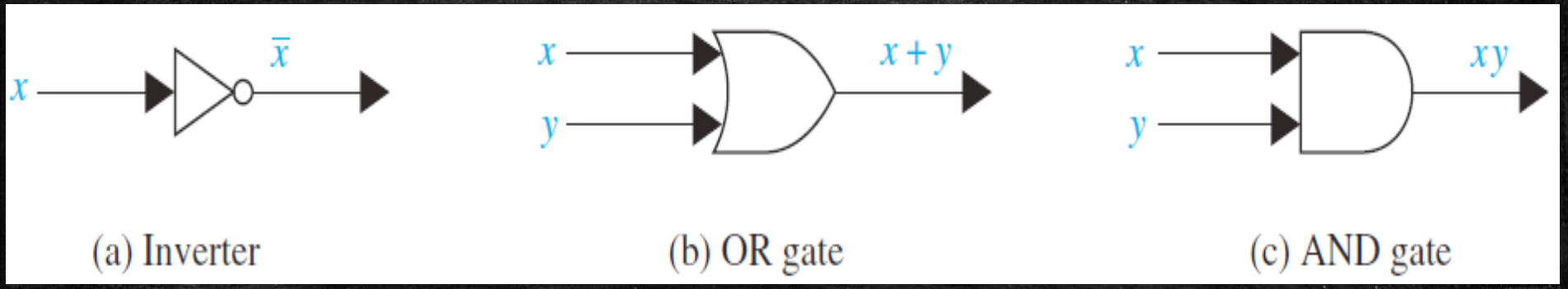
Boolean Algebra (cont.)

Assylbek Issakhov,
Ph.D., professor

Logic Gates

- 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 previous lecture. The basic elements of circuits are called **gates**. Each type of gate implements a Boolean operation.

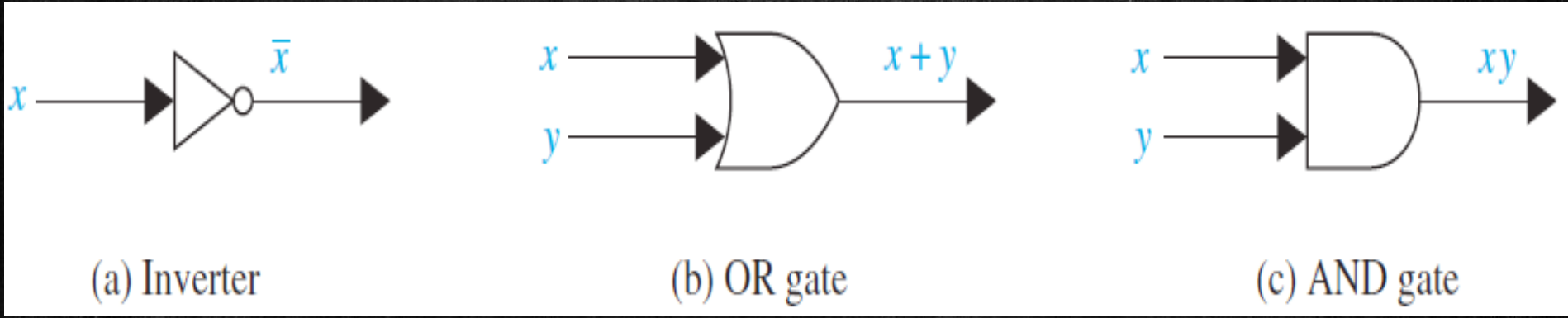
Basic Types of Gates



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

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 (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.

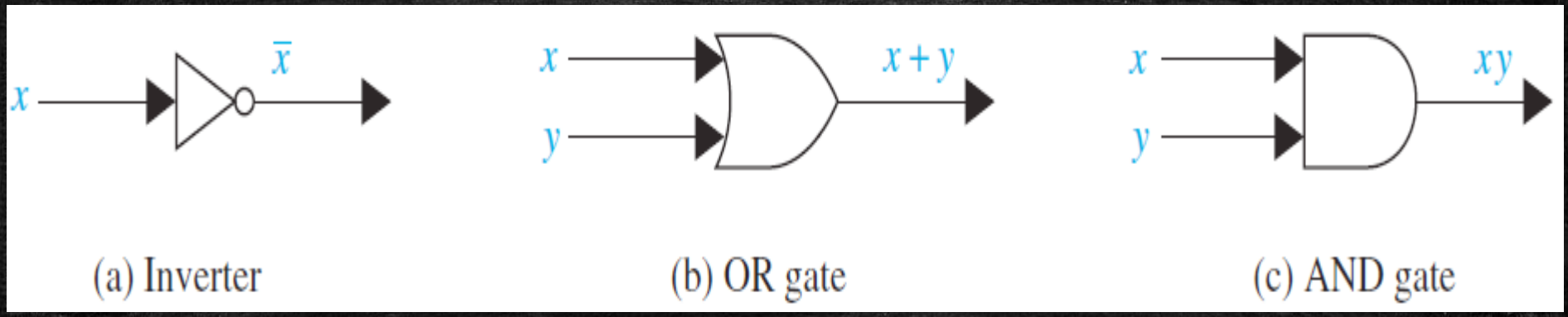
Basic Types of Gates



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

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 (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.

Basic Types of Gates



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

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 (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.

Basic Types of Gates

- 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.

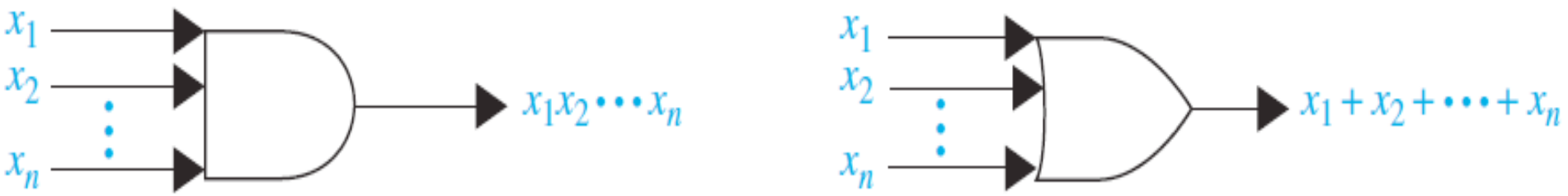


FIGURE 2 Gates with n Inputs.

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.

Combinations of Gates

- Note also that output from a gate may be used as input by one or more other elements, as shown in the next Figure 3. Both drawings in Figure 3 depict the circuit that produces the output $xy + \bar{x}y$.

Combinations of Gates

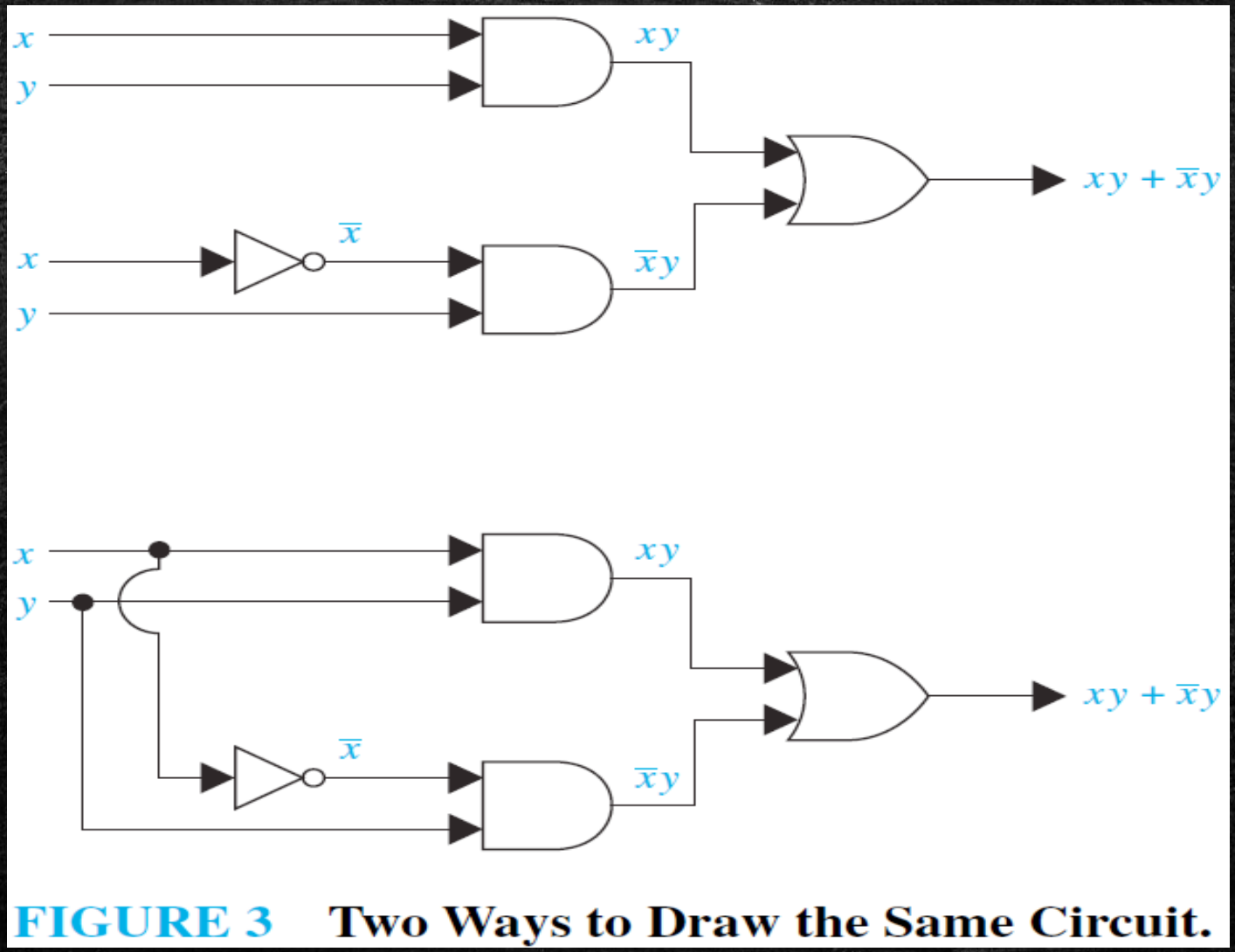
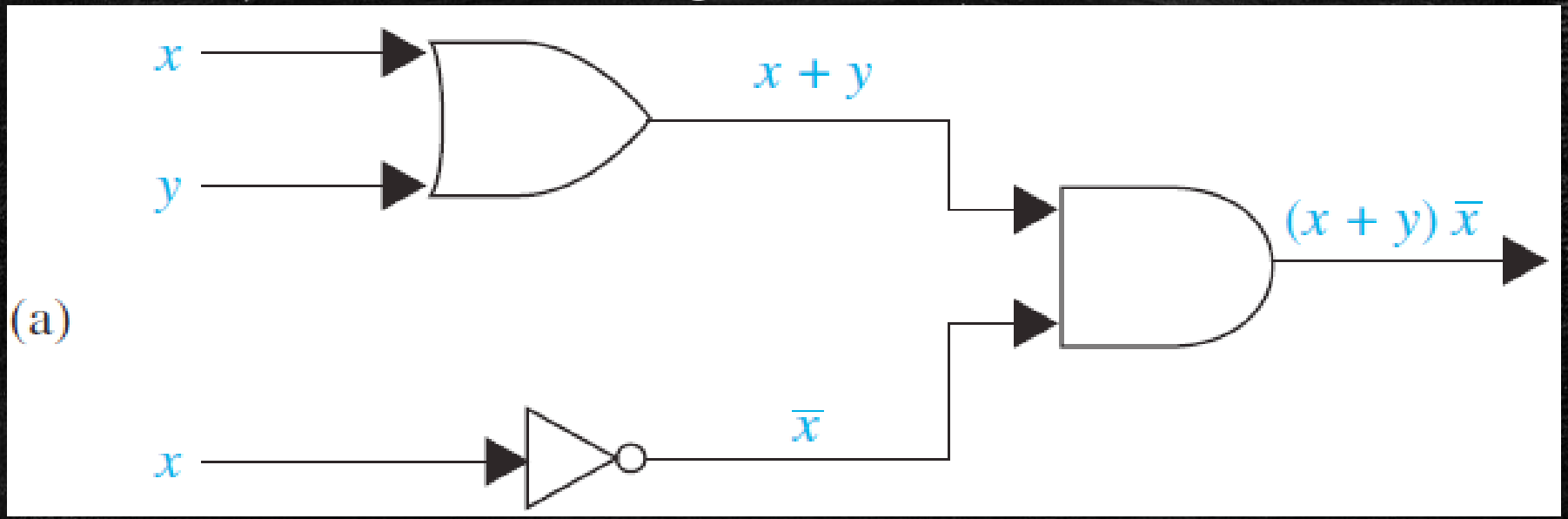


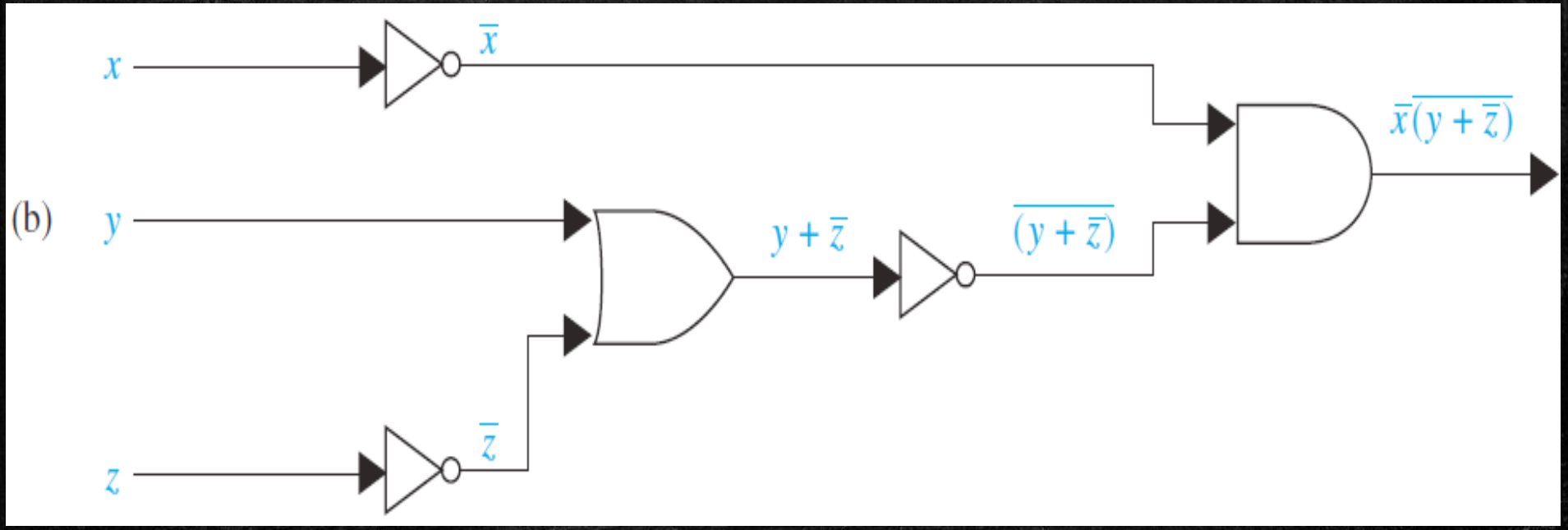
FIGURE 3 Two Ways to Draw the Same Circuit.

Combinations of Gates: Example

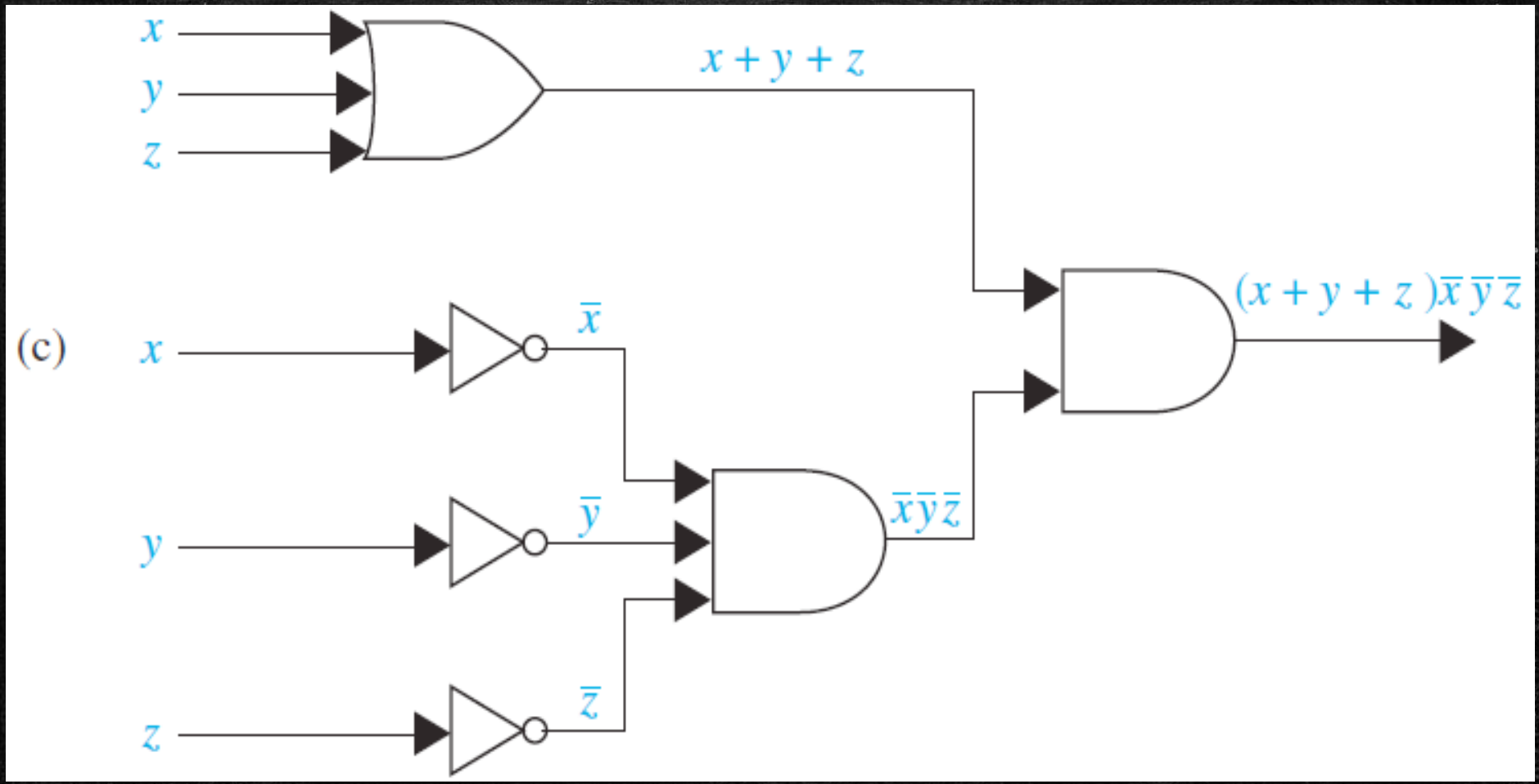
- Construct circuits that produce the following outputs: (a) $(x + y)\bar{x}$, (b) $\bar{x}\overline{(y + \bar{z})}$, and (c) $(x + y + z)(\bar{x}\bar{y}\bar{z})$.
- *Solution:* Circuits that produce these outputs are shown in the next Figures.



Combinations of Gates: Example



Combinations of Gates: Example



Another Examples of Circuits

- 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.

Another Examples of Circuits

- *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 $xy + xz + yz$. The circuit that implements this function is shown in the next Figure 5.

Another Examples of Circuits

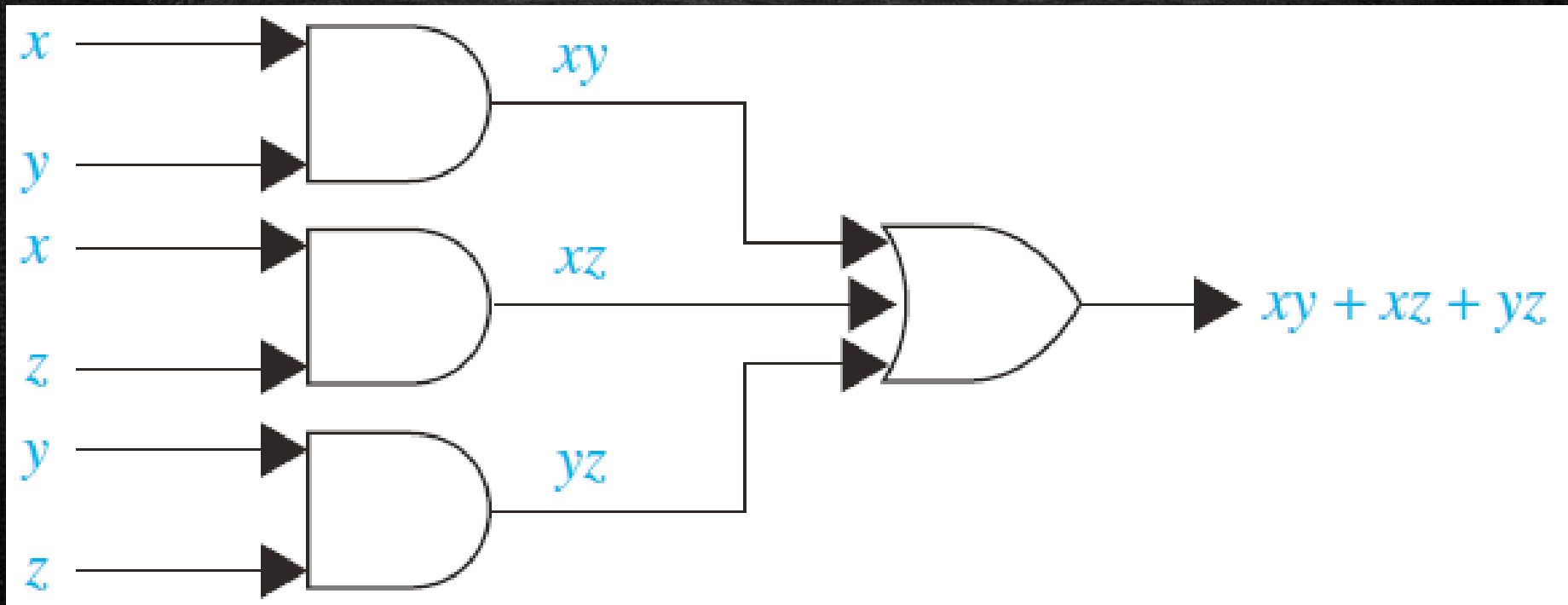


FIGURE 5 A Circuit for Majority Voting.

Another Examples of Circuits

- 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.

Another Examples of Circuits

- *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.

Another Examples of Circuits

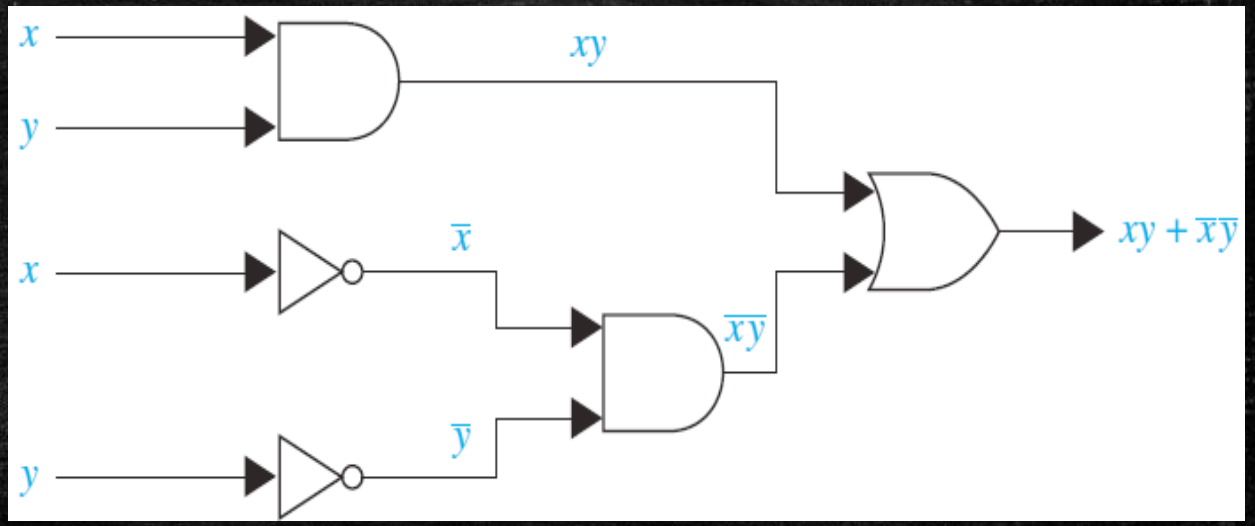
- *Solution (cont.):* We can arbitrarily decide that the light will be on when both switches are closed, so that $F(1,1) = 1$. This determines all the other values of F. When one of the two switches is opened, the light goes off, so $F(1,0) = F(0,1) = 0$. When the other switch is also opened, the light goes on, so $F(0,0) = 1$.

Another Examples of Circuits

- 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.

TABLE 1

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



Another Examples of Circuits

- *Solution (cont.):* 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 $F(1, 1, 1) = 1$.

Another Examples of Circuits

- *Solution (cont.):* 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.

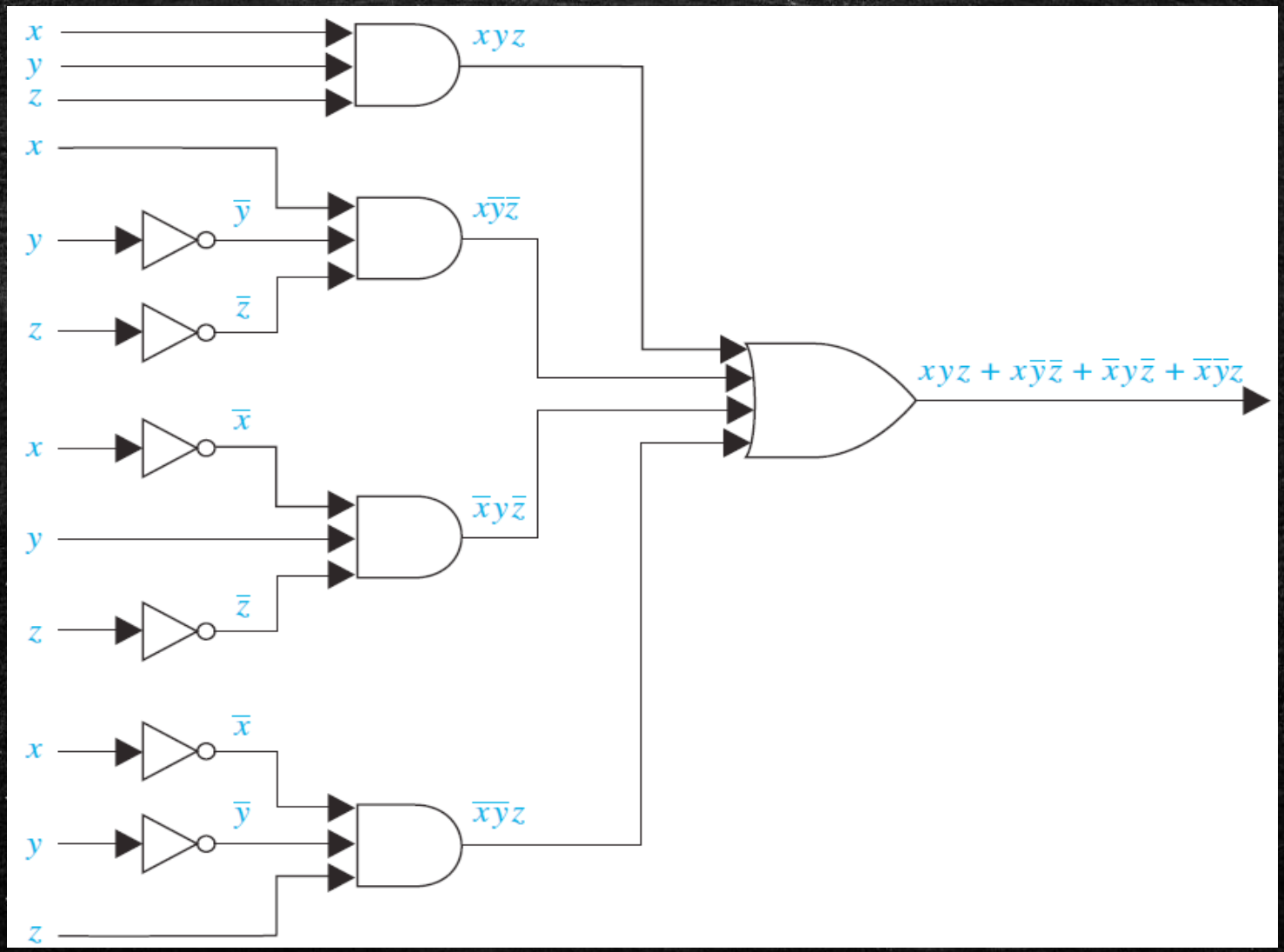
Another Examples of Circuits

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

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

Another Examples of Circuits



Adders

- 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.

Adders

- 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 the next Figure 8 computes the sum bit s and the carry bit c from the bits *x* and *y*.

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

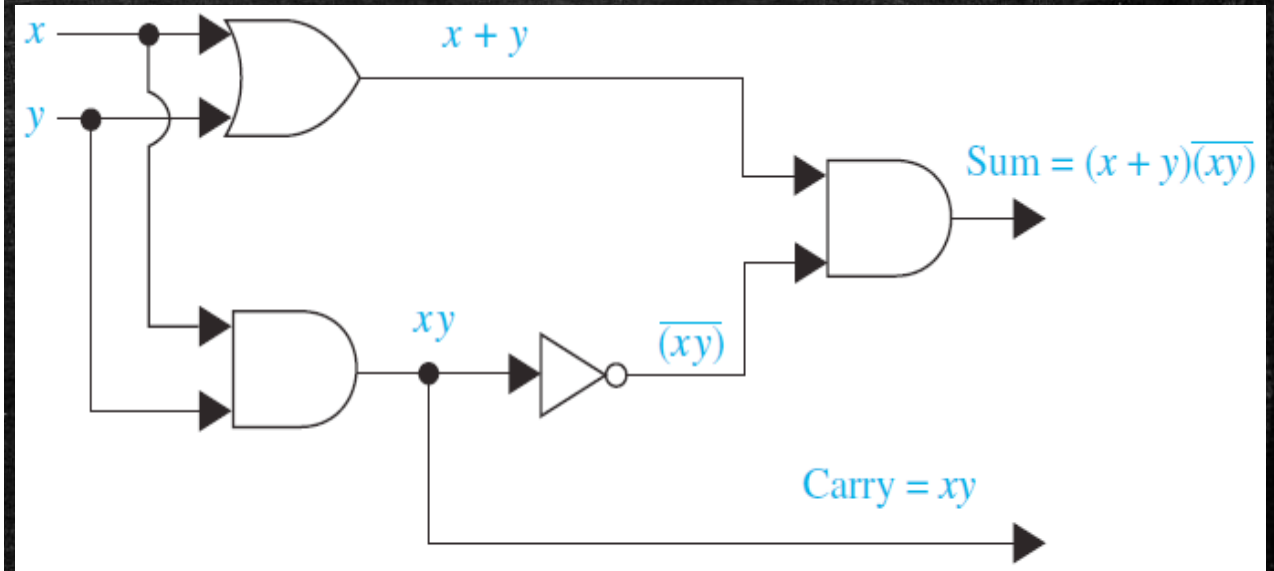


FIGURE 8 **The Half Adder.**

Adders

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

- 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.

Adders

- 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 the next Figure 9.

Adders

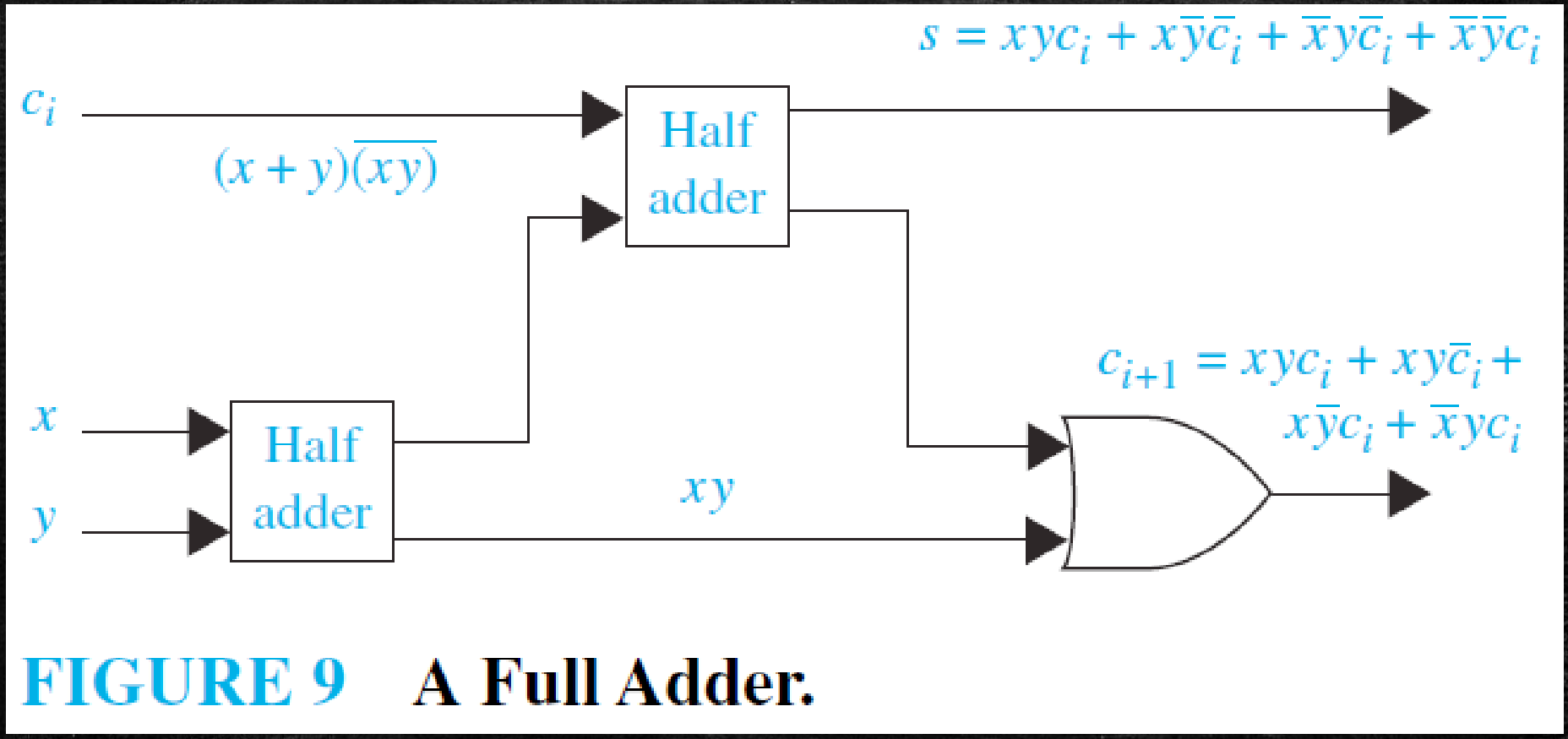


FIGURE 9 **A Full Adder.**

Adders

- 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.

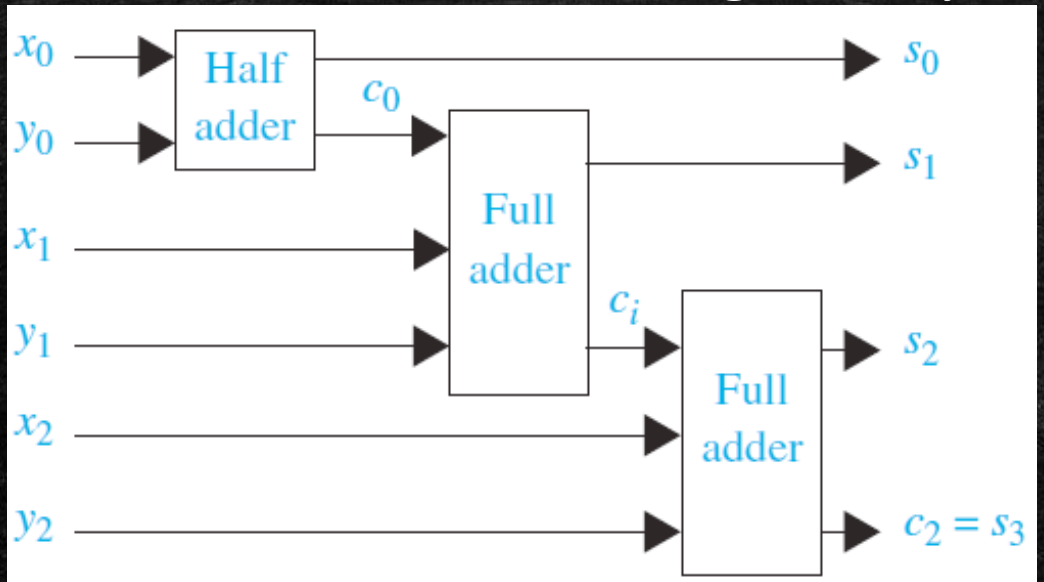
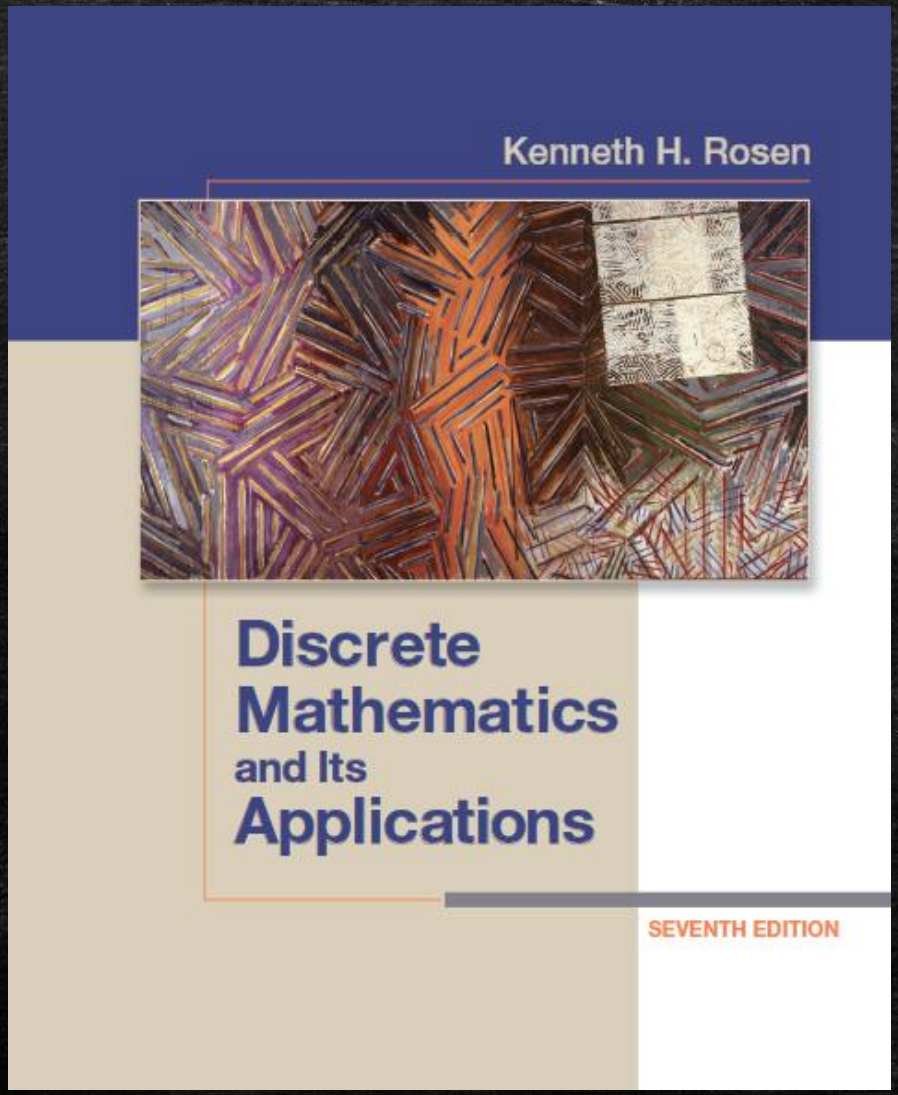


FIGURE 10 **Adding Two Three-Bit Integers with Full and Half Adders.**

HOMEWORK: Exercises 2, 4, 6, 16 on pp. 827-828



Minimization of Circuits

- 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.

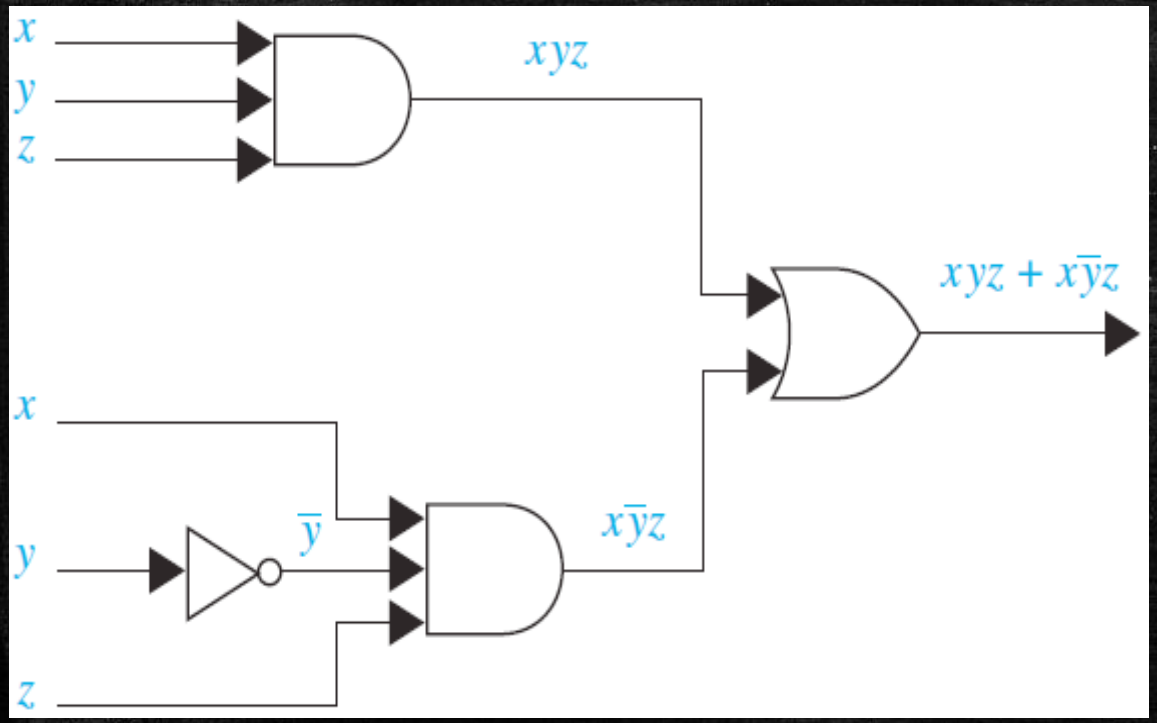
Minimization of Circuits

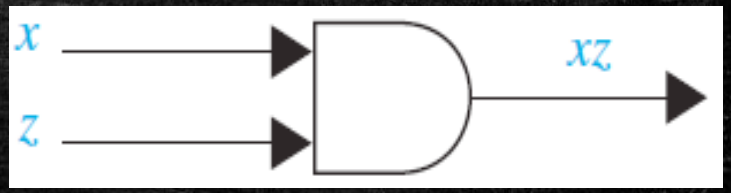
- Terms in a sum-of-products expansion that differ in just one variable, so that in one term this variable occurs and in the other term the complement of this variable occurs, can be combined. For instance, consider the circuit that has output 1 if and only if $x = y = z = 1$ or $x = z = 1$ and $y = 0$. The sum-of-products expansion of this circuit is $xyz + x\bar{y}z$. The two products in this expansion differ in exactly one variable, namely, y. They can be combined as

$$xyz + x\bar{y}z = (y + \bar{y})(xz) = 1 \cdot (xz) = xz.$$

Minimization of Circuits

- Hence, xz is a Boolean expression with fewer operators that represents the circuit. The second circuit uses only one gate, whereas the first circuit uses three gates and an inverter.





Minimization of Circuits

- This example shows that combining terms in the sum-of-products expansion of a circuit leads to a simpler expression for the circuit. We will describe two procedures that simplify sum-of-products expansions.
- The goal of both procedures is to produce Boolean sums of Boolean products that represent a Boolean function with the fewest products of literals such that these products contain the fewest literals possible among all sums of products that represent a Boolean function.

Minimization of Circuits

- Finding such a sum of products is called **minimization of the Boolean function**. Minimizing a Boolean function makes it possible to construct a circuit for this function that uses the fewest gates and fewest inputs to the AND gates and OR gates in the circuit, among all circuits for the Boolean expression we are minimizing.

Minimization of Circuits

- Until the early 1960s logic gates were individual components. To reduce costs it was important to use the fewest gates to produce a desired output. However, in the mid-1960s, integrated circuit technology was developed that made it possible to combine gates on a single chip. Even though it is now possible to build increasingly complex integrated circuits on chips at low cost, minimization of Boolean functions remains important.

Minimization of Circuits

- Reducing the number of gates on a chip can lead to a more reliable circuit and can reduce the cost to produce the chip. Also, minimization makes it possible to fit more circuits on the same chip. Furthermore, minimization reduces the number of inputs to gates in a circuit. This reduces the time used by a circuit to compute its output. Moreover, the number of inputs to a gate may be limited because of the particular technology used to build logic gates.

Minimization of Circuits

- The first procedure we will introduce, known as Karnaugh maps (or K-maps), was designed in the 1950s to help minimize circuits by hand. K-maps are useful in minimizing circuits with up to six variables, although they become rather complex even for five or six variables. The second procedure we will describe, the Quine-McCluskey method, was invented in the 1960s. It automates the process of minimizing combinatorial circuits and can be implemented as a computer program.

Karnaugh Maps

- To reduce the number of terms in a Boolean expression representing a circuit, it is necessary to find terms to combine. There is a graphical method, called a **Karnaugh map** or **K-map**, for finding terms to combine for Boolean functions involving a relatively small number of variables. The method we will describe was introduced by Maurice Karnaugh in 1953. His method is based on earlier work by E. W. Veitch. (This method is usually applied only when the function involves six or fewer variables.)

Karnaugh Maps

- K-maps give us a visual method for simplifying sum-of-products expansions; they are not suited for mechanizing this process. We will first illustrate how K-maps are used to simplify expansions of Boolean functions in two variables. We will continue by showing how K-maps can be used to minimize Boolean functions in three variables and then in four variables. Then we will describe the concepts that can be used to extend K-maps to minimize Boolean functions in more than four variables.

Karnaugh Maps

- There are four possible minterms in the sum-of-products expansion of a Boolean function in the two variables x and y. A K-map for a Boolean function in these two variables consists of four cells, where a 1 is placed in the cell representing a minterm if this minterm is present in the expansion. Cells are said to be adjacent if the minterms that they represent differ in exactly one literal. For instance, the cell representing $\bar{x}y$ is adjacent to the cells representing xy and $\bar{x}\bar{y}$. The four cells and the terms that they represent are shown in Figure 2.

Karnaugh Maps

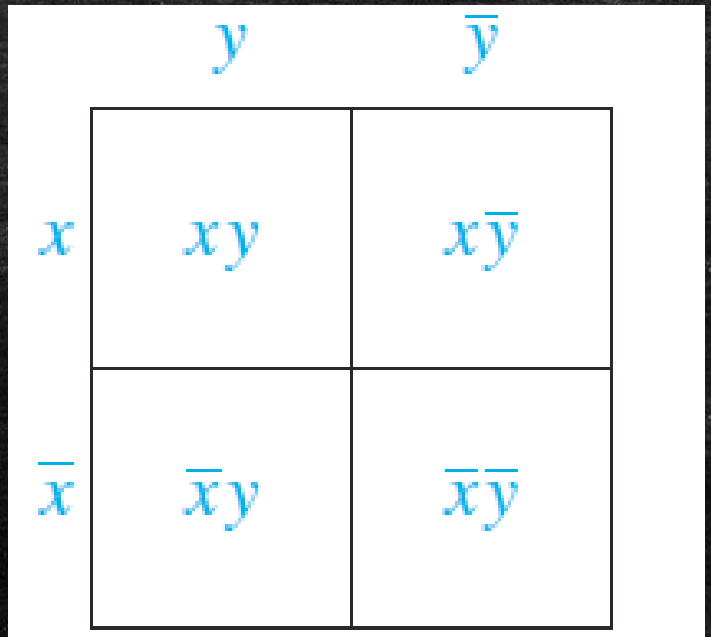


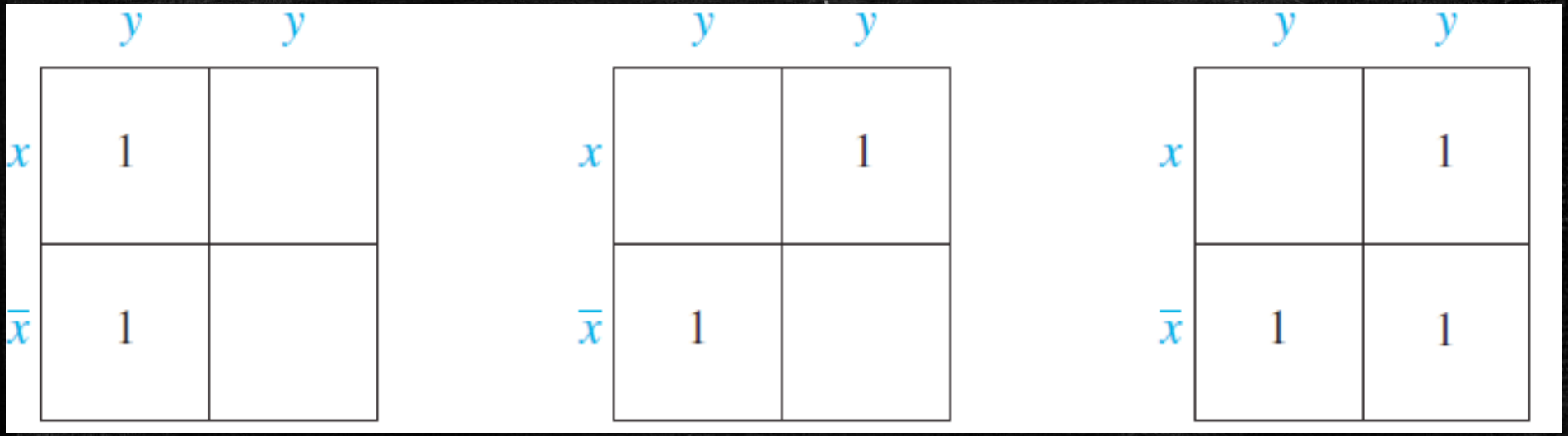
FIGURE 2
K-maps in Two Variables.

Karnaugh Maps: Example

- Find the K-maps for

  (a) $xy + \bar{x}y$, (b) $x\bar{y} + \bar{x}y$, and (c) $\bar{x}y + x\bar{y} + \bar{x}\bar{y}$.

- *Solution:* We include a 1 in a cell when the minterm represented by this cell is present in the sum-of-products expansion. The three K-maps are shown in Figure.



Karnaugh Maps

- We can identify minterms that can be combined from the K-map. Whenever there are 1s in two adjacent cells in the K-map, the minterms represented by these cells can be combined into a product involving just one of the variables. For instance, $x\bar{y}$ and $\bar{x}\bar{y}$ are represented by adjacent cells and can be combined into $\bar{y}$, because

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

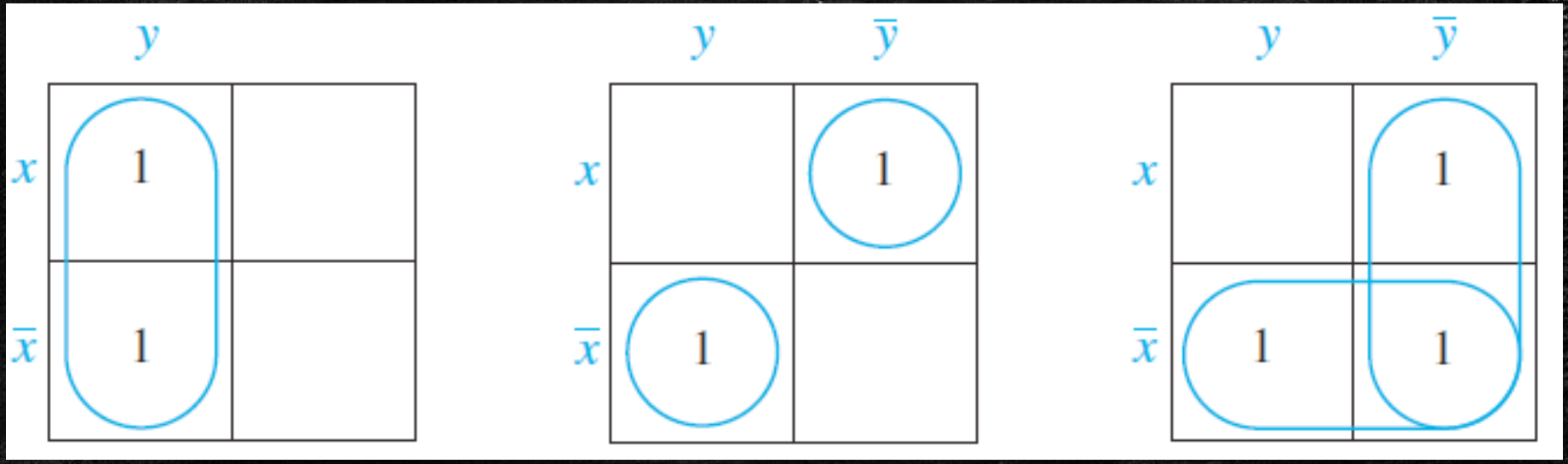
Karnaugh Maps

- Moreover, if 1s are in all four cells, the four minterms can be combined into one term, namely, the Boolean expression 1 that involves none of the variables. We circle blocks of cells in the K-map that represent minterms that can be combined and then find the corresponding sum of products. The goal is to identify the largest possible blocks, and to cover all the 1s with the fewest blocks using the largest blocks first and always using the largest possible blocks.

Karnaugh Maps

- Simplify the sum-of-products expansions given in previous Example.
- *Solution:* The grouping of minterms is shown in the Figure using the K-maps for these expansions. Minimal expansions for these sums-of-products are

(a) y, (b) $\bar{x}y + x\bar{y}$, and (c) $\bar{x} + \bar{y}$.

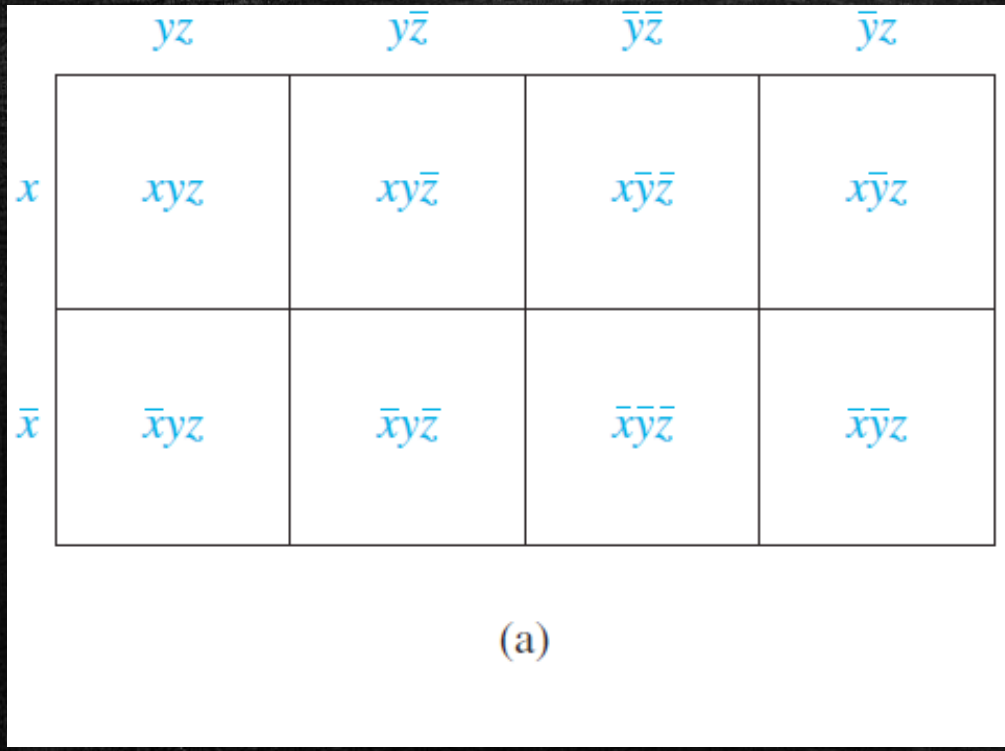


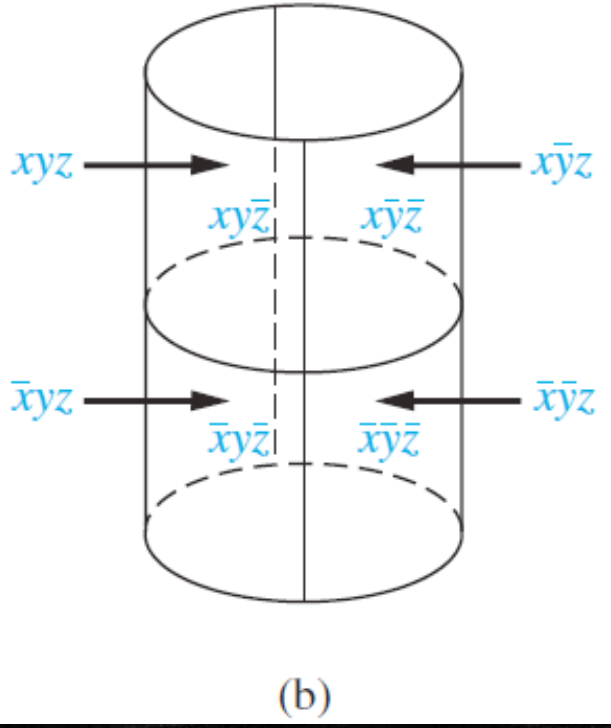
Karnaugh Maps

- A K-map in three variables is a rectangle divided into eight cells. The cells represent the eight possible minterms in three variables. Two cells are said to be adjacent if the minterms that they represent differ in exactly one literal. One of the ways to form a K-map in three variables is shown in the next Figure (a). This K-map can be thought of as lying on a cylinder, as shown in the next Figure (b).

Karnaugh Maps

On the cylinder, two cells have a common border if and only if they are adjacent.

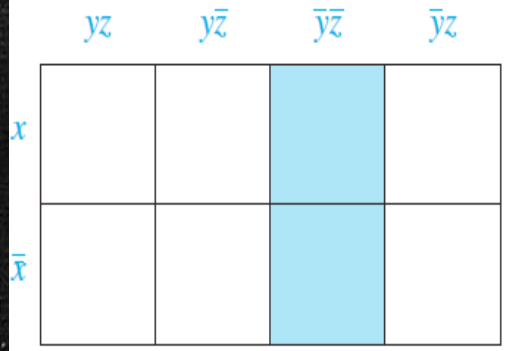




Karnaugh Maps

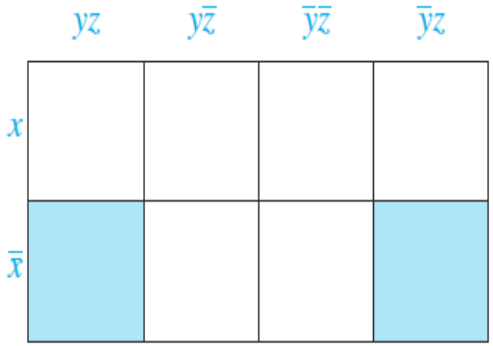
- To simplify a sum-of-products expansion in three variables, we use the K-map to identify blocks of minterms that can be combined. Blocks of two adjacent cells represent pairs of minterms that can be combined into a product of two literals; 2×2 and 4×1 blocks of cells represent minterms that can be combined into a single literal; and the block of all eight cells represents a product of no literals, namely, the function 1. In the next Figure, 1×2, 2×1, 2×2, 4×1, and 4×2 blocks and the products they represent are shown.

Karnaugh Maps



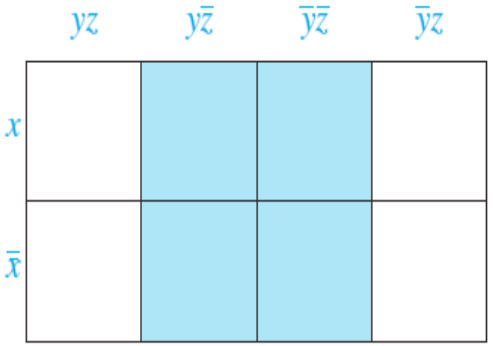
$\bar{y}\bar{z} = x\bar{y}\bar{z} + \bar{x}\bar{y}\bar{z}$

(a)



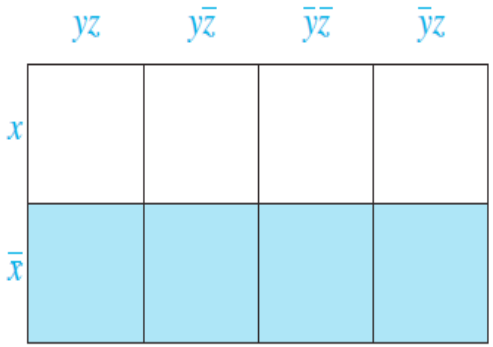
$\bar{x}z = \bar{x}yz + \bar{x}\bar{y}z$

(b)



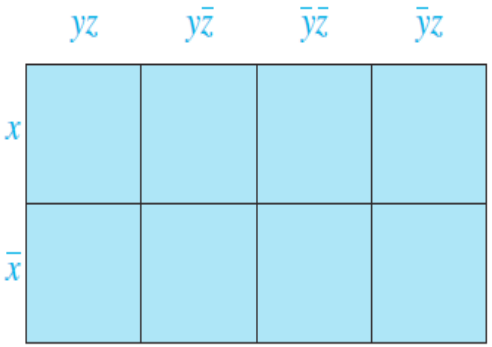
$\bar{z} = xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z}$

(c)



$\bar{x} = \bar{x}yz + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z$

(d)



$1 = xyz + xy\bar{z} + x\bar{y}\bar{z} + x\bar{y}z +$
$\bar{x}yz + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z$

(e)

Karnaugh Maps

- The product of literals corresponding to a block of all 1s in the K-map is called an **implicant** of the function being minimized. It is called a **prime implicant** if this block of 1s is not contained in a larger block of 1s representing the product of fewer literals than in this product.
- The goal is to identify the largest possible blocks in the map and cover all the 1s in the map with the least number of blocks, using the largest blocks first.

Karnaugh Maps

- The largest possible blocks are always chosen, but we must always choose a block if it is the only block of 1s covering a 1 in the K-map. Such a block represents an **essential prime implicant**. By covering all the 1s in the map with blocks corresponding to prime implicants we can express the sum of products as a sum of prime implicants. Note that there may be more than one way to cover all the 1s using the least number of blocks.

Karnaugh Maps: Example

- Use K-maps to minimize these sum-of-products expansions.

(a) $xy\bar{z} + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}\bar{y}\bar{z}$

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

(c) $xyz + xy\bar{z} + x\bar{y}z + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$

(d) $xy\bar{z} + x\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$

Karnaugh Maps: Example

- *Solution:* The K-maps for these sum-of-products expansions are shown in Figure 7. The grouping of blocks shows that minimal expansions into Boolean sums of Boolean products are (a) $x\bar{z} + \bar{y}\bar{z} + \bar{x}yz$, (b) $\bar{y} + \bar{x}z$, (c) $x + \bar{y} + z$, and (d) $x\bar{z} + \bar{x}\bar{y}$. In part (d) note that the prime implicants $x\bar{z}$ and $\bar{x}\bar{y}$ are essential prime implicants, but the prime implicant $\bar{y}\bar{z}$ is a prime implicant that is not essential, because the cells it covers are covered by the other two prime implicants.

Karnaugh Maps: Example

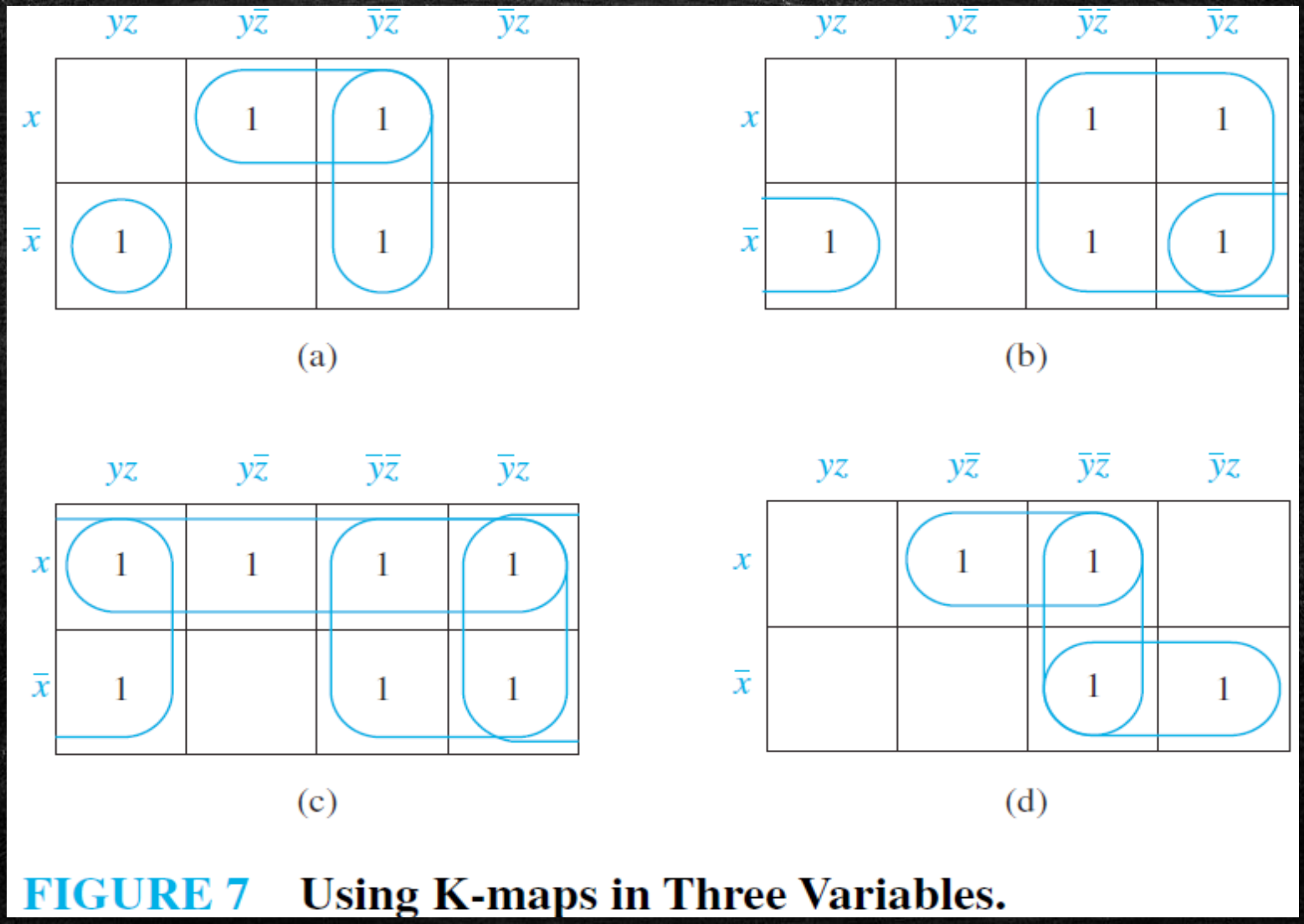
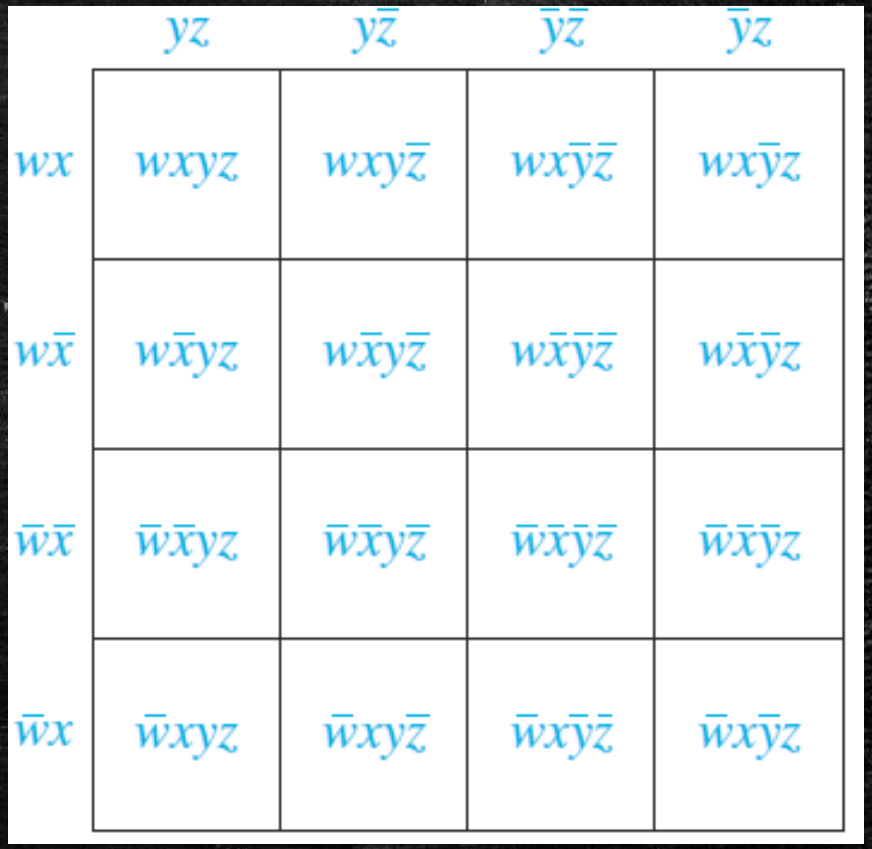


FIGURE 7 **Using K-maps in Three Variables.**

Karnaugh Maps

- A K-map in four variables is a square that is divided into 16 cells. The cells represent the 16 possible minterms in four variables. One of the ways to form a K-map in four variables is shown in Figure.



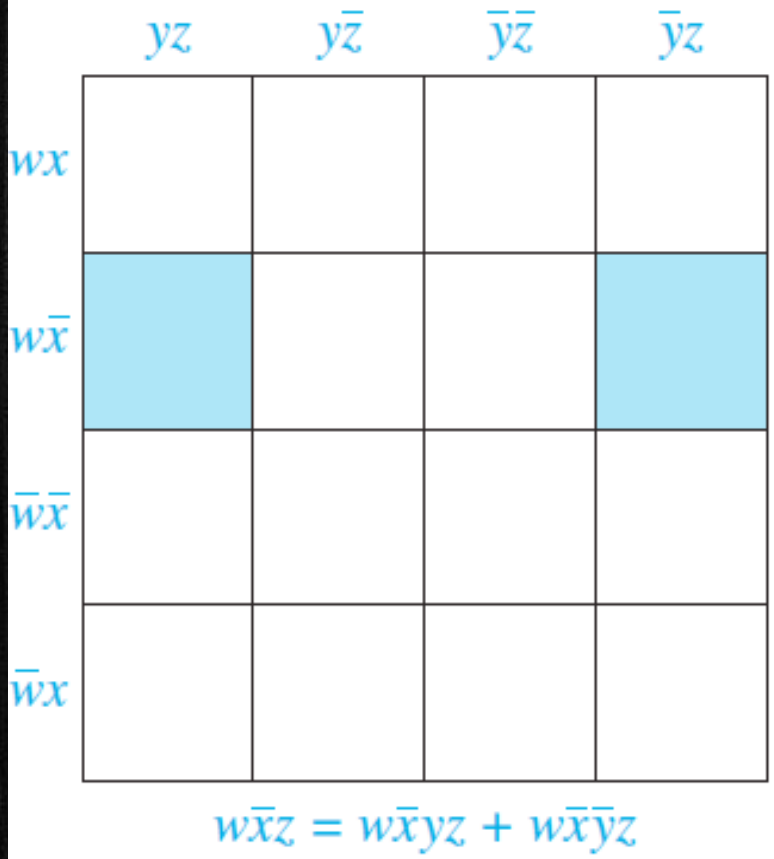
Karnaugh Maps

- Two cells are adjacent if and only if the minterms they represent differ in one literal. Consequently, each cell is adjacent to four other cells. The K-map of a sum-of-products expansion in four variables can be thought of as lying on a torus, so that adjacent cells have a common boundary.

Karnaugh Maps

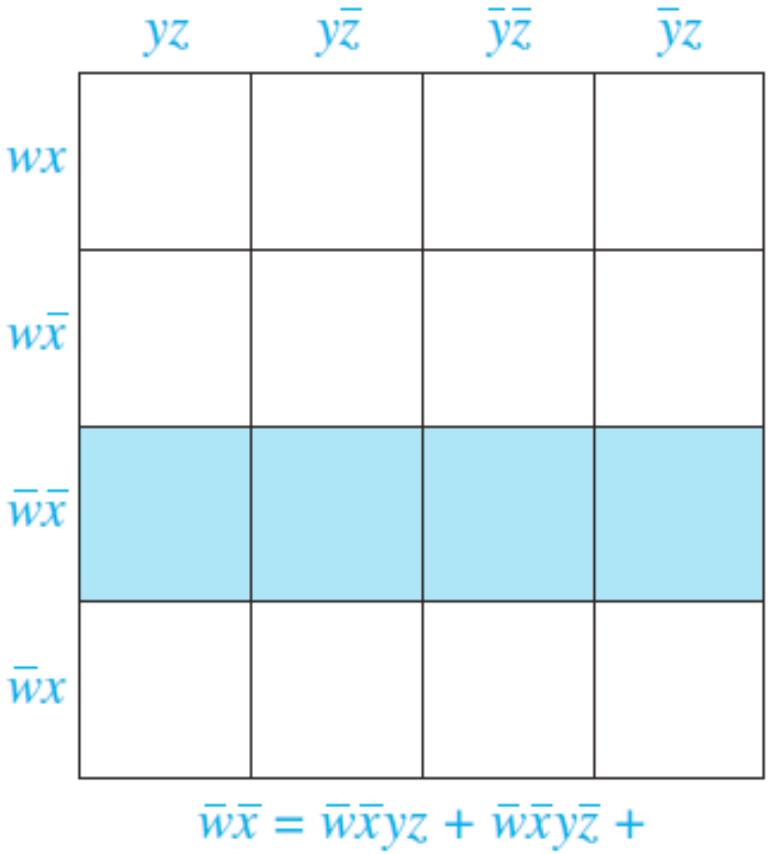
- The simplification of a sum-of-products expansion in four variables is carried out by identifying those blocks of 2, 4, 8, or 16 cells that represent minterms that can be combined. Each cell representing a minterm must either be used to form a product using fewer literals, or be included in the expansion. In the next Figure some examples of blocks that represent products of three literals, products of two literals, and a single literal are illustrated.

Karnaugh Maps



$w\bar{x}z = w\bar{x}yz + w\bar{x}\bar{y}z$

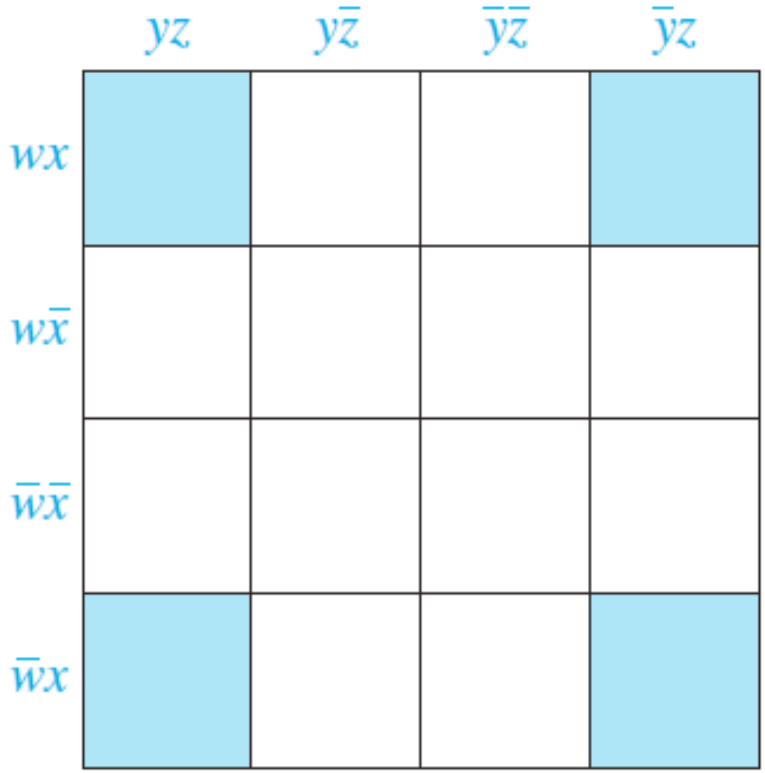
(a)



$\bar{w}\bar{x} = \bar{w}\bar{x}yz + \bar{w}\bar{x}y\bar{z} + \bar{w}\bar{x}\bar{y}\bar{z} + \bar{w}\bar{x}\bar{y}z$

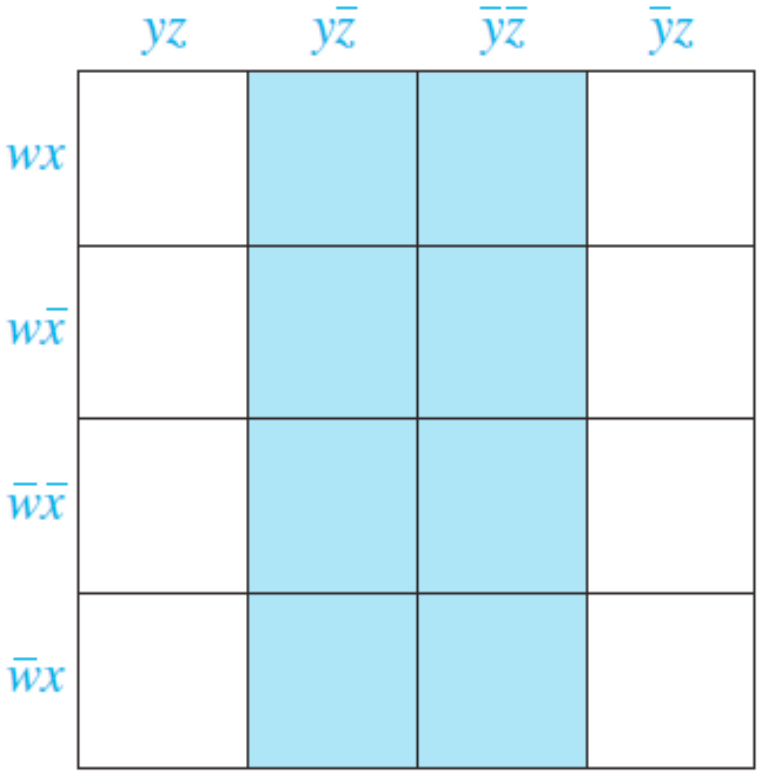
(b)

Karnaugh Maps



$x\bar{z} = wxyz + wx\bar{y}z + \bar{w}xyz + \bar{w}x\bar{y}z$

(c)



$\bar{z} = wxy\bar{z} + wx\bar{y}\bar{z} + w\bar{x}y\bar{z} + w\bar{x}\bar{y}\bar{z} + \bar{w}\bar{x}y\bar{z} + \bar{w}\bar{x}\bar{y}\bar{z} + \bar{w}xy\bar{z} + \bar{w}x\bar{y}\bar{z}$

(d)

Karnaugh Maps

- As is the case in K-maps in two and three variables, the goal is to identify the largest blocks of 1s in the map that correspond to the prime implicants and to cover all the 1s using the fewest blocks needed, using the largest blocks first. The largest possible blocks are always used.

Karnaugh Maps: Example

- Use K-maps to simplify these sum-of-products expansions.

(a) $wxyz + wxyz\bar{} + wxy\bar{}z\bar{} + wx\bar{}yz + wx\bar{}y\bar{}z + w\bar{}xy\bar{}z\bar{} + \bar{w}xy\bar{}z + \bar{w}x\bar{}yz + \bar{w}x\bar{}yz\bar{}$

(b) $wxy\bar{}z\bar{} + wx\bar{}yz + wx\bar{}yz\bar{} + wx\bar{}y\bar{}z\bar{} + \bar{w}xy\bar{}z\bar{} + \bar{w}x\bar{}yz\bar{} + \bar{w}x\bar{}y\bar{}z\bar{}$

(c) $wxyz\bar{} + wxy\bar{}z\bar{} + wx\bar{}yz + wx\bar{}yz\bar{} + wx\bar{}y\bar{}z\bar{} + \bar{w}xyz + \bar{w}xyz\bar{} + \bar{w}xy\bar{}z\bar{} + \bar{w}xy\bar{}z + \bar{w}x\bar{}yz\bar{} + \bar{w}x\bar{}y\bar{}z\bar{}$

Karnaugh Maps: Example

- *Solution:* The K-maps for these expansions are shown in the next Figures. Using the blocks shown leads to the sum of products

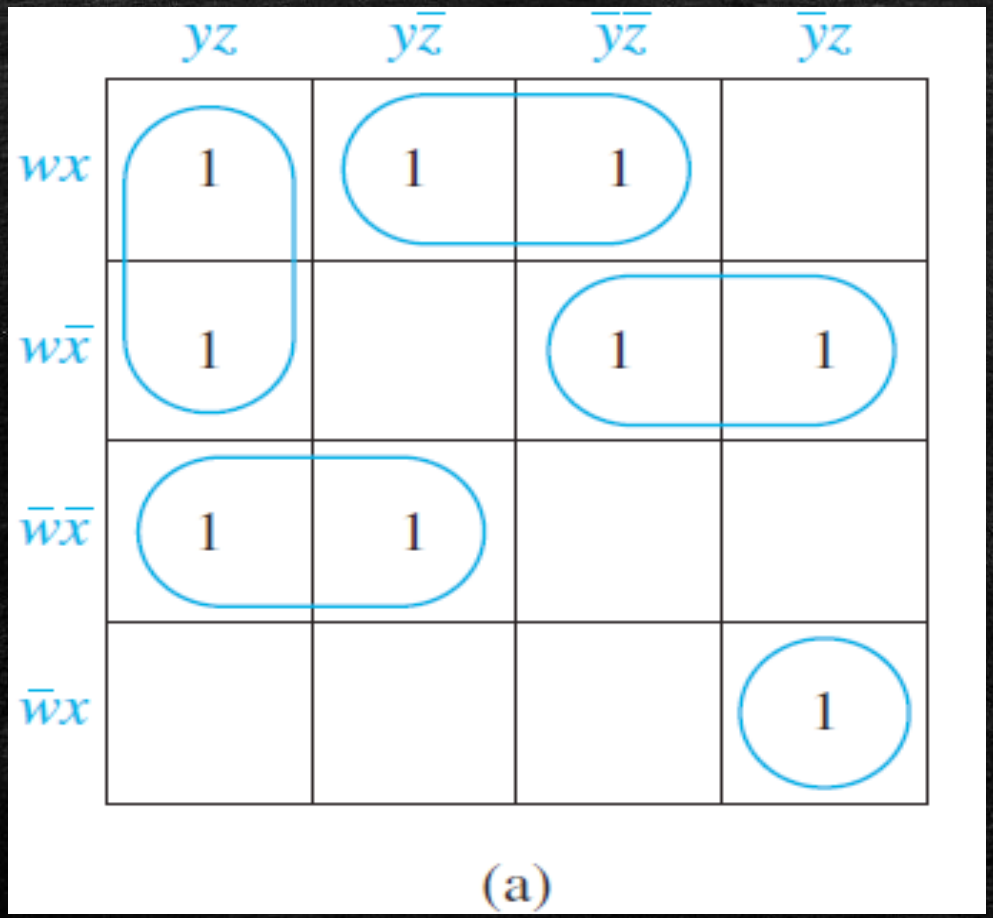
(a) $wyz + wx\bar{z} + w\bar{x}\bar{y} + \bar{w}\bar{x}y + \bar{w}x\bar{y}z,$

(b) $\bar{y}\bar{z} + w\bar{x}y + \bar{x}\bar{z},$

(c) $\bar{z} + \bar{w}x + w\bar{x}y.$

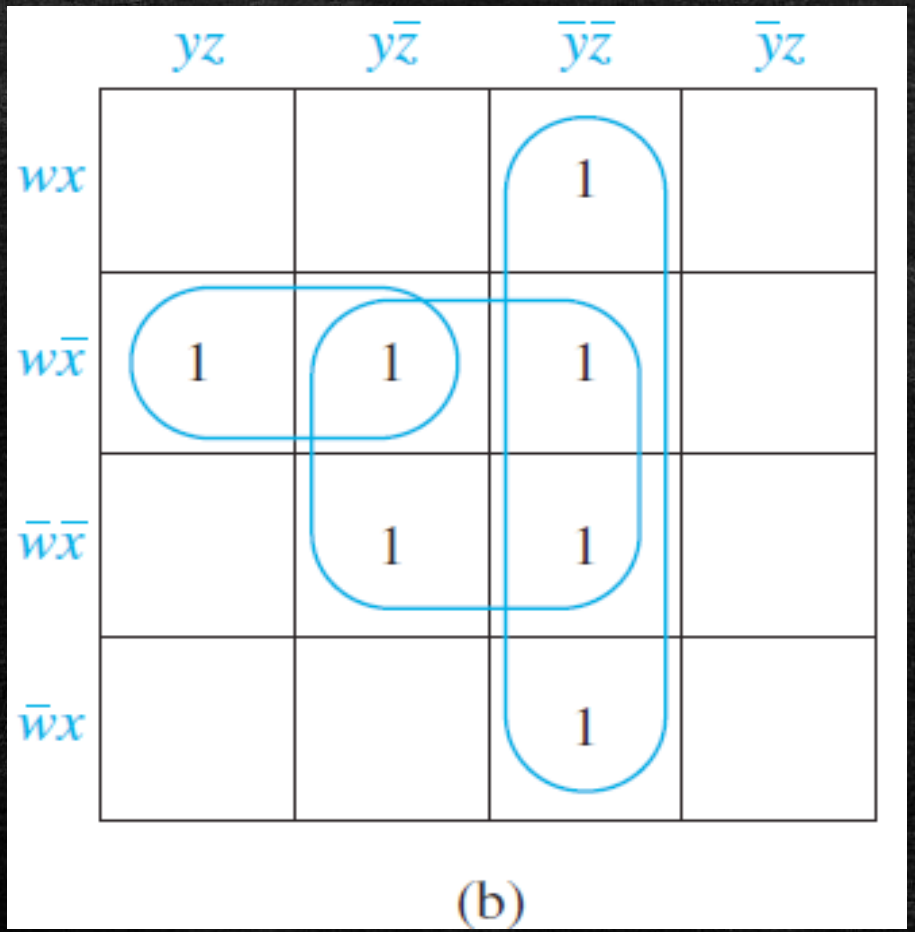
Karnaugh Maps: Example

(a) $wxyz + wxy\bar{z} + wx\bar{y}\bar{z} + w\bar{x}yz + w\bar{x}\bar{y}z + w\bar{x}\bar{y}\bar{z} + \bar{w}x\bar{y}z + \bar{w}\bar{x}yz + \bar{w}\bar{x}y\bar{z} = wyz + wx\bar{z} + w\bar{x}\bar{y} + \bar{w}\bar{x}y + \bar{w}x\bar{y}z.$



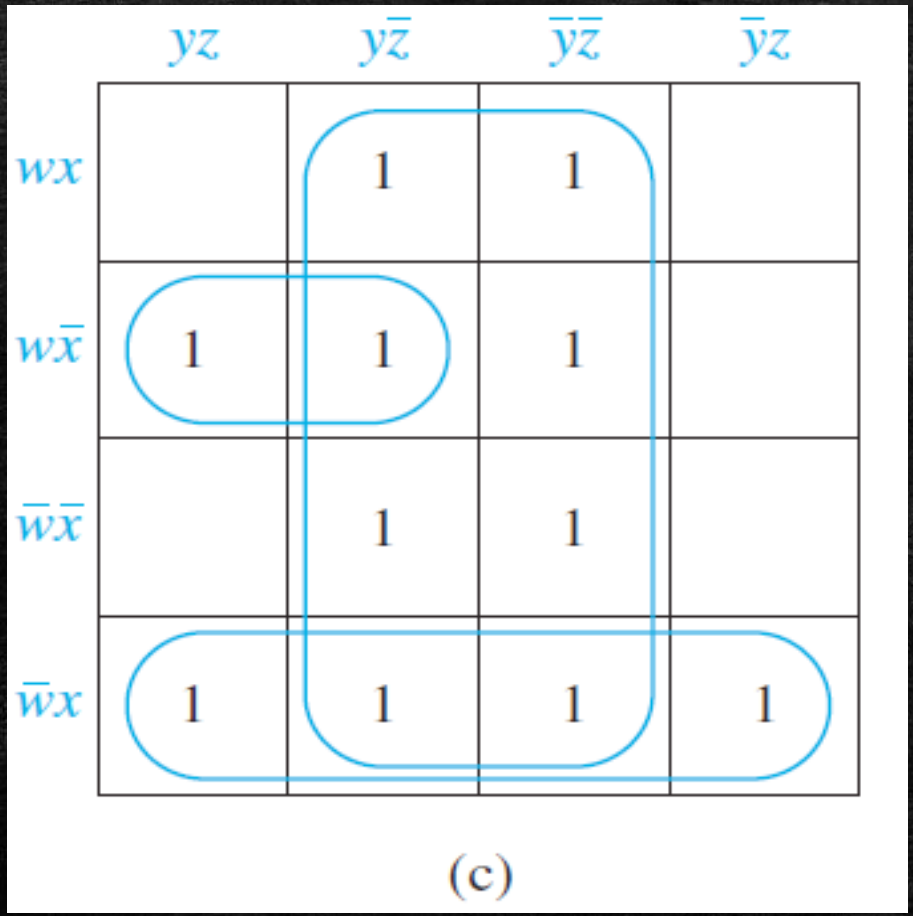
Karnaugh Maps: Example

(b) $wx\bar{y}\bar{z} + w\bar{x}yz + w\bar{x}y\bar{z} + w\bar{x}\bar{y}\bar{z} + \bar{w}x\bar{y}\bar{z} + \bar{w}\bar{x}yz + \bar{w}x\bar{x}\bar{y}\bar{z} = \bar{y}\bar{z} + w\bar{x}y + \bar{x}\bar{z}$



Karnaugh Maps: Example

(c) $wxyz' + wxy'z' + wx'yz + wx'yz' + wx'y'z' + w'xyz + w'xyz' + w'xy'z' + w'xy'z + w'x'yz' + w'x'y'z' = z' + w'x + wx'y.$



Karnaugh Maps

- K-maps can realistically be used to minimize Boolean functions with five or six variables, but beyond that, they are rarely used because they become extremely complicated. However, the concepts used in K-maps play an important role in newer algorithms. Furthermore, mastering these concepts helps you understand these newer algorithms and the computer-aided design (CAD) programs that implement them.

Karnaugh Maps

- As we develop these concepts, we will be able to illustrate them by referring back to our discussion of minimization of Boolean functions in three and in four variables.
- The K-maps we used to minimize Boolean func-s in two, three, and four variables are built using 2×2, 2×4, and 4×4 rectangles, respectively. Furthermore, corresponding cells in the top row and bottom row and in the leftmost column and rightmost column in each of thesecases are considered adjacent because they represent minterms differing in only one literal.

Karnaugh Maps

- We can build K-maps for minimizing Boolean functions in more than four variables in a similar way. We use a rectangle containing $2^{\lfloor n/2 \rfloor}$ rows and $2^{\lceil n/2 \rceil}$ columns. (These K-maps contain 2^n cells because $\lfloor n/2 \rfloor + \lceil n/2 \rceil = n$.) The rows and columns need to be positioned so that the cells representing minterms differing in just one literal are adjacent or are considered adjacent by specifying additional adjacencies of rows and columns.

Karnaugh Maps

- To help (but not entirely) achieve this, the rows and columns of a K-map are arranged using Gray codes, where we associate bit strings and products by specifying that a 1 corresponds to the appearance of a variable and a 0 with the appearance of its complement. For example, in a 10-dimensional K-map, the Gray code 01110 used to label a row corresponds to the product $\overline{x_1}x_2x_3x_4\overline{x_5}$.

The Quine–McCluskey Method

- We have seen that K-maps can be used to produce minimal expansions of Boolean functions as Boolean sums of Boolean products. However, K-maps are awkward to use when there are more than four variables. Furthermore, the use of K-maps relies on visual inspection to identify terms to group. For these reasons there is a need for a procedure for simplifying sum-of-products expansions that can be mechanized. The Quine-McCluskey method is such a procedure.

The Quine–McCluskey Method

- It can be used for Boolean functions in any number of variables. It was developed in the 1950s by W. V. Quine and E. J. McCluskey, Jr. Basically, the Quine-McCluskey method consists of two parts. The first part finds those terms that are candidates for inclusion in a minimal expansion as a Boolean sum of Boolean products. The second part determines which of these terms to actually use. We will use next Example to illustrate how, by successively combining implicants into implicants with one fewer literal, this procedure works.

The Quine-McCluskey Method: Example

- We will show how the Quine-McCluskey method can be used to find a minimal expansion equivalent to $xyz + x\bar{y}z + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$.
- We will represent the minterms in this expansion by bit strings. The first bit will be 1 if x occurs and 0 if $\bar{x}$ occurs. The second bit will be 1 if y occurs and 0 if $\bar{y}$ occurs. The third bit will be 1 if z occurs and 0 if $\bar{z}$ occurs. We then group these terms according to the number of 1s in the corresponding bit strings.

The Quine-McCluskey Method: Example

- This information is shown in Table 2.

TABLE 2

Minterm	*Bit String*	*Number of 1s*
xyz	111	3
$x\bar{y}z$	101	2
$\bar{x}yz$	011	2
$\bar{x}\,\bar{y}z$	001	1
$\bar{x}\,\bar{y}\,\bar{z}$	000	0

The Quine-McCluskey Method: Example

- Minterms that can be combined are those that differ in exactly one literal. Hence, two terms that can be combined differ by exactly one in the number of 1s in the bit strings that represent them. When two minterms are combined into a product, this product contains two literals. A product in two literals is represented using a dash to denote the variable that does not occur. For instance, the minterms $x\bar{y}z$ and $\bar{x}\bar{y}z$, represented by bit strings 101 and 001, can be combined into $\bar{y}z$, represented by the string – 01.

The Quine-McCluskey Method: Example

- All pairs of minterms that can be combined and the product formed from these combinations are shown in Table 3.

TABLE 3

			Step 1			Step 2		
	Term	Bit String		Term	String		Term	String
1	xyz	111	(1,2)	xz	1–1	(1,2,3,4)	z	– –1
2	$x\bar{y}z$	101	(1,3)	yz	–11			
3	$\bar{x}yz$	011	(2,4)	$\bar{y}z$	–01			
4	$\bar{x}\,\bar{y}z$	001	(3,4)	$\bar{x}z$	0–1			
5	$\bar{x}\,\bar{y}\,\bar{z}$	000	(4,5)	$\bar{x}\,\bar{y}$	00–			

The Quine-McCluskey Method: Example

- Next, all pairs of products of two literals that can be combined are combined into one literal. Two such products can be combined if they contain literals for the same two variables, and literals for only one of the two variables differ. In terms of the strings representing the products, these strings must have a dash in the same position and must differ in exactly one of the other two slots.

The Quine-McCluskey Method: Example

- We can combine the products yz and $\bar{y}z$, represented by the strings –11 and –01, into z, represented by the string – –1.We show all the combinations of terms that can be formed in this way in previous Table 3. In Table 3 we also indicate which terms have been used to form products with fewer literals; these terms will not be needed in a minimal expansion. The next step is to identify a minimal set of products needed to represent the Boolean function.

The Quine-McCluskey Method: Example

- We begin with all those products that were not used to construct products with fewer literals. Next, we form Table 4, which has a row for each candidate product formed by combining original terms, and a column for each original term; and we put an X in a position if the original term in the sum-of-products expansion was used to form this candidate product. In this case, we say that the candidate product **covers** the original minterm. We need to include at least one product that covers each of the original minterms.

The Quine-McCluskey Method: Example

- Consequently, whenever there is only one X in a column in the table, the product corresponding to the row this X is in must be used. From Table 4 we see that both z and $\bar{x}\bar{y}$ are needed. Hence, the final answer is $z + \bar{x}\bar{y}$.

TABLE 4

	xyz	$x\bar{y}z$	$\bar{x}yz$	$\bar{x}\,\bar{y}z$	$\bar{x}\,\bar{y}\,\bar{z}$
z	X	X	X	X	
$\bar{x}\,\bar{y}$				X	X

The Quine-McCluskey Method

- As was illustrated in previous Example, the Quine-McCluskey method uses this sequence of steps to simplify a sum-of-products expression:

1. Express each minterm in n variables by a bit string of length n with a 1 in the ith position if x_i occurs and a 0 in this position if $\overline{x_i}$ occurs.

2. Group the bit strings according to the number of 1s in them.

The Quine-McCluskey Method

3. Determine all products in $n-1$ variables that can be formed by taking the Boolean sum of minterms in the expansion. Minterms that can be combined are represented by bit strings that differ in exactly one position. Represent these products in $n-1$ variables with strings that have a 1 in the ith position if x_i occurs in the product, a 0 in this position if $\overline{x_i}$ occurs, and a dash in this position if there is no literal involving x_i in the product.

The Quine-McCluskey Method

4. Determine all products in $n - 2$ variables that can be formed by taking the Boolean sum of the products in $n - 1$ variables found in the previous step. Products in $n - 1$ variables that can be combined are represented by bit strings that have a dash in the same position and differ in exactly one position.

5. Continue combining Boolean products into products in fewer variables as long as possible.

6. Find all the Boolean products that arose that were not used to form a Boolean product in one fewer literal.

The Quine-McCluskey Method

7. Find the smallest set of these Boolean products such that the sum of these products represents the Boolean function. This is done by forming a table showing which minterms are covered by which products. Every minterm must be covered by at least one product. The first step in using this table is to find all essential prime implicants. Each essential prime implicant must be included because it is the only prime implicant that covers one of the minterms.

The Quine-McCluskey Method: Example in four variables

- Use the Quine–McCluskey method to simplify the sum-of-products expansion $wxyz + w\bar{x}yz + w\bar{x}y\bar{z} + \bar{w}xyz + \bar{w}xy\bar{z} + \bar{w}\bar{x}yz + \bar{w}\bar{x}\bar{y}z$.
- *Solution:* We first represent the minterms by bit strings and then group these terms together according to the number of 1s in the bit strings. This is shown in Table 5. All the Boolean products that can be formed by taking Boolean sums of these products are shown in Table 6.

The Quine-McCluskey Method: Example in four variables

TABLE 5

Term	*Bit String*	*Number of 1s*
$wxy\overline{z}$	1110	3
$w\overline{x}yz$	1011	3
$\overline{w}xyz$	0111	3
$w\overline{x}y\overline{z}$	1010	2
$\overline{w}x\overline{y}z$	0101	2
$\overline{w}\,\overline{x}yz$	0011	2
$\overline{w}\,\overline{x}\,\overline{y}z$	0001	1

The Quine-McCluskey Method: Example in four variables

TABLE 6

			Step 1			Step 2		
	Term	Bit String		Term	String		Term	String
1	$wxy\overline{z}$	1110	(1,4)	$wy\overline{z}$	1–10	(3,5,6,7)	$\overline{w}z$	0 – –1
2	$w\overline{x}yz$	1011	(2,4)	$w\overline{x}y$	101–			
3	$\overline{w}xyz$	0111	(2,6)	$\overline{x}yz$	–011			
4	$w\overline{x}y\overline{z}$	1010	(3,5)	$\overline{w}xz$	01–1			
5	$\overline{w}x\overline{y}z$	0101	(3,6)	$\overline{w}yz$	0–11			
6	$\overline{w}\,\overline{x}yz$	0011	(5,7)	$\overline{w}\,\overline{y}z$	0–01			
7	$\overline{w}\,\overline{x}\,\overline{y}z$	0001	(6,7)	$\overline{w}\,\overline{x}z$	00–1			

The Quine-McCluskey Method: Example in four variables

- The only products that were not used to form products in fewer variables are $\bar{w}xz$, $wy\bar{z}$, $w\bar{x}y$, and $\bar{x}yz$. In Table 7 we show the minterms covered by each of these products.

TABLE 7

	$wxy\bar{z}$	$w\bar{x}yz$	$\bar{w}xyz$	$w\bar{x}y\bar{z}$	$\bar{w}x\bar{y}z$	$\bar{w}\,\bar{x}yz$	$\bar{w}\,\bar{x}\,\bar{y}z$
$\bar{w}xz$			X		X	X	X
$wy\bar{z}$	X			X			
$w\bar{x}y$		X		X			
$\bar{x}yz$		X				X	

The Quine-McCluskey Method: Example in four variables

- To cover these minterms we must include $\bar{w}z$ and $wy\bar{z}$, because these products are the only products that cover $\bar{w}xyz$ and $wxy\bar{z}$, respectively. Once these two products are included, we see that only one of the two products left is needed. Consequently, we can take either $\bar{w}z + wy\bar{z} + w\bar{x}y$ or $\bar{w}z + wy\bar{z} + \bar{x}yz$ as the final answer.

HOMEWORK: Exercises 2, 4, 6, 12, 14, 22, 24 on pp. 841-842