CHAPTER 4

COMBINATIONAL LOGIC CIRCUITS

4.0 INTRODUCTION

In Chapter Three individual gates were investigated. This chapter will use those gates in combination to produce more complex logic functions. Techniques for simplifying these complex functions will also be covered.

Simplification of logic circuits is a responsibility of the designer. Simpler circuits are generally more economic and more reliable. The economy is achieved by using fewer integrated circuits while reliability is achieved by having fewer solder connections in the finished product.

4.1 OBJECTIVES

Upon completion of this chapter you should be able to:

- Simplify logic expressions.
- Simplify logic circuits.
- Use the Karnaugh map to simplify logic circuits and expressions.

4.2 SUM-OF-PRODUCT FORM

The sum-of-product form of a logic circuit output looks like the following examples:

$$x = A\overline{B}D + A\overline{BD}$$

$$z = \overline{A}B + C\overline{D} + EF + \overline{H}J$$

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

These examples show that the output of a logic circuit represented by x, z, or f are a logic one, or are true, when any of the logic products separated by the OR (+) designation are satisfied. The logic expressions completely define a logic circuit's operation in terms of the state of the logic inputs.

Logic equations may be formed directly from a truth table. These equations may also be simplified using Boolean algebra or more mechanical methods. Both types of simplification will be covered. The logic equations shown in the above examples are called "minterm" expressions.

Minterm expressions are logical equations where the logical product terms are separated by the logical sum operator. Minterm expressions are formed directly from truth tables. Minterm expressions are also called sum-of-product expressions.

4.3 DESIGNING COMBINATION CIRCUITS

Logic design begins with a problem statement. The problem statement is analyzed and translated into logic variable inputs. A truth table is then constructed to show when a logic one output is to be produced. Next a sum-of-product (minterm) logic equation is then produced. Then a circuit is drawn from the sum-of-product logic equation. These steps are illustrated by the following example.

Problem statement: An alarm is to be used in an automated ink bottling plant. A conveyer belt carries the empty ink bottles past the filling spout. The alarm is to sound if any of the following conditions occur:

A. The ink tank runs empty.

B. There are no bottles on the conveyor belt even if ink is in the tank.

C. There is ink in the tank, bottles on the conveyor belt, and electric power is lost.

The first step is to assign variables to the inputs.

I = ink in the tank

B = bottles on the conveyor belt

P = electric power is on

Next a truth table is constructed using these variables for inputs and indicating when the alarm is to ring by placing a one in the output, X, column. A minterm expression is then written. (See Table 4-1)

TABLE 4-1. Truth Table and Minterm Expression.

I	B	P	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$$X = \bar{I}\bar{B}\bar{P} + \bar{I}\bar{B}P + \bar{I}B\bar{P} + \bar{I}BP + I\bar{B}\bar{P} + I\bar{B}P + IB\bar{P}$$

Table 4-1 has a one in the output, X, for all cases where ink is not present ($\bar{I}$). In fact, the truth table shows that the alarm will not sound, X=0, when ink is present and bottles are present and power is on. Any other condition will sound the alarm.

Analyzing the minterm or sum-of-products expression shows that the alarm system may be directly implemented by using a seven input OR gate with each input being fed by a three input AND gate. This circuit implementation is shown in Figure 4-1.

FIGURE 4-1. Circuit Implementation of Minterm Expression.

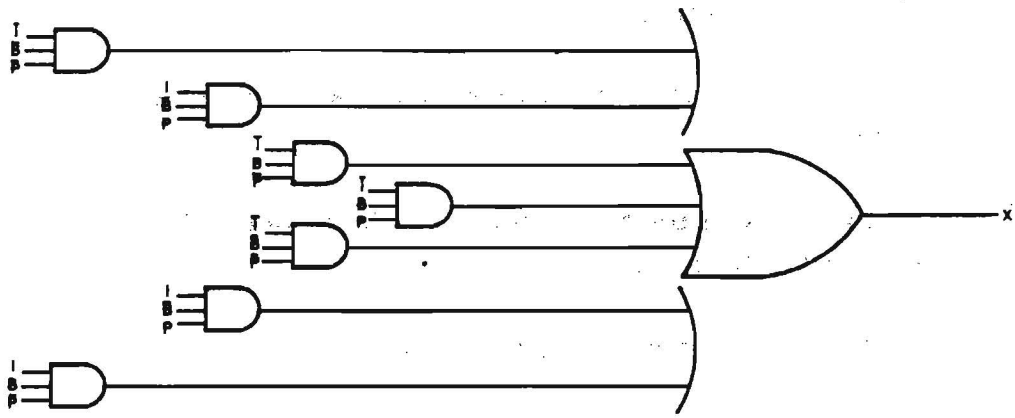


Figure 4-1 could be further complicated by including inverter circuits to form the "NOT" inputs. This circuit will fulfill the design objective of the problem, but may not be the simplest circuit.

4.4 BOOLEAN SIMPLIFICATION

One method of circuit or minterm simplification is to use Boolean algebra to remove logic redundancy. This method is based on the Boolean single and multivariable theorems. The Boolean theorems are summarized in Table 4-2.

TABLE 4-2. Boolean Theorems.

Single Variable Theorems:

(1) $X \cdot 0 = 0$	(5) $X + 0 = 0$
(2) $X \cdot 1 = X$	(6) $X + 1 = 1$
(3) $X \cdot X = X$	(7) $X + \overline{X} = X$
(4) $X \cdot \overline{X} = 0$	(8) $X + \overline{X} = 1$

Multivariable Theorems:

(9)	$X + Y = Y + X$
(10)	$X \cdot Y = Y \cdot X$
(11)	$X + (Y + Z) = (X + Y) + Z = X + Y + Z$
(12)	$X(Y \cdot Z) = (X \cdot Y)Z = X \cdot Y \cdot Z = XYZ$
(13a)	$X(Y + Z) = XY + XZ$
(13b)	$(W + X)(Y + Z) = WY + XY + WZ + XZ$
(14)	$X + XY = X$
(15)	$X + \overline{X}Y = X + Y$

These theorems may be applied to the solution of the example design problem of Table 4-1.

$$X = \bar{I}\bar{B}\bar{P} + \bar{I}B\bar{P} + \bar{I}BP + I\bar{B}\bar{P} + I\bar{B}P + IB\bar{P}$$

$$X = \bar{I}\bar{B}(\bar{P} + P) + \bar{I}B(\bar{P} + P) + I\bar{B}(\bar{P} + P) + IB\bar{P}$$

$$X = \bar{I}\bar{B} + \bar{I}B + I\bar{B} + IB\bar{P}$$

The first term of the original expression can be used again with the last term of the expression:

$$X = \bar{I}\bar{B} + \bar{I}B + I\bar{B} + \bar{P}(\bar{I}\bar{B} + IB)$$

$$X = \bar{I}\bar{B} + \bar{I}B + I\bar{B} + \bar{P}$$

Combining the first term with both the second and third terms give:

$$X = \bar{I}(\bar{B} + B) + \bar{B}(\bar{I} + I) + \bar{P}$$

$$X = \bar{I} + \bar{B} + \bar{P}$$

This final expression is logically equivalent to the original minterm expression. Figure 4-2 shows the final simplified circuit to implement the alarm system of the original problem. This solution is simpler, less expensive, and more reliable.

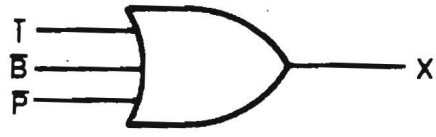


FIGURE 4-2. A Simplified Alarm Logic Circuit.

Boolean algebra can be used for logic circuit simplification, but most students find the Karnaugh map technique to be easier. The Karnaugh map technique will be discussed shortly.

4.5 DEMORGAN'S THEOREM

DeMorgan's theorem is important enough to command its own major heading in any digital text. DeMorgan's Theorem will allow the expression of logic equations in maxterm or product-of-sum form. (See Figure 4-3)

(A) $\overline{(X + Y)} = \bar{X} \cdot \bar{Y}$

(B) $\overline{X \cdot Y} = \bar{X} + \bar{Y}$

FIGURE 4-3. DeMorgan's Theorem.

Since there are only two logic operators besides the NOT function, DeMorgan's Theorem simply states that if an operator is NOTed it becomes the other. The OR operator NOTed becomes the AND operator and if the AND operator is NOTed it becomes the OR logic operator. The importance of this Theorem will become increasingly apparent in following discussions.

4.6 THE KARNAUGH MAP

The Karnaugh map or K-map technique is a graphical device to simplify logic equations or the output of truth tables following a simple orderly process. The K-map technique can be used for any number of variables, but becomes a little hard to handle when more than four input variables are considered. For this reason, the discussion of this technique will be limited to cases having no more than four input variables.

A K-map like a truth table displays the relationship between input variables and the desired or true output of a logic expression or truth table. The K-map presents this information as entries in boxes of a K-map rectangle. Figure 4-4 gives three examples. The examples become more complex as more input variables are involved. Note that each box in a K-map identifies a specific and unique combination of the input variables.

FIGURE 4-4. Logic Expressions, Truth Tables and K-Maps for Two, Three and Four Input Variables.

Logic Expression

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

Truth Table

A	B	X
0	0	0
0	1	1
1	0	0
1	1	1

K-Map

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

Logic Expression

$X = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

Truth Table

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

K-Map

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

Logic Expression

$X = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + AB\bar{C}D + ABCD$

FIGURE 4-4. Continued.

Truth Table

A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

K-Map

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

In viewing Figure 4-4, the following points should become apparent:

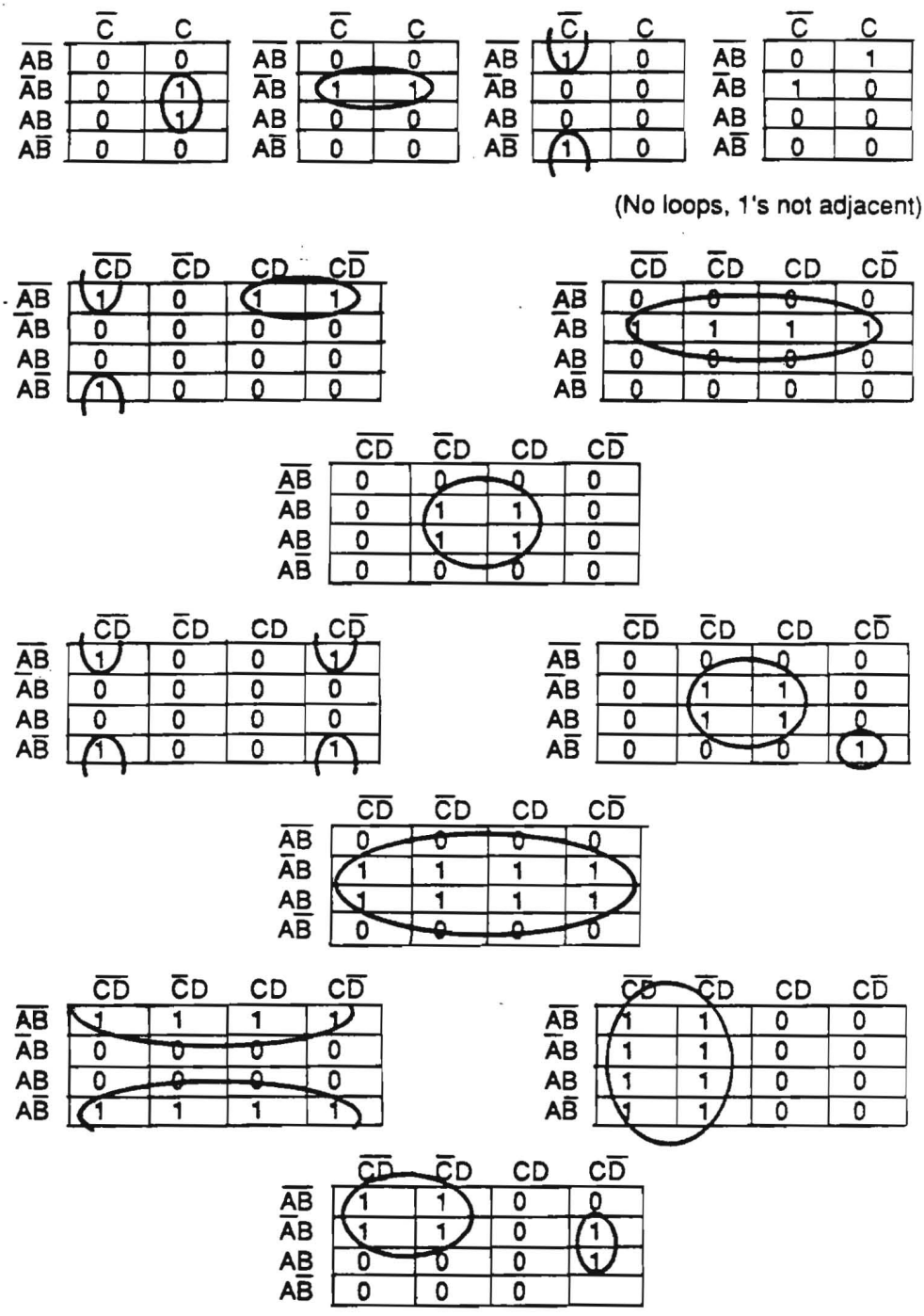
1. The logic equations, truth tables, and K-maps contain the same information.

2. The addition of an input variable doubles the number of entries in the truth tables and K-maps.

3. The K-maps are organized in a precise way. The entries across the top and down the side of the K-map are arranged so that only one variable changes. These patterns should be carefully and faithfully observed.

Once a K-map has been constructed for a problem. The entries may be looped. The loops are formed around adjacent 1's. The 1's may be looped in groups of one, two, four, or eight. Examples of looping are shown in Figure 4-5. Each loop of a K-map represents a single term in the simplified logic equation--larger and fewer loops result in the most simplification.

FIGURE 4-5. Examples of Looping.



Notice in the examples, that the edge boxes and corner boxes are adjacent. Also note that 1's in diagonal boxes are not adjacent. Any 1's not included in a loop, lead to a term in the final simplified logic expression.

To effectively use K-maps follow this procedure:

1. Construct the K-map from the original equation or truth table.

2. Carefully examine the K-map for adjacent 1's and loop the largest number of adjacent 1s (two, four, or eight).

3. Loop any pairs necessary to include any adjacent 1's that have not yet been included in a loop.

4. Loop any remaining single or isolated 1's.

5. Any variable appearing in a loop in both its true and complemented form is eliminated.

6. Form the simplified sum of products equation from all the terms generated by the loops.

Figure 4-6 shows some examples of the power of using the K-map technique. Both the original and simplified logic equation is given for each example.

Original Equation: $X = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + ABC\bar{D} + AB\bar{C}\bar{D}$

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

Two loops may be drawn: Only two terms will result.

Simplified Equation: $X = \bar{A}\bar{C} + ACD$

FIGURE 4-6. K-Map Simplification Examples.

Original Equation: $X = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + AB\bar{C}\bar{D} + AB\bar{C}D + A\bar{B}C\bar{D}$

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

Three loops may be drawn: Only three terms will result.

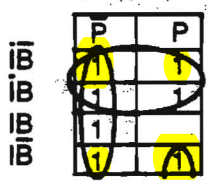
Simplified Equation: $X = B\bar{C} + \bar{A}B + A\bar{B}\bar{C}\bar{D}$

In the examples, each loop that is drawn results in a single term of the simplified equation. Ones may be used in more than a single loop as shown in the second example. Isolated ones become the most complex terms, as shown in the second example.

This technique can be applied to the ink factory alarm logic of the original example shown in Table 4-1. Figure 4-7 shows the original equation, the K-map with loops drawn, and the final simplified equation. Note that the simplified equation is the same as the one obtained by applying Boolean algebra.

FIGURE 4-7. K-Map Applied to Ink Factory Alarm Problem.

$$X = \bar{I}\bar{B}\bar{P} + \bar{I}\bar{B}P + \bar{I}B\bar{P} + \bar{I}BP + I\bar{B}\bar{P} + I\bar{B}P + IB\bar{P}$$



Simplified Equation: $X = \bar{I} + \bar{B} + \bar{P}$

One last point when dealing with K-maps. Sometimes a "don't care" output exists in a truth table. A "don't care" condition means that the combination of input variables controlling that output will never occur, therefore the output could be listed as either a 1 or 0 which ever would help the simplification process. The "don't care" condition is listed as X in the output of the truth table. When the K-map is drawn, the X can be changed to a 1 or 0 to expedite simplification. (See Figure 4-8)

FIGURE 4-8. Don't Care Conditions in K-Map Simplification.

Original Truth Table:

A	B	C	D	X
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	1
0	1	1	0	0
0	1	1	1	X
1	0	0	0	0
1	0	0	1	0
1	0	1	0	X
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

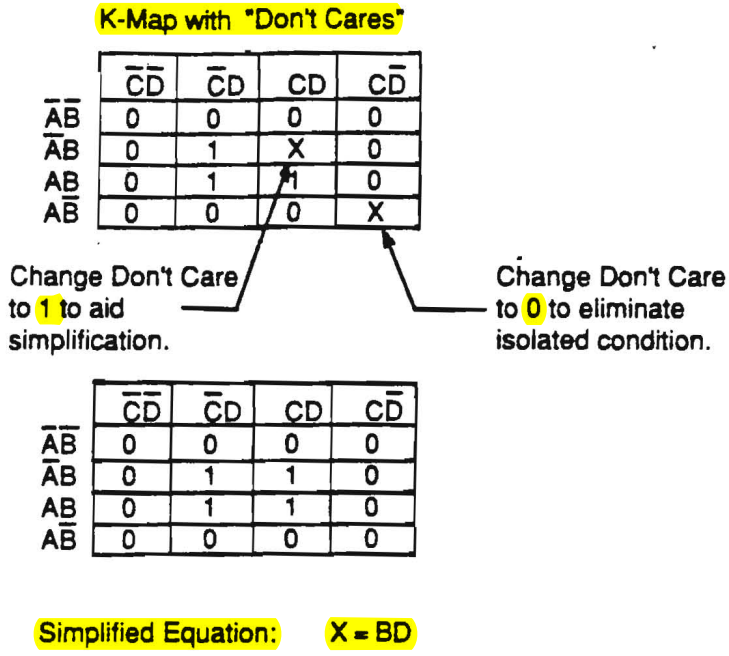


FIGURE 4-8. Continued.

Figure 4-8 shows how careful consideration in recognizing "don't care" conditions and later changing "don't cares" to ones or zeros can greatly simplify a logic design. The trick is to recognize early in the design any "don't care" conditions and identify them by using X instead of 1 or 0 in the truth table.

4.7 PRODUCT-OF-SUMS FORM

The opening section of this chapter discussed the sum-of-product form of equations. The implication was that there are other ways to express a logic equation. This other way is called the product-of-sums form. Figure 4-9 gives examples of this form for logic equations.

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

$$X = (\bar{I} + P + E) \cdot (\bar{I} + \bar{P} + E)$$

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

FIGURE 4-9. Examples of Product-of-Sums Logic Equation.

To create the product-of-sums form involves the use of DeMorgans Theorem. An example will be helpful in illustrating the concepts involved.

FIGURE 4-10. Steps in Creating the Product-of-Sums Logic Equation.

Typical Truth Table

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Sum-of-Products Form $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + AB\overline{C} + ABC$

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

Simplified Sum-of-Products Form $X = B\overline{C} + A\overline{C} + AB + \overline{A}\overline{B}C$

To create product-of sums form add an $\overline{X}$ column to the original truth table.

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

Sum-of-Products Form for $\overline{X}$ $\overline{X} = \overline{A}\overline{B}\overline{C} + \overline{A}BC + A\overline{B}C$

Note: This is the "NOT" of the above K-map.

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

Simplified Sum-of-Products for $\overline{X}$ $\overline{X} = \overline{A}\overline{B}\overline{C} + \overline{A}BC + A\overline{B}C$

(Same as original - no loops could be drawn)

Apply DeMorgan's Theorem:

$$\overline{X} = \overline{A}\overline{B}\overline{C} + \overline{A}BC + A\overline{B}C$$

Gives:

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

FIGURE 4-10. Continued.

In Figure 4-10, the final product-of-sums equation is logically equivalent to the original sum-of-products equation. Figure 4-11 shows how these two equations would be implemented.

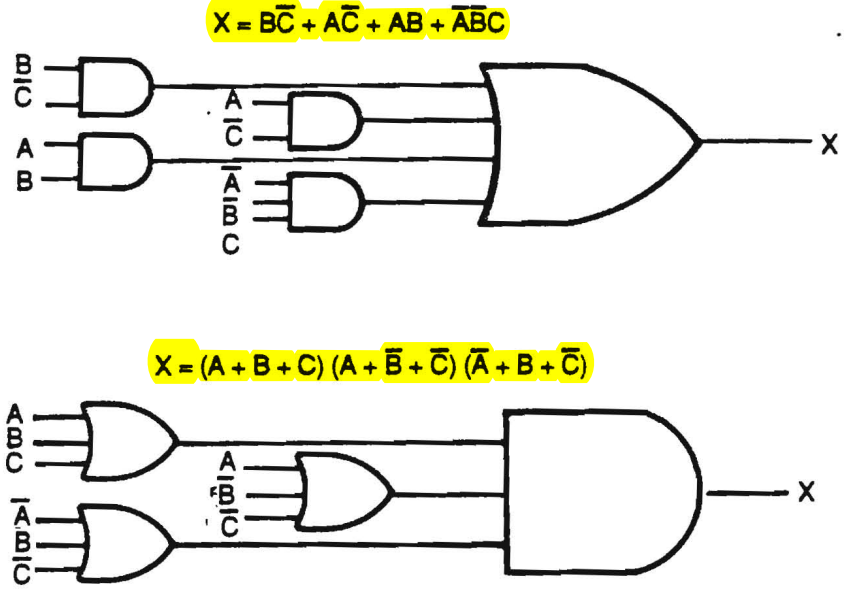


FIGURE 4-11. Implementation of the Equations in Figure 4-10.

The two logic circuits are equivalent.

The reason for interest in having two forms for logical equations is ease of implementation when using universal logic gates; NAND and NOR. The sum-of-product form is most easily implemented using all NAND gates, while the product-of-sums form is most easily implemented using all NOR gates. The examples given in Figures 4-10 and 4-11 are shown implemented using all NAND or all NOR gates.

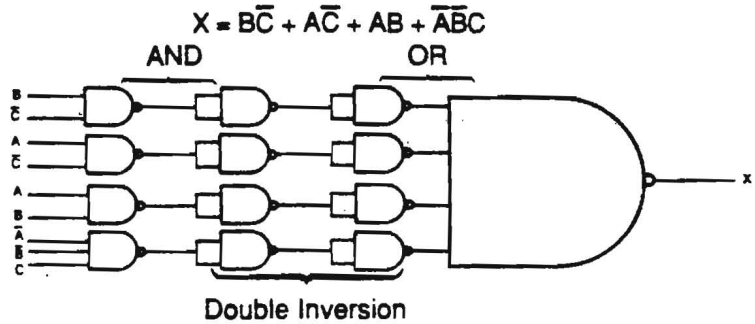
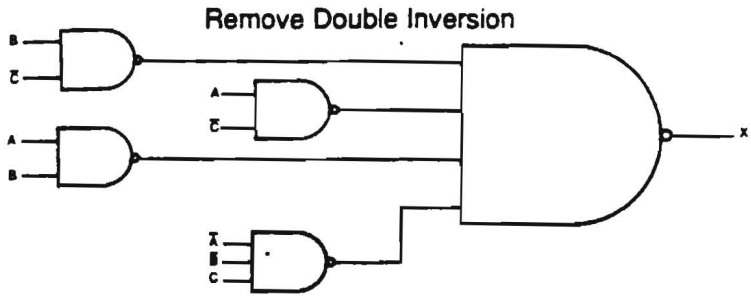


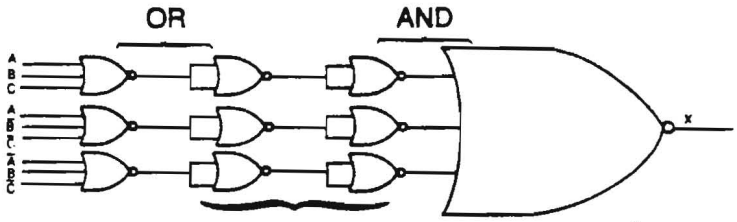
FIGURE 4-12. Logic Implementation Using Universal Gates.

FIGURE 4-12. Continued.



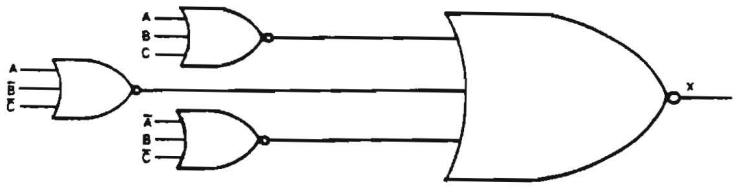
Final circuit using only NAND gates.

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



Double Inversion

Remove Double Inversion

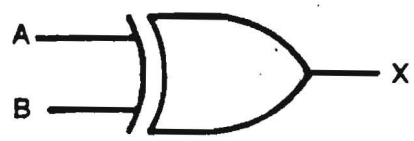


Final Circuit using only NOR gates.

The circuit using only NAND gates is logically equivalent to the circuit using only NOR gates. This can be verified by setting all possible input states to the two circuits and observe coincidence in the outputs.

4.8 THE EXCLUSIVE OR AND EXCLUSIVE NOR CIRCUITS

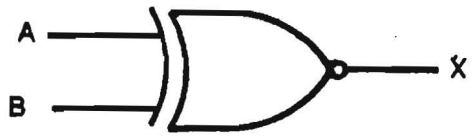
The final topic of this chapter deals with two gate structures that are not basic gate structures, but whose functions occur so frequently that they have earned their own symbols. These gate structures are often used in comparator circuits. Figure 4-13 indicates the symbols and truth tables for these logic gates.



XOR

Exclusive OR Gate

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0



XNOR

Exclusive NOR Gate.

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

FIGURE 4-13. The Exclusive OR Gate and Exclusive NOR Gate.

The output of the Exclusive OR gate is true only when the two inputs are different. The output of the Exclusive NOR gate is true only when the two inputs are equal. Each of these gates may be produced using AND, OR, and NOT gates.

4.9 SUMMARY

This chapter has introduced Boolean algebra and Karnaugh map techniques for simplifying logic circuits. Both single variable and multivariable theorems were covered as well as DeMorgan's theorem.

Product-of-sums and sum-of-products as two forms of logic gates were introduced. Each of these forms are more easily implemented by either NAND or NOR universal gates.

Two important gate functions; the Exclusive OR and the Exclusive NOR were introduced.

4.10 REVIEW QUESTIONS

1. Determine the simplified logic equation for each of the K-maps in Figure 4-14.

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

(a)

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

(b)

FIGURE 4-14. K-Maps.

FIGURE 4-14.
Continued.

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

(c)

2. Write the original equations used to form the K-maps of Figure 4-14.

(a) _____

(b) _____

(c) _____

3. Simplify the equations in question 2 using Boolean algebra-- show all steps.

Sketch the outputs for the inputs shown in Figure 4-15.

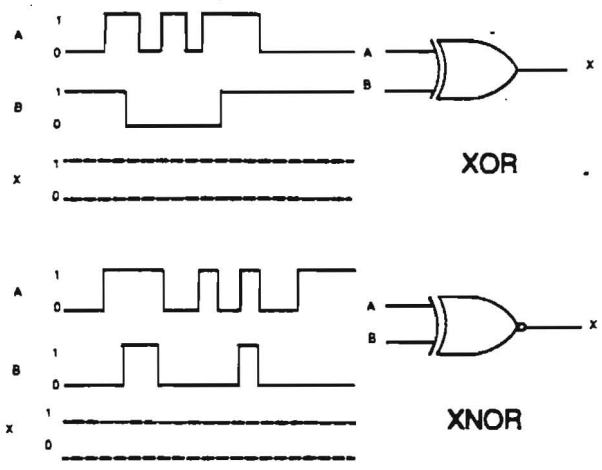


FIGURE 4-15. Outputs for XOR and XNOR Gates.

5. In the space below, show how Exclusive OR and Exclusive NOR circuits are constructed from AND, OR, and NOT gates.