4.1 INTRODUCTION

The complexity of digital logic gates to implement a Boolean function is directly related to the complexity of algebraic expression. Also, an increase in the number of variables results in an increase of complexity. Although the truth table representation of a Boolean function is unique, its algebraic expression may be of many different forms. Boolean functions may be simplified or minimized by algebraic means as described in Chapter 3. However, this minimization procedure is not unique because it lacks specific rules to predict the succeeding step in the manipulative process. The map method, first proposed by Veitch and slightly improvised by Karnaugh, provides a simple, straightforward procedure for the simplification of Boolean functions. The method is called Veitch diagram or Karnaugh map, which may be regarded either as a pictorial representation of a truth table or as an extension of the Venn diagram.

The Karnaugh map provides a systematic method for simplification and manipulation of a Boolean expression. The map is a diagram consisting of squares. For \( n \) variables on a Karnaugh map there are \( 2^n \) numbers of squares. Each square or cell represents one of the minterms. Since any Boolean function can be expressed as a sum of minterms, it is possible to recognize a Boolean function graphically in the map from the area enclosed by those squares whose minterms appear in the function. It is also possible to derive alternative algebraic expressions or simplify the expression with a minimum number of variables or literals and sum of products or product of sums terms, by analyzing various patterns. In fact, the map represents a visual diagram of all possible ways a function can be expressed in a standard form and the simplest algebraic expression consisting of a sum of products or product of sums can be selected. Note that the expression is not necessarily unique.

4.2 TWO-VARIABLE KARNAUGH MAPS

A two-variable Karnaugh map is shown in Figure 4.1. Since a two-variable system can form four minterms, the map consists of four cells—one for each minterm. The map has been
redrawn in Figure 4.1(b) to show the relationship between the squares and the two variables A and B. Note that, in the first row, the variable A is complemented, in the second row A is uncomplemented, in the first column variable B is complemented and in the second column B is uncomplemented.

The two-variable Karnaugh map is a useful way to represent any of the 16 Boolean functions of two variables as described in section 3.7, if the squares are marked with 1 whose minterms belong to a certain function. As an example, the function $AB$ has been shown in Figure 4.2(a). Since the function $AB$ is equal to the minterm $m_3$, a 1 is placed in the cell corresponding to $m_3$. Similarly, the function $A + B$ has three minterms of $A'B$, $AB'$, and $AB$, as

$$A + B = A (B + B') + B (A + A') = AB + AB' + AB + A'B = A'B + AB' + AB.$$ 

So the squares corresponding to $A'B$, $AB'$, and $AB$ are marked with 1 as shown in Figure 4.2(b).

4.3 THREE-VARIABLE KARNAUGH MAPS

Since, there are eight minterms for three variables, the map consists of eight cells or squares, which is shown in Figure 4.3(a). It may be noticed that the minterms are arranged, not according to the binary sequence, but according to the sequence similar to the reflected code, which means, between two consecutive rows or columns, only one single variable changes its logic value from 0 to 1 or from 1 to 0. Figure 4.3(b) shows the relationship between the squares and the variables. Two rows are assigned to $A'$ and A, and four columns to $B'C'$, $B'C$, $BC$, and $BC'$. The minterm $m_3$, for example, is assigned in the square corresponding to row 0 and column 11, thus making the binary number 011. Another way of analyzing the square $m_3$, is to consider it to be in the row $A'$ and column BC, as $m_3 = ABC$. Note that, each of the variables has four squares where its logic value is 0 and four squares with logic value 1.
The three-variable Karnaugh Map can be constructed in other ways, too. Figure 4.4(a) shows if variable C is assigned to the rows and variables A and B are assigned along the columns. Figure 4.4(b) demonstrates where variable A is along columns and variables B and C are along the rows. Corresponding minterms are shown in the figures.

To understand the usefulness of the map for simplifying the Boolean functions, we must observe the basic properties of the adjacent squares. Any two adjacent squares in the Karnaugh map differ by only one variable, which is complemented in one square and uncomplemented in one of the adjacent squares. For example, in Figure 4.3(a), m₁ and m₃ are placed at adjacent squares, where variable B is complemented at m₁ while it is uncomplemented at m₃. From the postulates of Boolean algebra, the sum of two minterms can be simplified to a single AND term consisting of less number of literals. As in the case of m₁ and m₃, m₁ + m₃ can be reduced to the term below.

\[ m₁ + m₃ = AB'C + ABC = AC (B' + B) = AC \]

So it can be observed, the variable which has been changed at the adjacent squares can be removed, if the minterms of those squares are ORed together.

**Example 4.1. Simplify the Boolean function**

\[ F = A'BC + A'BC' + AB'C' + AB'C. \]

**Solution.** First, a three-variable Karnaugh map is drawn and 1s are placed at the squares according to the minterms of the function as shown in Figure 4.5. Now two 1s
of adjacent squares are grouped together. As in the figure, \( A'BC \) and \( A'BC' \) are grouped together at the first row, and \( AB'C \) and \( ABC \) are grouped together. From the first row, the reduced term of \( ABC + A'BC' \) is \( A'B \), as \( C \) is the variable which changes its form. Similarly from the second row, \( AB'C + AB'C \) can be simplified to \( AB' \). Now, as further simplification is not possible for this particular Boolean function, the simplified sum of the product of the function can be written as,

\[ F = A'B + AB'. \]

![Figure 4.5](image)

**Example 4.2.** Simplify the expression \( F = A'BC + AB'C' + ABC + ABC' \).

**Solution.** The Karnaugh map for this function is shown in Figure 4.6. There are four squares marked with 1s, each for one of the minterms of the function.

![Figure 4.6](image)

In the third column, two adjacent squares are grouped together to produce the simplified term \( BC \). The other two 1s are placed at the first column and last column of the same second row. Note that these 1s or minterms can be combined to produce a reduced term. Here the \( B \) variable is changing its form, from uncomplemented to complemented. After combining these two minterms, we get the reduced term \( AC' \). This can be confirmed by applying the Boolean algebra, \( AB'C' + ABC = AC' \cdot (B + B') = AC' \).

Therefore, the final simplified expression can be written as,

\[ F = BC + AC'. \]

As in the previous examples, it is shown that two adjacent squares consisting of 1s can be combined to form reduced terms. Similarly, it is possible to combine four adjacent squares consisting of 1s, in the process of simplification of Boolean functions. Let us consider the next example.

**Example 4.3.** Simplify the expression \( F = A'B'C + A'BC + A'BC' + AB'C + ABC \).

**Solution.** The Karnaugh map is shown in Figure 4.7. The four adjacent squares comprising the minterms \( A'B'C \), \( A'BC \), \( AB'C \), and \( ABC \) can be combined. Here, it may observed that two of the variables \( A \) and \( B \) are changing their forms form uncomplemented to complemented. Therefore, these variables can be removed to form the reduced expression to \( C \).
SIMPLIFICATION AND MINIMIZATION OF BOOLEAN FUNCTIONS

Again, two adjacent squares comprising the minterms $A'B'C$ and $A'B'C'$ can be combined to produce the reduced term $A'B$. So the final simplified expression of the given function is

$$F = C + A'B.$$  

Note that squares that are already considered in one group, can be combined with other group or groups.

Example 4.4. Simplify the expression $F (A, B, C) = \sum (0, 2, 4, 5, 6)$.

The Karnaugh map is shown in Figure 4.8. Here, the minterms are given by their decimal-equivalent numbers. The squares according to those minterms are filled with 1s. $A'B'C'$, $ABC'$, $AB'C'$, and $ABC'$ are grouped to produce the reduced term of $C'$ and, $AB'C'$ and $ABC$ are grouped to produce the term $AB'$. So the final simplified expression may be written as

$$F = C' + AB'.$$

Note that four squares of the first column and last column may be combined just like the two squares combination explained in Example 4.2.

4.4 FOUR-VARIABLE KARNAUGH MAPS

Similar to the method used for two-variable and three-variable Karnaugh maps, four-variable Karnaugh maps may be constructed with 16 squares consisting of 16 minterms as shown in Figure 4.9(a). The same is redrawn in Figure 4.9(b) to show the relationship with the four binary variables. The rows and columns are numbered in a reflected code sequence, where only one variable is changing its form between two adjacent squares. The minterm of a particular square can be obtained by combining the row and column. As an example, the minterm of the second row and third column is $A'B'CD$ i.e., $m_{12}$.
Different four-variable Karnaugh maps can be redrawn, if the variables are assigned an other way. Figure 4.10(a) and 4.10(b) also demonstrate the location of minterms for four-variable Karnaugh maps when variables A and B are assigned along the columns and variables C and D are assigned along the rows.

<table>
<thead>
<tr>
<th></th>
<th>A' B'</th>
<th>A' B</th>
<th>A B</th>
<th>A B'</th>
</tr>
</thead>
<tbody>
<tr>
<td>A' B'</td>
<td>m_0</td>
<td>m_8</td>
<td>m_12</td>
<td>m_8</td>
</tr>
<tr>
<td>C' D</td>
<td>m_1</td>
<td>m_8</td>
<td>m_13</td>
<td>m_3</td>
</tr>
<tr>
<td>C' D</td>
<td>m_3</td>
<td>m_7</td>
<td>m_15</td>
<td>m_16</td>
</tr>
<tr>
<td>C' D</td>
<td>m_2</td>
<td>m_8</td>
<td>m_14</td>
<td>m_10</td>
</tr>
</tbody>
</table>

The minimization of four-variable Boolean functions using Karnaugh maps is similar to the method used to minimize three-variable functions. Two, four, or eight adjacent squares can be combined to reduce the number of literals in a function. The squares of the top and bottom rows as well as leftmost and rightmost columns may be combined. For example, m_0 and m_2 can be combined, as can m_4 and m_6, m_12 and m_14, m_10 and m_12, m_0 and m_8, m_1 and m_9, m_3 and m_11, and m_1 and m_10. Similarly, the four squares of the corners i.e., the minterms m_0, m_2, m_8, and m_10 can also be combined.

When two adjacent squares are combined, it is called a pair and represents a term with three literals.

Four adjacent squares, when combined, are called a quad and its number of literals is two.

If eight adjacent squares are combined, it is called an octet and represents a term with one literal.

If, in the case all sixteen squares can be combined, the function will be reduced to 1.

**Example 4.5.** Simplify the expression \( F (A, B, C, D) = m_1 + m_5 + m_{10} + m_{11} + m_{12} + m_{13} + m_{15} \).

**Solution.** The Karnaugh map for the above expression is shown in Figure 4.11.

<table>
<thead>
<tr>
<th></th>
<th>C'D</th>
<th>C'D</th>
<th>CD</th>
<th>C'D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A'B'</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A'B</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A'B'</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

From the figure, it can be seen that four pairs can be formed. The simplified expression may be written as, \( F = A'C'D + ABC' + ACD + AB'C \).

Note that the reduced expression is not a unique one, because if pairs are formed in different ways as shown in Figure 4.12, the simplified expression will be different. But both expressions are logically correct.
The simplified expression of the given function as per the Karnaugh map of Figure 4.12 is

\[ F = A'C'D + ABC' + ABD + AB'C. \]

\[
\begin{array}{cccc}
CD' & C'D & CD & CD' \\
A'B' & 1 & & \\
A'B & 1 & & \\
AB & 1 & 1 & 1 \\
AB' & 1 & 1 & 1 \\
\end{array}
\]

Figure 4.12

**Example 4.6.** Simplify the expression \( F(A, B, C, D) = m_7 + m_9 + m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15} \).

**Solution.** The Karnaugh map for the above expression is shown in Figure 4.13.

\[
\begin{array}{cccc}
CD' & C'D & CD & CD' \\
A'B' & & & \\
A'B & & 1 & \\
AB & 1 & 1 & 1 \\
AB' & 1 & 1 & 1 \\
\end{array}
\]

Figure 4.13

Three quads and one pair are formed as shown in the figure.

The simplified expression of the given function is,

\[ F = AB + AC + AD + BCD. \]

**Example 4.7.** Plot the logical expression \( F(A, B, C, D) = ABCD + AB'C'D' + AB'C + AB \) on a four-variable Karnaugh map. Obtain the simplified expression.

**Solution.** To form a Karnaugh map for a logical expression, the function is to be expanded to either canonical SOP form or canonical POS form. The canonical SOP form for the above expression can be obtained as follows.

\[
\begin{align*}
F(A, B, C, D) &= ABCD + AB'C'D' + AB'C + AB \\
&= ABCD + AB'C'D' + AB'C(D + D') + AB(C + C')(D + D') \\
&= ABCD + AB'C'D' + AB'CD + AB(C + C')(D + D') \\
&= ABCD + AB'C'D' + AB'CD + AB'CD' + ABCD + ABC'D + AB'CD' \\
&= ABCD + AB'C'D' + AB'CD + AB'CD' + ABCD + ABC'D + ABC'D' \\
&= \Sigma (8, 10, 11, 12, 13, 14, 15)
\end{align*}
\]

The Karnaugh map for the above expression is shown in Figure 4.14.
Three quads (one of them is a roll-over type formed with first column and fourth column) are formed. The simplified expression is

$$F = AB + AC + AD'.$$

**Example 4.8.** Simplify the expression $F(W, X, Y, Z) = \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$.

**Solution.** The Karnaugh map for the above function is shown in Figure 4.15. One octet and two quads are formed. The simplified expression is

$$F = Y' + W'Z' + XZ.$$

**Example 4.9.** Simplify the expression $F(W, X, Y, Z) = W'XY' + X'YZ' + WXYZ' + WXY'.

**Solution.** To obtain the minterms for the above expression, it needs to be expanded to the canonical SOP form as below.

$$F(W, X, Y, Z) = W'XY' + X'YZ' + WXYZ' + WXY'$$

$$= W'XY'(Z + Z') + X'YZ'(W + W') + WXYZ' + WXY'(Z + Z')$$

$$= W'X'Y'Z + WX'YZ' + WXY'Z + WXYZ' + WXYZ' + WXY'Z' + WXYZ + W'XY'Z'$$

The Karnaugh map for the above function is shown in Figure 4.16. One pair and two quads are formed (one quad consists of the four squares of the corners). The simplified expression is

$$F = X'Y' + X'Z' + WYZ'.$$
SIMPLIFICATION AND MINIMIZATION OF BOOLEAN FUNCTIONS

Figure 4.16

Note that, to form the Karnaugh map above like an expression, it is not always necessary to expand the Boolean expression as described above. For the term $W'X$, the squares $WX'YZ$ and $WX'YZ'$ are marked with 1s. For the term $XYZ'$, the squares $WXY'$ and $WXY'$ are marked with 1s. For the term $WXY'$, the squares $WX'YZ$ and $WXY'$ are marked with 1s. Lastly, the term $WXY'$ is the minterm itself, and is marked with 1.

After forming the Karnaugh map, SOP expression can be realized as above.

**Example 4.10.** Simplify the expression $F (W,X,Y,Z) = \Sigma (3, 4, 5, 7, 9, 13, 14, 15)$.

**Solution.** The Karnaugh map for the above function is shown in Figure 4.17.

Four pairs are formed. It may be noted that one quad can also be formed, but it is redundant as the squares contained by the quad are already covered by the pairs which are essential. The simplified expression may be written as

$$F = W'XY + WYZ + WY'Z + WXY.$$  

Figure 4.17

**Example 4.11.** Simplify the expression $F (W,X,Y,Z) = \Pi (0, 1, 4, 5, 6, 8, 9, 12, 13, 14)$.

**Solution.** The above expression is given in respect to the maxterms. In the Karnaugh map, 0s are to placed instead of 1s at the corresponding maxterm squares. The rest of the squares are filled with 1s.

The Karnaugh map for the above function is shown in Figure 4.18(a). There are two ways to achieve the minimized expression above. One way to is consider the 0s of the Karnaugh map. One octet and one quad has been formed with 0s. As we are considering the 0s, the simplified expression will be,

$$F' = Y' + XZ'.$$
Or, \[ F = (Y' + XZ)' = Y (X' + Z). \]

\[
\begin{array}{c|cccc}
& YZ' & YZ & YZ' & YZ' \\
\hline
WX' & 0 & 0 & 1 & 1 \\
WX & 0 & 0 & 1 & 0 \\
WX & 0 & 0 & 1 & 0 \\
WX' & 0 & 0 & 1 & 1 \\
\end{array}
\]

**Figure 4.18(a)**

The other way to achieve the minimized expression is to consider the 1s of the Karnaugh map as shown in Figure 4.18(b). Two quads are formed considering the 1s.

\[
\begin{array}{c|cccc}
& YZ' & YZ & YZ & YZ' \\
\hline
WX' & 0 & 0 & 1 & 1 \\
WX & 0 & 0 & 1 & 0 \\
WX & 0 & 0 & 1 & 0 \\
WX' & 0 & 0 & 1 & 1 \\
\end{array}
\]

**Figure 4.18(b)**

The minimized expression can be written as
\[
F = YZ + X'Y = Y(X' + Z).
\]

Note that the final expressions are the same in both cases.

**Example 4.12.** Obtain (a) the minimal sum of the products and (b) minimal product of the sums for the function \( F (W,X,Y,Z) = \sum (0, 1, 2, 5, 8, 9, 10). \)

**Solution.**

\[
\begin{array}{c|cccc}
& YZ' & YZ, & YZ & YZ' \\
\hline
WX' & 1 & 1 & 0 & 1 \\
WX & 0 & 1 & 0 & 0 \\
WX & 0 & 0 & 0 & 0 \\
WX' & 1 & 1 & 0 & 1 \\
\end{array}
\]

**Figure 4.19(a)**
(a) The Karnaugh map for the above function is shown in Figure 4.19(a). Two quads
and a pair are formed considering the 1s of the Karnaugh map.

The SOP expression of the above is \( F = X'Y' + XZ' + WY'Z. \)

(b) The Karnaugh map for the above function is shown in Figure 4.19(b). Three quads
are formed considering the 0s of the Karnaugh map.

The POS expression of above function can be derived as,
\[ F' = XZ' + WX + YZ. \]

Or,
\[ F = (X' + Z) (W' + X') (Y' + Z'). \]

4.5 FIVE-VARIABLE KARNAUGH MAPS

Karnaugh maps with more than four variables are not simple to use. The number of cells or
squares becomes excessively large and combining the adjacent squares becomes complex. The
number of cells or squares is always equal to the number of minterms. A five-variable Karnaugh
map contains 2⁵ or 32 cells, which are used to simplify any five-variable logic function. Figures
4.20(a) and 4.20(b) demonstrate the five-variable Karnaugh map and its minterms.
Figures 4.21, 4.22, and 4.23 also demonstrate five-variable Karnaugh maps, if the variables are assigned in different ways. The five-variable Karnaugh maps have properties similar to the two-, three-, or four-variable Karnaugh maps described earlier, i.e., adjacent squares can be grouped together. In addition to those, while making groups or combinations, in Figures 4.20 and 4.21, the 1st column with 4th column, 2nd column with 7th column, and 3rd column with 6th column can be combined together, as there is only one variable which is changing its form for those columns. Similarly, according to Figures 4.22 and 4.23, the 1st row with 4th row, 2nd row with 7th row, and 3rd row with 6th row can be combined together to get the terms of reduced literals.

\[
\begin{array}{cccccc}
A'B'C' & A'B'C & A'BC & A'B'C' & ABC' & ABC \\
C'D' & m_0 & m_4 & m_{12} & m_8 & m_{24} & m_{28} & m_{20} & m_{16} \\
C'D & m_1 & m_5 & m_{13} & m_9 & m_{25} & m_{29} & m_{21} & m_{17} \\
CD & m_3 & m_7 & m_{15} & m_{11} & m_{27} & m_{31} & m_{23} & m_{19} \\
CD' & m_2 & m_6 & m_{14} & m_{10} & m_{26} & m_{30} & m_{22} & m_{18} \\
\end{array}
\]

Figure 4.21

\[
\begin{array}{cccccc}
D'E' & D'E & DE & DE' & A'B' & AB' \\
A'B'C' & m_0 & m_1 & m_3 & m_2 & C'D'E' & m_0 & m_8 & m_{24} & m_{16} \\
A'B'C & m_4 & m_5 & m_7 & m_6 & C'D'E & m_1 & m_9 & m_{25} & m_{17} \\
A'BC & m_{11} & m_{12} & m_{15} & m_{14} & C'D'E & m_3 & m_{11} & m_{27} & m_{19} \\
A'B'C' & m_4 & m_5 & m_7 & m_{10} & C'D'E' & m_2 & m_{10} & m_{26} & m_{18} \\
ABC & m_{24} & m_{26} & m_{27} & m_{26} & CDE' & m_4 & m_{14} & m_{30} & m_{22} \\
ABC & m_{28} & m_{29} & m_{31} & m_{30} & CDE & m_7 & m_{15} & m_{31} & m_{23} \\
AB'C & m_{20} & m_{21} & m_{23} & m_{22} & CDE' & m_5 & m_{13} & m_{29} & m_{21} \\
AB'C' & m_5 & m_{12} & m_{17} & m_{19} & C'D'E' & m_4 & m_{12} & m_{28} & m_{20} \\
\end{array}
\]

Figure 4.22

Figure 4.23

4.6 SIX-VARIABLE KARNAUGH MAPS

Six-variable Karnaugh maps consist of $2^6$ or 64 squares or cells. Similar to the method described above, six-variable Karnaugh maps are formed with 64 minterms as demonstrated in Figure 4.24(a). Figure 4.24(b) also represents six-variable Karnaugh maps when the variables are assigned differently.
Apart from the properties described for two-, three- and four-variable Karnaugh maps that adjacent squares can be grouped together, similar to five-variable maps, the 1st column with 4th column, 2nd column with 7th column, 3rd column with 6th column, 1st row with 4th row, 2nd row with 7th row, and 3rd row with 6th row can be combined together to get the terms of reduced literals.

\[ \begin{array}{cccccccc}
D' & E & F' & D' & E & F & D' & E & F' \\
A'B'C' & m_0 & m_1 & m_4 & m_2 & m_6 & m_7 & m_5 & m_4 \\
A'B'C & m_3 & m_9 & m_{11} & m_{10} & m_{14} & m_{15} & m_{13} & m_{12} \\
A'B'C & m_4 & m_24 & m_25 & m_27 & m_26 & m_30 & m_31 & m_29 & m_{28} \\
A'B'C' & m_26 & m_{17} & m_{19} & m_{18} & m_{22} & m_{23} & m_{21} & m_{20} \\
A'B'C & m_{48} & m_{49} & m_{51} & m_{30} & m_{54} & m_{55} & m_{53} & m_{52} \\
A'B'C' & m_{56} & m_{57} & m_{59} & m_{58} & m_{62} & m_{63} & m_{61} & m_{60} \\
A'B'C & m_{40} & m_{41} & m_{43} & m_{42} & m_{46} & m_{47} & m_{45} & m_{44} \\
A'B'C' & m_{32} & m_{33} & m_{35} & m_{34} & m_{38} & m_{39} & m_{37} & m_{36} \\
\end{array} \]

Figure 4.24(a)

\[ \begin{array}{cccccccc}
A'B'C' & A'B'C & ABC & A'B'C' & ABC' & ABC & AB'C & AB'C' \\
D' & E & F' & D' & E & F & D' & E & F' \\
& m_0 & m_1 & m_{24} & m_{16} & m_{48} & m_{56} & m_{40} & m_{52} \\
D' & E & F' & m_1 & m_9 & m_{25} & m_{17} & m_{49} & m_{57} & m_{41} & m_{53} \\
D' & E & F' & m_3 & m_{11} & m_{27} & m_{19} & m_{51} & m_{59} & m_{43} & m_{55} \\
D' & E & F' & m_2 & m_{10} & m_{26} & m_{18} & m_{50} & m_{58} & m_{42} & m_{54} \\
DEF' & m_4 & m_{14} & m_{30} & m_{22} & m_{54} & m_{62} & m_{46} & m_{38} \\
DEF & m_7 & m_{15} & m_{31} & m_{23} & m_{55} & m_{63} & m_{47} & m_{39} \\
DEF & m_6 & m_{13} & m_{29} & m_{21} & m_{63} & m_{61} & m_{45} & m_{37} \\
DEF' & m_4 & m_{12} & m_{28} & m_{20} & m_{52} & m_{60} & m_{44} & m_{36} \\
\end{array} \]

Figure 4.24(b)

**Example 4.13.** Obtain the minimal sum of the products for the function

\[ F (A, B, C, D, E) = \Sigma (0, 2, 5, 7, 9, 11, 13, 15, 16, 18, 21, 23, 25, 27, 29, 31). \]

**Solution.** The five-variable Karnaugh map for the above function is shown in Figure 4.25.
An octet at the 6th and 7th column with 1st to 4th rows, one octet at the 2nd, 3rd, 6th, and 7th columns with 2nd and 3rd rows, and one quad at the 1st and 4th rows with 1st and 4th columns are formed. The minimized expression can be written as, 
\[ F = CE + BE + B'E'. \]

**Example 4.14.** Obtain the minimal sum of the products for the function 
\[ F(A, B, C, D, E) = \Sigma(0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31). \]

**Solution.** The five-variable Karnaugh map for the function is shown in Figure 4.26.

\[ F = BE + A'B'E' + AD'E. \]

### 4.7 DON'T-CARE COMBINATIONS

In certain digital systems, some input combinations never occur during the process of a normal operation because those input conditions are guaranteed never to occur. Such input combinations are called Don't-Care Combinations. The function output may be either 1 or 0 and these functions are called incompletely specified functions. These input combinations can be plotted on the Karnaugh map for further simplification of the function. The don't-care combinations are represented by \( d \) or \( x \) or \( \Phi \).
When an incompletely specified function, \( i.e. \), a function with don’t-care combinations is simplified to obtain minimal SOP expression, the value 1 can be assigned to the selected don’t care combinations. This is done to form groups like pairs, quads, etc., for further simplification. In each case, choice depends only on need to achieve simplification. Similarly, selected don’t care combinations may be assumed as 0s to form groups of 0s for obtaining the POS expression.

**Example 4.15.** Obtain the minimal sum of the products for the function

\[
F(A, B, C, D) = \Sigma (1,3,7,11,15) + \Phi(0,2,5).
\]

The Karnaugh map for the above function is shown in Figure 4.27.

\[\begin{array}{c|c|c|c|c}
A'B' & CD & CD' & C'D' \\
\hline
A'B & X & 1 & X \\
A'B & X & 1 & \\
AB & & 1 & \\
AB' & & 1 & \\
\end{array}\]

In the Karnaugh map of Figure 4.27, the minterm \( m_0 \) and \( m_2 \), i.e., \( A'B'C'D' \) and \( A'B'CD' \), are the don’t care terms which have been assumed as 1s, while making a quad. The simplified SOP expression of above function can be written as

\[
F = A'B' + CD.
\]

### 4.8 THE TABULATION METHOD

The Karnaugh map method is a very useful and convenient tool for simplification of Boolean functions as long as the number of variables does not exceed four (at the most six). But if the number of variables increases, the visualization and selection of patterns of adjacent cells in the Karnaugh map becomes complicated and difficult. The tabular method, also known as the Quine-McCluskey method, overcomes this difficulty. It is a specific step-by-step procedure to achieve guaranteed, simplified standard form of expression for a function.

The following steps are followed for simplification by the tabular or Quine-McCluskey method.

1. An exhaustive search is done to find the terms that may be included in the simplified functions. These terms are called *prime implicants*.
2. Form the set of prime implicants, essential prime implicants are determined by preparing a prime implicants chart.
3. The minterms that are not covered by the essential prime implicants, are taken into consideration by selecting some more prime implications to obtain an optimized Boolean expression.
### 4.8.1 Determination of Prime Implicants

The prime implicants are obtained by the following procedure:

1. Each minterm of the function is expressed by its binary representation.
2. The minterms are arranged according to increasing index (index is defined as the number of 1s in a minterm). Each set of minterms possessing the same index are separated by lines.
3. Now each of the minterms is compared with the minterms of a higher index. For each pair of terms that can combine, the new terms are formed. If two minterms are differed by only one variable, that variable is replaced by a ‘-’ (dash) to form the new term with one less number of literals. A line is drawn in when all the minterms of one set is compared with all the minterms of a higher index.
4. The same process is repeated for all the groups of minterms. A new list of terms is obtained after the first stage of elimination is completed.
5. At the next stage of elimination two terms from the new list with the ‘-’ of the same position differing by only one variable are compared and again another new term is formed with a less number of literals.
6. The process is to be continued until no new match is possible.
7. All the terms that remain unchecked *i.e.*, where no match is found during the process, are considered to be the prime implicants.

### 4.8.2 Prime Implicant Chart

1. After obtaining the prime implicants, a chart or table is prepared where rows are represented by the prime implicants and the columns are represented by the minterms of the function.
2. Crosses are placed in each row to show the composition of the minterms that makes the prime implicants.
3. A completed prime implicant table is to be inspected for the columns containing only a single cross. Prime implicants that cover the minterms with a single cross are called the essential prime implicants.

The above process to find the prime implicants and preparation of the chart can be illustrated by the following examples.

**Example 4.16.** Obtain the minimal sum of the products for the function

\[ F(A, B, C, D) = \sum (1, 4, 6, 7, 8, 9, 10, 11, 15). \]

**Solution.** The table in Figure 4.28 shows the step-by-step procedure the Quine-McCluskey method uses to obtain the simplified expression of the above function.

Column I consists of the decimal equivalent of the function or the minterms and column II is the corresponding binary representation. They are grouped according to their index *i.e.*, number of 1s in the binary equivalents. In column III, two minterms are grouped if they are differed by only a single variable and equivalent terms are written with a ‘-’ in the place where the variable changes its logic value. As an example, minterms 1 (0001) and 9 (1001) are grouped and written as 1,9 (− 001) and so on for the others. Also, the terms of column II, which are considered to form the group in column III, are marked with ‘√’.
The terms which are not marked with ‘√’ are the Prime implicants. To express the prime implicants algebraically, variables are to be considered as true form in place of 1s, as complemented form in place of 0s, and no variable if ‘−’ appears. Here the prime implicants are $B'C'D$, $A'B'D$, $A'BC$, $BCD$, $ACD$ (from column III), and $AB'$ (from column IV). So the Boolean expression of the given function can be written as

$$ F = AB' + B'C'D + A'B'D + A'BC + BCD + ACD. $$

But the above expression may not be of minimized form, as all the prime implicants may not be necessary. To find out the essential prime implicants, the following steps are carried out. A table or chart consisting of prime implicants and the decimal equivalent of minterms as given in the expression, as in Figure 4.29 is prepared.
In the table, the prime implicants are listed in the 1st column and Xs are placed against the corresponding minterms. The completed prime implicant table is now inspected for the columns containing only a single X. As in Figure 4.29, the minterm 1 is represented by only a single prime implicant B’C’D, and only a single X in that column, it should be marked as well as the corresponding column should be marked. Similarly, the prime implicants AB’ and AB’D’ are marked. These are the essential prime implicants as they are absolutely necessary to form the minimized Boolean expression. Now all the other minterms corresponding to these prime implicants are marked at the end of the columns i.e., the minterms 1, 4, 6, 8, 9, 10, and 11 are marked. Note that the terms ABC, BCD, and ACD are not marked. So they are not the essential prime implicants. However, the minterms 7 and 15 are still unmarked and both of them are covered by the term BCD and are included in the Boolean expression. Therefore, the simplified Boolean expression of the given function can be written as

$$F = AB’ + B’C’D + A’BD’ + BCD.$$ 

The simplified expressions derived in the preceding example are in the sum of products form. The Quine-McClusky method can also be adopted to derive the simplified expression in product of sums form. In the Karnaugh map method the complement of the function was considered by taking 0s from the initial list of the minterms. Similarly the tabulation method or Quine-McClusky method may be carried out by considering the 0s of the function to derive the sum of products form. Finally, by making the complement again, we obtain the simplified expression in the form of product of sums.

A function with don’t-care conditions can be simplified by the tabulation method with slight modification. The don’t-care conditions are to be included in the list of minterms while determining the prime implicants. This allows the derivation of prime implicants with the least number of literals. But the don’t-care conditions are excluded in the list of minterms when the prime implicants table is prepared, because these terms do not have to be covered by the selected prime implicants.

### 4.9 MORE EXAMPLES

**Example 4.17.** Obtain the minimal sum of the products for the function $F (A,B,C,D) = \Sigma (1, 2, 3, 7, 8, 9, 10, 11, 14, 15)$ by the Quine-McClusky method.

**Solution.** The first step is to find out the prime implicants as described by the table in Figure 4.30.

The prime implicants are B’D, B’C, AB’, CD, and AC. The prime implicant table is prepared as in Figure 4.31.

<table>
<thead>
<tr>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal equivalent</td>
<td>Binary equivalent</td>
<td>ABCD</td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SIMPLIFICATION AND MINIMIZATION OF BOOLEAN FUNCTIONS

Figure 4.30

<table>
<thead>
<tr>
<th>Prime Implicants</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>√ B'D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>√ B'C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>√ AB'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>√ CD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>√ AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

All the prime implicants are essential. So the simplified Boolean expression of the given function is

\[ F = B'D + B'C + AB' + CD + AC. \]

Example 4.18. Using the Quine-McClusky method, obtain the minimal sum of the products expression for the function \( F(A, B, C, D) = \Sigma (1, 3, 4, 5, 9, 10, 11) + \Phi (6, 8). \)

Solution. The prime implicants are obtained from the table in Figure 4.32.
<table>
<thead>
<tr>
<th>$I$</th>
<th>$II$</th>
<th>$III$</th>
<th>$IV$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal equivalent</td>
<td>Binary equivalent</td>
<td>$ABCD$</td>
<td>$ABCD$</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 1</td>
<td>√</td>
<td>1,3 00–1</td>
</tr>
<tr>
<td>4</td>
<td>0 1 0 0</td>
<td>√</td>
<td>1,5 0–01</td>
</tr>
<tr>
<td>8</td>
<td>1 0 0 0</td>
<td>√</td>
<td>1,9 –001</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4,6 01–0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8,10 10–0</td>
</tr>
<tr>
<td>3</td>
<td>0 0 1 1</td>
<td>√</td>
<td>3,11 –011</td>
</tr>
<tr>
<td>5</td>
<td>0 1 0 1</td>
<td>√</td>
<td>9,11 10–1</td>
</tr>
<tr>
<td>6</td>
<td>0 1 1 0</td>
<td>√</td>
<td>10,11 101–</td>
</tr>
<tr>
<td>9</td>
<td>1 0 0 1</td>
<td>√</td>
<td>9,11 101–</td>
</tr>
<tr>
<td>10</td>
<td>1 0 1 0</td>
<td>√</td>
<td>9,11 101–</td>
</tr>
<tr>
<td>11</td>
<td>1 0 1 1</td>
<td>√</td>
<td>9,11 101–</td>
</tr>
</tbody>
</table>

Figure 4.32

The prime implicants are $A'C'D, A'BC', A'BD', B'D$, and $AB'$. The prime implicant table is prepared as in Figure 4.33.

<table>
<thead>
<tr>
<th>Prime Implicants</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A'C'D$</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A'BC'$</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A'BD'$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sqrt{B'D}$</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sqrt{AB'}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\sqrt{\checkmark}$</td>
<td>$\checkmark$</td>
</tr>
</tbody>
</table>

Figure 4.33
From the table, we obtain the essential prime implicants $B'D$ and $AB'$. The minterms 4 and 5 are not marked in the table. The term $A'BC'$ is considered, which covers both the minterms 4 and 5. So the simplified Boolean expression for the given function is

$$ F = A'BC' + B'D + AB'. $$

**Example 4.19.** Using the Karnaugh map method obtain the minimal sum of the products and product of sums expressions for the function

$$ F(A,B,C,D) = \Sigma (1, 3, 4, 5, 6, 7, 9, 12, 13). $$

**Solution.** The Karnaugh map for the above function is in Figure 4.34. To obtain the SOP expression, 1s of the Karnaugh map are considered.

The simplified Boolean expression for the function is

$$ F = A'B + BC' + C'D + A'D. $$

To derive the POS expression, the 0s of the Karnaugh map are considered as in Figure 4.35.

From the Karnaugh map we obtain $F' = AC + B'D'$. So the POS expression for the above function is

$$ F = (AC + B'D')' = (AC)' + (B'D')' = (A' + C') \cdot (B + D). $$
Example 4.20. Using the Karnaugh map method obtain the minimal sum of the products and product of sums expressions for the function

\[ F(A, B, C, D) = \Sigma (1, 5, 6, 7, 11, 12, 13, 15). \]

Solution. The Karnaugh map for the above function is in Figure 4.36. To obtain the SOP expression, 1s of the Karnaugh map are considered.

The simplified Boolean expression for the function is

\[ F = A' \overline{C} \overline{D} + A'BC + ABC' + ACD = B(A' \overline{C} + AC') + D(A'C' + AC). \]

\[ = B(\overline{A} \oplus C) + D(A' \oplus C'). \]

To derive the POS expression, the 0s of the Karnaugh map are considered as in Figure 4.37.

From the Karnaugh map we obtain

\[ F' = A'CD' + A'BC + AB'C' + ACD'. \]

So the POS expression for the above function is

\[ F = (A + C + D)(A + B + C')(A' + B + C)(A' + C' + D). \]

Example 4.21. Using the Karnaugh map method obtain the minimal sum of the products expression for the function

\[ F(A, B, C, D) = \Sigma (0, 2, 3, 6, 7) + d (8, 10, 11, 15). \]

Solution. The Karnaugh map for the above function is in Figure 4.38.
The simplified Boolean expression for the function is
\[ F = A'C + B'D'. \]

**Example 4.22.** Using the Karnaugh map method obtain the minimal product of the sums expression for the function given in example 4.21.

**Solution.** To derive the POS expression, the 0s of the Karnaugh map are considered as in Figure 4.39.

The simplified Boolean expression for the function is
\[ F' = A + C'D + BC'. \]

So \[ F = A' (C + D') (B' + C). \]

**Example 4.23.** Using the Karnaugh map method obtain the minimal sum of the products expression for the function
\[ F(A,B,C,D) = \Sigma (1, 5, 7, 13, 14, 15, 17, 18, 21, 22, 25, 29) + d(6, 9, 19, 23, 30). \]

**Solution.** The Karnaugh map for the above function is in Figure 4.40.
The simplified Boolean expression for the function is
\[ F = D'E + A'CD + AB'D. \]

**Example 4.24.** Using the Quine-McClusky method obtain the minimal sum of the products expression for the function
\[ F(A,B,C,D,E) = \Sigma (0, 2, 3, 5, 7, 9, 11, 13, 14, 16, 18, 24, 26, 28, 30). \]

**Solution.** The prime implicants are obtained from the table in Figure 4.41.

<table>
<thead>
<tr>
<th>I</th>
<th>Decimal equivalent</th>
<th>Binary equivalent</th>
<th>ABCDE</th>
<th>ABCDE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A  B  C  D  E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0  0  0  0  0</td>
<td>√</td>
<td>0,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>000–0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0  0  0  1  0</td>
<td>√</td>
<td>0,16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>–0000</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>0  0  0  0  0</td>
<td>√</td>
<td>16,18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24,26</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0  0  0  1  1</td>
<td>√</td>
<td>2,3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0001–</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0  0  1  0  1</td>
<td>√</td>
<td>2,18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>–0010</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0  1  0  0  1</td>
<td>√</td>
<td>16,18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100–0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0  0  0  1  0</td>
<td>√</td>
<td>16,24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1–000</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1  1  0  0  0</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0  0  1  1  1</td>
<td>√</td>
<td>3,7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00–11</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0  1  0  1  1</td>
<td>√</td>
<td>3,11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0–011</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0  1  1  0  1</td>
<td>√</td>
<td>5,7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>001–1</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0  1  1  1  0</td>
<td>√</td>
<td>5,13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0–101</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>1  1  0  1  0</td>
<td>√</td>
<td>9,13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01–01</td>
</tr>
</tbody>
</table>
The prime implicant table is prepared as in Figure 4.42.

The essential prime implicants are $B'C'E'$, $AB'E'$, $A'C'DE$, $A'BD'E$, and $BCDE'$, as each of them represent at least one minterm which is not represented by any of the other prime implicants. The term $A'BC'E$ may be considered to include minterms 5 and 7.

\[
\begin{array}{cccccccccccc}
\hline
\text{Prime Implicants} & 0 & 2 & 3 & 5 & 7 & 9 & 11 & 13 & 14 & 16 & 24 & 26 & 28 & 30 \\
\hline
\vee B'C'E' & X & X & X & X & X & X & X & X & X & X & X & X & X & X \\
\vee A'C'E' & X & X & X & X & X & X & X & X & X & X & X & X & X & X \\
\vee A'BD'E & X & X & X & X & X & X & X & X & X & X & X & X & X & X \\
\vee BCDE' & X & X & X & X & X & X & X & X & X & X & X & X & X & X \\
\hline
\end{array}
\]

The simplified expression of the function is

\[ F = B'C'E + ABE' + A'C'DE + A'BD'E + BCDE'. \]

### 4.10 VARIABLE-ENTERED KARNAUGH MAPS

There is another method of simplification of Boolean functions which is not widely used, but certainly has some importance from an academic point of view. Earlier in this section we have already discussed five-variable and six-variable Karnaugh maps, which are a little complex and difficult while making pairs, quads, or octets. *Variable-entered Karnaugh maps* may be used in cases where the number of variables exceeds four. It is the useful extension of normal Karnaugh maps as discussed earlier. In variable-entered Karnaugh maps, one or more Boolean variables can be used as map entries along with 1s, 0s, and don’t-cares. The variables associated with the entries in these maps are called *map-entered variables.*
The significance of variable-entered maps is that they provide for map compression. Normally an $n$th order Karnaugh map is associated with the Boolean functions of $n$ number of variables. However, if one of the Boolean variables is used as a map-entered variable, the Karnaugh map will be reduced to an order of $n-1$. In general, if the number of map-entered variables is $m$ of the total number of variables $n$, the Karnaugh map of the order of $n-m$ will suffice to associate in the necessary simplification process. ($n > m$).

A useful application for variable-entered maps arises in the problems that have infrequently appearing variables. In such situations it is convenient to have the functions of the infrequently appearing variables as the entries within the map, allowing a high-order Boolean function to be represented by a low-order map.

### 4.10.1 Construction of Variable-entered Maps

To understand the construction of variable-entered Karnaugh maps, consider the generic truth table in Figure 4.43, where the functional value for row $i$ is denoted by $F_i$. From the truth table the Karnaugh map is constructed in Figure 4.44. The entries within the cells are the $F_i$'s, which, in turn, correspond to the 0s, 1s, and don’t-cares that normally appear in the last column of the truth table.

Alternatively, the generic minterm canonical formula for the truth table of Figure 4.43 can be written as

$$F(A, B, C) = F_0 \cdot A'B'C' + F_1 \cdot A'B'C + F_2 \cdot A'BC' + F_3 \cdot A'BC + F_4 \cdot ABC' + F_5 \cdot ABC + F_6 \cdot ABC' + F_7 \cdot ABC.$$ 

Using Boolean algebra, this expression can be modified as

$$F(A, B, C) = A' \cdot B' \cdot (F_0 \cdot C' + F_1 \cdot C) + A' \cdot B \cdot (F_2 \cdot C' + F_3 \cdot C) + AB' \cdot (F_4 \cdot C' + F_5 \cdot C) + AB \cdot (F_6 \cdot C' + F_7 \cdot C).$$

Since this equation consists of four combinations of $A$ and $B$ variables in their complemented and uncomplemented form, a map can be prepared from the equation by using $A$ and $B$ variables as the row and column labels and the terms within the parentheses as cell entries. This is illustrated in Figure 4.45. Now too may noticed that the Karnaugh
map of order three has been reduced to the order of two. Hence, the map compression is achieved. In this example, A and B are called map variables, and C is treated as a map-entered variable as it is appearing in the cell entries.

\[
\begin{array}{ccc}
B' & B \\
\end{array}
\]

Figure 4.45

The above expression may also be written as

\[
\]

Here A and C are the map variables and B may be used as a map-entered variable, if the Karnaugh map, according to this expression, is to be formed.

Also, the same equation may be expressed as

\[
\]

Where B and C are the map variables, and A is the map-entered variable for its Karnaugh map.

It may be noted that the Karnaugh map for the above expression can be further compressed in respect of its order, if the expression is rewritten as below.

\[
\]

In this case, A is the map variable, and B and C are the map-entered variable. The Karnaugh map can be constructed as in Figure 4.46.

\[
\begin{array}{ccc}
A' & A \\
\end{array}
\]

Figure 4.46

Thus higher order Karnaugh maps can be reduced to lower order maps using one or more variables within the cells. However, the degree of difficulty in interpreting compressed maps lies in the complexities of the entered function. Alternatively, variable-entered maps may be derived by partitioning of the truth table. The truth table of Figure 4.43 may be reconstructed like Figure 4.47, where rows are paired such that they correspond to equal values of A and B. The two possible values of the C variable appear within each pair and the last column of the truth table consists of single variable functions corresponding to C. Since within each of the partitions A and B possess the same value, the partitioned truth table can now be used to form the variable-entered map. This means that for each combination of A and B variables, the cell entries become the function of the C variable.

We have so far discussed the map entries as single-variable functions. The map entries functions may be generalized as \( F_i.V' + F_j.V \) where \( F_i \) and \( F_j \) are the functions \( F_0 \), \( F_1 \), ..., \( F_7 \). Now, with the assumption of completely specified Boolean function and truth table,
i.e., there are no don't-care conditions, it may be noted that values of \( F_i \) and \( F_j \) are restricted to only 0s and 1s. The table in Figure 4.48 illustrates the four possible value assignments to \( F_i \) and \( F_j \) and corresponding map entries in respect to \( V \), where \( V \) is generalized as a map-entered variable. In addition to these, the don't-care conditions are to be considered as map entries wherever necessary.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>( F_i )</th>
<th>( F_j ).V + ( F_j.V )</th>
<th>Map entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( F_0 )</td>
<td>( F_0.C' + F_1.C )</td>
<td>0 0 0 + 0 = 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( F_1 )</td>
<td>( F_2 .C' + F_3 .C )</td>
<td>0 1 0 + ( V ) = ( V )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( F_2 )</td>
<td>( F_4 .C' + F_6 .C )</td>
<td>1 0 ( V' + 0 ) = ( V' )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( F_3 )</td>
<td>( F_5 .C )</td>
<td>1 1 ( V' + V = 1 )</td>
</tr>
</tbody>
</table>

Now let us consider a practical example of a Boolean function according to the truth table of Figure 4.49. The Boolean expression of the function is

\[
F(A, B, C) = A'B'C' + A'B'C + A'BC' + AB'C.
\]

The expression may be modified as

\[
F(A, B, C) = A'B'(C' + C) + A'BC' + AB'C
\]
\[
= A'B'(1) + A'B(C') + AB'(C).
\]
The expressions within the parentheses correspond to the single variable map entries. With C as the map-entered variable, the Karnaugh map is constructed as in Figure 4.50.

Another example may be cited here for the Boolean expression as below.

\[ F(X,Y,A,B,C) = X'A'B'C' + A'B'C + A'BC + YABC' + ABC \]

In this expression, variables X and Y appear infrequently and may be considered as map-entered variables. The Karnaugh map is constructed as in Figure 4.51.

![Figure 4.51](image)

**4.10.2 Formation of Minimal Sums and Products with Map-entered Variables**

Just like normal Karnaugh maps, minimal sum terms or product terms can be obtained from variable-entered Karnaugh maps by grouping and subgrouping of cells. However, while obtaining a minimal sum, it is necessary to form groups involving the map-entered variable in addition to 1s. Similarly, the map-entered variables are to be considered for formation of group in addition to 0s, while obtaining minimal products. We know from the Boolean algebra that \( V + V' = 1 \), where \( V \) is the generalized notation of a map-entered variable. Therefore a 1 in a cell can be grouped with \( V \) as well as \( V' \). This may be clarified by considering the Karnaugh map of Figure 4.47. The 1 at cell \( A'B' \) may be represented as \( C + C' \) (here the map-entered variable is \( C \)). The map can be reconstructed as in Figure 4.52(a). Now the 1 can be grouped with \( C \) as well as \( C' \), if they are placed at adjacent cells. This has been demonstrated in Figure 4.52(b). The minimal terms are obtained as \( B'C \) (where two Cs are grouped) and \( A'C' \) (for the group of two Cs).

![Figure 4.52(a)](image)

![Figure 4.52(b)](image)

![Figure 4.52(c)](image)
Therefore, the final minimal sum of the products expression is
\[ F(A, B, C) = B' C + A' C'. \]

The same technique may be applied to derive the minimal product of the sums expression, where groups or subgroups are formed with 0s as from Boolean algebra \( VV' = 0 \). This has been shown in Figure 4.52(c). Therefore, the product of the sums expression of function is
\[ F'(A, B, C) = (B + C'). (A + C). \]

This may be summarized to a two-step procedure for obtaining minimal sums for completely specified Boolean functions from a variable-entered map as stated below.

1. Consider each map entry having literal \( V \) and \( V' \). Form the groups with the maximum number of elements involving the literal \( V \) using cells containing 1s as don't-care cells and \( V' \) as 0-cells. Next form the groups involving the literal \( V' \) using cells containing 1s as don't-care cells and the cells containing literal \( V \) as 0-cells.
2. After formation of groups involving \( V \) and \( V' \), form the groups of cells containing 1s but not completely covered in step 1. One approach for doing this is to let all the cells containing the literals \( V \) and \( V' \) become 0 and all 1s that were completely covered at step 1 become don't-care cells. Another way for the not completely covered 1-cells is to use \( V \)-cells and \( V' \)-cells from step 1 that ensures that all the 1-cells are completely covered.

**Example 4.25.** Let us consider the Boolean function that has the Karnaugh map as per figure 4.53(a).

At the first step, the cell with D is grouped with adjacent 1 as in Figure 4.53(b). For this the minimal term is obtained as \( A'B'D \). Next \( D' \) is grouped with adjacent 1s as a quad, which is demonstrated in Figure 4.53(c). The minimal term is obtained as \( CD' \). Now, note that the 1-cell at \( A'B'C \) is grouped with \( D \) as well as \( D' \). Therefore, it can be stated that this 1-cell is completely covered. But 1s at \( ABC \) and \( A'B'C \) are considered in the grouping of \( D' \) only, so these 1-cells are not completely covered and they are grouped again separately as shown in Figure 4.53(d). The minimal term obtained is \( BC \). So, considering all the minimal terms obtained, the final expression is derived as
F(A,B,C,D) = A'B'D + CD' + BC.

In the above example, the minimal terms are determined in three steps. This is done for illustration purposes and in practice minimal terms may be determined in a single-step only.

**Example 4.26.** Find the sum of the product expression for the Boolean function whose variable-entered Karnaugh map is shown in Figure 4.54(a).

\[
F (A,B,C,D) = A'B'D + CD' + BC.
\]

**Solution.** The required sum of the product expression is

\[ F = A'D' + C'D' + B'D' + ACD. \]

The 1-cell has been grouped with D as well as D' to make it completely covered.

### 4.10.3 Variable-entered MAPS with Don’t-care Conditions

So far variable-entered mapping has been discussed for the Boolean functions that are completely specified. However, incompletely specified Boolean functions *i.e.*, those having don’t-care conditions, commonly occur in logic design process. It is possible to generalize the construction and reading of variable-entered maps with don’t-care conditions.

Let us assume again that map entries in a variable-entered map correspond to single-variable functions. Previously it was shown that the map entries functions may be generalized as \( F_i \cdot V' + F_j \cdot V \) where \( F_i \) and \( F_j \) are the functions \( F_0, F_1, \ldots \). \( F_i \) and \( F_j \) may have the values of 0, 1, or don’t-care. A table in Figure 4.55 lists the nine possible assignments to \( F_i \) and \( F_j \), the evaluation of \( F_i \cdot V' + F_j \cdot V \) for each case, and the corresponding entries for a variable-entered map. The don’t-cares are denoted by X in the expressions.

<table>
<thead>
<tr>
<th>( F_i )</th>
<th>( F_j )</th>
<th>( F_i \cdot V' + F_j \cdot V )</th>
<th>Map entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.V' + 0.V = 0+0 = 0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.V' + 1.V = 0+1 = 1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>0.V' + X.V = 0+X.V = X.V</td>
<td>V, 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1.V' + 0.V = V'+0 = V'</td>
<td>V'</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.V' + 1.V = V'+V = 1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>1.V' + X.V = V'+X.V</td>
<td>V', 1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>X.V' + 0.V = X.V'+0 = X.V'</td>
<td>V', 0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>X.V' + 1.V = X.V'+V</td>
<td>V, 1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X.V' + X.V</td>
<td>X</td>
</tr>
</tbody>
</table>

**Figure 4.55**
It may be noted from the table in Figure 4.55 that there are some double entries in the map, which signifies that these cells can have both values. The first part of the double entry is referred to as the literal part and the second part is the constant part. The process of reading a variable-entered map with don't-care conditions is more complex, since the double entry cells in the map provide flexibility. Again, two-step process may be adopted for the formation of groups and subgroups to obtain the minimal terms.

1. Form an optimal collection or group for all entries that consist of only a single literal, *i.e.*, V or V', using 1s, Xs and double entries having a 1 constant part as don't-cares. In addition, double entries having a 0 constant part can be used as don't-cares for the formation of groups that agree with the literal part of that double entry.

2. Form a step 2 as follows,
   (a) Replace the single literal entries, *i.e.*, V and V' by 0.
   (b) Retain the single 0 and X entries.
   (c) Replace each single 1 entry with an X, if it was completely covered in step 1, otherwise retain a single 1 entry.
   (d) Replace the double entries having a 0 as a constant part, *i.e.*, V,0 and V',0 as 0.
   (e) Replace each double entry having a 1 as a constant part by X, if the cell was used in step 1 to form at least one group agreeing with the literal part, otherwise replace the double entry having a 1 constant part by a 1. (It should be noted that the second case corresponds to the cell not being covered at all or only used in association with the complement of the literal part of a double entry).

The resulting step 2 map only has 0, 1, and X entries. Minimal terms are to be obtained considering the X-cells, also. This may be illustrated by considering the practical example that follows.

**Example 4.27.** Consider the Boolean function

\[ F(A,B,C,D) = \sum(3, 5, 6, 8, 9, 10) + \Phi(4, 11, 12, 14, 15). \]

The corresponding truth table is presented in Figure 4.56. Variables A, B, and C are considered as map variables, and D is considered as a map-entered variable. In the truth table, columns are provided for the function \( F_i.V + F_i.V \) and map-entry values. The variable-entered Karnaugh map is shown in Figure 4.57(a). Figure 4.57(b) demonstrates how map entries are grouped at step 1. Here, one D entry is grouped with 1, X, and (D',1), as (D',1) may be interpreted as don't-care while grouping with D. Thus the minimal term is formed as CD.

At the next step, replace (D',1) at the cell position AB'C with 1 as this was once grouped with D. Replace (D,1) of cell position A'BC' with 1 and (D',0) of cell position ABC with 0. The Karnaugh map is reconstructed with changes as shown in Figure 4.57(c). Minimal terms are obtained from this map as AB' and A'B.

So the final expression for the given Boolean function can be written as

\[ F(A,B,C,D) = CD + AB' + A'B. \]
### Simplification and Minimization of Boolean Functions

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X.D’+D</td>
<td>D,1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>D’+D</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D’+D</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>D’+X.D</td>
<td>D’,1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X.D’+0</td>
<td>D’,0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X.D’+X.D</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Figure 4.56**

- **Figure 4.57(a)**

- **Figure 4.57(b)**

- **Figure 4.57(c)**
Example 4.28. Find the Boolean expression for the following function using variable-entered map technique.

\[ F(A,B,C,D) = \Sigma (0, 4, 5, 6, 13, 14, 15) + \Phi (2, 7, 8, 9) \]

Solution. Let A, B, and C be map variables and D be a map-entered variable. The truth table with map entry is prepared as in Figure 4.58(a). The Karnaugh map according to the truth table is formed in Figure 4.58(b). At first step, group formation consisting of single literal is done. The minimal terms thus obtained are \( A'D' \) and \( BD \). (\( D', 1 \) may be considered as don't-care while grouped with \( D \), as once it has been considered as \( D' \).)

At the next step, the Karnaugh map is reconstructed in Figure 4.58(c). Here the single literal entries \( D \) and \( D' \) are replaced by 0s as they are already used. (\( D', 0 \)) at cell position \( A'B'C \) is replaced by 0. (\( D', 1 \)) at cell position \( A'B'C \) is replaced by \( X \) as this has been grouped with both \( D \) as well as \( D' \) and similarly 1 at cell position \( A'B'C \) replaced by \( X \) as this also has been grouped with \( D \) as well as \( D' \). The minimal term obtained from this map is \( BC \).

Therefore, the final expression of the given Boolean function may be derived as

\[ F(A,B,C,D) = A'D' + BD + BC. \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>F</th>
<th>( F', V' + F, V )</th>
<th>Map entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( D'0 )</td>
<td>D'</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( D'0 )</td>
<td>D',0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( X.D'0 )</td>
<td>D',0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( D'+D )</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( D'+D )</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( D'+X.D )</td>
<td>D',1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>( D'+X.D )</td>
<td>D',1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>( X.D'+X.D )</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>( X.D'+X.D )</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0+0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( 0+0 )</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( 0+D )</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( 0+D )</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( D'+D )</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( D'+D )</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.58(a)
The above examples demonstrate how a single variable is used as a map-entered variable. However, more than one variable may used as map-entered variables. But these increase the complexities and difficulties in the simplification process.

**4.11 CONCLUDING REMARKS**

Two methods of Boolean function simplification are shown in this chapter. The criterion for simplification is to minimize the number of literals in the sum of products or product of sums forms. Both the map and the tabulation methods are restricted in their capabilities as they are useful for simplifying the Boolean functions expressed in the standard forms. Although this is a disadvantage, it is not very critical, as most of the applications prefer the standard forms over any other form. The gate implementation of standard expressions requires no more than two levels of gates. Expressions not in the standard forms are implemented with more than two levels.

It should be observed that the reflected code sequence chosen for the maps is not unique. It is possible to draw the maps assigning different code sequences to rows and columns keeping reflected code sequence in mind. This is already shown in this chapter. Also, the simplified expression for a function may not be unique, if pairs, quads, etc., are considered differently. The map method is preferable because of its simplicity when the number of variables is restricted to four, at the most five. For more than five variables, grouping of binary sequences leads to confusion and error.

The tabulation method has the distinct advantage at this point, as a step-by-step procedure is followed to minimize the literals. Moreover, this formal procedure is suitable for computer mechanization. But the tabulation process always starts with the minterm list of the function. If the function is not in this form, it is to be converted and the list of minterms is to be prepared.

In this chapter, we have considered the simplifications of functions with many input variables and a single output. However, some digital circuits have more than one output. Such circuits with multiple outputs may sometimes have common terms among the various functions which can be utilized to form common gates during the implementation. This results in further simplification which is not found in the simplification process if done separately. There exists an extension of the tabulation process for multiple-output functions. However, the method is too specialized and very tedious for human manipulation.

**REVIEW QUESTIONS**

1. What are the don't-care conditions?
2. Explain the terms (a) prime implicant, and (b) essential prime implicant.
3. What are the advantages of the tabulation method?
4.4 Draw a Karnaugh map for a four-variable Ex-OR function and derive its expression.

4.5 How does a Karnaugh map differ from a truth table?

4.6 What kind of network is developed by sum of the products?

4.7 Using a Karnaugh map, simplify the following functions and implement them with basic gates.

(a) \( F(A, B, C, D) = \Sigma (0, 2, 3, 6, 7, 8, 10, 11, 12, 15) \)

(b) \( F(A, B, C, D) = \Sigma (0, 2, 3, 5, 7, 8, 13) + d(1, 6, 12) \)

(c) \( F(A, B, C, D) = \Sigma (1, 7, 9, 10, 12, 13, 14, 15) + d(4, 5, 8) \)

(d) \( F(A, B, C, D) = \pi (0, 8, 10, 11, 14) + d(6) \)

(e) \( F(A, B, C, D) = \pi (2, 8, 11, 15) + d(3, 12, 14) \)

(f) \( F(W, X, Y, Z) = \pi (0, 2, 6, 11, 13, 15) + d(1, 9, 10, 14) \)

4.8 Prepare a Karnaugh map for the following functions.

(a) \( F = ABC + ABC + B'C' \)

(b) \( F = A + B + C' \)

(c) \( Y = AB + B'CD \)

4.9 Using the Karnaugh map method, simplify the following functions, obtain their sum of the products form, and product of the sums form. Realize them with basic gates.

(a) \( F(W, X, Y, Z) = \Sigma (1, 3, 4, 5, 6, 7, 9, 12, 13) \)

(b) \( F(W, X, Y, Z) = \Sigma (1, 5, 6, 7, 11, 12, 13, 15) \)

4.10 Determine the don’t-care conditions for the Boolean expression \( BE + B'D'E' \), which is the simplified version of the expression \( AB'E + BCDE + BC'D'E + A'B'DE' + B'C'DE' \).

4.11 Obtain the sum of the products expressions for the following functions and implement them with NAND gates as well as NOR gates.

(a) \( F = \Sigma (1, 4, 7, 8, 9, 11) + d(0, 3, 5) \)

(b) \( F = \Sigma (0, 2, 3, 5, 6, 7, 8, 9) + \phi(10, 11, 12, 13, 14, 15) \)

4.12 A combinational switching network has four inputs \( A, B, C, \) and \( D \), and one output \( Z \). The output is to be 0, if the input combination is a valid Excess-3 coded decimal digit. If any other combinations of inputs appear, the output is to be 1. Implement the network using basic gates.

4.13 Design a circuit similar to problem 4.7 for BCD digits.

4.14 Using the Quine-McCluskey method obtain all the prime implicants, essential prime implicants, and minimized Boolean expression for the following functions.

(a) \( F(A, B, C, D, E) = \Sigma (4, 5, 6, 7, 9, 10, 14, 19, 26, 30, 31) \)

(b) \( F(A, B, C, D) = \Sigma (7, 9, 12, 13, 14, 15) + d(4, 11) \)

(c) \( F(A, B, C, D, E) = \Sigma (1, 3, 6, 10, 11, 12, 14, 15, 17, 19, 20, 22, 24, 29, 30) \)

(d) \( F(A, B, C, D) = \Sigma (4, 5, 8, 9, 12, 13) + d(0, 3, 7, 10, 11) \)

4.15 Determine the simplified expression for each of the following functions using variable-entered maps where \( A, B, \) and \( C \) are map variables.

(a) \( F(A, B, C, D) = \Sigma (2, 3, 5, 12, 14) + d(0, 4, 8, 10, 11) \)

(b) \( F(A, B, C, D) = \Sigma (1, 5, 6, 7, 9, 11, 12, 13) + d(0, 3, 4) \)

(c) \( F(A, B, C, D) = \Sigma (1, 5, 7, 10, 11) + d(2, 3, 6, 13) \)

(d) \( F(A, B, C, D) = \Sigma (5, 6, 7, 12, 13, 14) + d(3, 8, 9) \)