9. Show that the Boolean sum $y_1 + y_2 + \cdots + y_n$, where $y_i = x_i$ or $y_i = \bar{x}_i$, 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 $y_i = \bar{x}_i$. 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 $^-$.

a) $x + y + z$ **b)** $x + \bar{y}(\bar{x} + z)$

c) $\overline{x + \bar{y}}$ **d)** $\bar{x}(x + \bar{y} + \bar{z})$

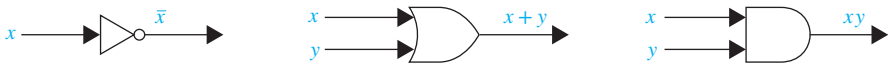
13. Express each of the Boolean functions in Exercise 12 using the operators $+$ and $^-$.

12.3 Logic Gates

12.3.1 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 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



(a) Inverter (b) OR gate (c) AND gate

FIGURE 1 **Basic types of gates.**

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.

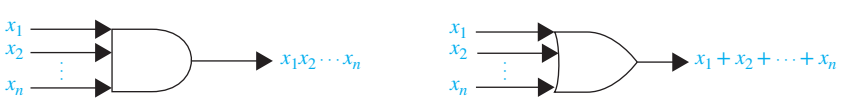


FIGURE 2 **Gates with n inputs.**

12.3.2 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. ◀

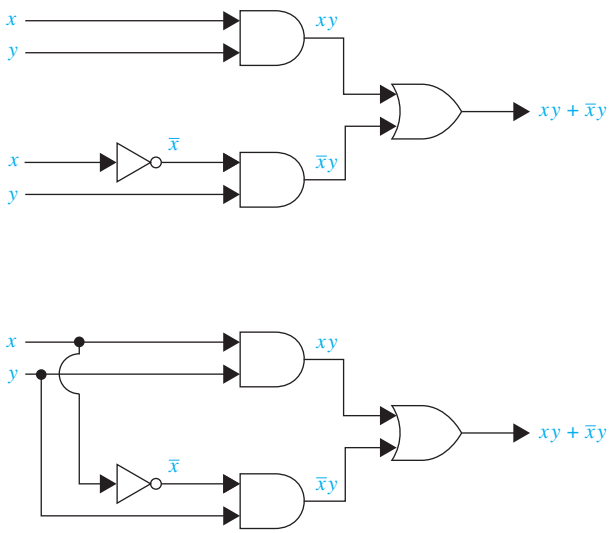


FIGURE 3 Two ways to draw the same circuit.

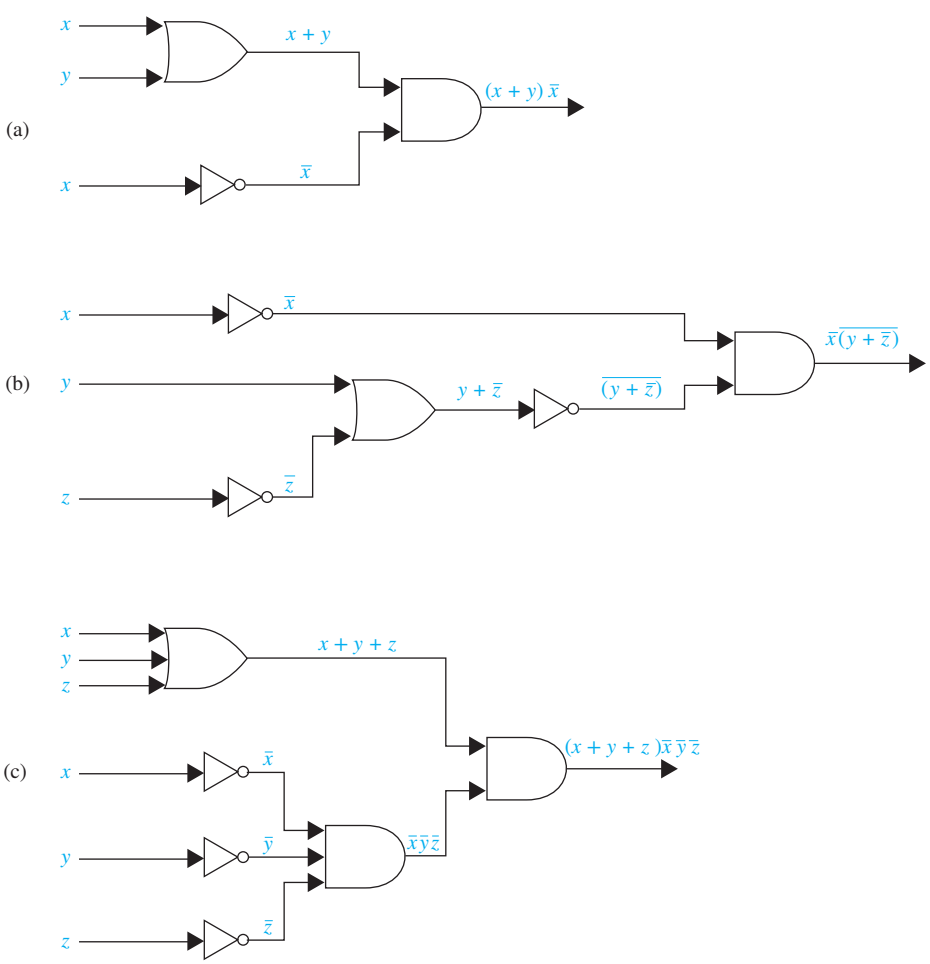


FIGURE 4 Circuits that produce the outputs specified in Example 1.

12.3.3 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. ◀

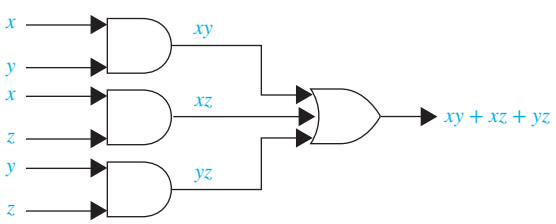


FIGURE 5 **A circuit for majority voting.**

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 $F(x, y) = xy + \bar{x}\,\bar{y}$. This function is implemented by the circuit shown in Figure 6.

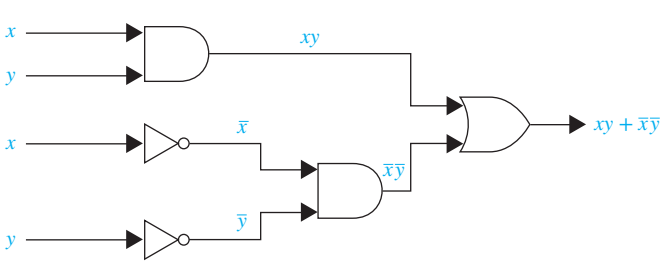


FIGURE 6 **A circuit for a light controlled by two switches.**

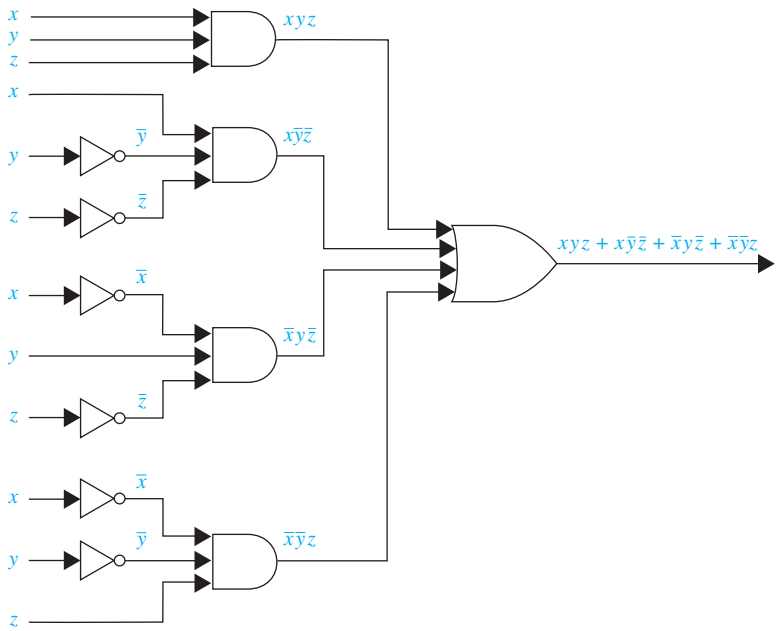


FIGURE 7 **A circuit for a fixture controlled by three switches.**

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 $F(x, y, z) = xyz + x\bar{y}\,\bar{z} + \bar{x}yz + \bar{x}\,\bar{y}z$. The circuit shown in Figure 7 implements this function. ◀

Links ❯

12.3.4 Adders

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 $s = x\bar{y} + \bar{x}y = (x + y)\overline{(xy)}$. 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.

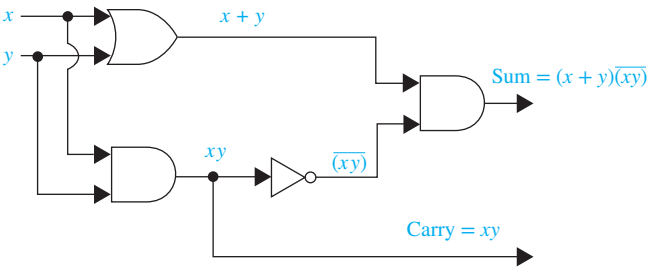


FIGURE 8 **The half adder.**

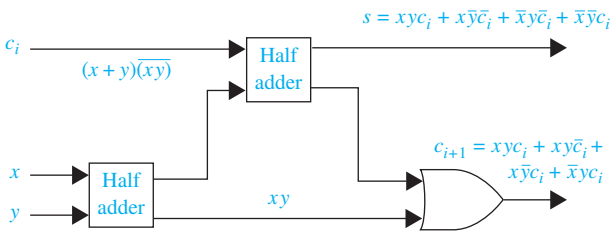


FIGURE 9 **A full adder.**

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
0	1	0	1	0
0	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 $xyc_i + x\bar{y}\,\bar{c}_i + \bar{x}y\bar{c}_i + \bar{x}\,\bar{y}c_i$ and $xyc_i + xy\bar{c}_i + x\bar{y}c_i + \bar{x}yc_i$, 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.

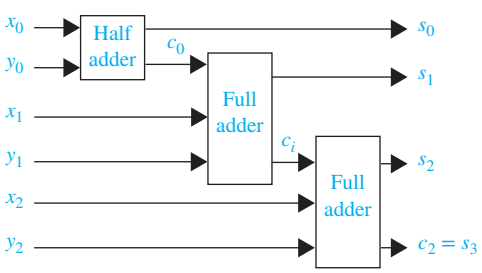
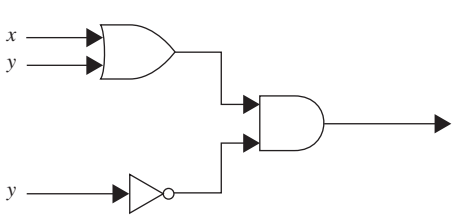


FIGURE 10 **Adding two three-bit integers with full and half adders.**

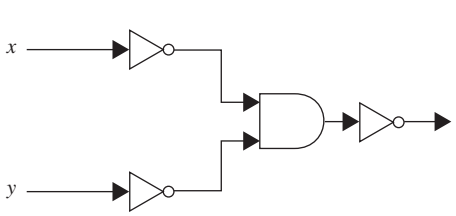
Exercises

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

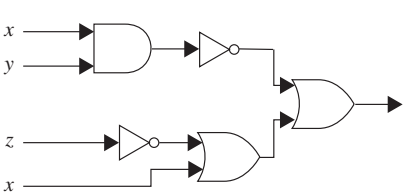
1.



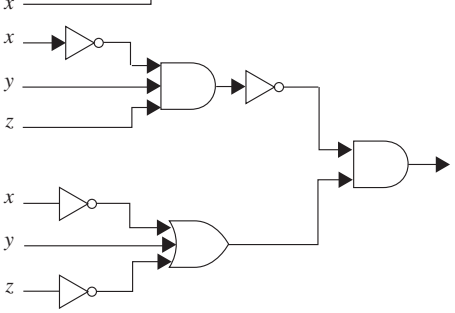
2.



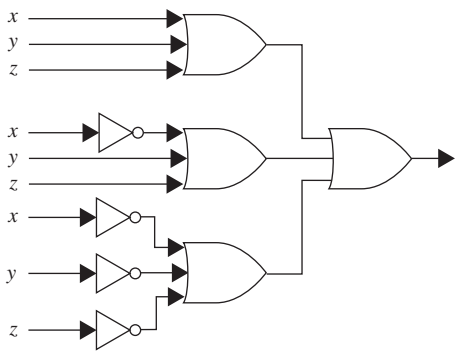
3.



4.



5.



6. Construct circuits from inverters, AND gates, and OR gates to produce these outputs.

a) $\bar{x} + y$ **b)** $\overline{(x + y)}x$

c) $xyz + \bar{x}\bar{y}\bar{z}$ **d)** $\overline{(\bar{x} + z)(y + \bar{z})}$

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.

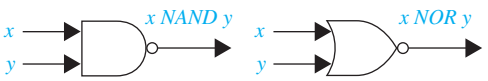
11. Construct a circuit for a full subtractor using AND gates, OR gates, and inverters. A **full subtractor** has two bits and a borrow as input, and produces as output a difference bit and a borrow.

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) $\bar{x}$ **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

12.4.1 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 may contain many more terms than are necessary. Terms in a sum-of-products expansion that differ in just one variable, so that in one term this variable occurs and in the other term the complement of this variable occurs, can be combined. For instance, consider the circuit that has