KARNAUGH MAPS

Boolean expression can be simplified using Karnaugh maps (K-maps) by the following procedure:

- Plot function on map using either
  a) Equation in sum-of-products (SOP) form
  or
  b) Truth table

- Form the fewest number of largest possible rectangles of adjacent (vertically or horizontally) 1's consisting of $1, 2, 4, 8, \ldots 2^n$ squares. These rectangles may overlap if necessary.

  The product term for the rectangle consists of those variables with a common value. The variable that changes values is not needed in the term.

- Write the resulting expression by ORing together the product terms for each rectangle to form a SOP expression.

Note: Remember that K-Maps are spherical (i.e. they wrap around themselves). Therefore the top row is adjacent to the bottom row and the columns furthest to the right and left are adjacent.

TABLE 1-6
American Standard Code for Information Interchange (ASCII)

	$B_7B_6B_5$							
$B_4B_3B_2B_1$	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P		p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

TABLE 2-3
Basic Identities of Boolean Algebra

1. $X + 0 = X$	2. $X \cdot 1 = X$	
3. $X + 1 = 1$	4. $X \cdot 0 = 0$	
5. $X + X = X$	6. $X \cdot X = X$	
7. $X + \overline{X} = 1$	8. $X \cdot \overline{X} = 0$	
9. $\overline{\overline{X}} = X$		
10. $X + Y = Y + X$	11. $XY = YX$	Commutative
12. $X + (Y + Z) = (X + Y) + Z$	13. $X(YZ) = (XY)Z$	Associative
14. $X(Y + Z) = XY + XZ$	15. $X + YZ = (X + Y)(X + Z)$	Distributive
16. $\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17. $\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan

2

DIGITAL CIRCUITS

2-1 BINARY LOGIC AND GATES

Digital circuits are hardware components that manipulate binary information. The circuits are constructed with electronics parts such as transistors, diodes, and resistors. Each circuit is referred to as a *gate*. The designer of a digital system does not have to be concerned with the internal construction of the individual gates but only with their external logical properties. Each gate performs a specific logical operation, and the output from one gate is applied to the inputs of other gates, in sequence, to form the required digital circuit.

In order to describe the operational properties of digital circuits, it is necessary to introduce a mathematical notation that specifies the operation of each gate. This mathematical system is a binary logic system known as *Boolean algebra*. The name of the algebra is in honor of the English mathematician George Boole, who in 1854 published a book introducing the mathematical theory of logic. Today Boolean algebra is used to describe the interconnection of digital gates and to transform circuit diagrams to algebraic expressions. We will first introduce the concept of binary logic and show its relationship to digital gates and binary signals. We will then present the properties of Boolean algebra together with other design methods for dealing with various aspects of digital circuits and systems.

Binary Logic

Binary logic deals with variables that take on two discrete values and with operations that assume logical meaning. The two values the variables take may be called by different names, but for our purpose it is convenient to think in terms of binary

values and assign 1 and 0 to each variable. The variables are designated by letters of the alphabet such as A, B, C, X, Y, Z. There are three logical operations associated with the binary variables called AND, OR, and NOT.

1. AND. This operation is represented by a dot or by the absence of an operator. For example, $X \cdot Y = Z$ or $XY = Z$ is read "X AND Y is equal to Z." The logical operation AND is interpreted to mean that $Z = 1$ if and only if $X = 1$ and $Y = 1$; otherwise $Z = 0$. (Remember that X, Y, and Z are binary variables and can be equal to 1 or 0 and nothing else.)
2. OR. This operation is represented by a plus symbol. For example $X + Y = Z$ is read "X OR Y is equal to Z", meaning that $Z = 1$ if $X = 1$ or $Y = 1$ or if both $X = 1$ and $Y = 1$. Only if $X = 0$ and $Y = 0$, is $Z = 0$.
3. NOT. This operation is represented by a bar over the variable. For example, $\overline{X} = Z$ is read "X NOT is equal to Z", meaning that Z is what X is not. In other words, if $X = 1$, then $Z = 0$; but if $X = 0$, then $Z = 1$. The NOT operation is also referred to as the *complement* operation, since it changes a 1 to 0 and a 0 to 1.

Binary logic resembles binary arithmetic, and the operations AND and OR have similarities to multiplication and addition, respectively. In fact, the symbols used for AND and OR are the same as those used for multiplication and addition. However, binary logic should not be confused with binary arithmetic. One should realize that an arithmetic variable designates a number that may consist of many digits. A logic variable is always either a 1 or a 0. The possible binary values for the logical OR operation are as follows:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 1$$

These resemble binary addition except for the last operation. In binary logic we have $1 + 1 = 1$ (read "one OR one is equal to one"), but in binary arithmetic we have $1 + 1 = 10$ (read "one plus one is equal to two"). To avoid ambiguity, the symbol $\vee$ is sometimes used for the OR operation instead of the + symbol. But as long as arithmetic and logic operations are not mixed, each can use the + symbol with its own independent meaning.

The binary values for the AND operation are

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

This is identical to binary multiplication provided we use only a single bit. Logical AND is sometimes referred to as *logical multiplication* and logical OR as *logical addition*.

TABLE 2-1
Truth Tables for the Three Logical Operations

AND			OR			NOT	
X	Y	$X \cdot Y$	X	Y	$X + Y$	X	$\bar{X}$
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

For each combination of the values of binary variables such as X and Y, there is a value of Z specified by the definition of the logical operation. These definitions may be listed in a compact form in a *truth table*. A truth table is a table of combinations of the binary variables showing the relationship between the values that the variables take and the result of the operation. The truth tables for the operations AND, OR, and NOT are shown in Table 2-1. The tables list all possible values for the variables and the results of the operation. These tables clearly demonstrate the definition of the three operations.

Logic Gates

Logic gates are electronic circuits which operate on one or more input signals to produce an output signal. Electrical signals such as voltages or currents exist throughout a digital system in either of two recognizable values. Voltage operated circuits respond to two separate voltage levels which represent a binary variable equal to logic-1 or logic-0. For example, a particular digital system may define logic-0 as a signal equal to 0 volts, and logic-1 as a signal equal to 4 volts. In practice, each voltage level has an acceptable range as shown in Figure 2-1. The input terminals of digital circuits accept binary signals within the allowable range and respond at the output terminals with binary signals that fall within the specified range. The intermediate region between the allowed regions is crossed only during state transition. Any desired information for computing or control can be operated upon by passing binary signals through various combinations of logic gates with each signal representing a particular binary variable.

The graphic symbols used to designate the three types of gates are shown in

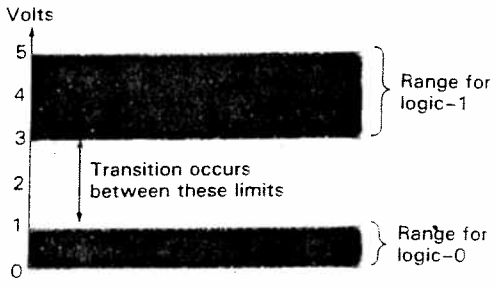


FIGURE 2-1
Example of Binary Signals

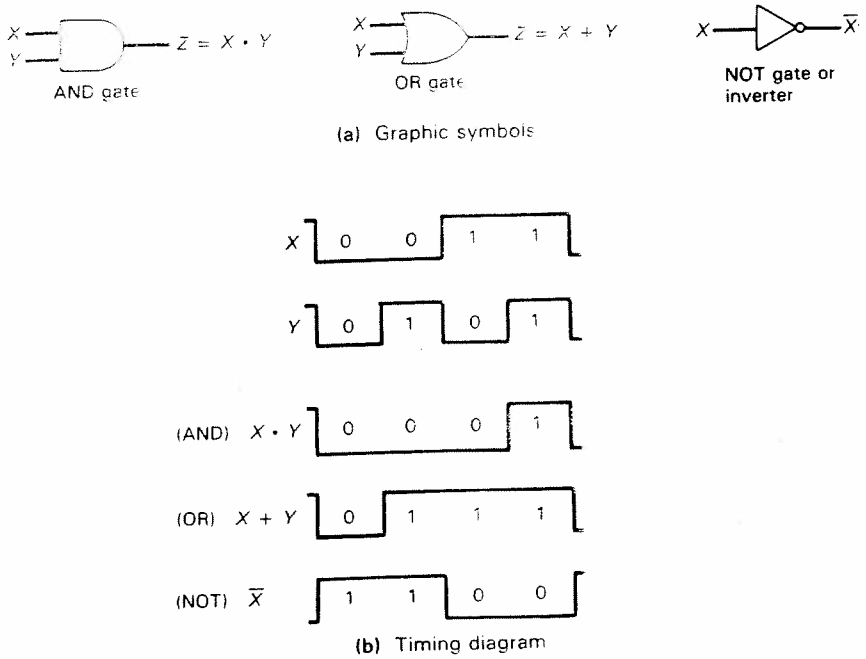


FIGURE 2-2
Digital Logic Gates

Figure 2-2(a). The gates are blocks of hardware that produce the equivalent of logic-1 and logic-0 output signals if input logic requirements are satisfied. The input signals X and Y may exist in the AND and OR gates in one of four possible states: 00, 01, 10, or 11. These input signals are shown in Figure 2-2(b) together with the corresponding output signal for each gate. The timing diagrams illustrate the response of each gate to the four possible input signal combinations. The horizontal axis of the timing diagram represents time and the vertical axis shows the signal as it changes between the two possible voltage levels. The low level represents logic-0 and the high level represents logic-1. The AND gate responds with a logic-1 output signal when both input signals are logic-1. The OR gate responds with logic-1 output signal if any input signal is logic-1. The NOT gate is commonly referred to as an *inverter*. The reason for this name is apparent from the signal response in the timing diagram where it is shown that the output signal inverts the logic sense of the input signal.

AND and OR gates may have more than two inputs. An AND gate with three inputs and an OR gate with four inputs are shown in Figure 2-3. The three-input

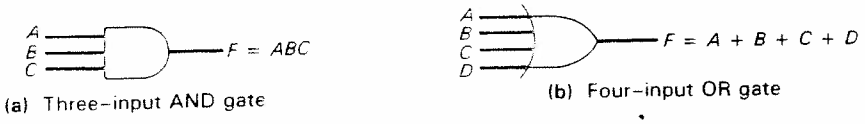


FIGURE 2-3
Gates with Multiple Inputs

AND gate responds with a logic-1 output if all three inputs are logic-1. The output produces a logic-0 if any input is logic-0. The four-input OR gate responds with a logic-1 if any input is logic-1; its output becomes a logic-0 only when all inputs are logic-0.

2-2 BOOLEAN ALGEBRA

Boolean algebra is an algebra that deals with binary variables and logic operations. The variables are designated by letters of the alphabet, and the three basic logic operations are AND, OR, and complement. A Boolean function consists of an algebraic expression formed with binary variables, the constants 0 and 1, the logic operation symbols, parentheses, and an equal sign. For a given value of the binary variables, the Boolean function can be equal to either 1 or 0. Consider as an example the following Boolean function:

$$F = X + \overline{Y}Z$$

The function F is equal to 1 if X is equal to 1 or if both $\overline{Y}$ and Z are equal to 1. Otherwise F is equal to 0. The complement operation dictates that when $\overline{Y} = 1$ then $Y = 0$. Therefore, we can say that $F = 1$ if $X = 1$ or if $Y = 0$ and $Z = 1$. A Boolean function expresses the logical relationship between binary variables. It is evaluated by determining the binary value of the expression for all possible values of the variables.

A Boolean function can be represented in a truth table. A truth table is a list of combinations of 1's and 0's assigned to the binary variables and a column that shows the value of the function for each binary combination. The number of rows in the truth table is 2^n, where n is the number of variables in the function. The binary combinations for the truth table are obtained from the binary numbers by counting from 0 through $2^n - 1$. Table 2-2 shows the truth table for the function listed above. There are eight possible binary combinations for assigning bits to the three variables X, Y, and Z. The column labeled F contains either 0 or 1 for each of these combinations. The table shows that the function is equal to 1 when $X = 1$ or when $YZ = 01$. Otherwise it is equal to 0.

TABLE 2-2
Truth Table for the Function $F = X + \overline{Y}X$

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

A Boolean function can be transformed from an algebraic expression into a circuit diagram composed of logic gates. The logic circuit diagram for F is shown in Figure 2-4. There is an inverter for input Y to generate the complement $\overline{Y}$. There is an AND gate for the term $\overline{Y}Z$ and an OR gate that combines the two terms. In logic circuit diagrams, the variables of the function are taken as the inputs of the circuit and the binary variable F is taken as the output of the circuit.

There is only one way that a Boolean function can be represented in a truth table. However, when the function is in algebraic form, it can be expressed in a

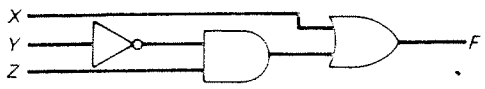


FIGURE 2-4
Logic Circuit Diagram for $F = X + \overline{Y}Z$

variety of ways. The particular expression used to designate the function will also dictate the interconnection of gates in the logic circuit diagram. By manipulating a Boolean expression according to Boolean algebra rules, it is sometimes possible to obtain a simpler expression for the same function and thus reduce the number of gates in the circuit. To see how this is done, it is necessary first to study the basic rules of the algebra.

Basic Identities of Boolean Algebra

Table 2-3 lists the most basic identities of Boolean algebra. The notation is simplified by omitting the symbol · for the AND whenever it does not lead to confusion. The first nine identities show the relationship between a single variable X, its complement $\overline{X}$, and the binary constants 0 and 1. The next five identities, 10 through 14, are similar to ordinary algebra. The last three, 15 through 17, do not apply in ordinary algebra but are very useful in manipulating Boolean expressions.

The basic rules listed in Table 2-3 have been arranged in two columns. The two parts demonstrate the property of duality of Boolean algebra. The *dual* of an algebraic expression is obtained by interchanging OR and AND operations and replacing 1's by 0's and 0's by 1's. An equation in one column of the table can be obtained from the corresponding equation in the other column by taking the dual of the expressions on both sides of the equal sign. For example, relation 2 is the dual of relation 1 because the OR has been replaced by an AND and the 0 by 1.

The nine identities involving a single variable can be easily verified by substituting both possible values for X. For example, to show that $X + 0 = X$, let $X = 0$ to obtain $0 + 0 = 0$, and then let $X = 1$ to obtain $1 + 0 = 1$. Both equations are true according to the definition of the OR logic operation. Any expression can be substituted for the variable X in all the Boolean equations listed in the table. Thus, by identity 3 and with $X = AB + C$ we obtain

$$AB + C + 1 = 1$$

Note that equation 9 states that double complementation restores the variable to its original value. Thus if $X = 0$ then $\overline{X} = 1$ and $\overline{\overline{X}} = 0 = X$.

The commutative laws state that the order in which the variables are written will not affect the result when using the OR and AND operations. The associative laws state that the result of forming an operation among three variables is inde-

TABLE 2-3
Basic Identities of Boolean Algebra

1. $X + 0 = X$	2. $X \cdot 1 = X$	
3. $X + 1 = 1$	4. $X \cdot 0 = 0$	
5. $X + X = X$	6. $X \cdot X = X$	
7. $X + \overline{X} = 1$	8. $X \cdot \overline{X} = 0$	
9. $\overline{\overline{X}} = X$		
10. $X + Y = Y + X$	11. $XY = YX$	Commutative
12. $X + (Y + Z) = (X + Y) + Z$	13. $X(YZ) = (XY)Z$	Associative
14. $X(Y + Z) = XY + XZ$	15. $X + YZ = (X + Y)(X + Z)$	Distributive
16. $\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17. $\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan

pendent of the order that is taken and therefore, the parentheses can be removed altogether.

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

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

These two laws and the first distributive law are well known from ordinary algebra so they should not impose any difficulty. The second distributive law given by identity 15 is the dual of the ordinary distributive law and is very useful in manipulating Boolean functions.

$$X + YZ = (X + Y)(X + Z)$$

This equation can be used for other combination of variables. Consider the expression $(A + B)(A + CD)$. Letting $X = A$, $Y = B$, and $Z = CD$, and applying the second distributive law we obtain

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

The last two identities in Table 2-3 are referred to as DeMorgan's theorem.

$$\overline{(X + Y)} = \overline{X} \cdot \overline{Y} \quad \text{and} \quad \overline{(X \cdot Y)} = \overline{X} + \overline{Y}$$

This is a very important theorem and is used to obtain the complement of an expression. DeMorgan's theorem can be verified by means of truth tables that assign all the possible binary values to X and Y. Table 2-4 shows two truth tables that verify the first part of DeMorgan's theorem. In A, we evaluate $\overline{(X + Y)}$ for all possible values of X and Y. This is done by first evaluating $X + Y$ and then taking its complement. In B, we evaluate $\overline{X}$ and $\overline{Y}$ and then AND them together. The result is the same for the four binary combinations of X and Y which verifies the identity of the equation.

Note the order in which the operations are performed when evaluating an expression. The complement over a single variable is evaluated, then the AND operation, and then the OR operation, just as we do in ordinary algebra with multiplication and addition. A complement over an expression such as $\overline{(X + Y)}$ is considered as specifying NOT $(X + Y)$ so the value within the parentheses is evaluated first and then the complement of the result is taken. It is customary to exclude the parentheses when complementing an expression since a bar is drawn over the entire expression. Thus $\overline{(X + Y)}$ is expressed as $\overline{X + Y}$ when designating the complement of $(X + Y)$.

TABLE 2-4
Truth Tables to Verify DeMorgan's Theorem

A. X	Y	$X + Y$	$\overline{(X + Y)}$	B. X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X} \cdot \overline{Y}$
0	0	0	1	0	0	1	1	1
0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	0	1	0
1	1	1	0	1	1	0	0	0

DeMorgan's theorem can be extended to three or more variables. The general DeMorgan's theorem can be expressed as follows:

$$\overline{X_1 + X_2 + X_3 + \ldots + X_n} = \overline{X}_1\overline{X}_2\overline{X}_3 \ldots \overline{X}_n$$

$$\overline{X_1X_2X_3 \ldots X_n} = \overline{X}_1 + \overline{X}_2 + \overline{X}_3 + \ldots + \overline{X}_n$$

The logic operation changes from OR to AND or from AND to OR. In addition, the complement is removed from the entire expression and placed instead over each variable. For example,

$$\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z} \quad \text{and} \quad \overline{A + B + C + D} = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}.$$

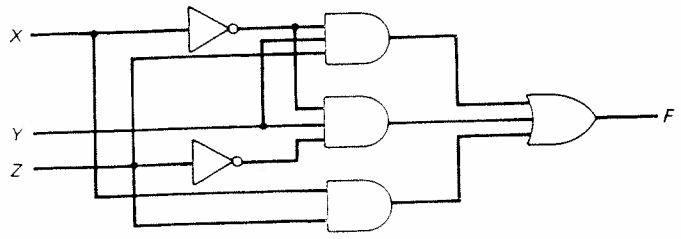
Algebraic Manipulation

Boolean algebra is a useful tool for simplifying digital circuits. Consider for example the following Boolean function:

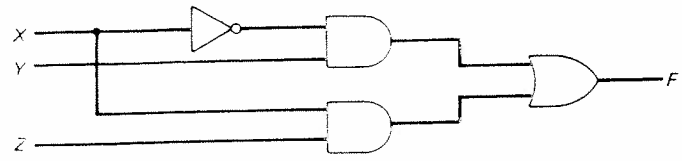
$$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$$

The implementation of this function with logic gates is shown in Figure 2-5(a). Input variables X and Z are complemented with inverters to obtain $\overline{X}$ and $\overline{Z}$. The three terms in the expression are implemented with three AND gates. The OR gate forms the logical sum of the three terms. Now consider the possible simplification of the function by applying some of the identities listed in Table 2-3.

$$\begin{aligned} F &= \overline{X}YZ + \overline{X}Y\overline{Z} + XZ \\ &= \overline{X}Y(Z + \overline{Z}) + XZ && \text{by identity 14} \\ &= \overline{X}Y \cdot 1 + XZ && \text{by identity 7} \\ &= \overline{X}Y + XZ && \text{by identity 2} \end{aligned}$$



(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$



(b) $F = \overline{X}Y + X\overline{Z}$

FIGURE 2-5
Implementation of Boolean Function with Gates

TABLE 2-5
Truth Table for Boolean Function

X	Y	Z	F
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	0
1	1	1	1

The function is reduced to only two terms and can be implemented with gates as shown in Figure 2-5(b). It is obvious that the circuit in (b) is simpler than the one in (a); yet both implement the same function. It is possible to use a truth table to verify that the two expressions are equivalent, as shown in Table 2-5. The function as expressed in Figure 2-5(a) is equal to 1 when $XYZ = 011$, or when $XYZ = 010$, or when $XZ = 11$. This produces four 1's for F in the table. The function as expressed in Figure 2-5(b) is equal to 1 when $XY = 01$, or when $XZ = 11$. This produces the four 1's in the table. Since both expressions produce the same truth table, they are said to be equivalent. Therefore, the two circuits have the same output for all possible input binary combinations of the three variables. Each implements the same function but the one with fewer gates is preferable because it requires fewer components.

When a Boolean expression is implemented with logic gates, each term requires a gate and each variable within the term designates an input to the gate. We define a *literal* as a single variable within a term that may or may not be complemented. The function of Figure 2-5(a) has three terms and eight literals; the one in Figure 2-5(b) has two terms and four literals. By reducing the number of terms, the number of literals, or both, in a Boolean expression, it is sometimes possible to obtain a simpler circuit. The manipulation of Boolean algebra consists primarily of reducing an expression for the purpose of obtaining a simpler circuit. Unfortunately, there are no specific rules that guarantee a good result. The only method available is a cut-and-try procedure employing the basic relations and other manipulations that become familiar with use. The following examples illustrate a few of the possibilities.

a. $X + XY = X(1 + Y) = X$
b. $XY + X\overline{Y} = X(Y + \overline{Y}) = X$
c. $X + \overline{X}Y = (X + \overline{X})(X + Y) = X + Y$

Note that the intermediate step $X = X \cdot 1$ has been omitted when X is factored out in the first equation. The relationship $1 + Y = 1$ is useful for eliminating redundant terms, as is done with the term XY in the first equation. The relation $Y + \overline{Y} = 1$ is useful for combining two terms, as is done in the second equation. The two terms to be combined must contain the same variable, but the variable must be complemented in one term and not complemented in the other. The third equation is simplified by means of the second distribution law (identity 15 in Table 2-3).

The following are three more examples of Boolean expression simplification.

d. $X(X + Y) = X + XY = X(1 + Y) = X$
e. $(X + Y)(X + \overline{Y}) = X + Y\overline{Y} = X$
f. $X(\overline{X} + Y) = X\overline{X} + XY = XY$

Note that the intermediate steps $XX = X = X \cdot 1$ has been omitted during the manipulation of equation d. The expression in e is simplified by means of the second distributive law. Here again we omit the intermediate step $Y\overline{Y} = 0$ and $X + 0 = X$.

The last three equations are the dual of the first three equations. Remember that the dual of an expression is obtained by changing AND to OR and OR to AND throughout (and 1's to 0's and 0's to 1's if they appear in the expression).

The duality principle of Boolean algebra states that a Boolean equation remains valid if we take the dual of the expressions on both sides of the equal sign. Therefore, equations d, e, and f can be verified by taking the dual of equations a, b, and c, respectively.

The consensus theorem, shown below, is sometimes useful when simplifying Boolean expressions.

$$XY + \overline{X}Z + YZ = XY + \overline{X}Z$$

It shows that the third term YZ is redundant and can be eliminated. Note that Y and Z are also associated with X and $\overline{X}$ in the first two terms. The proof of the equation is obtained by first ANDing YZ with $(X + \overline{X}) = 1$.

$$\begin{aligned} XY + \overline{X}Z + YZ &= XY + \overline{X}Z + YZ(X + \overline{X}) \\ &= XY + \overline{X}Z + XYZ + \overline{X}YZ \\ &= XY + XYZ + \overline{X}Z + \overline{X}YZ \\ &= XY(1 + Z) + \overline{X}Z(1 + Y) \\ &= XY + \overline{X}Z \end{aligned}$$

The dual of the consensus theorem is

$$(X + Y)(\overline{X} + Z)(Y + Z) = (X + Y)(\overline{X} + Z)$$

The following example shows how the consensus theorem can be applied when manipulating a Boolean expression.

$$\begin{aligned} (A + B)(\overline{A} + C) &= A\overline{A} + AC + \overline{A}B + BC \\ &= AC + \overline{A}B + BC \\ &= AC + \overline{A}B \end{aligned}$$

Note that $A\overline{A} = 0$ and $0 + AC = AC$. The redundant term here is BC.

Complement of a Function

The complement of a function, F, is obtained from an interchange of 1's to 0's and 0's to 1's in the values of F in the truth table. The complement of a function can be derived algebraically by applying DeMorgan's theorem. Remember that the generalized form of this theorem states that the complement of an expression is obtained by interchanging AND and OR operations and complementing each variable.

Example 2-1

Find the complement of the following two functions: $F_1 = \overline{X}Y\overline{Z} + \overline{X}\overline{Y}Z$ and $F_2 = X(\overline{Y}\overline{Z} + YZ)$.

Applying DeMorgan's theorem as many times as necessary the complements are obtained as follows:

$$\begin{aligned} \overline{F}_1 &= \overline{\overline{X}Y\overline{Z} + \overline{X}\overline{Y}Z} = (\overline{\overline{X}Y\overline{Z}}) \cdot (\overline{\overline{X}\overline{Y}Z}) \\ &= (X + \overline{Y} + Z)(X + Y + \overline{Z}) \end{aligned}$$

$$\overline{F}_2 = \overline{X(\overline{Y}\overline{Z} + YZ)} = \overline{X} + \overline{(\overline{Y}\overline{Z} + YZ)}$$
$$= \overline{X} + (\overline{\overline{Y}\overline{Z}} \cdot \overline{YZ})$$
$$= \overline{X} + (Y + Z)(\overline{Y} + \overline{Z}) \quad \blacksquare$$

A simpler method for deriving the complement of a function is to take the dual of the function and complement each literal. This method follows from the generalized DeMorgan's theorem. Remember that the dual of an expression is obtained by interchanging AND and OR operations and 1's and 0's.

Example 2-2 Find the complement of the functions in Example 2-1 by taking their dual and complementing each literal

$F_1 = \overline{X}Y\overline{Z} + \overline{X}\overline{Y}Z$

The dual of F_1 $(\overline{X} + Y + \overline{Z})(\overline{X} + \overline{Y} + Z)$

Complement each literal $(X + \overline{Y} + Z)(X + Y + \overline{Z}) = \overline{F}_1$

$F_2 = X(\overline{Y}\overline{Z} + YZ)$

The dual of F_2 $X + (\overline{Y} + \overline{Z})(Y + Z)$

Complement each literal $\overline{X} + (Y + Z)(\overline{Y} + \overline{Z}) = \overline{F}_2$

2-3 STANDARD FORMS

A Boolean function can be written in a variety of ways when expressed algebraically. There are, however, a few algebraic expressions that are considered to be in standard form. The standard forms facilitate the simplification procedures of Boolean expressions and frequently result in a more desirable gating circuits.

The standard forms contain terms referred to as *product* terms and *sum* terms. An example of a product term is $X\overline{Y}Z$. This is a logical product consisting of an AND operation among several variables. An example of a sum term is $\overline{X} + Y + Z$. This is a logical sum consisting of an OR operation among the variables. It must be realized that the words *product* and *sum* do not imply arithmetic operations when dealing with Boolean algebra. Instead they specify *logical* operations equivalent to the Boolean operations of AND and OR, respectively.

Minterms and Maxterms

It has been shown that a truth table defines a Boolean function. An algebraic expression representing the function is derived from the table by finding the logical sum of all product terms for which the function assumes the binary value of 1. A product term in which all the variables appear exactly once either complemented or uncomplemented is called a *minterm*. Its characteristic property is that it shows

TABLE 2-6
Minterms and Maxterms for Three Variables

			Minterms		Maxterms	
X	Y	Z	Product Term	Symbol	Sum Term	Symbol
0	0	0	$\overline{X}\overline{Y}\overline{Z}$	m_0	$X + Y + Z$	M_0
0	0	1	$\overline{X}\overline{Y}Z$	m_1	$X + Y + \overline{Z}$	M_1
0	1	0	$\overline{X}Y\overline{Z}$	m_2	$X + \overline{Y} + Z$	M_2
0	1	1	$\overline{X}YZ$	m_3	$X + \overline{Y} + \overline{Z}$	M_3
1	0	0	$X\overline{Y}\overline{Z}$	m_4	$\overline{X} + Y + Z$	M_4
1	0	1	$X\overline{Y}Z$	m_5	$\overline{X} + Y + \overline{Z}$	M_5
1	1	0	$XY\overline{Z}$	m_6	$\overline{X} + \overline{Y} + Z$	M_6
1	1	1	XYZ	m_7	$\overline{X} + \overline{Y} + \overline{Z}$	M_7

exactly one combination of the binary variables in a truth table. There are 2^n distinct minterms for n variables. The four minterms for the two variables X and Y are $\overline{X}\overline{Y}$, $\overline{X}Y$, $X\overline{Y}$, and XY. The eight minterms for the three variables X, Y, and Z are listed in Table 2-6. The binary numbers from 000 to 111 are listed under the variables. Each minterm is obtained from the product term of exactly three variables with each variable being complemented if the corresponding bit of the binary number is 0 and uncomplemented if it is 1. A symbol for each minterm is also shown in the table and is of the form m_j, where the subscript j denotes the decimal equivalent of the binary number of the mintern. The list of minterms for any given n variables can be formed in a similar manner from a list of the binary numbers from 0 through $2^n - 1$.

A sum term that contains all the variables in complemented or uncomplemented form is called a *maxterm*. Again, it is possible to formulate 2^n maxterms with n variables. The eight maxterms for three variables are listed in Table 2-6. Each maxterm is obtained from the logical sum of the three variables with each variable being complemented if the corresponding bit is 1 and uncomplemented if 0. The symbol for a maxterm is M_j, where j denotes the binary number of the maxterm. Note that a minterm and maxterm with the same subscript number are the complements of each other, that is $M_j = \overline{m}_j$. For example, for $j = 3$, we have

$$\overline{m}_3 = \overline{\overline{X}YZ} = X + \overline{Y} + \overline{Z} = M_3$$

A Boolean function can be expressed algebraically from a given truth table by forming the logical sum of all the minterms which produce a 1 in the function. Consider the Boolean function F in Table 2-7(a). The function is equal to 1 for each of the following binary combinations of the variables X, Y and Z: 000, 010, 101, and 111. These combinations correspond to minterms 0, 2, 5, and 7. The function F can be expressed algebraically as the logical sum of these four minterms.

$$F = \overline{X}\overline{Y}\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}Z + XYZ = m_0 + m_2 + m_5 + m_7$$

This can be further abbreviated by listing only the decimal subscripts of the minterms.

$$F(X, Y, Z) = \Sigma m(0, 2, 5, 7)$$

TABLE 2-7
Boolean Functions of Three Variables

(a) X	Y	Z	F	$\overline{F}$	(b) X	Y	Z	E
0	0	0	1	0	0	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	1	0	0	1	0	1
0	1	1	0	1	0	1	1	0
1	0	0	0	1	1	0	0	1
1	0	1	1	0	1	0	1	1
1	1	0	0	1	1	1	0	0
1	1	1	1	0	1	1	1	0

The symbol Σ stands for the logical sum (Boolean OR) of the minterms. The numbers following it are the minterms of the function. The letters in parentheses following F form a list of the variables in the order taken when the minterms are converted to product terms.

Now consider the complement of a Boolean function. The binary values of $\overline{F}$ in Table 2-7(a) are obtained by changing 1's to 0's and 0's to 1's in the values of F. Taking the logical sum of the minterms of $\overline{F}$ we obtain

$$\overline{F} = \overline{X}\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XY\overline{Z} = m_1 + m_3 + m_4 + m_6$$

or in abbreviated form

$$\overline{F}(X, Y, Z) = \Sigma m(1, 3, 4, 6)$$

Note that the minterm numbers for $\overline{F}$ are the ones missing from the list of the minterms of F. We now take the complement of $\overline{F}$ to obtain F.

$$F = \overline{m_1 + m_3 + m_4 + m_6} = \overline{m}_1 \cdot \overline{m}_3 \cdot \overline{m}_4 \cdot \overline{m}_6$$

$$= M_1 \cdot M_3 \cdot M_4 \cdot M_6 \qquad (\text{Since } \overline{m}_j = M_j)$$

$$= (X + Y + \overline{Z})(X + \overline{Y} + \overline{Z})(\overline{X} + Y + Z)(\overline{X} + \overline{Y} + Z)$$

This shows the procedure for expressing a Boolean function in product of maxterms. The abbreviated form for the product of maxterms form is

$$F(X, Y, Z) = \Pi M(1, 3, 4, 6)$$

The symbol Π denotes the logical product (Boolean AND) of the maxterms listed in parentheses. Note that the decimal numbers included in the product of maxterms will always be the same as the minterm list of the complemented function (1, 3, 4, 6 in the above example). Maxterms are seldom used when dealing with Boolean functions as we can always replace them with the minterm list of $\overline{F}$. The following is a summary of the most important properties of minterms:

1. There are 2^n minterms for n Boolean variables. They can be evaluated from the binary numbers from 0 to $2^n - 1$.
2. Any Boolean function can be expressed as a logical sum of minterms.

3. The complement of a function contains those minterms not included in the original function.
4. A function that included all the 2^n minterms is equal to logic-1.

A function that is not in sum of minterms can be converted to the sum of minterms form by means of a truth table, since the truth table always specifies the minterms of the function. Consider for example the Boolean function

$$E = \overline{Y} + \overline{X}\overline{Z}$$

The expression is not in sum of minterms because each term does not contain all three variables X, Y, and Z. The truth table for this function is listed in Table 2-7(b). From the truth table we obtain the minterms of the function.

$$E(X, Y, Z) = \Sigma m(0, 1, 2, 4, 5)$$

The minterms for the complement of E are

$$\overline{E}(X, Y, Z) = \Sigma m(3, 6, 7)$$

Note that the total number of minterms in E and $\overline{E}$ is equal to eight since the function has three variables and three variables can produce a total of eight minterms. With four variables, there will be a total of 16 minterms, and for two variables, there will be four minterms. An example of a function that includes all the minterms is

$$G(X, Y) = \Sigma m(0, 1, 2, 3) = 1$$

Since G is a function of two variables and contains all four minterms, it will always equal to logic-1.

Sum of Products

The sum of minterms form is a standard algebraic expression that is obtained directly from a truth table. The expression so obtained contains the maximum number of product terms and the maximum number of literals in each term. This is because, by definition, each minterm must include all the variables of the function complemented or uncomplemented. Once the sum of minterms is obtained from the truth table, the next step is to try to simplify the expression to see if it is possible to reduce the number of product terms and the number of literals in the terms. The result is a simplified expression in sum of products. The sum of products is an alternate standard form of expression that contains product terms with one, two or any number of literals. An example of a Boolean function expressed in sum of products is

$$F = \overline{Y} + \overline{X}Y\overline{Z} + XY$$

The expression has three product terms. The first term has one literal, the second has three literals, and the third has two literals.

The logic diagram of a sum of products expression consists of a group of AND gates followed by a single OR gate. This configuration pattern is shown in Figure

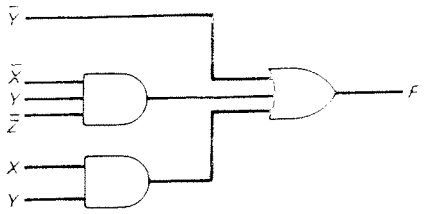


FIGURE 2-6
Sum of Products Implementation

2-6. Each product term requires an AND gate except for a term with a single literal. The logical sum is formed with an OR gate that receives its inputs from the outputs of the AND gates or the single variable. It is assumed that the input variables are directly available in their complement, so inverters are not included in the diagram. The AND gates followed by the OR gate forms a circuit configuration referred to as a *two-level* implementation.

If an expression is not in sum of products form, it can be converted to the standard form by means of the distributive laws. Consider the expression

$$F = AB + C(D + E)$$

This is not in sum of products form because the term $(D + E)$ is part of a product but is not a single variable. The expression can be converted to a sum of products by removing the parentheses.

$$F = AB + C(D + E) = AB + CD + CE$$

The implementation of this function is shown in Figure 2-7. The function is implemented in a nonstandard form in (a). This requires two AND gates and two OR gates. There are three levels of gating in this circuit. The expression is implemented in sum of products form in (b). This circuit requires three AND gates and an OR gate and uses two levels of gating. In general, a two-level implementation is preferred because it produces the least amount of delay time through the gates when the signal propagates from the inputs to the output.

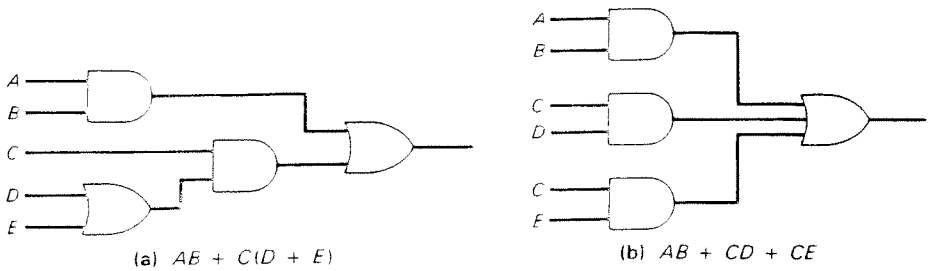


FIGURE 2-7
Three- and Two-Level Implementation

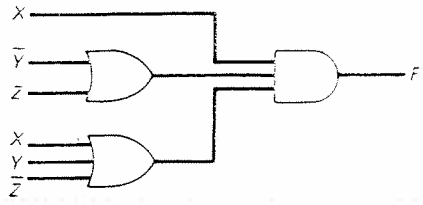


FIGURE 2-8
Product of Sums Implementation

Product of Sums

Another standard form of expressing Boolean functions algebraically is the product of sums. It is obtained by forming the logical product of sum terms. Each logical sum term may have any number of literals. An example of a function expressed in product of sums is

$$F = X(\overline{Y} + Z)(X + Y + \overline{Z})$$

This expression has three sum terms of one, two, and three literals. The sum terms perform an OR operation and the product is an AND operation.

The gate structure of the product of sums expression consists of a group of OR gates for the sum terms (except for a single literal term) followed by an AND gate. This is shown in Figure 2-8. This standard type of expression results in a two-level gating structure.

2-4 MAP SIMPLIFICATION

The complexity of the digital logic gates that implement a Boolean function is directly related to the algebraic expression from which the function is implemented. Although the truth table representation of a function is unique, when expressed algebraically, the function can appear in many different forms. Boolean expressions may be simplified by algebraic manipulation as discussed in Section 2-2. However, this procedure of simplification is awkward because it lacks specific rules to predict each succeeding step in the manipulative process and it is difficult to determine whether the simplest expression has been achieved. The map method provides a straightforward procedure for simplifying Boolean functions of up to four variables. Maps for larger number of variables can be drawn but are more cumbersome to use. The map is also known as the Karnaugh map or K-map.

The map is a diagram made up of squares with each square representing one minterm of the function. Since any Boolean function can be expressed as a sum of minterms, it follows that a Boolean function is recognized graphically in the map from the area enclosed by those squares whose minterms are included in the function. In fact, the map presents a visual diagram of all possible ways a function may be expressed in a standard form. By recognizing various patterns, the user can derive alternate algebraic expressions for the same function, from which the simplest can be selected.

The simplified expressions produced by the map are always in one of the two standard forms: either in sum of products or in product of sums. It will be assumed that the simplest algebraic expression is one with a minimum number of terms and with the fewest possible number of literals in each term. This produces a circuit logic diagram with a minimum number of gates and the minimum number of inputs to the gates. We will see subsequently that the simplest expression is not necessarily unique. It is sometimes possible to find two or more expressions that satisfy the simplification criteria. In that case, either solution would be satisfactory. This section covers only the sum of products simplification. In the next section we will show how to obtain the product of sums simplification.

Two-Variable Map

There are four minterms for a Boolean function with two variables. Hence, the two-variable map consists of four squares, one for each minterm, as shown in Figure 2-9. The map is redrawn in (b) to show the relationship between the squares and the two variables X and Y. The 0 and 1 marked on the left side and the top of the map designate the values of the variables. Variable X appears complemented in row 0 and uncomplemented in row 1. Similarly, Y appears complemented in column 0 and uncomplemented in column 1.

A function of two variables can be represented in a map by marking the squares that correspond to the minterms of the function. As an example, the function XY is shown in Figure 2-10(a). Since XY is equal to minterm m_3, a 1 is placed inside the square that belongs to m_3. Figure 2-10 (b) shows the map for the logical sum of three minterms.

$$m_1 + m_2 + m_3 = \overline{X}Y + X\overline{Y} + XY = X + Y$$

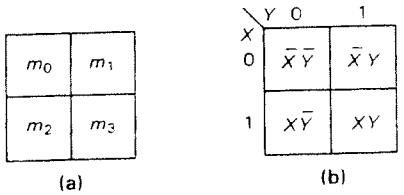


FIGURE 2-9
Two-Variable Map

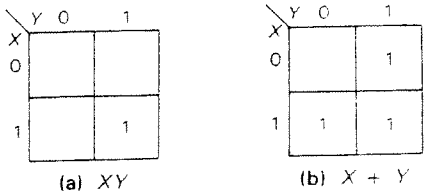


FIGURE 2-10
Representation of Functions in the Map

The simplified expression $X + Y$ is determined from the two-square area for variable X in the second row and the two-square area for Y in the second column. Together, these two areas enclose the three squares belonging to X or Y. This simplification can be justified by algebraic manipulation.

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

The exact procedure for combining squares in the map will be clarified in the examples that follow.

Three-Variable Map

There are eight minterms for three binary variables. Therefore, a three-variable map consists of eight squares as shown in Figure 2-11. The map drawn in part (b) is marked with binary numbers in each row and each column to show the binary values of the minterms. Note that the numbers along the columns do not follow the binary count sequence. The characteristic of the listed sequence is that only one bit changes in value from one adjacent column to the next.

A minterm square can be located in the map in two ways. We can memorize the numbers listed in Figure 2-11 (a) for each minterm location, or we can refer to the binary numbers along the rows and columns. For example, the square assigned to m_5 corresponds to row 1 and column 01. When these two numbers are concatenated, they give the binary number 101, whose decimal equivalent is 5. Another way of looking at square $m_5 = X\overline{Y}Z$ is to consider it to be in the row marked X and the column belonging to $\overline{Y}Z$ (column 01). Note that there are four squares where each variable is equal to 1 and four squares where each is equal to 0. The variable appears uncomplemented in the four squares where it is equal to 1 and complemented in the four squares where it is equal to 0. For convenience, we write the variable name with the letter symbol along the four squares where it is uncomplemented.

To understand the usefulness of the map for simplifying Boolean functions, we must recognize the basic property possessed by adjacent squares. Any two adjacent squares placed horizontally or vertically (but not diagonally) correspond to minterms which differ in only a single variable. The single variable appears uncomplemented in one square and complemented in the other. For example, m_5 and m_7 lie in two adjacent squares. Variable Y is complemented in m_5 and uncomplemented

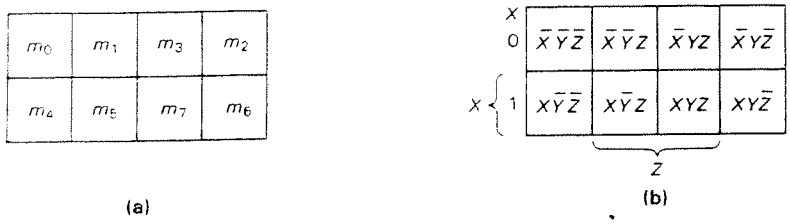


FIGURE 2-11
Three-Variable Map

in m_7, while the other two variables are the same in both squares. The logical sum of two adjacent minterms can be simplified into a single product term of two variables.

$$m_5 + m_7 = X\overline{Y}Z + XYZ = XZ(\overline{Y} + Y) = XZ$$

Here the two squares differ by variable Y, which can be removed when the logical sum (OR) of the two minterms is formed. Thus, any two minterms in adjacent squares that are ORed together produce a product term of two variables.

Example 2-3

Simplify the Boolean function

$$F(X, Y, Z) = \Sigma m(2, 3, 4, 5)$$

First, a 1 is marked in each minterm that represents the function. This is shown in Figure 2-12 where the squares for minterms 010, 011, 100, and 101 are marked with 1's. The next step is to find possible adjacent squares. These are indicated in the map by two rectangles, each enclosing two 1's. The upper right rectangle represents the area enclosed by $\overline{X}Y$. This is determined by observing that the two-square area is in row 0, corresponding to $\overline{X}$, and the last two columns, corresponding to Y. Similarly, the lower left rectangle represents the product term $X\overline{Y}$. (The second row represents X and the two left columns represent $\overline{Y}$.) The logical sum of these two product terms gives the simplified expression.

$$F = \overline{X}Y + X\overline{Y}$$

■

There are cases where two squares in the map are considered to be adjacent even though they do not touch each other. In Figure 2-11, m_0 is adjacent to m_2 and m_4 is adjacent to m_6 because the minterms differ by one variable. This can be readily verified algebraically.

$$m_0 + m_2 = \overline{X}\overline{Y}\overline{Z} + \overline{X}Y\overline{Z} = \overline{X}\overline{Z}(\overline{Y} + Y) = \overline{X}\overline{Z}$$

$$m_4 + m_6 = X\overline{Y}\overline{Z} + XY\overline{Z} = X\overline{Z}(\overline{Y} + Y) = X\overline{Z}$$

Consequently, we must modify the definition of adjacent squares to include this and other similar cases. This is done by considering the map as being drawn on a surface where the right and left edges touch each other to form adjacent squares.

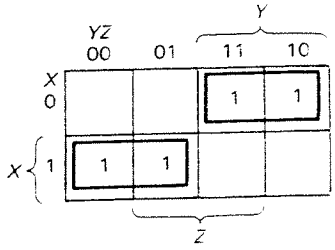


FIGURE 2-12
Map for Example 2-3: $F(X, Y, Z) = \Sigma m(2, 3, 4, 5) = \overline{X}Y + X\overline{Y}$

Consider now any combination of four adjacent squares in the three-variable map. Any such combination represent the logical sum of four minterms and results in an expression of only one literal. As an example, the logical sum of the four adjacent minterms 0, 2, 4, and 6 reduces to a single literal term $\overline{Z}$.

$$\begin{aligned} m_0 + m_2 + m_4 + m_6 &= \overline{X}\overline{Y}\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XY\overline{Z} \\ &= \overline{X}\overline{Z}(\overline{Y} + Y) + X\overline{Z}(\overline{Y} + Y) \\ &= \overline{X}\overline{Z} + X\overline{Z} = \overline{Z}(\overline{X} + X) = \overline{Z} \end{aligned}$$

The number of adjacent squares that may be combined must always represent a number that is a power of two such as 1, 2, 4, and 8. As a larger number of adjacent squares are combined, we obtain a product term with fewer literals.

One square represents a minterm of three literals.

Two adjacent squares represent a product term of two literals.

Four adjacent squares represent a product term of one literal.

Eight adjacent squares encompasses the entire map and produces a function which is always equal to logic-1.

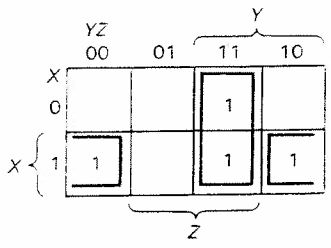
Example 2-4 Simplify the two Boolean functions

$$F_1(X, Y, Z) = \Sigma m(3, 4, 6, 7)$$

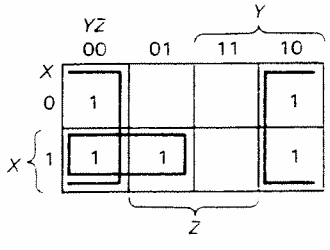
$$F_2(X, Y, Z) = \Sigma m(0, 2, 4, 5, 6)$$

The map for F_1 is shown in Figure 2-13(a). There are four squares marked with 1's, one for each minterm of the function. Two adjacent squares are combined in the third column to give a two-literal term YZ. The remaining two squares with 1's are also adjacent by the new definition and are shown in the diagram with their values enclosed in half rectangles. These two squares when combined, give the two-literal term $X\overline{Z}$. The simplified function becomes

$$F_1 = YZ + X\overline{Z}$$



(a) $F_1(X, Y, Z) = \Sigma m(3, 4, 6, 7)$
$= YZ + X\overline{Z}$



(b) $F_2(X, Y, Z) = \Sigma m(0, 2, 5, 6)$
$= \overline{Z} + X\overline{Y}$

FIGURE 2-13
Maps for Example 2-4

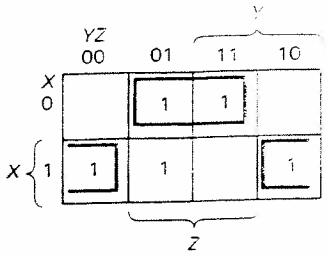


FIGURE 2-14
$F(X, Y, Z) = \Sigma m(1, 3, 4, 5, 6)$
$= \overline{X}Z + X\overline{Z} + X\overline{Y}$
$= \overline{X}Z + X\overline{Z} + \overline{Y}Z$

The map for F_2 is shown in Fig. 2-13(b). First we combine the four adjacent squares in the first and last columns to give the single literal term $\overline{Z}$. The remaining single square representing minterm 5 is combined with an adjacent square that is already being used once. This is not only permissible but rather desirable since the two adjacent squares give the two-literal term $X\overline{Y}$ while the single square represents the three-literal minterm $X\overline{Y}Z$. The simplified function is

$$F_2 = \overline{Z} + X\overline{Y}$$

■

There are occasions when there are two alternate ways of combining squares to produce equally simplified expressions. An example of this is demonstrated in the map of Figure 2-14. Minterms 1 and 3 are combined to give the term $\overline{X}Z$ and minterms 4 and 6 produce the term $X\overline{Z}$. However, there are two ways that the square of minterm 5 can be combined with another adjacent square to produce a third two-literal term. Combining it with minterm 4 gives the term $X\overline{Y}$. We could choose instead to combine it with minterm 1 to give the term $\overline{Y}Z$. Each of the two possible simplified expressions listed in Figure 2-14 has three terms of two literals each; so there are two possible simplified solutions for this function.

If a function is not expressed as a sum of minterms, we can use the map to obtain the minterms of the function and then simplify the function. It is necessary to have the algebraic expression in sum of products, from which each product term is plotted in the map. The minterms of the function are then read directly from the map. Consider the following Boolean function.

$$F = \overline{X}Z + \overline{X}Y + X\overline{Y}Z + YZ$$

Three product terms in the expression have two literals and are represented in a three-variable map by two squares each. The two squares corresponding to the first term, $\overline{X}Z$, are found in Figure 2-15 from the coincidence of $\overline{X}$ (first row) and Z (two middle columns) to give squares 001 and 011. Note that when marking 1's in the squares, it is possible to find a 1 already placed there from a preceding term. This happens with the second term $\overline{X}Y$ which has 1's in squares 011 and 010, but square 011 is common with the first term $\overline{X}Z$ so only one 1 is marked in it. Continuing in this fashion we find that the function has five minterms as indicated by the five 1's in the map of Figure 2-15. The minterms are read directly from the

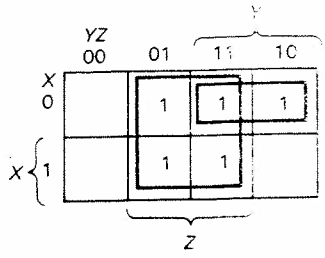


FIGURE 2-15
$F(X, Y, Z) = \Sigma m(1, 2, 3, 5, 7) = Z + \bar{X}Y$

map to be 1, 2, 3, 5, and 7. The function as originally given has too many product terms. It can be simplified to only two terms.

$$F = Z + \bar{X}Y$$

Four-Variable Map

There are 16 minterms for four binary variables and therefore, a four-variable map consists of 16 squares as shown in Figure 2-16. The minterm assignment in each square is indicated in part (a) of the diagram. The map is redrawn in (b) to show the relationship of the four variables. The rows and columns are numbered in a special sequence so that only one bit of the binary number changes in value between any two adjacent squares. The minterms corresponding to each square can be obtained from the concatenation of the row number with the column number. For example, the numbers in the third row (11) and the second column (01), when concatenated, give the binary number 1101, the binary equivalent of 13. Thus, the square in the third row and second column represents minterm m_{13}. In addition, each variable is marked in the map to show the eight squares where it appears

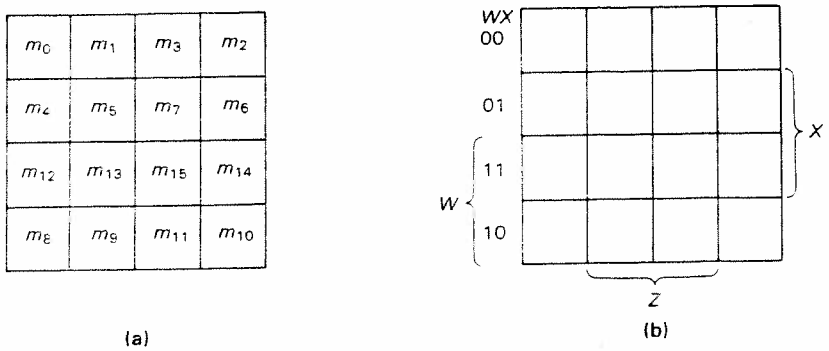


FIGURE 2-16
Four-Variable Map

uncomplemented. The other eight squares where no label is indicated correspond to the variable being complemented. Thus, W appears complemented in the first two rows and uncomplemented in the second two rows.

The map simplification of four-variable functions is similar to the method used to simplify three-variable functions. Adjacent squares are defined to be squares next to each other. In addition, the map is considered to lie on a surface with the top and bottom edges, as well as the right and left edges, touching each other to form adjacent squares. For example, m_0 and m_2 are two adjacent squares, as are m_3 and m_{11}. The combination of squares that can be taken during the simplification process in the four-variable map is as follows:

One square represents a minterm of four literals.
Two adjacent squares represent a product term of three literals.
Four adjacent squares represent a product term of two literals.
Eight adjacent squares represent a product term of one literal.
Sixteen squares produce a function which is always equal to logic-1.

No other combination of adjacent squares can be used. The following examples show the procedure for simplifying four-variable Boolean functions.

Example 2-5 Simplify the Boolean function

$$F(W, X, Y, Z) = \Sigma m(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

The minterms of the function are marked with 1's in the map of Figure 2-17. Eight adjacent squares in the two left columns are combined to form the one literal term $\overline{Y}$. The remaining three 1's cannot be combined together to give a simplified term. They must be combined as two or four adjacent squares. The top two 1's on the right are combined with the top two 1's on the left to give the term $\overline{W}\overline{Z}$. Note again that it is permissible to take the same square more than once. We are now left with a square marked with 1 in the third row and fourth column (minterm 1100). Instead of taking this square alone which will give a term of four literals,

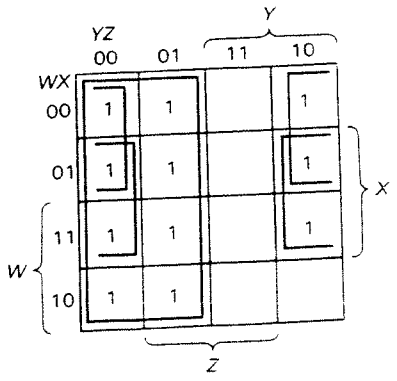


FIGURE 2-17
Map for Example 2-5: $F = \overline{Y} + \overline{W}\overline{Z} + X\overline{Z}$

we combine it with squares already used to form an area of four adjacent squares in the two middle rows and the two end columns, giving the term $X\overline{Z}$. The simplified expression is the logical sum of the three terms.

$$F = \overline{Y} + \overline{W}\overline{Z} + X\overline{Z}$$

■

Example 2-6 Simplify the Boolean function

$$F = \overline{A}\overline{B}\overline{C} + \overline{B}C\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\overline{C}$$

This function has four variables A, B, C, and D. It is expressed in sum of products with three terms of three literals each and one term of four literals. The area in the map covered by this function is shown in Figure 2-18. Each term of three literals is represented in the map with two squares. $\overline{A}\overline{B}\overline{C}$ is represented in squares 0000 and 0001, $\overline{B}C\overline{D}$ in squares 0010 and 1010, and $A\overline{B}\overline{C}$ in squares 1000 1001. The term with four literals is minterm 0110. The function is simplified in the map by taking the 1's in the four corners to give the term $\overline{B}\overline{D}$. This is possible because these four squares are adjacent when the map is drawn on a surface with top and bottom or left and right edges touching one another. The two 1's in the top row are combined with the two 1's in the bottom row to give the term $\overline{B}\overline{C}$. The remaining 1 in square 0110 is combined with its adjacent square 0010 to give the term $\overline{A}C\overline{D}$. The simplified function is

$$F = \overline{B}\overline{D} + \overline{B}\overline{C} + \overline{A}C\overline{D}$$

■

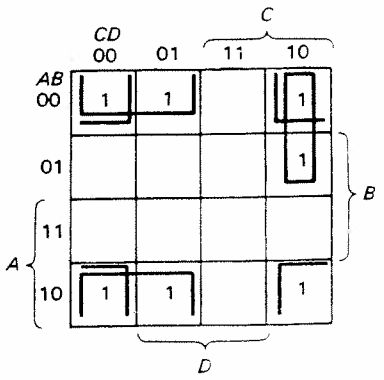


FIGURE 2-18
Map for Example 2-6; $F = \overline{B}\overline{D} + \overline{B}\overline{C} + \overline{A}C\overline{D}$

2-5 MAP MANIPULATION

When combining adjacent squares in a map, it is necessary to ensure that all the minterms of the function are included. At the same time it is necessary to minimize the number of terms in the simplified function by avoiding any redundant terms whose minterms are already covered by other terms. In this section we consider a