Discrete Mathematics (151)

Department of Mathematics
College of Sciences
King Saud University

Chapter 6 : Boolean Algebra

6.1. Boolean Functions (12.1 in book)

Boolean Functions

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

- The complement of an element, denoted with a bar, is defined by
$$\overline{0} = 1 \text{ and } \overline{1} = 0$$
- The Boolean sum, denoted by $+$ or by OR, has the following values:
$$1 + 1 = 1, 1 + 0 = 1, 0 + 1 = 1, 0 + 0 = 0.$$
- The Boolean product, denoted by . or by AND, has the following values:
$$1.1 = 1, 1.0 = 0, 0.1 = 0, 0.0 = 0.$$

When there is no danger of confusion, the symbol . 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.

Boolean Functions

Example 1

Find the value of $1.0 + \overline{(0 + 1)}$.

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

$1.0 + \overline{(0 + 1)} = 0 + \overline{1} = 0 + 0 = 0$

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.

Boolean Functions

Example 2

Translate $1.0 + \overline{(0 + 1)}$, 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 $(T \wedge F) \vee \neg(T \vee F) \equiv F$.

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

Example 3

Translate the logical equivalence $(T \wedge T) \vee \neg F \equiv T$ 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 $(1.1) + \overline{0} = 1$.

Boolean Expressions and Boolean Functions

Let $B = \{0, 1\}$. Then $B_n = \{(x_1, x_2, \ldots, x_n) | x_i \in B \text{ for } 1 \leq i \leq n\}$ 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.

TABLE 1

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

Example 4

The function $F(x, y) = x\bar{y}$ 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.

Boolean Functions

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 $\overline{E_1}$, $(E_1 E_2)$, 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 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.

Boolean Functions

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

Boolean Functions

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 $xy.1$ are equivalent. The **complement** of the Boolean function F is the function $\overline{F}$, where $\overline{F}(x_1, \cdots, x_n) = \overline{F(x_1, \cdots, x_n)}$.
Let F and G be Boolean functions of degree n. The **Boolean sum** $F + G$ and the **Boolean product** FG are defined by
$(F + G)(x_1, \cdots, x_n) = F(x_1, \cdots, x_n) + G(x_1, \cdots, x_n), (FG)(x_1, \cdots, x_n) = F(x_1, \cdots, x_n)G(x_1, \cdots, x_n)$.

Boolean Functions

Identities of Boolean Algebra

There are many identities in Boolean algebra. The most important of these are displayed in Table 5. These identities are particularly useful in simplifying the design of circuits. Each of the identities in Table 5 can be proved using a table.We will prove one of the distributive laws in this way in Example 6.

TABLE 5 Boolean Identities.

Identity	*Name*
$\overline{\overline{x}} = x$	Law of the double complement
$x + x = x$ $x \cdot x = x$	Idempotent laws
$x + 0 = x$ $x \cdot 1 = x$	Identity laws
$x + 1 = 1$ $x \cdot 0 = 0$	Domination laws
$x + y = y + x$ $xy = yx$	Commutative laws
$x + (y + z) = (x + y) + z$ $x(yz) = (xy)z$	Associative laws
$x + yz = (x + y)(x + z)$ $x(y + z) = xy + xz$	Distributive laws
$\overline{(xy)} = \overline{x} + \overline{y}$ $\overline{(x + y)} = \overline{x}\,\overline{y}$	De Morgan's laws
$x + xy = x$ $x(x + y) = x$	Absorption laws
$x + \overline{x} = 1$	Unit property
$x\overline{x} = 0$	Zero property

Example 6 (8 in book)

Show that the distributive law $x(y+z) = xy + xz$ is valid.

Solution: The verification of this identity is shown in Table 6. The identity holds because the last two columns of the table agree.

TABLE 6 Verifying One of the Distributive Laws.

x	y	z	$y+z$	xy	xz	$x(y+z)$	$xy+xz$
1	1	1	1	1	1	1	1
1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	1
1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0

Example 7 (9 in book)

Translate the distributive law $x + yz = (x + y)(x + z)$ in Table 5 into a logical equivalence.

Solution: To translate a Boolean identity into a logical equivalence, we change each Boolean variable into a propositional variable. Here we will change the Boolean variables x, y, and z into the propositional variables p, q, and r. Next, we change each Boolean sum into a disjunction and each Boolean product into a conjunction. (Note that 0 and 1 do not appear in this identity and 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 of chapter 1.

Boolean Functions

Identities in Boolean algebra can be used to prove further identities. We demonstrate this in Example 8.

Example 8 (10 in book)

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.)

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

$$
\begin{aligned}
x(x+y) &= (x+0)(x+y) && \text{Identity law for the Boolean sum} \\
&= x+0.y && \text{Distributive law of the Boolean sum over the Boolean product} \\
&= x+y.0 && \text{Commutative law for the Boolean product} \\
&= x+0 && \text{Domination law for the Boolean product} \\
&= x && \text{Identity law for the Boolean sum.}
\end{aligned}
$$

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 9 (11 in book)

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

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

Boolean Functions

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.

Example 10 (12 in book)

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.

Boolean Functions

The Abstract Definition of a Boolean Algebra

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:

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

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

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

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

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

6.2. Representing Boolean Functions (12.2 in book)

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 ., $+$, and $\bar{}$.
- 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.

Representing Boolean Functions

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 $y = 0$.

Representing Boolean Functions

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

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 $xy\bar{z}$ has the value 1 if and only if $x = y = 1$ and $z = 0$. Similarly, the product $\bar{x}y\bar{z}$ has the value 1 if and only if $x = z = 0$ and $y = 1$. The Boolean sum of these two products, $xy\bar{z} + \bar{x}y\bar{z}$, represents G, because it has the value 1 if and only if $x = y = 1$ and $z = 0$, or $x = z = 0$ and $y = 1$.

Representing Boolean Functions

DEFINITION 1

A **literal** is a Boolean variable or its complement. A **minterm** of the Boolean variables $x_1, x_2, \cdots, x_n$ 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.

Representing Boolean Functions

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.

Representing Boolean Functions

Example 3

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

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

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

Representing Boolean Functions

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

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

Representing Boolean Functions

CSP, CPS

- We denote by **CSP** form (**Complete sum-of-product**), is obtained by the previous methods.
- We denote by **CPS** form (**Complete product-of-sum**), is obtained by giving the CSP for the complement of the function, and we take the complement of the CSP give the CPS.

Example

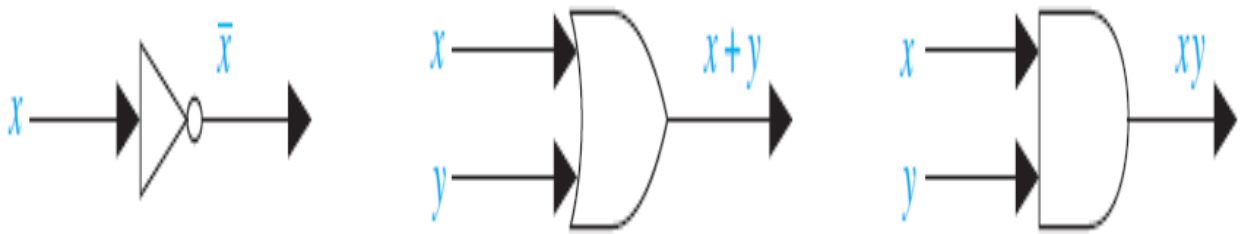
Find the CSP form and the CPS form, for the functions $f(x, y, z) = x + \overline{y}(\overline{x} + z)$ and $g(x, y, z) = \overline{x}(x + \overline{y} + y\overline{z})$.

Solution:

6.3. Logic Gates (12.3 in book)

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 in Sections 6.1 and 6.2. The basic elements of circuits are called **gates**. 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**.



Logic Gates

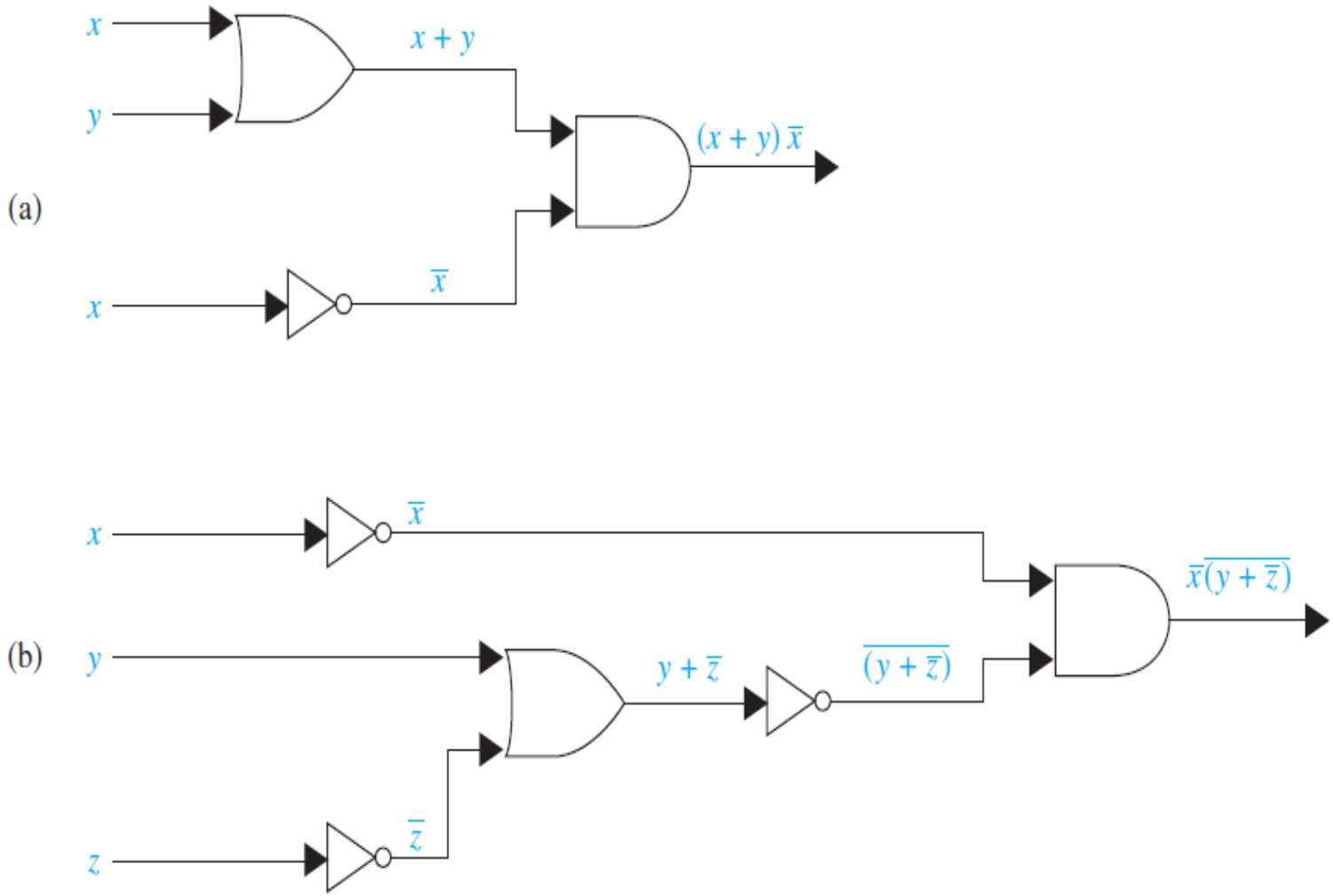
Combinations of Gates Combinational circuits can be constructed using a combination of inverters, OR gates, and AND gates. When combinations of circuits are formed, some gates may share inputs. This is shown in one of two ways in depictions of circuits. One method is to use branchings that indicate all the gates that use a given input. The other method is to indicate this input separately for each gate.

Example 1

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

Solution: Circuits that produce these outputs are shown in Figure 2.

Logic Gates


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

Logic Gates

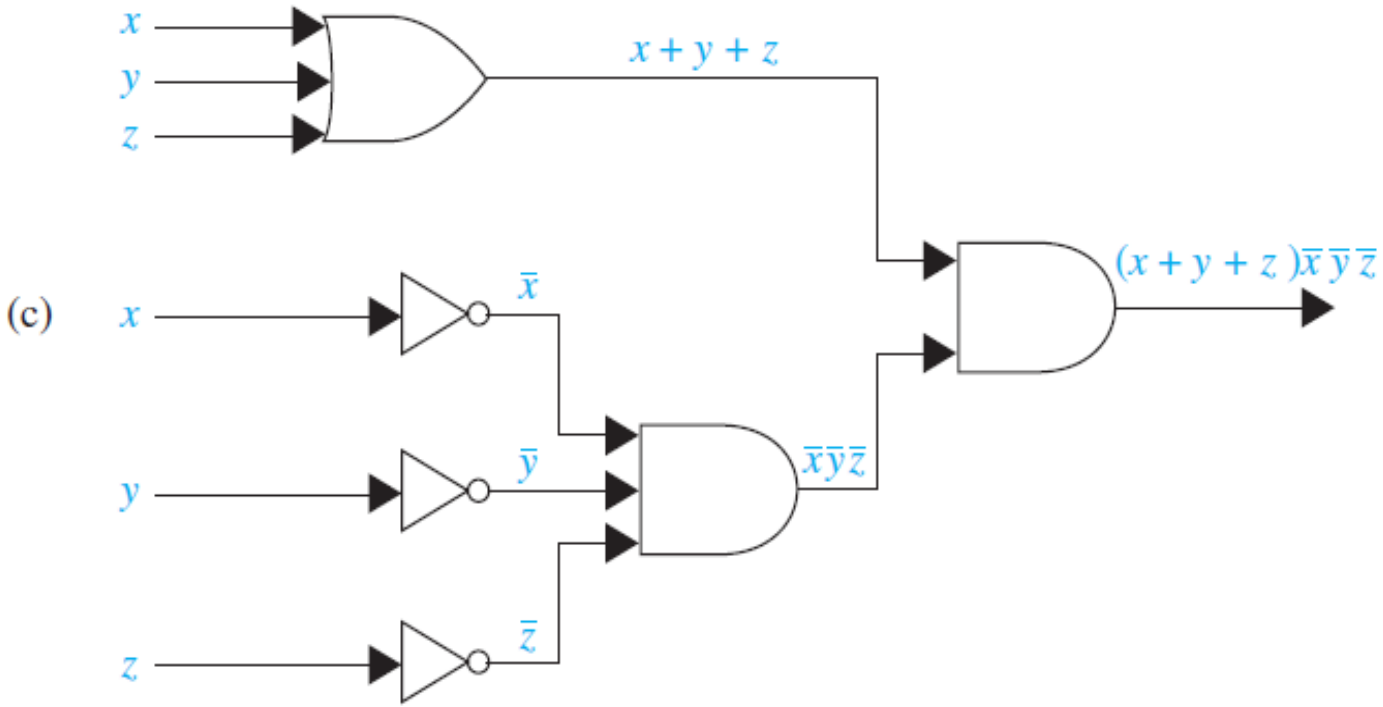


Figure 2: Circuits that Produce the Outputs Specified in Example 1.

Logic Gates

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.

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 Figure 3.

Logic Gates

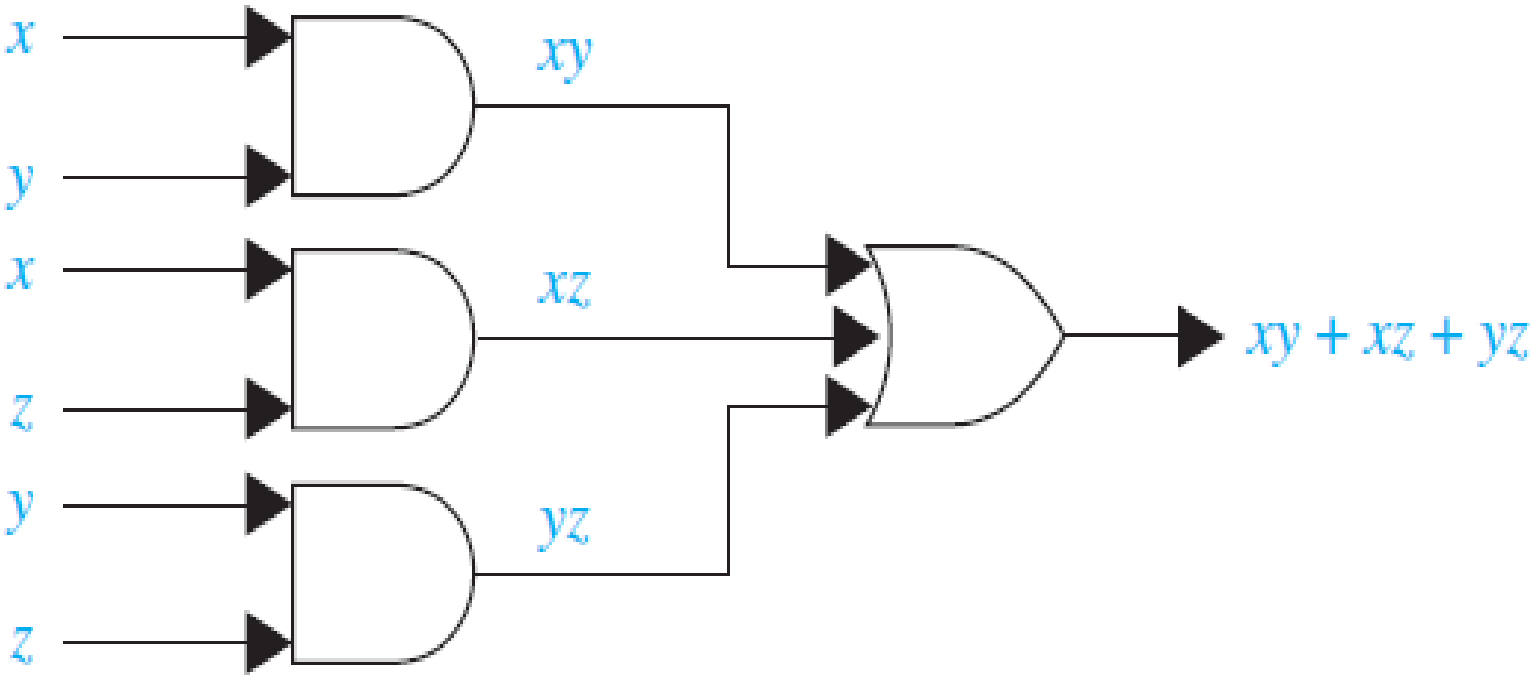


Figure 3: A Circuit for Majority Voting.

6.4. Minimization of Circuits (12.4 in book)

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

Minimization of Circuits

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 $xyz + x\bar{y}z = (y + \bar{y})(xz) = 1.(xz) = xz$.
Hence, xz is a Boolean expression with fewer operators that represents the circuit. We show two different implementations of this circuit in Figure 4. The second circuit uses only one gate, whereas the first circuit uses three gates and an inverter.

Minimization of Circuits

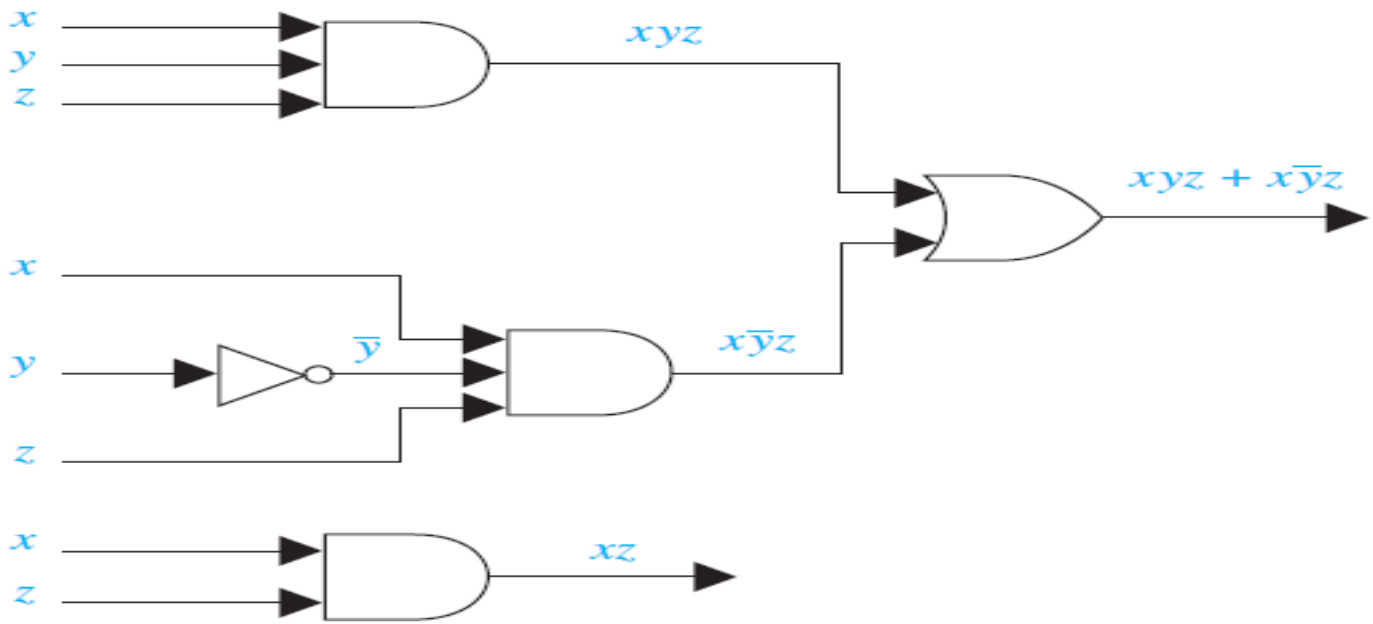


Figure 4: Two Circuits with the Same Output.

Minimization of Circuits

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.
- 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.

Minimization of Circuits

- 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 $\overline{x}y$ is adjacent to the cells representing xy and $x\overline{y}$. The four cells and the terms that they represent are shown in Figure 5.

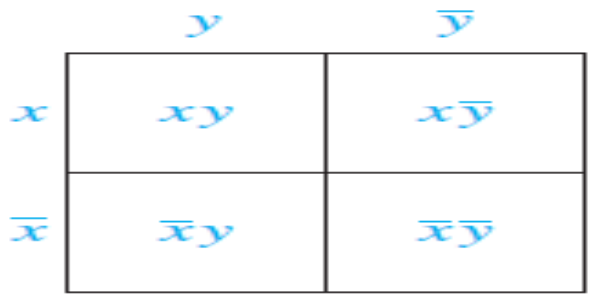


Figure 5: K-maps in Two Variables.

Minimization of Circuits

Example 1

Find the K-maps for (a) $xy + \overline{x}y$, (b) $x\overline{y} + \overline{x}y$, and (c) $x\overline{y} + \overline{x}y + \overline{x}\,\overline{y}$.
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 6.

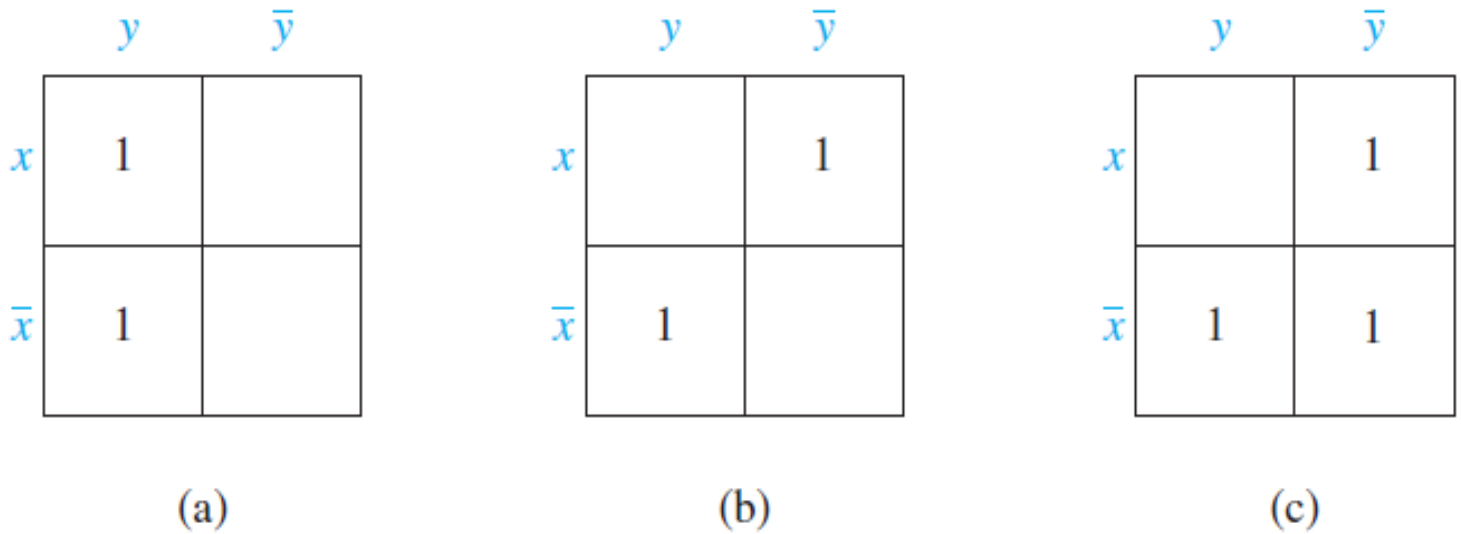


Figure 6: K-maps for the Sum-of-Products Expansions in Example 1.

Minimization of Circuits

Example 2

Simplify the sum-of-products expansions given in Example 1.

Solution: The grouping of minterms is shown in Figure 7 using the K-maps for these expansions. Minimal expansions for these sums-of-products are (a) y, (b) $x\overline{y} + \overline{x}y$, and (c) $\overline{x} + \overline{y}$

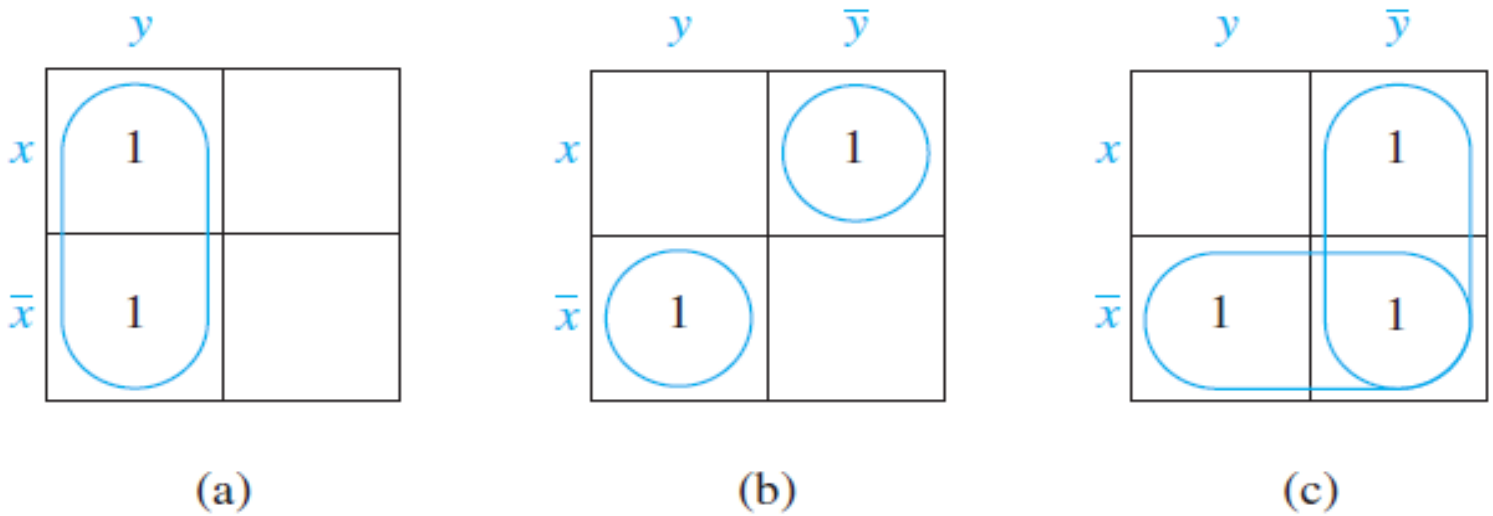


Figure 7: Simplifying the Sum-of-Products Expansions from Example 2.

Minimization of Circuits

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 Figure 8.

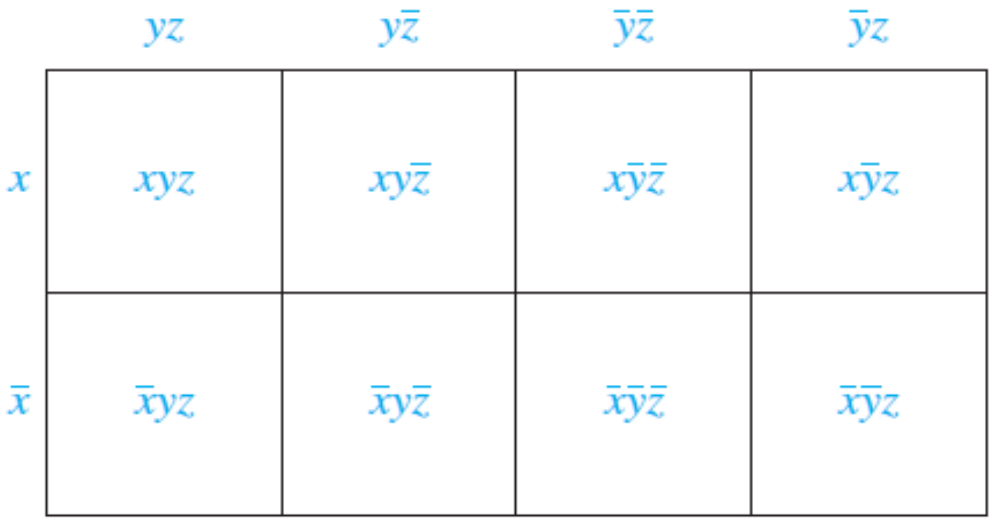


Figure 8: K-maps in Three Variables.

Minimization of Circuits

Example 3

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

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

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

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

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

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) $x\overline{z} + \overline{y}\ \overline{z} + \overline{x}yz$, (b) $\overline{y} + \overline{x}z$, (c) $x + \overline{y} + z$, and (d)$x\overline{z} + \overline{x}\ \overline{y}$. In part (d) note that the prime implicants $x\overline{z}$ and $\overline{x}\ \overline{y}$ are essential prime implicants, but the prime implicant $\overline{y}\ \overline{z}$ is a prime implicant that is not essential, because the cells it covers are covered by the other two prime implicants.

Minimization of Circuits

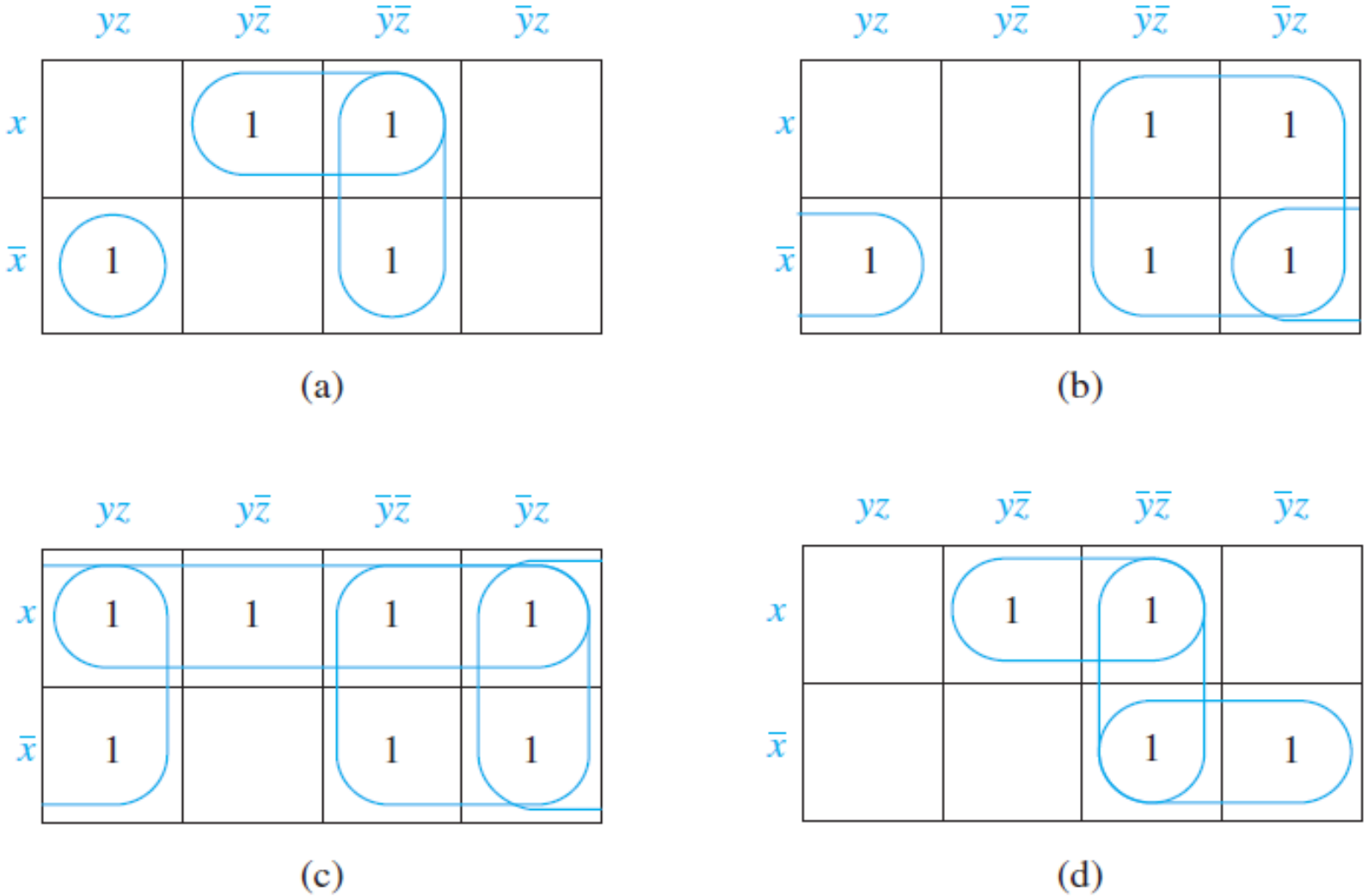


Figure 9: Using K-maps in Three Variables.

Minimization of Circuits

MSP, MPS

- We denote by **MSP** form (**Minimal sum-of-product**), is obtained using K-maps method.
- We denote by **CPS** form (**Minimal product-of-sum**), is obtained by giving the CSP for the complement of the function, and we take the complement of the CSP give the CPS.

Example

Find the MSP form and the MPS form, for the functions $f(x, y, z) = x + \overline{y}(\overline{x} + z)$ and $g(x, y, z) = \overline{x}(x + \overline{y} + y\overline{z})$.

Solution:

Exercices

Exercice

Let

	zw	zw'	z'w'	z'w
xy		1		1
xy'	1	1	1	1
x'y'	1		1	
x'y				

Be the Karnaugh -map for $f(x, y, z, w)$.

1. Find **MSP(f)** and **MPS(f)**.
2. Construct a minimal circuit using (AND-OR) gates, with $f(x, y, z, w)$ output.
3. Use **NAND** gates to construct circuits with $f(x, y, z, w)$ output.
4. Use **NOR** gates to construct circuits with $f(x, y, z, w)$ output.

Exercices

Exercice

Let

	zw	zw'	z'w'	z'w
xy		1	1	
xy'	1	1	1	
x'y'		1	1	
x'y	1	1	1	1

Be the Karnaugh -map for $f(x, y, z, w)$.

1. Find **MSP(f)** and **MPS(f)**.
2. Construct a minimal circuit using (AND-OR) gates, with $f(x, y, z, w)$ output.
3. Use **NAND** gates to construct circuits with $f(x, y, z, w)$ output.
4. Use **NOR** gates to construct circuits with $f(x, y, z, w)$ output.