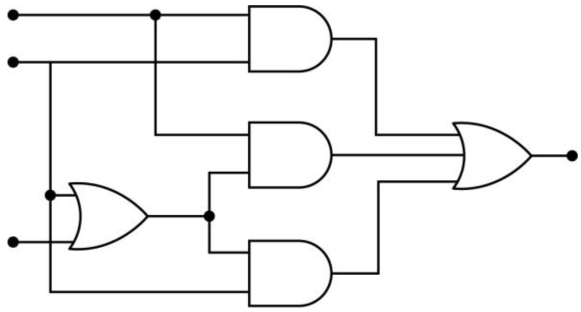

Logic circuits have many applications. Often we need to simplify complex logic circuits like the one shown above. In this unit, you will learn how to simplify such logic circuits using Boolean algebra.

UNIT 8

LOGIC CIRCUITS

Structure

STUDY GUIDE

In this unit, you will learn about logic circuits that are more complex than the ones you studied in Unit 7. You will learn Boolean algebra to simplify such circuits. This will require practice. You will learn how to convert truth tables into equivalent logic circuits. We advise you to treat this unit like a workbook and do every step yourself on a separate piece of paper. Do all solved Examples yourself, solve all SAQs and Terminal Questions on your own to learn the concepts of this unit properly.

> ***"In questions of science, the authority of a thousand is not worth the humble reasoning of a single individual."***
>
> ***Galileo***

8.1 INTRODUCTION

In Unit 7, you have studied the AND, OR and NOT logic gates and their combinations, NAND, NOR, XOR and XNOR gates. These gates are the building blocks of most digital circuits. Usually the logic circuits used in digital electronics are far more in number. While designing a digital circuit for an application, it is important to keep the number of logic gates minimum to minimise power consumption and reduce the complexities of the circuit. Boolean algebra (the algebra used to handle binary numbers) provides a way of reducing logic circuitry from complex to simpler circuits, expressing logic circuits symbolically and manipulating them.

Therefore, we begin our discussion by explaining Boolean algebra in Sec. 8.2. Then in Sec. 8.3, we use it to simplify logic circuits. You will learn the methods for doing this exercise. Finally, in Sec. 8.4, we explain the methods of converting a given truth table into an equivalent logic circuit through the sum of products and Karnaugh map.

In the next unit, you will learn applications of logic circuits in addition and subtraction of binary numbers.This is the basis of all arithmetic operations in computers.

Expected Learning Outcomes

After studying this unit, you should be able to:

- apply the rules of Boolean algebra to write Boolean equations for any given logic circuit and its truth table;
- write Boolean theorems and apply them for simplifying Boolean equations and logic circuits;
- obtain a truth table from a given Boolean expression; and
- obtain an equivalent logic circuit from a given truth table.

8.2 BOOLEAN ALGEBRA

In this section, you will learn about Boolean algebra, which provides the methodology for reducing a complex digital circuit into a simple one. Boolean algebra was invented by the mathematician George Boole in 1854. It is a branch of algebra in which there are only two values of variables: true and false denoted by 1 and 0, respectively. You will appreciate while studying this section why Boolean algebra forms the backbone of digital circuits and their analysis. We begin by stating the rules and postulates of Boolean algebra.

8.2.1 Rules and Postulates of Boolean Algebra

In Unit 7, you have learnt about various logic gates and their input-output relationships. We were using Boolean algebra without naming it there. So, let us formally state the rules of Boolean algebra.

1. In Boolean algebra, a variable can have only the value 0 or 1.
2. The Boolean equation for the word equation *Y* = NOT *A* (corresponding to the NOT gate) is:

$$Y = \overline{A} \quad (8.1)$$

The bar over *A* stands for the NOT operation NOT A and Eq. (8.1) is read as "*Y* equals NOT *A*" or "*Y* equals the complement of *A*". So,

if *A* is 0, then $Y = \overline{A} = \overline{0} = 1$ and

If *A* is 1, then $Y = \overline{A} = \overline{1} = 0$

3. The Boolean equation for the word equation *Y* = *A* OR *B* is:

$$Y = A + B \quad (8.2)$$

which is read as "*Y* equals *A* OR *B*". Note that this is the standard way of writing the input-output relationship for an OR gate. So, the symbol + in Boolean algebra stands for the OR operation (read the margin remark). Therefore, if

A is 0 and *B* is 0, then $Y = 0 + 0 = 0$

A is 0 and *B* is 1, then $Y = 0 + 1 = 1$

A is 1 and *B* is 0, then $Y = 1 + 0 = 1$

A is 1 and *B* is 1, then $Y = 1 + 1 = 1$

4. The Boolean equation for the word equation *Y* = *A* AND *B* is:

$$Y = A \,.\, B \quad \text{or} \quad Y = AB \quad (8.3)$$

which is read as "*Y* equals *A* AND *B*". Note that Eq. (8.3) is the standard way of writing the input-output relationship for an AND gate. So, the symbol **.** or no symbol between the variables in Boolean algebra stands for the AND operation. Therefore, if

A is 0 and *B* is 0, then $Y = 0 \,.\, 0 = 0$

A is 0 and *B* is 1, then $Y = 0 \,.\, 1 = 0$

A is 1 and *B* is 0, then $Y = 1 \,.\, 0 = 0$

A is 1 and *B* is 1, then $Y = 1 \,.\, 1 = 1$

You should not be confused by the use of the symbols of addition and multiplication in Boolean algebra. Whereas in decimal system, the symbols + and **.** stand for addition and multiplication, in Boolean algebra or in the discussion on logic circuits, these symbols stand for the OR and AND operations defined by Eqs. (8.2 and 8.3).

We are now ready to write equations for logic circuits using these rules of Boolean algebra. But before studying further, we would like you to learn the following notation that we use now.

Notation to be used

We use **italics for the bits that are involved in an operation**. For example, if we wish to perform the OR or AND operations on three bits, we use italicized letters as follows:

$$Y = A + B + C$$

or $$Y = A \,.\, B \,.\, C$$

But if we wish to write a word using bits, we do not italicise the bits. For example, we write:

a 4 bit word as ABCD

So, if A is 1, B is 1, C is 0 and D is 0, then $ABCD$ represents the AND operation on the 4 bits and is equal to:

$$A.\ B.\ C.\ D = 1.1.\ 0.0 = 0$$

But ABCD is the notation for a 4-bit word. For the values above, it is: 1100.

Let us summarise the Boolean expressions for the AND, OR and NOT gates in Table 8.1.

Table 8.1: Boolean expressions for AND, OR and NOT gates

AND operation	OR operation	NOT operation
$0 . 0 = 0$	$0 + 0 = 0$	$\bar{0} = 1$
$0 . 1 = 0$	$0 + 1 = 1$	$\bar{1} = 0$
$1 . 0 = 0$	$1 + 0 = 1$	
$1 . 1 = 1$	$1 + 1 = 1$	

The operations given in Table 8.1 constitute the ten postulates of Boolean algebra. Note that each of these postulates describes the input-output relationship of the concerned logic gate. You can now learn how to write and use a Boolean equation for a digital circuit. Go through the example below.

EXAMPLE 8.1 : WRITING BOOLEAN EQUATION

Write the Boolean equation for the following logic circuit.

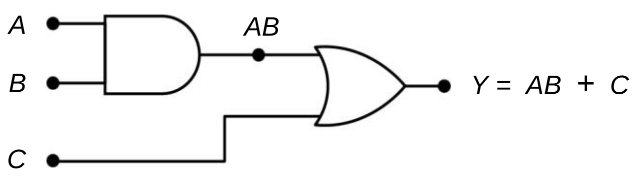

Fig. 8.1: Digital circuit comprising AND and OR gates.

SOLUTION ■ Note from Fig. 8.1 that A and B are the inputs of AND gate. C is one input of the OR gate and the output of the AND gate, i.e., $A . B$ (AB) is the other input. So, the output of the OR gate is:

$$Y = AB + C$$

This is the Boolean equation for the circuit in Fig. 8.1.

Let us use Table 8.1 to find the output Y of the circuit shown in Fig. 8.1 for different values of the inputs A, B and C. For example,

For $A = 1$, $B = 1$ and $C = 1$,

$$Y = 1 .1 + 1 = 1 \text{ (since } 1.1 = 1 \text{ and } 1 + 1 = 1 \text{ from Table 8.1)}$$

Similarly, for $A = 0$, $B = 1$ and $C = 1$,

$$Y = 0 .1 + 1 = 1 \text{ (since } 0.1 = 0 \text{ and } 0 + 1 = 1 \text{ from Table 8.1)}$$

You could practice this for other values of the inputs and construct the truth table for the logic circuit of Fig. 8.1. Solve SAQ 1 before studying further.

SAQ 1 - **Boolean equation for a given circuit**

a) Draw the truth table for the logic circuit of Fig. 8.1.

b) Write the Boolean equation for the logic circuit of Fig. 8.1 if the AND gate and OR gate are interchanged in it.

Using Table 8.1, we now write several Boolean theorems or identities and the de Morgan's theorems that are used in Boolean algebra.

8.2.2 Boolean Laws and Theorems

It is worthwhile to recall what you have learnt in Unit 7 about the outputs of various primary gates:

A. i) Output of an AND gate is 1 only when all the inputs are 1.

ii) Output of an AND gate is 0 when all or any of the inputs is 0.

B. i) Output of an OR gate is 0 when all the inputs are 0.

ii) Output of an OR gate is 1 when either of the inputs or all the inputs are 1.

C. Output of a NOT gate is inversion of its input.

From these statements and postulates of Table 8.1, we derive the following Boolean theorems and laws:

From AND function:

1. $X . 0 = 0$

2. $0 . X = 0$

3. $X . 1 = X$

4. $1 . X = X$

From OR function:

5. $X + 0 = X$

6. $0 + X = X$

7. $X + 1 = 1$

8. $1 + X = 1$

Combination of a variable with itself or its complement:

9. $X \cdot X = X$

10. $X \cdot \overline{X} = 0$

11. $X + X = X$

12. $X + \overline{X} = 1$

From double complementation:

13. $\overline{\overline{X}} = X$

Commutative laws for multiplication and addition:

These laws show that the order in which two variables are ORed or ANDed together makes no difference.

14. $X \cdot Y = Y \cdot X$

15. $X + Y = Y + X$

Associative laws for addition and multiplication:

These laws show that while ORing or ANDing several variables, it makes no difference in what order the variables are grouped.

16. $X + (Y + Z) = (X + Y) + Z = X + Y + Z$

17. $X (YZ) = (XY) Z = XYZ$

Distributive laws:

18. $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$

19. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$

20. $(W + X) \cdot (Y + Z) = WY + XY + WZ + XZ$

Did you note here that the commutative, associative and distributive laws of Boolean algebra are similar to ordinary algebra? So, it should not be difficult for you to remember those.

Absorption laws:

These laws have no counterpart in ordinary algebra.

21. $X + X.Y = X$

22. $X . (X + Y) = X$

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

24. $X . (\overline{X} + Y) = XY$

De Morgan's First and Second Theorems:

De Morgan's first theorem states that the complement of a sum is equal to the product of complements:

25. $\overline{X + Y} = \overline{X} . \overline{Y}$

De Morgan's second theorem states that the complement of a product is equal to the sum of complements:

26. $\overline{X.Y} = \overline{X} + \overline{Y}$

We advise you to memorise all the laws and theorems stated in this section. All these are used to simplify logic circuits. Let us take up a simple application of these laws and theorems to simplify Boolean expressions. You will be using such applications in Secs. 8.3 and 8.4.

EXAMPLE 8.2 : APPLYING BOOLEAN THEOREMS

Simplify the following Boolean expressions:

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

b) $ABC + A\overline{B} + AB\overline{C}$

SOLUTION ■ We simplify the expressions in stages, two at a time.

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

We first simplify the product of the first two terms:

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

$= 0 + AB + B\overline{A} + B$ (using 10 and 9)

$= (A + \overline{A})B + B = B.1 + B = B$ (using 12 and 11)

$\therefore (A + B)(\overline{A} + B)(\overline{B} + C) = B(\overline{B} + C) = B\overline{B} + BC = BC$ (using 10)

b) $ABC + A\overline{B} + AB\overline{C} = ABC + AB\overline{C} + A\overline{B}$

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

$= AB.1 + A\overline{B} = A(B + \overline{B}) = A$ (using 12 in both steps)

We now introduce the concepts of fundamental products, minterms and maxterms in Boolean algebra used for simplying logic circuits.

8.2.3 Fundamental Products, Minterms and Maxterms

Fundamental products in Boolean algebra are defined as the products that produce a high (1) output.

For example, suppose we have two inputs *A* and *B*. Then all AND operations on *A* and *B* that produce 1 as the output are fundamental products of *A* and *B*. So, we can have four possible combinations of *A* and *B* and the corresponding fundamental products are shown in Table 8.2.

Table 8.2: Fundamental products of two inputs

A	*B*	Fundamental product	Output
0	0	$\overline{A}\,\overline{B}$	1
0	1	$\overline{A}B$	1
1	0	$A\overline{B}$	1
1	1	AB	1

Do you recognize the pattern in Table 8.2 above?

Whenever the input is zero, we take its complement in the product.

What will be the fundamental products for 3 inputs? Let us find out.

There will be 8 combinations with corresponding fundamental products shown in Table 8.3.

Table 8.3: Fundamental products of three inputs

A	*B*	*C*	Fundamental product	Output
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	1
0	0	1	$\overline{A}\,\overline{B}C$	1
0	1	0	$\overline{A}B\overline{C}$	1
0	1	1	$\overline{A}BC$	1
1	0	0	$A\overline{B}\,\overline{C}$	1
1	0	1	$A\overline{B}C$	1
1	1	0	$AB\overline{C}$	1
1	1	1	ABC	1

Minterm also known as the **Sum of Product** (SOP) refers to the products of the variables that are separated by a plus sign. The variables can be complemented or uncomplemented.

A sum of products (minterm) is obtained as follows: For each row of the truth table for which the output is 1, the Boolean term is the product of variables that are equal to 1 and the complement of variable that are equal to 0. The sum of these products is the desired minterm. For example, refer to the three-input truth table for a given logic circuit presented in Table 8.4:

Table 8.4: Minterms for three inputs

S. No.	*A*	*B*	*C*	Output
1.	0	0	0	0
2.	0	0	1	1
3.	0	1	0	1
4.	0	1	1	1
5.	1	0	0	1
6.	1	0	1	0
7.	1	1	0	0
8.	1	1	1	0

So, in Table 8.4, the outputs at serial numbers 2 to 5 are 1. The Boolean term for each of these outputs is:

$\bar{A}\bar{B}C \quad \bar{A}B\bar{C} \quad \bar{A}BC \quad A\bar{B}\bar{C}$

Therefore, the sum of product (minterm) is:

$$\bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C}$$

And the sum of product Boolean equation is:

$$Y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C}$$

Maxterm also known as the **Product of Sum** (POS) refers to the sum of variables that are separated by a multiplication sign. The variables can be complemented or uncomplemented.

A product of sum expression is obtained as follows: For each row of the truth table for which the output is 0, the Boolean term is the sum of the variables that are equal to 0 and the complement of the variables that are equal to 1. The product of these sums is the desired maxterm. For example, refer to the truth table 8.4 again. Consider all rows in which the output is 0. There are 4 such rows. So, the 4 Boolean terms are:

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

Therefore, the product of sum (maxterm) is:

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

And the product of sum Boolean equation is:

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

Before studying further, you may like to solve SAQ 2 based on Secs. 8.2.2 and 8.2.3.

SAQ 2 – **Applying Boolean laws and theorems**

a) Simplify the Boolean expression $A\overline{B}C + AB\overline{C} + ABC$.

b) Determine the minterm (SOP) and maxterm (POS) Boolean equations for the truth tables of 2-input NAND and NOR gates.

8.3 SIMPLIFICATION OF LOGIC CIRCUITS

So far, you have learnt that a logic circuit can be expressed in the form of Boolean expression which, in turn, can be simplified using Boolean laws. You will now learn how to simplify complex logic circuits. Let us set out the rules for simplifying logic circuits and corresponding Boolean expressions, and then apply them.

8.3.1 Rules for Simplifying Boolean Equations

A Boolean expression can be simplified in either of the two forms: a) Sum of Product (SOP), or b) Product of Sum (POS). Since the SOP form is used most commonly, we shall limit our discussion in this section to only SOP. The aim of simplifying a logic circuit is to minimize the operation symbols and hence the number of logic gates in it or the variables in the corresponding Boolean expression. Many a times we get more than one simplified form of a logic circuit or a Boolean expression for the circuit, each being equivalent in the number of gates and variables to be used. In final analysis, we will use the **Minimum Sum of Product** (MSP) form for the Boolean expression, which is written without brackets. The corresponding logic circuit would then be the simplest.

For example, consider the reduced expression $A(B + C)$ which is written in MSP form $AB + AC$. While the reduced expression requires one AND gate and one OR gate, the MSP expression requires two AND gates and one OR gate. Thus, in this case the MSP expression is not the simplest. The fundamental rule of simplification is that the Boolean expression for a logic circuit must be

a) **reduced as much as possible**, and

b) **written without brackets**.

For the simplification of Boolean expressions for logic circuits, the Boolean operations should be carried out in the following order:

1) Inversion of single variables.
2) All operations with brackets.
3) AND operations before OR operations.
4) OR operations.
5) If an expression is with a bar, then before inverting perform all operations.

Let us consider an example for simplifying logic circuits.

EXAMPLE 8.3 : SIMPLIFYING LOGIC CIRCUITS

Simplify the following logic circuit for which the Boolean equation is given by

$$Y = \overline{AB} + (\overline{A} + \overline{B})\overline{C} \qquad \text{(i)}$$

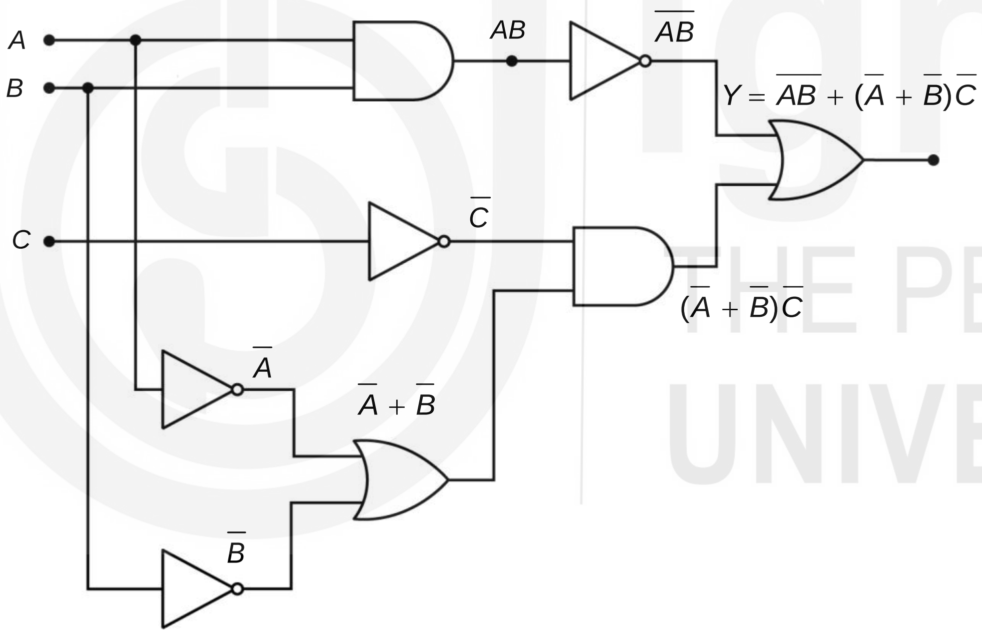

Fig. 8.2: **Digital circuit for** $Y = \overline{AB} + (\overline{A} + \overline{B})\overline{C}$.

SOLUTION ■ We will obtain the MSP expression for the Boolean equation (i) using Boolean laws and theorems.

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

$= (\overline{A} + \overline{B})\overline{C} + (\overline{A} + \overline{B})$ using de Morgan's 2^{nd} theorem 26

$= (\overline{A} + \overline{B})(\overline{C} + 1)$ taking $(\overline{A} + \overline{B})$ common

$= (\overline{A} + \overline{B}).1 = \overline{A} + \overline{B}$ using 7 and 3

So, the MSP expression is: $Y = \overline{A} + \overline{B}$

Fig. 8.3 shows the simplified logic circuit for the MSP so obtained.

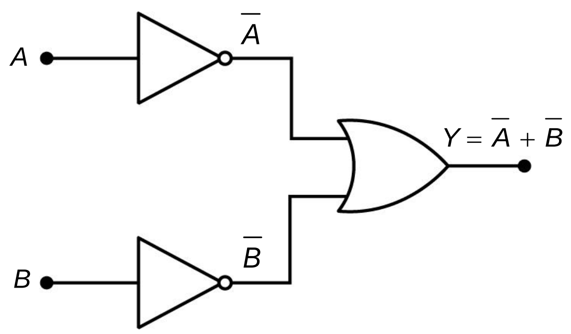

Fig. 8.3: **Digital circuit for** $Y = \overline{A} + \overline{B}$.

Let us consider another example on simplifying logic circuits.

EXAMPLE 8.4 : SIMPLIFYING LOGIC CIRCUITS

Simplify the following logic circuit for which the Boolean equation is given by

$$Y = AB + A(B + C) + B(B + C) \qquad \text{(i)}$$

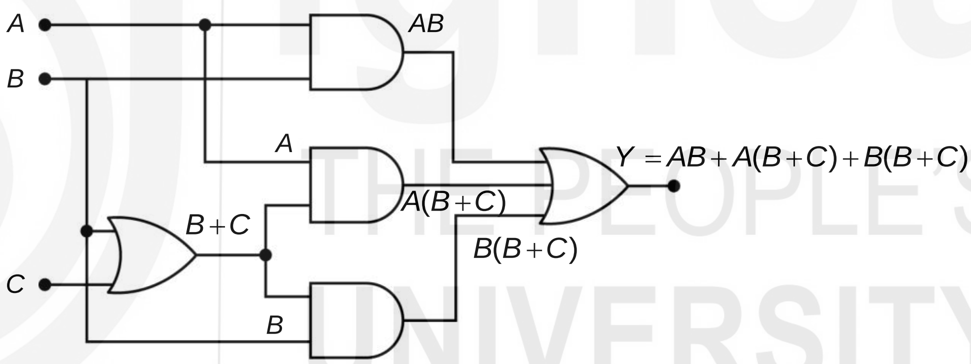

Fig. 8.4: **Digital circuit for** $Y = AB + A(B+C) + B(B+C)$.

SOLUTION ■ We will obtain the MSP expression for the Boolean equation (i) using Boolean laws and theorems.

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

$$= AB + AB + AC + BB + BC$$

$$= AB + AB + AC + B + BC \qquad \text{using 9}$$

$$= AB + AC + B + BC \qquad \text{using 11}$$

$$= AB + AC + B(1 + C) \qquad \text{taking } B \text{ common}$$

$$= AB + AC + B.1 \qquad \text{using 8}$$

$$= (A + 1)B + AC \quad = 1.B + AC \qquad \text{using 7}$$

The simplified logic circuit for the MSP so obtained is shown in Fig. 8.5

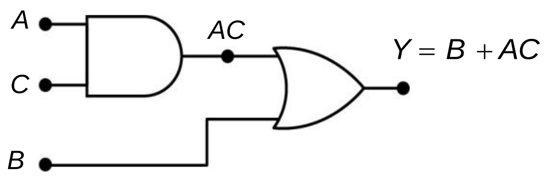

Fig. 8.5: **Digital circuit for** $Y = B + AC$.

Let us consider yet another example.

EXAMPLE 8.5 : SIMPLIFYING LOGIC CIRCUITS

Simplify the following logic circuit for which the Boolean equation is given by

$$Y = \overline{A}C + AB(\overline{B} + C) \qquad \text{(i)}$$

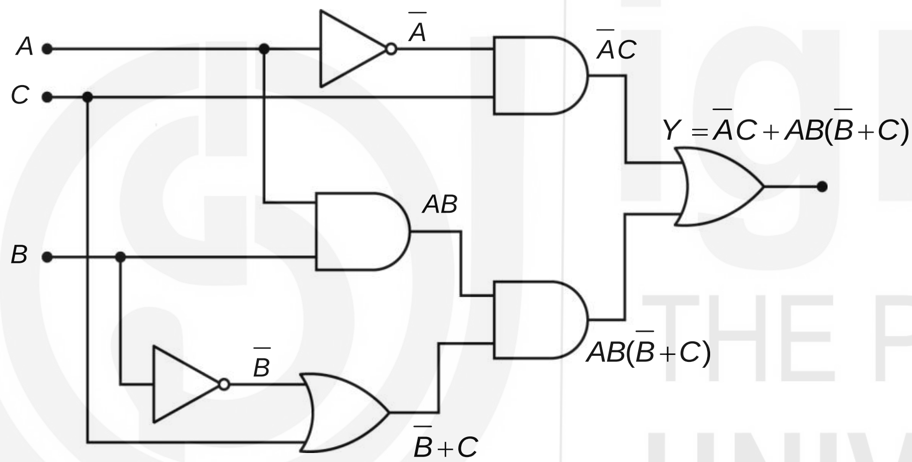

Fig. 8.6: **Digital circuit for** $Y = \overline{A}C + AB(\overline{B} + C)$.

SOLUTION ■ We will obtain the MSP expression for the Boolean equation (i) using Boolean laws and theorems.

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

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

$= \overline{A}C + A.0 + ABC$ using 10

$= \overline{A}C + ABC$ using 1

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

$= (\overline{A} + B)C$ using 23

$= \overline{A}C + BC$

So, the MSP expression is: $Y = \overline{A}C + BC$

The simplified logic circuit for the MSP so obtained is shown in Fig. 8.7:

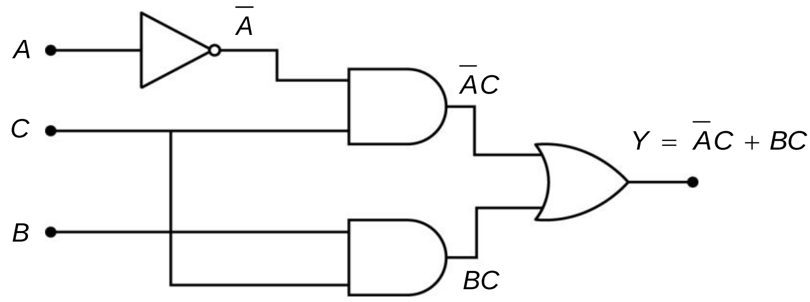

Fig. 8.7: **Digital circuit for** $Y = \overline{A}C + BC$.

You should now solve SAQ 3 to simplify a Boolean equation for a logic circuit on your own.

SAQ 3 - Simplifying Boolean equations

Obtain the MSP expression for the Boolean equation:

$Y = A\overline{B}\overline{C} + AB\overline{C} + ABC.$

In this section, you have learnt the rules for simplifying Boolean equations to obtain simple logic circuits having a minimum number of logic gates from complex logic circuits. To sum up, you have used the following steps to do so:

1) Write the logic operation in the form of a Boolean expression.
2) Obtain the simplified Boolean expression using Boolean laws and theorems.
3) Draw the digital circuit of the reduced Boolean expression having minimum number of gates.

8.3.2 Obtaining a Truth Table from a Boolean Expression

You have constructed truth tables from the Boolean expressions for various logic gates in Unit 7. So, you are familiar with the simplest method of obtaining a truth table from a Boolean expression. You substitute the values of the inputs in all possible combinations.

For example, you know that for 2 inputs, you will have 4 combinations, for 3 inputs, 8 combinations, and so on. You have learnt that for *n* inputs there will be 2^n combinations. And for each combination of input values, you found the value of output and put it in the corresponding position in the truth table.

Let us illustrate this method again with the help of an example.

Suppose you have to construct the truth table for the following Boolean expression:

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

There are 3 inputs, and so there will be 8 combinations as shown in Table 8.5:

Table 8.5: Obtaining truth table from Boolean expression

A	*B*	*C*	*Y*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

We have obtained the output for each input combination as we did in Unit 7. For example, in row 5, when *A* is 1, and *B* and *C* are 0, we get

$$Y = 1.0 + 1.(0+0) + 0.(0+0)$$

$$= 0 + 1.0 + 0.0$$

$$= 0 + 0 + 0 = 0$$

We obtain the output values for all input combinations in the same way.That is how we get the truth table 8.5.

You may like to verify the *Y* values in Table 8.5. Solve SAQ 4.

***SAQ 4* - Obtaining truth table from a Boolean expression**

Verify the output values in the 2nd , 3rd and 8th rows in Table 8.5.

We will now explain another method of obtaining a truth table from a Boolean expression that involves reasoning.

Let us ask: When will the output of the Boolean expression be 1?

For example, consider the MSP expression:

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

The output *Y* in this expression will be 1 as long as either $A\overline{C}$ or *BC* is 1. So, you should put *Y* = 1 for all entries of $A\overline{C} = 1$ and/or $BC = 1$.

So, in the truth table 8.6, you should put the output as 1 in rows 5 and 7, and 4 and 8. For all other rows *Y* is 0.

Table 8.6 is then the truth table for the given Boolean expression.

Table 8.6: Truth table for $Y = A\overline{C} + BC$

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Hence, it is better to use the method of reasoning to obtain the truth table for a given Boolean expression. This is just a two-step method:

1. Obtain the MSP form of the given Boolean expression, and
2. Reason out which of the truth table entries should be 1 for each product or term in the MSP expression.

Let us illustrate this method with the help of an example.

EXAMPLE 8.6 : OBTAINING TRUTH TABLE

Obtain the truth table for the Boolean equation:

$$Y = A + AB + BCD \qquad \text{(i)}$$

SOLUTION ■ We will obtain the MSP expression for the Boolean equation (i) using Boolean laws and theorems.

$$Y = A + AB + BCD$$

$$= A(1 + B) + BCD$$

$$= A.1 + BCD$$

$$= A + BCD$$

So, the MSP expression is: $Y = A + BCD$

From reasoning, we can see that $Y = 1$ whenever A is 1 or the product BCD is 1.

Therefore, in the truth table for the equation, we put $Y = 1$ for all entries of $A = 1$ (i.e., the entries 9 to 16 in Table 8.7) and all entries of $BCD = 1$ (i.e., the entries 8 and 16 in Table 8.7).

Put $Y = 0$ for all other entries in Table 8.7.

Table 8.7: Truth table for $Y = A + BCD$

	A	B	C	D	Y
1.	0	0	0	0	0
2.	0	0	0	1	0
3.	0	0	1	0	0
4.	0	0	1	1	0
5.	0	1	0	0	0
6.	0	1	0	1	0
7.	0	1	1	0	0
8.	0	1	1	1	1
9.	1	0	0	0	1
10.	1	0	0	1	1
11.	1	0	1	0	1
12.	1	0	1	1	1
13.	1	1	0	0	1
14.	1	1	0	1	1
15.	1	1	1	0	1
16.	1	1	1	1	1

Solve SAQ 5 to practice this method of obtaining the truth table from Boolean expression.

***SAQ 5* – Obtaining truth table from a Boolean expression**

Obtain the truth table for $Y = AB + BC + CA$.

8.4 CONVERSION OF TRUTH TABLE INTO EQUIVALENT LOGIC CIRCUIT

In the last section of this unit, we explain how to convert a given truth table into a Boolean expression and hence into the corresponding logic circuit. We will use two methods to do so: Sum of Product Method and Karnaugh Maps.

8.4.1 Sum of Product Method

Let us explain this method of converting a truth table into a Boolean expression with the help of an example. Consider the truth table 8.8.

Table 8.8: Given truth table

	A	B	C	Y
1.	0	0	0	0
2.	0	0	1	0
3.	0	1	0	0
4.	0	1	1	0
5.	1	0	0	1
6.	1	0	1	0
7.	1	1	0	1
8.	1	1	1	1

We write down the Boolean expression that describes this truth table by writing the Boolean expression for each line in the truth table where the output is 1, as a product. Then we sum the products. Note that the entries 5, 7 and 8 in the table have output 1. What is the Boolean expression for each entry?

To obtain the Boolean expression, we need only to write a product term for each entry with output 1. Therefore, we proceed as follows:

Entry 5: $Y = 1$ for $A = 1, B = 0, C = 0$

Hence, $Y = A\overline{B}\overline{C}$

because the output of an AND gate is 1 only when all its inputs are 1. Similarly,

Entry 7: $Y = 1$ for $A = 1, B = 1, C = 0$

Hence, $Y = AB\overline{C}$

Entry 8: $Y = 1$ for $A = 1, B = 1, C = 1$

Hence, $Y = ABC$

Now we sum these products using the OR logic. Therefore, the Boolean expression for the truth table is:

$$Y = A\overline{B}\overline{C} + AB\overline{C} + ABC \qquad \text{(SOP)}$$

We can simplify this Boolean expression as follows:

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

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

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

$$= A(B + \overline{C}) \qquad \text{(using 23)}$$

Therefore, the MSP expression is:

$$Y = AB + A\overline{C}$$

You can easily draw the logic circuit for this MSP expression. Draw it before studying further in SAQ 6.

SAQ 6 – Obtaining logic circuit from a truth table

Draw the logic circuit for $Y = AB + A\overline{C}$ obtained for Table 8.8.

Let us summarise the sum of product method:

1. Combine with an AND operation, all the input variables for the entries for which the output is 1.
2. Select the input variable with its complement or without so that the product of the input values (ANDed values of the inputs) gives 1. You can see that these are the **fundamental products**.
3. Assemble the products with the OR operation.
4. The sum of product so obtained may not be an MSP expression. Use Boolean laws and theorems to obtain the MSP expression.

You may like to practice this method from step 1. Solve SAQ 7.

SAQ 7 – Obtaining logic circuit from a truth table

Obtain the Boolean expression for the truth table given in Table 8.9.

Table 8.9: Given truth table

	A	*B*	*C*	*Y*
1.	0	0	0	1
2.	0	0	1	0
3.	0	0	1	0
4.	0	1	1	0
5.	1	0	0	1
6.	1	0	1	0
7.	1	1	0	0
8.	1	1	1	1

We now explain another method for converting a given truth table into a Boolean expression for a logic circuit.

8.4.2 Karnaugh Map

Let us begin by explaining how to construct a two variable Karnaugh map for a 2-input truth table such as Table 8.10.

Table 8.10: Given truth table

A	*B*	*Y*
0	0	0
0	1	1
1	0	0
1	1	1

Let us construct the Karnaugh map for Table 8.10 as an example. We first draw a square like the one shown in Fig. 8.8a and write the input variables and their complements as shown there. We put the first variable in the vertical column on the left of the square and the second variable in the horizontal row on top of the square. Also note the order in which we have put the input variables and their complements. In the vertical column $\overline{A}$ (complement of *A*) comes before *A*. In the horizontal row, $\overline{B}$ is placed before *B*. You should always remember this sequence.

Now we look for the output 1 in Table 8.10. Note that the first output 1 appears when the input *A* is 0 and *B*, 1. The fundamental product for this is $\overline{A}B$ because for the input values of *A* (0) and *B* (1), $\overline{A}B$ is equal to 1. So, we put the first output 1 as shown in Fig. 8.8b. The second output 1 corresponds to the fundamental product *AB*. So, we put the second output 1 as shown in Fig. 8.8c.

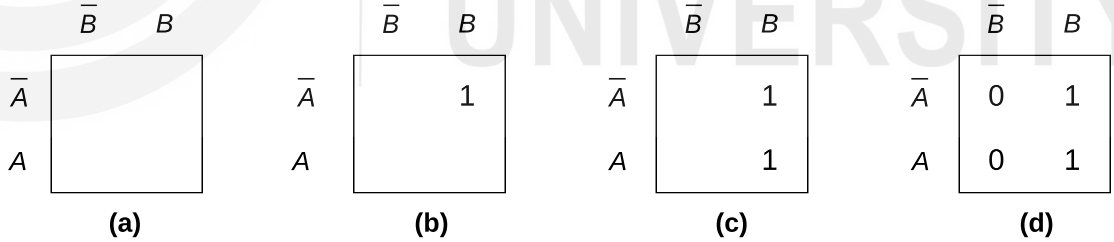

Fig. 8.8: Constructing Karnaugh map for 2-input truth table.

The last step is to enter 0 in all remaining spaces in the Karnaugh map. Thus, we get the final Karnaugh map as shown in Fig. 8.8d. We can write the sum of products (Boolean expression) for the output by just looking at the Karnaugh map (Fig. 8.8d). It is: $Y = \overline{A}B + AB.$ You can draw the logic circuit yourself. Let us explain this method further by taking a 3-input truth table such as Table 8.12. But before studying further, solve SAQ 8 to check your understanding.

Table 8.11

A	*B*	*Y*
0	0	0
1	0	1
1	0	0
1	1	1

***SAQ 8* – Obtaining logic circuit by drawing Karnaugh map**

Draw the Karnaugh map for Table 8.11.

Consider Table 8.12. Let us draw the Karnaugh map for it.

Table 8.12: Given truth table

A	*B*	*C*	*Y*
0	0	0	0
0	0	1	1
0	0	1	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

We begin by drawing Fig. 8.9a. Note the order of the variables and their complements and always remember it. Note that the order $\bar{A}\bar{B}$, $\bar{A}B, AB$ and $A\bar{B}$ follows the order 00, 01, 11 and 10. It is done so that only one variable changes from uncomplemented to complemented form or vice-versa. Next, we look for all outputs 1 in Table 8.12. These are in the rows 2, 4 and 8. We write the corresponding fundamental products: $\bar{A}\bar{B}C$, $\bar{A}BC$ and ABC. We enter the outputs 1 in the Karnaugh map as shown in Fig. 8.9b. Then we enter 0 in the remaining spaces as shown in Fig. 8.9c.

	$\bar{C}$	C
$\bar{A}\bar{B}$		
$\bar{A}B$		
AB		
$A\bar{B}$		

(a)

	$\bar{C}$	C
$\bar{A}\bar{B}$		1
$\bar{A}B$		1
AB		1
$A\bar{B}$		

(b)

	$\bar{C}$	C
$\bar{A}\bar{B}$	0	1
$\bar{A}B$	0	1
AB	0	1
$A\bar{B}$	0	0

(c)

Fig. 8.9: Constructing Karnaugh map for 3-input truth table.

We can write the sum of products (Boolean expression) for the output by just looking at the Karnaugh map (Fig. 8.9c). It is: $Y = \bar{A}\bar{B}C + \bar{A}BC + ABC$. Karnaugh maps are useful also because they give us a handy tool for simplifying logic circuits. But we will not be dealing with that aspect here. We now summarise what you have learnt in this unit.

8.5 SUMMARY

Concept	Description
Boolean algebra	■ All logic gates and circuits work in binary mode, that is, the inputs and outputs in these circuits can have values only 0 or 1. Boolean algebra is

used to describe the input-output relationships of logic circuits because in Boolean algebra, a variable can have only the value 0 or 1. The rules, laws and theorems of Boolean algebra are obtained from the truth tables of the three primary gates, AND, OR and NOT. **Fundamental products** in Boolean algebra are defined as the products that produce a high (1) output. **Minterm** also known as the **Sum of Product** refers to the products of the variables that are separated by a plus sign. **Maxterm** also known as the **Product of Sum** refers to the sum of variables that are separated by a multiplication sign.

Simplification of logic circuits

- A digital circuit can be expressed as a Boolean expression and a logic circuit can be obtained from a Boolean expression. Boolean laws and theorems can be used to simplify the Boolean expression for a logic circuit and thus obtain a simplified logic circuit. In such applications, a Boolean expression is first simplified to give a simpler circuit. The fundamental rule of simplification is that the Boolean expression for a logic circuit must be

  a) **reduced as much as possible**, and

  b) **written without brackets**.

  For the simplification of Boolean expressions for logic circuits, the Boolean operations should be carried out in the following order:

  1. Inversion of single variables.
  2. All operations with brackets.
  3. AND operations before OR operations.
  4. OR operations.
  5. If an expression is with a bar, then before inverting perform all operations.

  A truth table can be obtained from a Boolean expression without reference to its logic circuit.

Conversion of truth table into Boolean expression and equivalent logic circuit

- A Boolean expression and the corresponding logic circuit can be obtained from a given truth table using two methods: Sum of Product (SOP) and Karnaugh map. In both methods, the Boolean expression is obtained in the form of sum of products, which can further be simplified to get the Minimum-Sum-of-The-Product (MSP) form. The MSP expression is used to draw the final logic circuit.

8.6 TERMINAL QUESTIONS

1. Simplify the expression $Y = A\bar{B}D + A\bar{B}\bar{D}$ to its MSP form.
2. Simplify the expression $Y = BCD + A\bar{B}CD$ to its MSP form.
3. Simplify the expression $Y = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D}$ to its MSP form.
4. Simplify the expression $Y = \overline{(A + BC).(D + FG)}$ to its MSP form.
5. Write the truth table for the MSP form obtained in Terminal Question 2.
6. Write the truth table for $Y = \bar{A}B + A\bar{C}$.

7. Write the Boolean expression for the truth table given in Table 8.13 using the SOP method and simplify it.

Table 8.13

A	*B*	*C*	*Y*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

8. Write the Boolean expression for the truth table given in Table 8.14 using the SOP method.

Table 8.14

A	*B*	*C*	*Y*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

9. Write the Boolean expression for the truth table 8.15 and simplify it.

Table 8.15

A	*B*	*C*	*Y*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

10. Write the Boolean expression for the truth table given in Table 8.16 using the Karnaugh map.

Table 8.16

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

8.7 SOLUTIONS AND ANSWERS

Self-Assessment Questions

1. a) Truth table for $Y = AB + C$ is given by Table 8.17:

Table 8.17

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

b) $Y = (A + B)\, C$

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

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

$= A(B + C) = AB + AC$ (using 23)

b) Truth tables for 2-input NAND and NOR gates are given below:

Table 8.18: Truth table for 2-input NAND gate

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Minterm: $Y = \bar{A}\bar{B} + \bar{A}B + A\bar{B}$

Maxterm: $Y = \bar{A} + \bar{B}$

Table 8.19: Truth table for 2-input NOR gate

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Minterm: $Y = \bar{A}\bar{B}$

Maxterm: $Y = (A + \bar{B})(\bar{A} + B)\,(\bar{A} + \bar{B})$

$= (A\bar{A} + AB + \bar{A}\bar{B} + B\bar{B})(\bar{A} + \bar{B})$

$= (AB + \bar{A}\bar{B})(\bar{A} + \bar{B})$ using 10

$= A\bar{A}B + AB\bar{B} + \bar{A}\bar{A}\bar{B} + \bar{A}\bar{B}\bar{B}$

$= \bar{A}\bar{B}$ using 9, 10 and 11

$= \overline{(A + B)}$ using 25

3. $A\bar{B}\bar{C} + AB\bar{C} + ABC = A\bar{B}\bar{C} + AB(\bar{C} + C)$

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

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

4. For the Boolean expression: $Y = AB + A(B + C) + B(B + C)$

in the second row of Table 8.5, $A = 0$, $B = 0$ and $C = 1$. Hence,

$Y = 0.0 + 0.(0 + 1) + 0.(0 + 1)$

$= 0 + 0.1 + 0.1$

$= 0 + 0 + 0 = 0$

In the third row of Table 8.5, $A = 0, B = 1$ and $C = 0$. Hence,

$$Y = 0.1 + 0.(1+0) + 1.(1+0)$$

$$= 0 + 0.1 + 1.1 = 0 + 0 + 1 = 1$$

In the eighth row of Table 8.5, $A = 1, B = 1$ and $C = 1$. Hence,

$$Y = 1.1 + 1.(1+1) + 1.(1+1)$$

$$= 1 + 1.1 + 1.1 = 1 + 1 + 1 = 1$$

5. We use the reasoning method that $Y = 1$ when either or all of AB, BC and CA are 1. Thus, we get the truth table 8.20:

Table 8.20

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

6. See Fig. 8.10.

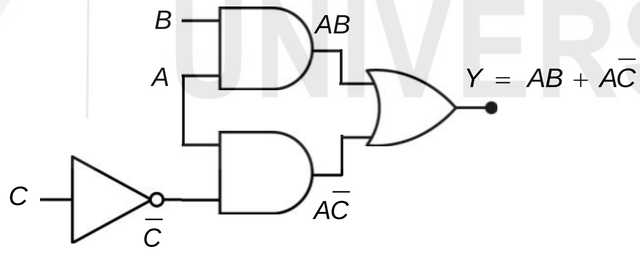

Fig. 8.10

7. $Y = \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} + ABC$

8. See Fig. 8.11.

(a)

	$\overline{B}$	B
$\overline{A}$		
A		

(b)

	$\overline{B}$	B
$\overline{A}$		
A	1	

(c)

	$\overline{B}$	B
$\overline{A}$		
A	1	1

(d)

	$\overline{B}$	B
$\overline{A}$	0	0
A	1	1

Fig. 8.11: Karnaugh map for Table 8.11.

Terminal Questions

1. $Y = A\overline{B}D + A\overline{B}\,\overline{D}$

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

   $= A\overline{B}.1 = A\overline{B}$

2. $Y = BCD + A\overline{B}CD$

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

   $= (B + A)CD = BCD + ACD$

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

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

4. We use de Morgan's first and second theorems to simplify this expression.

   $Y = \overline{(A + BC).\,(D + FG)}$

   $= \overline{(A + BC)} + \overline{(D + FG)}$

   $= \overline{A}.\overline{BC} + \overline{D}.\overline{FG}$

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

5. See Table 8.21 for the truth table of *BCD* + *ACD*.

Table 8.21

A	*B*	*C*	*D*	*Y*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

6. See Table 8.22.

Table 8.22

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

7. $Y = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}C + ABC$

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

$= \overline{A}\overline{C} + AC$

8. $Y = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + ABC$

9. $Y = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}C + AB\overline{C}$

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

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

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

10. See Fig. 8.12 for the Karnaugh map.

(a)

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

(b)

	$\overline{C}$	C
$\overline{A}\overline{B}$	1	1
$\overline{A}B$	1	1
AB		
$A\overline{B}$		

(c)

	$\overline{C}$	C
$\overline{A}\overline{B}$	1	1
$\overline{A}B$	1	1
AB	0	0
$A\overline{B}$	0	0

Fig. 8.12: Constructing Karnaugh map for TQ 10.

The Boolean expression is: $Y = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC$