This example also demonstrates that an expression with the minimum number of literals is not necessarily unique. Sometimes the designer is confronted with a choice between two terms with an equal number of literals, with either choice resulting in a minimized expression.

3-9 THE TABULATION METHOD

The map method of simplification is convenient as long as the number of variables does not exceed five or six. As the number of variables increases, the excessive number of squares prevents a reasonable selection of adjacent squares. The obvious disadvantage of the map is that it is essentially a trial-and-error procedure which relies on the ability of the human user to recognize certain patterns. For functions of six or more variables, it is difficult to be sure that the best selection has been made.

The tabulation method overcomes this difficulty. It is a specific step-by-step procedure that is guaranteed to produce a simplified standard-form expression for a function. It can be applied to problems with many variables and has the advantage of being suitable for machine computation. However, it is quite tedious for human use and is prone to mistakes because of its routine, monotonous process. The tabulation method was first formulated by Quine (3) and later improved by McCluskey (4). It is also known as the Quine-McCluskey method.

The tabular method of simplification consists of two parts. The first is to find by an exhaustive search all the terms that are candidates for inclusion in the simplified function. These terms are called *prime-implicants*. The second operation is to choose among the prime-implicants those that give an expression with the least number of literals.

3-10 DETERMINATION OF PRIME-IMPLICANTS*

The starting point of the tabulation method is the list of minterms that specify the function. The first tabular operation is to find the prime-implicants by using a matching process. This process compares each minterm with every other minterm. If two minterms differ in only one variable, that variable is removed and a term with one less literal is found. This process is repeated for every minterm until the exhaustive search is completed. The matching-process cycle is repeated for those new terms just found. Third and further cycles are continued until a single pass through a cycle yields no further elimination of literals. The remaining terms and all the terms that did not match during the process comprise the prime-implicants. This tabulation method is illustrated by the following example.

*This section and the next may be omitted without loss of continuity.

EXAMPLE 3-13: Simplify the following Boolean function by using the tabulation method:

$$F = \Sigma(0, 1, 2, 8, 10, 11, 14, 15)$$

Step 1: Group binary representation of the minterms according to the number of 1's contained, as shown in Table 3-5, column (a). This is done by grouping the minterms into five sections separated by horizontal lines. The first section contains the number with no 1's in it. The second section contains those numbers that have only one 1. The third, fourth, and fifth sections contain those binary numbers with two, three, and four 1's, respectively. The decimal equivalents of the minterms are also carried along for identification.

Step 2: Any two minterms which differ from each other by only one variable can be combined, and the unmatched variable removed. Two minterm numbers fit into this category if they both have the same bit value in all positions except one. The minterms in one section are compared with those of the next section down only, because two terms differing by more than one bit cannot match. The minterm in the first section is compared with each of the three minterms in the second section. If any two numbers are the same in every position but one, a check is placed to the right of both minterms to show that they have been used. The resulting term,

TABLE 3-5 Determination of prime-implicants for Example 3-13

(a)		(b)		(c)	
	w x y z		*w x y z*		*w x y z*
0	0 0 0 0 √	0, 1	0 0 0 −	0, 2, 8, 10	− 0 − 0
		0, 2	0 0 − 0 √	0, 8, 2, 10	− 0 − 0
1	0 0 0 1 √	0, 8	− 0 0 0 √	10, 11, 14, 15	1 − 1 −
2	0 0 1 0 √			10, 14, 11, 15	1 − 1 −
8	1 0 0 0 √	2, 10	− 0 1 0 √		
		8, 10	1 0 − 0 √		
10	1 0 1 0 √				
		10, 11	1 0 1 − √		
11	1 0 1 1 √	10, 14	1 − 1 0 √		
14	1 1 1 0 √				
		11, 15	1 − 1 1 √		
15	1 1 1 1 √	14, 15	1 1 1 − √		

together with the decimal equivalents, is listed in column (b) of the table. The variable eliminated during the matching is denoted by a dash in its original position. In this case m_0 (0000) combines with m_1 (0001) to form (000 –). This combination is equivalent to the algebraic operation:

$$m_0 + m_1 = w'x'y'z' + w'x'y'z = w'x'y'$$

Minterm m_0 also combines with m_2 to form (00-0) and with m_8 to form (-000). The result of this comparison is entered into the first section of column (b). The minterms of sections two and three of column (a) are next compared to produce the terms listed in the second section of column (b). All other sections of (a) are similarly compared and subsequent sections formed in (b). This exhaustive comparing process results in the four sections of (b).

Step 3: The terms of column (b) have only three variables. A 1 under the variable means it is unprimed, a 0 means it is primed, and a dash means the variable is not included in the term. The searching and comparing process is repeated for the terms in column (b) to form the two-variable terms of column (c). Again, terms in each section need to be compared only if they have dashes in the same position. Note that the term (000-) does not match with any other term. Therefore, it has no check mark at its right. The decimal equivalents are written on the left-hand side of each entry for identification purposes. The comparing process should be carried out again in column (c) and in subsequent columns as long as proper matching is encountered. In the present example, the operation stops at the third column.

Step 4: The unchecked terms in the table form the prime-implicants. In this example we have the term $w'x'y'$ (000-) in column (b), and the terms $x'z'$(-0-0) and wy (1-1-) in column (c). Note that each term in column (c) appears twice in the table, and as long as the term forms a prime-implicant, it is unnecessary to use the same term twice. The sum of the prime-implicants gives a simplified expression for the function. This is because each checked term in the table has been taken into account by an entry of a simpler term in a subsequent column. Therefore, the unchecked entries (prime-implicants) are the terms left to formulate the function. For the present example, the sum of prime-implicants gives the minimized function in sum of products:

$$F = w'x'y' + x'z' + wy$$

It is worth comparing this answer with that obtained by the map method. Figure 3-27 shows the map simplification of this function. The combinations of

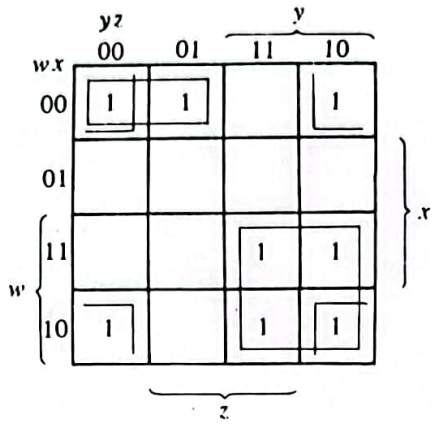

Figure 3-27 Map for the function of Example 3-13; $F = w'x'y' + x'z' + wy$

adjacent squares give the three prime-implicants of the function. The sum of these three terms is the simplified expression in sum of products.

It is important to point out that Example 3-13 was purposely chosen to give the simplified function from the sum of prime-implicants. In most other cases, the sum of prime-implicants does not necessarily form the expression with the minimum number of terms. This is demonstrated in Example 3-14.

The tedious manipulation that one must undergo when using the tabulation method is reduced if the comparing is done with decimal numbers instead of binary. A method will now be shown that uses subtraction of decimal numbers instead of the comparing and matching of binary numbers. We note that each 1 in a binary number represents the coefficient multiplied by a power of 2. When two minterms are the same in every position except one, the minterm with the extra 1 must be larger than the number of the other minterm by a power of 2. Therefore, two minterms can be combined if the number of the first minterm differs by a power of 2 from a second larger number in the next section down the table. We shall illustrate this procedure by repeating Example 3-13.

As shown in Table 3-6, column (a), the minterms are arranged in sections as before, except that now only the decimal equivalents of the minterms are listed. The process of comparing minterms is as follows: Inspect every two decimal numbers in adjacent sections of the table. If the number in the section below is *greater* than the number in the section above by a power of 2 (i.e., 1, 2, 4, 8, 16, etc.), check both numbers to show that they have been used, and write them down in column (b). The pair of numbers transferred to column (b) includes a third number in parentheses that designates the power of 2 by which the numbers differ. The number in parentheses tells us the position of the dash in the binary notation. The result of all comparisons of column (a) is shown in column (b).

The comparison between adjacent sections in column (b) is carried out in a similar fashion, except that only those terms with the same number in parentheses are compared. The pair of numbers in one section must differ by a power of 2 from the pair of numbers in the next section. And the numbers in the next section

TABLE 3-6 Determination of prime-implicants of Example 3-13 with decimal notation

(a)	(b)	(c)
0 √	0, 1 (1)	0, 2, 8, 10 (2, 8)
	0, 2 (2) √	0, 2, 8, 10 (2, 8)
1 √	0, 8 (8) √	
2 √		10, 11, 14, 15 (1, 4)
8 √	2, 10 (8) √	10, 11, 14, 15 (1, 4)
	8, 10 (2) √	
10 √		
	10, 11 (1) √	
11 √	10, 14 (4) √	
14 √		
	11, 15 (4) √	
15 √	14, 15 (1) √	

below must be *greater* for the combination to take place. In column (c), write all four decimal numbers with the two numbers in parentheses designating the positions of the dashes. A comparison of Tables 3-5 and 3-6 may be helpful in understanding the derivations in Table 3-6.

The prime-implicants are those terms not checked in the table. These are the same as before, except that they are given in decimal notation. To convert from decimal notation to binary, convert all decimal numbers in the term to binary and then insert a dash in those positions designated by the numbers in parentheses. Thus 0, 1 (1) is converted to binary as 0000, 0001; a dash in the first position of either number results in (000-). Similarly, 0, 2, 8, 10 (2, 8) is converted to the binary notation from 0000, 0010, 1000, and 1010, and a dash inserted in positions 2 and 8, to result in (-0-0)

EXAMPLE 3-14: Determine the prime-implicants of the function:

$$F(w, x, y, z) = \Sigma(1, 4, 6, 7, 8, 9, 10, 11, 15)$$

The minterm numbers are grouped in sections as shown in Table 3-7, column (a). The binary equivalent of the minterm is included for the purpose of counting the number of 1's. The binary numbers in the first section have only one 1; in the second section, two 1's; etc. The minterm numbers are compared by the decimal method and a match is found if the number in the section below is greater than that in the section above. If the number in the section below is smaller than the

TABLE 3-7 Determination of prime-implicants for Example 3-14

(a)		(b)		(c)
0001	1 √	1, 9	(8)	8, 9, 10, 11 (1, 2)
0100	4 √	4, 6	(2)	8, 9, 10, 11 (1, 2)
1000	8 √	8, 9	(1) √	
		8, 10	(2) √	
0110	6 √			
1001	9 √	6, 7	(1)	
1010	10 √	9, 11	(2) √	
		10, 11	(1) √	
0111	7 √			
1011	11 √	7, 15	(8)	
		11, 15	(4)	
1111	15 √			

Prime-implicants

Decimal	Binary $w\ x\ y\ z$	Term
1, 9 (8)	− 0 0 1	$x'y'z$
4, 6 (2)	0 1 − 0	$w'xz'$
6, 7 (1)	0 1 1 −	$w'xy$
7, 15 (8)	− 1 1 1	xyz
11, 15 (4)	1 − 1 1	wyz
8, 9, 10, 11 (1, 2)	1 0 − −	wx'

one above, a match is not recorded even if the two numbers differ by a power of 2. The exhaustive search in column (a) results in the terms of column (b), with all minterms in column (a) being checked. There are only two matches of terms in column (b). Each gives the same two-literal term recorded in column (c). The prime-implicants consist of all the unchecked terms in the table. The conversion from the decimal to the binary notation is shown at the bottom of the table. The prime-implicants are found to be $x'y'z$, $w'xz'$, $w'xy$, xyz, wyz, and wx'.

The sum of the prime-implicants gives a valid algebraic expression for the function. However, this expression is not necessarily the one with the minimum number of terms. This can be demonstrated from inspection of the map for the

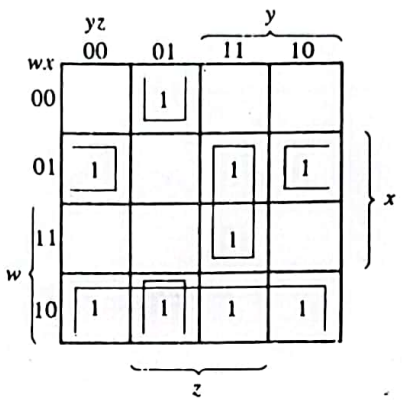

Figure 3-28 Map for the function of Example 3-14; $F = x'y'z + w'xz' + xyz + wx'$

function of Example 3-14. As shown in Fig. 3-28, the minimized function is recognized to be:

$$F = x'y'z + w'xz' + xyz + wx'$$

which consists of the sum of four of the six prime-implicants derived in Example 3-14. The tabular procedure for selecting the prime-implicants that give the minimized function is the subject of the next section.

3-11 SELECTION OF PRIME-IMPLICANTS

The selection of prime-implicants that form the minimized function is made from a prime-implicant table. In this table, each prime-implicant is represented in a row and each minterm in a column. Crosses are placed in each row to show the composition of minterms that make the prime-implicants. A minimum set of prime-implicants is then chosen that covers all the minterms in the function. This procedure is illustrated in Example 3-15.

EXAMPLE 3-15: Minimize the function of Example 3-14. The prime-implicant table for this example is shown in Table 3-8. There are six rows, one for each prime-implicant (derived in Example 3-14), and nine columns, each representing one minterm of the function. Crosses are placed in each row to indicate the minterms contained in the prime-implicant of that row. For example, the two crosses in the first row indicate that minterms 1 and 9 are contained in the prime-implicant $x'y'z$. It is advisable to include the decimal equivalent of the prime-implicant in each row, as it conveniently gives the minterms contained in it. After all the crosses have been marked, we proceed to select a minimum number of prime-implicants.

The completed prime-implicant table is inspected for columns

TABLE 3-8 Prime-implicant table for Example 3-15

		1	4	6	7	8	9	10	11	15
$\surd x'y'z$	1, 9	X					X			
$\surd w'xz'$	4, 6		X	X						
$w'xy$	6, 7			X	X					
xyz	7, 15				X					X
wyz	11, 15								X	X
$\surd wx'$	8, 9, 10, 11					X	X	X	X	
		$\surd$	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$	$\surd$	

containing only a single cross. In this example, there are four minterms whose columns have a single cross: 1, 4, 8, and 10. Minterm 1 is covered by prime-implicant $x'y'z$, i.e., the selection of prime-implicant $x'y'z$ guarantees that minterm 1 is included in the function. Similarly, minterm 4 is covered by prime-implicant $w'xz'$, and minterms 8 and 10, by prime-implicant wx'. Prime-implicants that cover minterms with a single cross in their column are called *essential prime-implicants*. To enable the final simplified expression to contain all the minterms, we have no alternative but to include essential prime-implicants. A check mark is placed in the table next to the essential prime-implicants to indicate that they have been selected.

Next we check each column whose minterm is covered by the selected essential prime-implicants. For example, the selected prime-implicant $x'y'z$ covers minterms 1 and 9. A check is inserted in the bottom of the columns. Similarly, prime-implicant $w'xz'$ covers minterms 4 and 6, and wx' covers minterms 8, 9, 10, and 11. Inspection of the prime-implicant table shows that the selection of the essential prime-implicants covers all the minterms of the function except 7 and 15. These two minterms must be included by the selection of one or more prime-implicants. In this example, it is clear that prime-implicant xyz covers both minterms and is therefore the one to be selected. We have thus found the minimum set of prime-implicants whose sum gives the required minimized function:

$$F = x'y'z + w'xz' + wx' + xyz$$

The simplified expressions derived in the preceding examples were all in the sum of products form. The tabulation method can be adapted to give a simplified expression in product of sums. As in the map method, we have to start with the complement of the function by taking the 0's as the initial list of minterms. This list contains those minterms not included in the original function which are

numerically equal to the maxterms of the function. The tabulation process is carried out with the 0's of the function and terminates with a simplified expression in sum of products of the complement of the function. By taking the complement again, we obtain the simplified product of sums expression.

A function with don't-care conditions can be simplified by the tabulation method after a slight modification. The don't-care terms are included in the list of minterms when the prime-implicants are determined. This allows the derivation of prime-implicants with the least number of literals. The don't-care terms are not included in the list of minterms when the prime-implicant table is set up, because don't-care terms do not have to be covered by the selected prime-implicants.

3-12 CONCLUDING REMARKS

Two methods of Boolean function simplification were introduced in this chapter. The criterion for simplification was taken to be the minimization of the number of literals in sum of products or product of sums expressions. Both the map and the tabulation methods are restricted in their capabilities since they are useful for simplifying only Boolean functions expressed in the standard forms. Although this is a disadvantage of the methods, it is not very critical. Most applications prefer the standard forms over any other form. We have seen from Fig. 3-15 that the gate implementation of expressions in standard form consists of no more than two levels of gates. Expressions not in the standard form are implemented with more than two levels. Humphrey (5) shows an extension of the map method that produces simplified multilevel expressions.

One should recognize that the reflected-code sequence chosen for the maps is not unique. It is possible to draw a map and assign a binary reflected-code sequence to the rows and columns different from the sequence employed here. As long as the binary sequence chosen produces a change in only one bit between adjacent squares, it will produce a valid and useful map.

Two alternate versions of the three-variable maps which are often found in the digital logic literature are shown in Fig. 3-29. The minterm numbers are written

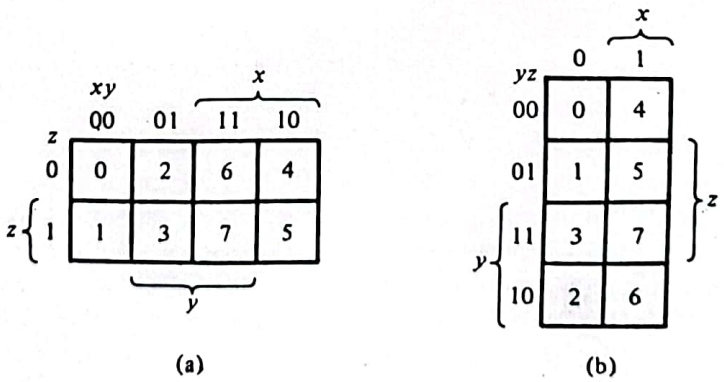

Figure 3-29 Variations of the three-variable map