ ◀

It is also possible to find a Boolean expression that represents a Boolean function by taking a Boolean product of Boolean sums. The resulting expansion is called the **conjunctive normal form** or **product-of-sums expansion** of the function. These expansions can be found from sum-of-products expansions by taking duals. How to find such expansions directly is described in Exercise 10.

**TABLE 2**

| $x$ | $y$ | $z$ | $x+y$ | $\overline{z}$ | $(x+y)\overline{z}$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |

## Functional Completeness

Every Boolean function can be expressed as a Boolean sum of minterms. Each minterm is the Boolean product of Boolean variables or their complements. This shows that every Boolean function can be represented using the Boolean operators $\cdot$, $+$, and $^{-}$. Because every Boolean function can be represented using these operators we say that the set $\{\cdot, +, ^{-}\}$ is **functionally complete**. Can we find a smaller set of functionally complete operators? We can do so if one of the three operators of this set can be expressed in terms of the other two. This can be done using one of De Morgan's laws. We can eliminate all Boolean sums using the identity

$$x + y = \overline{\overline{x}\,\overline{y}},$$

which is obtained by taking complements of both sides in the second De Morgan law, given in Table 5 in Section 12.1, and then applying the double complementation law. This means that the set $\{\cdot, ^{-}\}$ is functionally complete. Similarly, we could eliminate all Boolean products using the identity

$$xy = \overline{\overline{x} + \overline{y}},$$

which is obtained by taking complements of both sides in the first De Morgan law, given in Table 5 in Section 12.1, and then applying the double complementation law. Consequently $\{+, ^{-}\}$ is functionally complete. Note that the set $\{+, \cdot\}$ is not functionally complete, because it is impossible to express the Boolean function $F(x) = \overline{x}$ using these operators (see Exercise 19).

We have found sets containing two operators that are functionally complete. Can we find a smaller set of functionally complete operators, namely, a set containing just one operator? Such sets exist. Define two operators, the $\mid$ or ***NAND*** operator, defined by $1 \mid 1 = 0$ and $1 \mid 0 = 0 \mid 1 = 0 \mid 0 = 1$; and the $\downarrow$ or ***NOR*** operator, defined by $1 \downarrow 1 = 1 \downarrow 0 = 0 \downarrow 1 = 0$ and $0 \downarrow 0 = 1$. Both of the sets $\{\mid\}$ and $\{\downarrow\}$ are functionally complete. To see that $\{\mid\}$ is functionally complete, because $\{\cdot, ^{-}\}$ is functionally complete, all that we have to do is show that both of the operators $\cdot$ and $^{-}$ can be expressed using just the $\mid$ operator. This can be done as

$$\overline{x} = x \mid x,$$
$$xy = (x \mid y) \mid (x \mid y).$$

The reader should verify these identities (see Exercise 14). We leave the demonstration that $\{\downarrow\}$ is functionally complete for the reader (see Exercises 15 and 16).

## Exercises

**1.** Find a Boolean product of the Boolean variables $x$, $y$, and $z$, or their complements, that has the value 1 if and only if

a) $x = y = 0, z = 1$. b) $x = 0, y = 1, z = 0$.
c) $x = 0, y = z = 1$. d) $x = y = z = 0$.

**2.** Find the sum-of-products expansions of these Boolean functions.

a) $F(x, y) = \overline{x} + y$ b) $F(x, y) = x\,\overline{y}$
c) $F(x, y) = 1$ d) $F(x, y) = \overline{y}$

**3.** Find the sum-of-products expansions of these Boolean functions.

a) $F(x, y, z) = x + y + z$
b) $F(x, y, z) = (x + z)y$
c) $F(x, y, z) = x$
d) $F(x, y, z) = x\,\overline{y}$

**4.** Find the sum-of-products expansions of the Boolean function $F(x, y, z)$ that equals 1 if and only if

a) $x = 0$. b) $xy = 0$.
c) $x + y = 0$. d) $xyz = 0$.

**5.** Find the sum-of-products expansion of the Boolean function $F(w, x, y, z)$ that has the value 1 if and only if an odd number of $w$, $x$, $y$, and $z$ have the value 1.

**6.** Find the sum-of-products expansion of the Boolean function $F(x_1, x_2, x_3, x_4, x_5)$ that has the value 1 if and only if three or more of the variables $x_1, x_2, x_3, x_4$, and $x_5$ have the value 1.

Another way to find a Boolean expression that represents a Boolean function is to form a Boolean product of Boolean sums of literals. Exercises 7–11 are concerned with representations of this kind.

**7.** Find a Boolean sum containing either $x$ or $\overline{x}$, either $y$ or $\overline{y}$, and either $z$ or $\overline{z}$ that has the value 0 if and only if

a) $x = y = 1, z = 0$. b) $x = y = z = 0$.
c) $x = z = 0, y = 1$.

**8.** Find a Boolean product of Boolean sums of literals that has the value 0 if and only if $x = y = 1$ and $z = 0$, $x = z = 0$ and $y = 1$, or $x = y = z = 0$. [*Hint:* Take the Boolean product of the Boolean sums found in parts (a), (b), and (c) in Exercise 7.]

**9.** Show that the Boolean sum $y_1 + y_2 + \cdots + y_n$, where $y_i = x_i$ or $y_i = \overline{x}_i$, has the value 0 for exactly one combination of the values of the variables, namely, when $x_i = 0$ if $y_i = x_i$ and $x_i = 1$ if $y_i = \overline{x}_i$. This Boolean sum is called a **maxterm**.

**10.** Show that a Boolean function can be represented as a Boolean product of maxterms. This representation is called the **product-of-sums expansion** or **conjunctive normal form** of the function. [*Hint:* Include one maxterm in this product for each combination of the variables where the function has the value 0.]

**11.** Find the product-of-sums expansion of each of the Boolean functions in Exercise 3.

**12.** Express each of these Boolean functions using the operators $\cdot$ and $\overline{\phantom{x}}$.

a) $x + y + z$ b) $x + \overline{y}(\overline{x} + z)$
c) $\overline{x + \overline{y}}$ d) $\overline{x}(x + \overline{y} + \overline{z})$

**13.** Express each of the Boolean functions in Exercise 12 using the operators $+$ and $\overline{\phantom{x}}$.

**14.** Show that

a) $\overline{x} = x \mid x$. b) $xy = (x \mid y) \mid (x \mid y)$.
c) $x + y = (x \mid x) \mid (y \mid y)$.

**15.** Show that

a) $\overline{x} = x \downarrow x$.
b) $xy = (x \downarrow x) \downarrow (y \downarrow y)$.
c) $x + y = (x \downarrow y) \downarrow (x \downarrow y)$.

**16.** Show that $\{\downarrow\}$ is functionally complete using Exercise 15.

**17.** Express each of the Boolean functions in Exercise 3 using the operator $\mid$.

**18.** Express each of the Boolean functions in Exercise 3 using the operator $\downarrow$.

**19.** Show that the set of operators $\{+, \cdot\}$ is not functionally complete.

**20.** Are these sets of operators functionally complete?
a) $\{+, \oplus\}$ b) $\{\overline{\phantom{x}}, \oplus\}$ c) $\{\cdot, \oplus\}$

# 12.3 Logic Gates

## Introduction

Links

Boolean algebra is used to model the circuitry of electronic devices. Each input and each output of such a device can be thought of as a member of the set $\{0, 1\}$. A computer, or other electronic device, is made up of a number of circuits. Each circuit can be designed using the rules of Boolean algebra that were studied in Sections 12.1 and 12.2. The basic elements of circuits

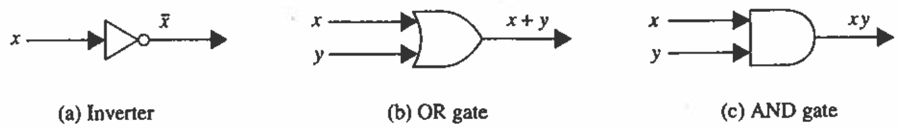



**FIGURE 1 Basic Types of Gates.**

are called **gates**, and were introduced in Section 1.2. Each type of gate implements a Boolean operation. In this section we define several types of gates. Using these gates, we will apply the rules of Boolean algebra to design circuits that perform a variety of tasks. The circuits that we will study in this chapter give output that depends only on the input, and not on the current state of the circuit. In other words, these circuits have no memory capabilities. Such circuits are called **combinational circuits** or **gating networks.**

We will construct combinational circuits using three types of elements. The first is an **inverter**, which accepts the value of one Boolean variable as input and produces the complement of this value as its output. The symbol used for an inverter is shown in Figure 1(a). The input to the inverter is shown on the left side entering the element, and the output is shown on the right side leaving the element.

The next type of element we will use is the **OR gate**. The inputs to this gate are the values of two or more Boolean variables. The output is the Boolean sum of their values. The symbol used for an OR gate is shown in Figure 1(b). The inputs to the OR gate are shown on the left side entering the element, and the output is shown on the right side leaving the element.

The third type of element we will use is the **AND gate**. The inputs to this gate are the values of two or more Boolean variables. The output is the Boolean product of their values. The symbol used for an AND gate is shown in Figure 1(c). The inputs to the AND gate are shown on the left side entering the element, and the output is shown on the right side leaving the element.

We will permit multiple inputs to AND and OR gates. The inputs to each of these gates are shown on the left side entering the element, and the output is shown on the right side. Examples of AND and OR gates with $n$ inputs are shown in Figure 2.

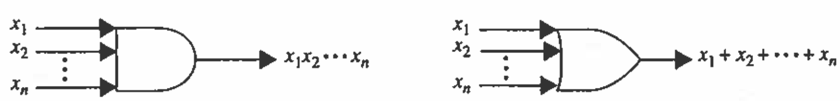



**FIGURE 2 Gates with *n* Inputs.**

## Combinations of Gates

Combinational circuits can be constructed using a combination of inverters, OR gates, and AND gates. When combinations of circuits are formed, some gates may share inputs. This is shown in one of two ways in depictions of circuits. One method is to use branchings that indicate all the gates that use a given input. The other method is to indicate this input separately for each gate. Figure 3 illustrates the two ways of showing gates with the same input values. Note also that output from a gate may be used as input by one or more other elements, as shown in Figure 3. Both drawings in Figure 3 depict the circuit that produces the output $xy + \bar{x}y$.

**EXAMPLE 1** Construct circuits that produce the following outputs: (a) $(x + y)\bar{x}$, (b) $\bar{x}\,\overline{(y + \bar{z})}$, and (c) $(x + y + z)(\bar{x}\,\bar{y}\,\bar{z})$.

*Solution:* Circuits that produce these outputs are shown in Figure 4. ◀

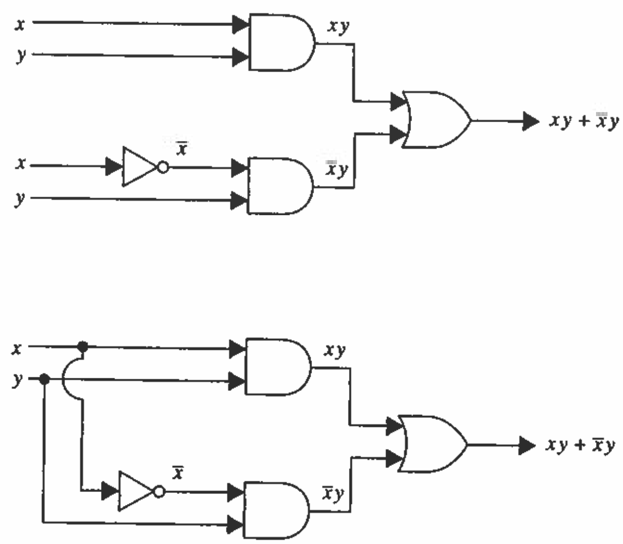



**FIGURE 3 Two Ways to Draw the Same Circuit.**

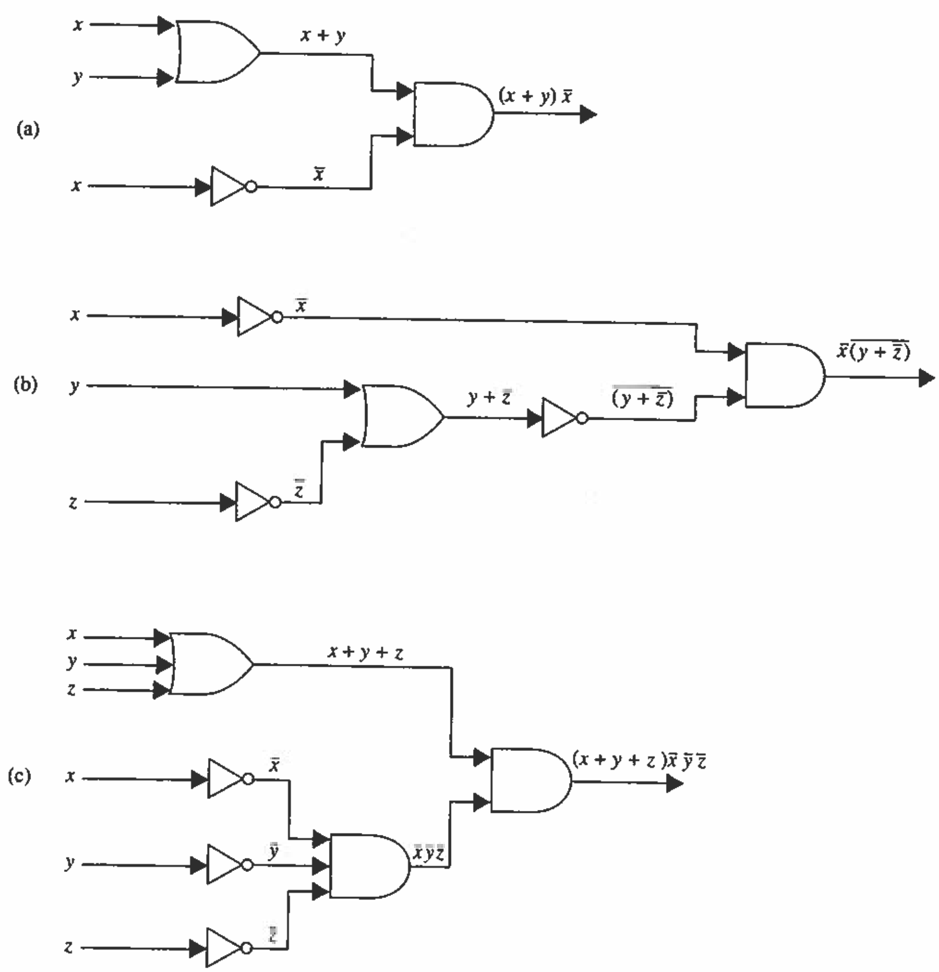



**FIGURE 4 Circuits that Produce the Outputs Specified in Example 1.**

## Examples of Circuits

We will give some examples of circuits that perform some useful functions.

**EXAMPLE 2** A committee of three individuals decides issues for an organization. Each individual votes either yes or no for each proposal that arises. A proposal is passed if it receives at least two yes votes. Design a circuit that determines whether a proposal passes.

Extra Examples

*Solution:* Let $x = 1$ if the first individual votes yes, and $x = 0$ if this individual votes no; let $y = 1$ if the second individual votes yes, and $y = 0$ if this individual votes no; let $z = 1$ if the third individual votes yes, and $z = 0$ if this individual votes no. Then a circuit must be designed that produces the output 1 from the inputs $x$, $y$, and $z$ when two or more of $x$, $y$, and $z$ are 1. One representation of the Boolean function that has these output values is $xy + xz + yz$ (see Exercise 12 in Section 12.1). The circuit that implements this function is shown in Figure 5. ◀

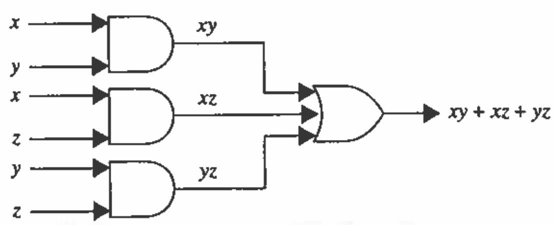



**FIGURE 5 A Circuit for Majority Voting.**

**EXAMPLE 3** Sometimes light fixtures are controlled by more than one switch. Circuits need to be designed so that flipping any one of the switches for the fixture turns the light on when it is off and turns the light off when it is on. Design circuits that accomplish this when there are two switches and when there are three switches.

**TABLE 1**

| $x$ | $y$ | $F(x, y)$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

*Solution:* We will begin by designing the circuit that controls the light fixture when two different switches are used. Let $x = 1$ when the first switch is closed and $x = 0$ when it is open, and let $y = 1$ when the second switch is closed and $y = 0$ when it is open. Let $F(x, y) = 1$ when the light is on and $F(x, y) = 0$ when it is off. We can arbitrarily decide that the light will be on when both switches are closed, so that $F(1, 1) = 1$. This determines all the other values of $F$. When one of the two switches is opened, the light goes off, so $F(1, 0) = F(0, 1) = 0$. When the other switch is also opened, the light goes on, so $F(0, 0) = 1$. Table 1 displays these values. Note that $F(x, y) = xy + \overline{x}\,\overline{y}$. This function is implemented by the circuit shown in Figure 6.

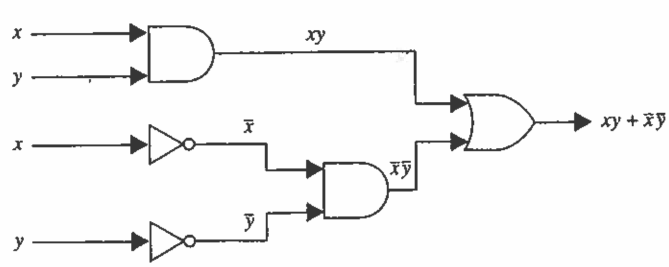



**FIGURE 6 A Circuit for a Light Controlled by Two Switches.**

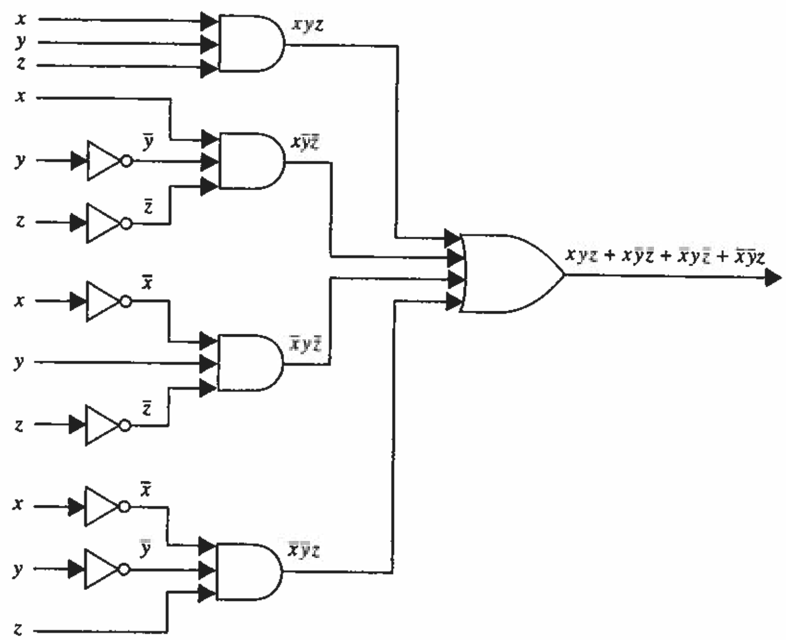



**FIGURE 7 A Circuit for a Fixture Controlled by Three Switches.**

**TABLE 2**

| $x$ | $y$ | $z$ | $F(x, y, z)$ |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 |

We will now design a circuit for three switches. Let $x$, $y$, and $z$ be the Boolean variables that indicate whether each of the three switches is closed. We let $x = 1$ when the first switch is closed, and $x = 0$ when it is open; $y = 1$ when the second switch is closed, and $y = 0$ when it is open; and $z = 1$ when the third switch is closed, and $z = 0$ when it is open. Let $F(x, y, z) = 1$ when the light is on and $F(x, y, z) = 0$ when the light is off. We can arbitrarily specify that the light be on when all three switches are closed, so that $F(1, 1, 1) = 1$. This determines all other values of $F$. When one switch is opened, the light goes off, so $F(1, 1, 0) = F(1, 0, 1) = F(0, 1, 1) = 0$. When a second switch is opened, the light goes on, so $F(1, 0, 0) = F(0, 1, 0) = F(0, 0, 1) = 1$. Finally, when the third switch is opened, the light goes off again, so $F(0, 0, 0) = 0$. Table 2 shows the values of this function.

The function $F$ can be represented by its sum-of-products expansion as $F(x, y, z) = xyz + x\overline{y}\,\overline{z} + \overline{x}y\overline{z} + \overline{x}\,\overline{y}z$. The circuit shown in Figure 7 implements this function. ◀

## Adders

Links

**TABLE 3 Input and Output for the Half Adder.**

| *Input* | | *Output* | |
|---|---|---|---|
| $x$ | $y$ | $s$ | $c$ |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 |

We will illustrate how logic circuits can be used to carry out addition of two positive integers from their binary expansions. We will build up the circuitry to do this addition from some component circuits. First, we will build a circuit that can be used to find $x + y$, where $x$ and $y$ are two bits. The input to our circuit will be $x$ and $y$, because these each have the value 0 or the value 1. The output will consist of two bits, namely, $s$ and $c$, where $s$ is the sum bit and $c$ is the carry bit. This circuit is called a **multiple output circuit** because it has more than one output. The circuit that we are designing is called the **half adder**, because it adds two bits, without considering a carry from a previous addition. We show the input and output for the half adder in Table 3. From Table 3 we see that $c = xy$ and that $s = x\overline{y} + \overline{x}y = (x + y)\overline{(xy)}$. Hence, the circuit shown in Figure 8 computes the sum bit $s$ and the carry bit $c$ from the bits $x$ and $y$.

We use the **full adder** to compute the sum bit and the carry bit when two bits and a carry are added. The inputs to the full adder are the bits $x$ and $y$ and the carry $c_i$. The outputs are the sum bit $s$ and the new carry $c_{i+1}$. The inputs and outputs for the full adder are shown in Table 4.

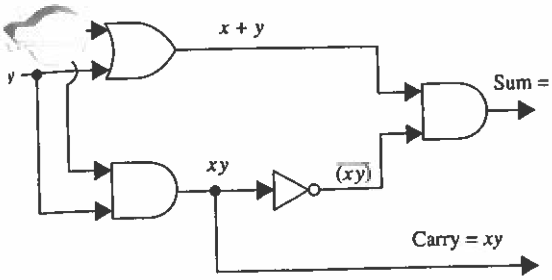



**FIGURE 8 The Half Adder.**

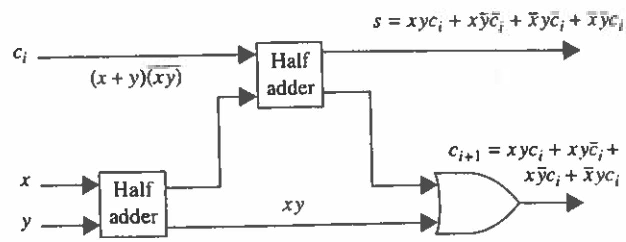



**FIGURE 9 A Full Adder.**

**TABLE 4 Input and Output for the Full Adder.**

| Input | | | Output | |
|---|---|---|---|---|
| $x$ | $y$ | $c_i$ | $s$ | $c_{i+1}$ |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| [illegible] | [illegible] | 0 | 1 | 0 |
| [illegible] | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |

The two outputs of the full adder, the sum bit $s$ and the carry $c_{i+1}$, are given by the sum-of-products expansions $xyc_i + x\overline{y}\,\overline{c}_i + \overline{x}y\overline{c}_i + \overline{x}\,\overline{y}c_i$ and $xyc_i + xy\overline{c}_i + x\overline{y}c_i + \overline{x}yc_i$, respectively. However, instead of designing the full adder from scratch, we will use half adders to produce the desired output. A full adder circuit using half adders is shown in Figure 9.

Finally, in Figure 10 we show how full and half adders can be used to add the two three-bit integers $(x_2x_1x_0)_2$ and $(y_2y_1y_0)_2$ to produce the sum $(s_3s_2s_1s_0)_2$. Note that $s_3$, the highest-order bit in the sum, is given by the carry $c_2$.

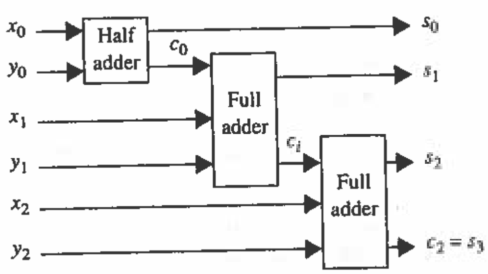
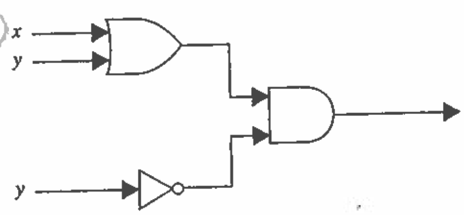



**FIGURE 10 Adding Two Three-Bit Integers with Full and Half Adders.**

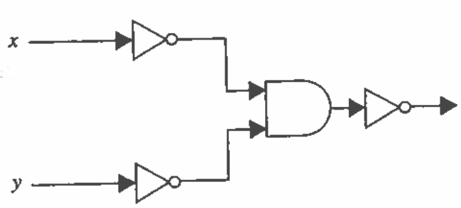
## Exercises

In Exercises 1–5 find the output of the given circuit.

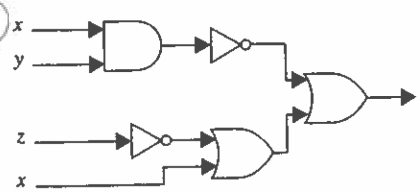
**1.**



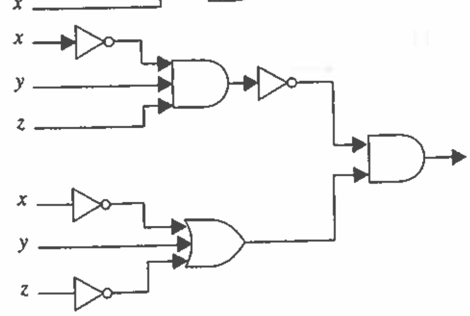

**2.**

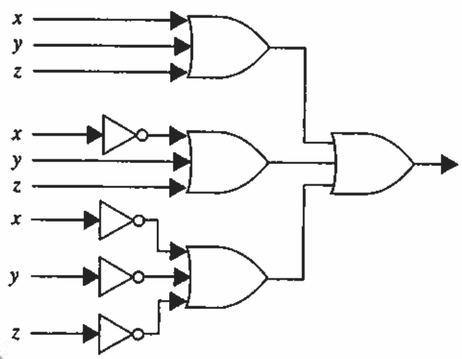



**3.**




**4.**



5. 


6. Construct circuits from inverters, AND gates, and OR gates to produce these outputs.
   a) $\overline{x} + y$
   b) $\overline{(x + y)x}$
   c) $xyz + \overline{x}\,\overline{y}\,\overline{z}$
   d) $\overline{(\overline{x} + z)(y + \overline{z})}$

7. Design a circuit that implements majority voting for five individuals.

8. Design a circuit for a light fixture controlled by four switches, where flipping one of the switches turns the light on when it is off and turns it off when it is on.

9. Show how the sum of two five-bit integers can be found using full and half adders.

10. Construct a circuit for a half subtractor using AND gates, OR gates, and inverters. A **half subtractor** has two bits as input and produces as output a difference bit and a borrow.

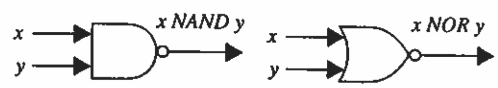
11. Construct a circuit for a full subtractor using AND gates, OR gates, and inverters. A **full subtractor** has two bits and a borrow as input, and produces as output a difference bit and a borrow.

12. Use the circuits from Exercises 10 and 11 to find the difference of two four-bit integers, where the first integer is greater than the second integer.

*13. Construct a circuit that compares the two-bit integers $(x_1x_0)_2$ and $(y_1y_0)_2$, returning an output of 1 when the first of these numbers is larger and an output of 0 otherwise.

*14. Construct a circuit that computes the product of the two-bit integers $(x_1x_0)_2$ and $(y_1y_0)_2$. The circuit should have four output bits for the bits in the product.

Two gates that are often used in circuits are NAND and NOR gates. When NAND or NOR gates are used to represent circuits, no other types of gates are needed. The notation for these gates is as follows:




*15. Use NAND gates to construct circuits with these outputs.
   a) $\overline{x}$
   b) $x + y$
   c) $xy$
   d) $x \oplus y$

*16. Use NOR gates to construct circuits for the outputs given in Exercise 15.

*17. Construct a half adder using NAND gates.

*18. Construct a half adder using NOR gates.

A **multiplexer** is a switching circuit that produces as output one of a set of input bits based on the value of control bits.

19. Construct a multiplexer using AND gates, OR gates, and inverters that has as input the four bits $x_0, x_1, x_2$, and $x_3$ and the two control bits $c_0$ and $c_1$. Set up the circuit so that $x_i$ is the output, where $i$ is the value of the two-bit integer $(c_1c_0)_2$.

The **depth** of a combinatorial circuit can be defined by specifying that the depth of the initial input is 0 and if a gate has $n$ different inputs at depths $d_1, d_2, \ldots, d_n$, respectively, then its outputs have depth equal to $\max(d_1, d_2, \ldots, d_n) + 1$; this value is also defined to be the depth of the gate. The depth of a combinatorial circuit is the maximum depth of the gates in the circuit.

20. Find the depth of
   a) the circuit constructed in Example 2 for majority voting among three people.
   b) the circuit constructed in Example 3 for a light controlled by two switches.
   c) the half adder shown in Figure 8.
   d) the full adder shown in Figure 9.

# 12.4 Minimization of Circuits

## Introduction

The efficiency of a combinational circuit depends on the number and arrangement of its gates. The process of designing a combinational circuit begins with the table specifying the output for each combination of input values. We can always use the sum-of-products expansion of a circuit to find a set of logic gates that will implement this circuit. However, the sum-of-products expansion