

3

INTRODUCTION

- The minimization of combinational expression is considered to be one of the major steps in the digital design process.
- This emphasis on minimization stems from the time when logic gates were very expensive and required a considerable amount of physical space and power.
- However with the advent of Integrated circuits (SSI, MSI, LSI and VLSI) the traditional minimization process has lessened somewhat, there is still a reasonable degree of correlation between minimizing gate count and reduced package count.
- It is very easy to understand that the complexity of logical implementation of a Boolean function is directly related to the complexity of algebraic expression from which it is implemented.
- Although the truth table representation of a Boolean function is unique, but when expressed algebraically, it can appear in many different forms.
- Because of the reason mentioned above, the objective of this chapter is to develop an understanding of how modern reduction techniques have evolved from the time consuming mathematical approach (Theorem reduction) to quick graphical techniques called ‘mapping’ and ‘tabular method’ for large number of variable in the combinational expression.

MINIMIZATION USING POSTULATES AND THEOREM OF BOOLEAN ALGEBRA

- The ones of major interest are theorems:

6 (a) $A + AB = A$	(b) $A(A + B) = A$	Absorption
7 (a) $A + A'B = A + B$	(b) $A(A' + B) = AB$	
8 (a) $AB + AB' = A$	(b) $(A + B)(A + B') = A$	Logic Adjacency
- Theorem 6 and 7 have special significance when applied to expression in standard form, whereas theorem 8 is of particular importance in simplifying canonical form expression.
- Theorem 6 can be applied only when an expression is in a ‘standard form’, that is, one that has at least one term which is not a MIN or MAX term.

Example 1

$$F = CD + AB'C + ABC' + BCD$$

$$\begin{array}{ccc} \uparrow & \text{---} & \uparrow \\ & \text{Thm 6} & \therefore \\ & = CD + AB'C + ABC' \end{array}$$

Select one of the smaller terms and examine the larger terms which contain this smaller term.

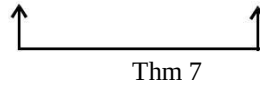
— for application of Theorem 7, the larger terms are scanned looking for of application the smaller in its complemented form.

Example 2

$$F = AB + BEF + A'CD + B'CD$$

$$\triangleright AB + BEF + CD (A' + B')$$

$$\triangleright AB + BEF + CD (AB)' \rightarrow \text{Using Demorgan's Theorem}$$



$$= AB + BEF + CD \quad \therefore$$

— Theorem 8 is the basis of our next minimization technique i.e., Karnaugh map method. It has a word statement—‘If any two terms in a canonical or standard form expression vary in only one variable, and that variable in one term is the complement of the variable in the other term then of the variable is superfluous to both terms.

Example 3

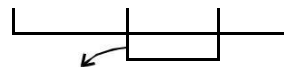
$$F = A'B'C' + A'B'C + ABC' + AB'C$$



$$F = A'B' + AC$$

Example 4

$$F = A'B'C' + AB'C' + ABC' + A'B'C$$



$$= A'B' + AC'$$

- By now it should become obvious that another techniques is needed because this tech-niques backs specific rules to predict each succeeding step in manipulative process.
- Therefore, if we could develop some graphical technique whereby the application of
- ‘Theorem 8’ the logical adjacency theorem is made obvious and where the desired grouping could be plainly displayed, we would be in mind better position to visualize the proper application of theorem.

MINIMIZATION USING KARNAUGH MAP (K-MAP) METHOD

- In 1953 Maurice Karnaugh developed K-map in his paper titled ‘The map method for synthesis of combinational logic circuits.
- The map method provides simple straight forward procedure for minimizing Boolean functions that may be regarded as pictorial form of truth table. K-map orders and displays the minterms in a geometrical pattern such that the application of the logic adjacency theorem becomes obvious.
 - The K-map is a diagram made up of squares. Each square represents one minterm.
 - Since any function can be expressed as a sum of minterms, it follows that a Boolean function can be recognized from a map by the area enclosed by those squares. Whose minterms are included in the operation.
 - By various patterns, we can derive alternative algebraic expression for the same operation, from which we can select the simplest one. (One that has minimum member of literals).

Now, let us start with a two variable K map.

Two and Three Variable K Map

- If we examine a two variable truth table, Fig. 3.1(a) we can make some general obser-vations that support the geometric layout of K Map shown in Fig. 3.1(b).

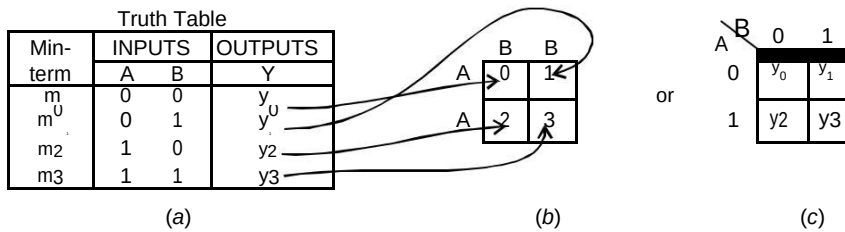


Fig. 3.1 (a) (b) and (c)

- The four squares (0, 1, 2, 3) represent the four possible combinations of A and B in a two variable truth table. Square 1 in the K-map; then, stands for $A'B'$, square 2 for $A'B$, and so forth.
- The map is redrawn in Fig. 3.1(c) to show the relation between the squares and the two variables.
- The 0's and 1's marked for each row and each column designate the values of variables A and B respectively.
- A appears primed in row 0 and unprimed in row 1.
- Similarly B appears primed in column 0 and unprimed in column 1.

Now let us map a Boolean function $Y = A + B$.

Method I—(i) Draw the truth table of given function. [Fig. 3.2(a)]

Min-term	A	B	Y
m_0	0	0	0
m_1	0	1	1
m_2	1	0	1
m_3	1	1	1

Fig. 3.2(a)

(ii) Draw a two variable K map and fill those squares with a 1 for which the value of minterm in the function is equal to 1. [Fig. 3.2(b)]



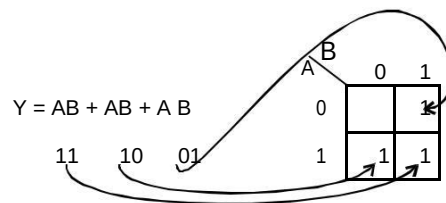
Fig. 3.2(b)

The empty square in the map represents the value of minterms [m_0 (or $A'B'$)] that is equal to zero in the function. Thus, actually this empty square represents zero.

Method II—(i) Find all the minterms of the function $Y = A + B$.

$$\begin{aligned}
 Y &= A + B = A(B + B') + B(A + A') = \\
 &AB + AB' + AB + A'B \\
 Y &= AB + AB' + A'B
 \end{aligned}$$

(ii) Draw a two variable K map using this sum of minterms expression.



— These K map, is nothing more than an interesting looking Truth-Table, and it simply provide a graphical display of ‘implicants’ (minterms) involved in any SOP canonical or stand-ard form expression.

- Now examine a three variable truth table shown in Fig. 3.3 (a).
Truth Table

Min term	INPUTS			OUTPUT
	A	B	C	Y
m_0	0	0	0	y_0
m_1	0	0	1	y_1
m_2	0	1	0	y_2
m_3	0	1	1	y_3
m_4	1	0	0	y_4
m_5	1	0	1	y_5
m_6	1	1	0	y_6
m_7	1	1	1	y_7

Fig. 3.3 (a)

- Here we need a K-map with 8 squares represent all the combination (Minterms) of input variables A, B and C distinctly. A three-variable map is shown in Fig. 3.3 (b).

A \ BC	00	01	11	10
0	m0	m1	m3	m2
1	m4	m5	m7	m6

Fig. 3.3 (b)

- It is very important to realize that in the three variable K-map of Fig. 3.3 (b), the minterms are not arranged in a binary sequence, but similar to ‘Gray Code’ sequence.
- The gray code sequence is a unit distance sequence that means only one bit changes in listing sequence.
- Our basic objective in using K-map is the simplify the Boolean function to minimum number of literals.
- The gray code sequencing greatly helps in applying. ‘Logic Adjacency theorem’ to adjacent squares that reduces number of literals.

- The map in Fig. 3.3 (b) is redrawn in Fig. 3.3 (c) that will be helpful to make the pictures clear.

		BC			
		00	01	11	10
A	0	A B C	A B C	A B C	A B C
	1	AB C	AB C	ABC	ABC

Fig. 3.3 (c)

- We can see that any two adjacent squares in the map differ by only one variable, which is primed in one square and unprimed in other.
- For example $m_3 (A'BC)$ and $m_7 (ABC)$ are two adjacent squares. Variable A is primed in m_3 and unprimed in m_7 , whereas the other two variables are same in both squares. Now applying 'logic adjacency' theorem, it is simplified to a single AND term of only two literals. To clarify this, consider the sum of m_3 and $m_7 \rightarrow m_3 + m_7 = A'BC + ABC = BC(A + A') = BC$.

Boolean Expression Minimization Using K-Map

- Construct the K-map as discussed. Enter 1 in those squares corresponding to the minterms for which function value is 1. Leave empty the remaining squares.
- *Now in following steps the square means the square with a value 1.*
- Examine the map for squares that can not be combined with any other squares and form group of such single squares.
- Now, look for squares which are adjacent to only one other square and form groups containing only two squares and which are not part of any group of 4 or 8 squares. A group of two squares is called a *pair*.
- Next, group the squares which result in groups of 4 squares but are not part of an 8-squares group. A group of 4 squares is called a *quad*.
- Group the squares which result in groups of 8 squares. A group of 8 squares is called *octet*.
- Form more pairs, quads and outlets to include those squares that have not yet been grouped, and use only a minimum no. of groups.
- There can be overlapping of groups if they include common squares.
 - Omit any redundant group.
 - Form the logical sum of all the terms generated by each group.

Using Logic Adjacency Theorem we can conclude that,

- a group of two squares eliminates one variable,
- a group of four squares eliminates two variable and a group of eight squares eliminates three variables.

Now let us do some examples to learn the procedure.

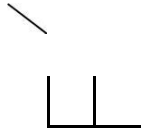
Example 1. Simplify the boolean for $F = AB + AB' + A'B$. Using two variable K-map.

This function can also be written as

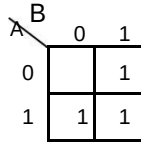
$$F(A, B) = \Sigma(1, 2, 3)$$

Solution. Step 1. Make a two variable K-map and enter 1 in squares corresponding to minterms present in the expression and leave empty the remaining squares.

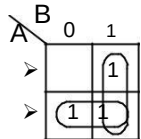
A B	0	1
0	m0	m1
1	m2	m3



Step 2. There are no 1's which are not adjacent to other 1's. So this step is discarded.



Step 3. m_1 is adjacent to m_3 , therefore, forms a group of two squares and is not part of any group of 4 squares. [A group of 8 squares is not possible in this case].



Similarly m_2 is also adjacent to m_3 therefore forms another group of two squares and is not a part of any group of 4 squares.

Step 4 and 5. Discarded because these are no quad or octet.

Step 6. All the 1's have already been grouped.

There is an overlapping of groups because they include common minterm m_3 .

Step 7. There is no redundant group.

Step 8. The terms generated by the two groups are 'OR' operated together to obtain the expression for F as follows:

F = A	+	B
↓		↓
From		From group
group m_2 m_3		m_1 m_3
↓		↓
This row is		This column is
corresponding		corresponding to
to the value		the value of B is
of A is equal		equal to 1.
to 1.		

Qs: Simply the following Boolean function by first expressing it in sum of minterms.

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

Minimization in Products of Sums Form

- So far, we have seen in all previous examples that the minimized function were expressed in sum of products form. With a minor modification, product of sums (POS) form can be obtained. The process is as follows:
 - Draw map as for SOP; mark the 0 entries.
 - The 1's places in the squares represents minterms of the function.
 - The minterms not included in the function denote the complement of the function.
 - Thus the complement of a function is represented on the map by the squares marked by 0's.
 - Group 0 entries as you group 1 entries for a SOP reading, to determine the simplified SOP expression for F' .
 - Use De Morgan's theorem on F' to produce the simplified expression in POS form.

Example. Given the following Boolean function:

$$F = A'C + A'B + AB'C + BC$$

Find the simplified products of sum (POS) expression.

Solution. Step 1. We draw a three variable K-map. From the given function we observe that minterms 1, 2, 3, 5, and 7 are having the value 1 and remaining minterms i.e., 0, 4 and 6 are 0. So we mark the 0 entries.

		BC			
A		00	01	11	10
	0	0	1	1	1
1	0	1		1	

Step 2. Minterms 0 and 4 forms a pair, giving value = $B'C'$. Similarly minterms 4 and 6 forms a second pair giving value = AC' . Therefore we get $F' = AC' + B'C'$.

Step 3. Applying De Morgan's theorem automatically converts SOP expression into POS expression, giving the value of F

$$\begin{aligned}
 (F')' &= [AC' + B'C']' \\
 &\Rightarrow [(AC')' \cdot (B'C')'] F = \\
 &(A + C) \cdot (B + C)
 \end{aligned}$$

Four Variable K-Map

- Let us examine a four variable truth table shown is Fig. 3.4. We used a K-map with 16 squares to represent all the minterms of input variables A, B, C and D distinctly.
- A four-variable K-map is shown in Fig. 3.5.

Truth Table

Minterm	Inputs				Output Y
	A	B	C	D	
m_0	0	0	0	0	y_0
m_1	0	0	0	1	y_1
m_2	0	0	1	0	y_2
m_3	0	0	1	1	y_3
m_4	0	1	0	0	y_4
m_5	0	1	0	1	y_5
m_6	0	1	1	0	y_6
m_7	0	1	1	1	y_7
m_8	1	0	0	0	y_8
m_9	1	0	0	1	y_9
m_{10}	1	0	1	0	y_{10}
m_{11}	1	0	1	1	y_{11}
m_{12}	1	1	0	0	y_{12}

Minterm	Inputs				Output Y
	A	B	C	D	
m_{13}	1	1	0	1	y_{13}
m_{14}	1	1	1	0	y_{14}
m_{15}	1	1	1	1	y_{15}

Fig. 3.4

BC AB	00	01	11	10
00	m ₀	m ₁	m ₃	m ₂
01	m ₄	m ₅	m ₇	m ₆
11	m ₁₂	m ₁₃	m ₁₅	m ₁₄
10	m ₈	m ₉	m ₁₁	m ₁₀

Fig. 3.5

- The rows and column are numbered in a reflected-code sequence, with only one digit changing value between two adjacent rows or columns.
- The minimization process is similar as well have done in a three variable K-Map. However the definition of adjacency can further be extended. Considering the map to be on a surface with the top and bottom edges, as well as sight and left edges, touching each other of form adjacent squares.
- For example m_0 is adjacent to m_2 , m_4 as well as to m_8 , similarly m_3 is adjacent to m_1 , m_2 , m_7 as will as to m_{11} and so on.

Example 1. Simplify the given fraction.

$$F = ABCD + AB'C'D' + AB'C + AB$$

Solution.

Step 1. The given function is consisting of four variables A, B, C and D. We draw a four variable K-map. The first two terms in the function have fours literals and are repeated in a four variable map by one square each. The square corresponding to first term ABCD is equivalent to minterm 1111. (m_{15}). Similarly the square for second term $AB'C'D'$ is equivalent to minterm 1000 (m_8) the third term in the function has three literals and is represented in a four var map by two adjacent squares. AB' corresponds to 4th row (i.e. 10) in the map and C corresponds to last two columns (i.e. 11 and 10) in the map. The last term AB has two (A term with only one literal is represented by 8 adjacent square in map. Finally a 1 in all 16 squares give $F = 1$. It means all the minterms are having the value equal to 1 in the function). Literals and is represented by 4 adjacent squares. Here, AB simply corresponds to 3rd row (i.e., $AB = 11$).

CD AB	00	01	11	10
00	m ₀	m ₁	m ₃	m ₂
01	m ₄	m ₅	m ₇	m ₆
11	m ₁₂	m ₁₃	m ₁₅	m ₁₄
10	m ₈	m ₉	m ₁₁	m ₁₀

AB AB'C'D' ABCD AB'C

Now, let us redraw the map first for simplification purpose.

Step 2. Is discarded. (No 1's which are not adjacent to other 1's)

Step 3. Discard (No pairs which are not part of any larger group)

Step 4. There are three quads.

Minterms 8, 10, 12, 14 from first quad.

Minterms 12, 13, 14, 15 form second quad

and Minterms 10, 11, 14, 15 form third quad.

Step 5. Discarded. (No octets)

Step 6. Discarded (All 1's have been grouped.)

Step 7. Discard (No redundant term)

Step 8. The terms generated by three groups are 'OR' operated as follow

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

$\downarrow$
From
First
group

$\downarrow$
Second
group

$\downarrow$
Third
group.

AB \ CD	00	01	11	10
00				
01				
11	1	1	1	1
10	1		1	1

Prime and Essential Implicants

- So far we have seen a method for drawing and minimising Karnaugh maps in such a way that unnecessary (redundant) groupings can be avoided.
- Now let us establish some important definitions that will be used to a systematic procedure for combining squares in the process of K-map minimization.
- To do this, consider a function defined as $F(A, B, C, D) = \Sigma(0, 1, 2, 3, 5, 7, 8, 9, 10, 13, 15)$.
- Now we will analyze the grouping shown in the 4 variable map in Fig. 3.6.

AB \ CD	00	01	11	10
00	m0 1	m1 1	m3 1	m2 1
01	m4 1	m5 1	m7 1	m6 1
11	m12 1	m13 1	m15 1	m14 1
10	m8 1	m9 1	m11 1	m10 1

Fig. 3.6

- Here, we see that all realistic groupings are shown. Note further that each group is sufficiently large that it can not be completely covered by any other simple grouping. Each of these five groupings is defined as a Prime implicant.

I group → covering minterms → 0, 2, 8, and 10, → $B'D'$ II group →

covering minterms → 0, 1, 2, and 3 → $A'B'$ III group → covering

minterms → 1, 5, 9, and 13 → $C'D'$ IV group → covering minterms

→ 1, 3, 5 and 7 → $A'D'$ V group → covering minterms → 5, 7, 13

and 15 → BD IV group → covering minterms → 0, 1, 8, 9 → $B'C'$

Don't care Map Entries

- Many times in digital system design, some input combinations must be considered as cases that “Just don't happen”, and there are cases when the occurrence of particular combinations will have no effect on the system, and if those combinations do occur, “you don't care”.
- For example, consider the case where the outputs of a 4-bit binary counter; which happens to have a possible range from 0000 to 1111, is to be converted to a decimal display having the range of 0, 1, 2, ..., 9.
- The converter might be a digital system having binary 4-bit inputs and decimal upto as shown Fig. 3.7.
- In this particular case, the input combinations 1001, 1010, 1011, 1100, 1101, 1110, 1111 are to be considered by combinations that just can not be accepted by the digital system if it is function properly.
- Therefore, when this digital system is being designed, these minterms in the map are treated in a special way.
- That is a d or a × (cross) is entered into each square to signify “don't care” MIN/MAX terms.
- Reading a map, or grouping a map with don't care entries is a simple process.
- Group the don't care (d or ×) with a 1 grouping if and only if this grouping will result in greater simplification; otherwise treat it as if it were a 0 entry.

Example. Simplify the following Boolean function.

$$F(A, B, C, D) = \sum (0, 1, 2, 10, 11, 14) \\ d(5, 8, 9)$$

Solution. The K-map for the given function is shown in Fig. 3.8 with entries X (don't care) in squares corresponding to combinations 5, 8 and 9.

CD \ AB	00	01	11	10
00	1	1		1
01		X		
11				1
10	X	X	1	1

Fig. 3.8

As discussed above, the 1's and d's (Xs) are combined in order to enclose the maximum number of adjacent squares with 1. As shown in K-map in Fig. 3.8, by combining 1's and d's (Xs), three quads can be obtained. The X in square 5 is left free since it does not contribute in increasing the size of any group. Therefore, the

I Quad covers minterms 0, 2, 10 and d8

➤ Quad covers minterms 10, 11 and d8, d9. III

Quad covers minterms 0, 1 and d8, d9. A pair covers minterms 10 and 14.

So,

$$F = B'D' + AB' + B'C' + ACD' \\ \text{Due to I quad} \quad \text{II quad} \quad \text{III Quad} \quad \text{Due to Pair.}$$

MINIMIZATION USING QUINE-McCLUSKEY (TABULAR) METHOD

The K-map method is suitable for simplification of Boolean functions up to 5 or 6 variables. As the number of variables increases beyond this, the visualization of adjacent squares is difficult as the geometry is more involved.

The 'Quine-McCluskey' or 'Tabular' method is employed in such cases. This is a systematic step by step procedure for minimizing a Boolean expression in standard form.

Procedure for Finding the Minimal Expression

- Arrange all minterms in groups, such that all terms in the same group have same number of 1's in their binary representation. Start with the least number of 1's and continue with grouping of increasing number of 1's, the number of 1's in each term is called the index of that term i.e., all the minterms of same index are placed in a same group. The lowest value of index is zero. Separate each group by a thick line. This constitutes the I stage.
- Compare every term of the lowest index (say i) group with each term in the successive group of index (say, $i + 1$).
- If two minterms differ in only one variable, that variable should be removed and a dash (–) is placed at the position, thus a new term with one less literal is formed. If such a situation occurs, a check mark (✓) is placed next to both minterms.
- After all pairs of terms with indices i and $(i + 1)$ have been considered, a thick line is drawn under the last terms.
- When the above process has been repeated for all the groups of I stage, one stage of elimination have been completed.
- This constitutes the II stage.
- The III stage of elimination should be repeated of the newly formed groups of second stage. In this stage, two terms can be compared only when they have dashes in same positions.
- The process continues to next higher stages until no further comparisons are possible. (i.e., no further elimination of literals).
- All terms which remain unchecked (No ✓ sign) during the process are considered to be prime implicants (PIs). Thus, a set of all PIs of the function is obtained.
- From the set of all prime implicants, a set of essential prime implicants (EPIs) must be determined by preparing prime implicant chart as follow.
 - (a) The PIs should be represented in rows and each minterm of the function in a column.
 - (b) Crosses should be placed in each row corresponding to minterms that makes the PIs.
 - (c) A complete PIs chart should be inspected for columns containing only a single cross. PIs that cover minterms with a single cross in their column are called EPIs.
- The minterms which are not covered by the EPIs are taken into consideration and a minimum cover is obtained from the remaining PIs.

Now to clarify the above procedure, let's do an example step by step.

Example 1. Simplify the given function using tabular method.

$$F(A, B, C, D) = \sum (0, 2, 3, 6, 7, 8, 10, 12, 13)$$

Solution. 1. The minterms of the function are represented in binary form. The binary represented are grouped into a number of sections in terms of the number of 1's index as shown in Table of Fig. 3.15.

Minterms	Binary ABCD	No. of 1's	Minterms Group	Index	Binary ABCD
m_0	0 0 0 0	0	m_0	0	0 0 0 0 ✓
m_2	0 0 1 0	1	m_2		0 0 1 0 ✓
m_3	0 0 1 1	2	m_8	1	1 0 0 0 ✓
m_6	0 1 1 0	2	m_3		0 0 1 1 ✓
m_7	0 1 1 1	3	m_6	2	0 1 1 0 ✓
m_8	1 0 0 0	1	m_{10}		1 0 1 0 ✓
m_{10}	1 0 1 0	2	m_{12}		1 1 0 0 ✓
m_{12}	1 1 0 0	2	m_7		0 1 1 1 ✓
m_{13}	1 1 0 1	3	m_{13}	3	1 1 0 1 ✓

Fig. 3.15

- Compare each binary term with every term in the adjacent next higher category. If they differ only by one position put a check mark and copy the term into the next column with (–) in the place where the variable is unmatched, which is shown in next Table of Fig. 3.16.

Minterm Group	A	B	C	D	
0, 2	0	0	–	0	✓
0, 8	–	0	0	0	✓
2, 3	0	0	1	–	✓
2, 6	0	–	1	0	✓
2, 10	–	0	1	0	✓
8, 10	1	0	–	0	✓
8, 12	1	–	0	0	PI
3, 7	0	–	1	1	✓
6, 7	0	1	1	–	✓
12, 13	1	1	0	–	PI

Fig. 3.16

Minterm Group	A	B	C	D	
0, 2, 8, 10	–	0	–	0	PI
0, 8, 2, 10	–	0	–	0	PI eliminated
2, 3, 6, 7	0	–	1	–	PI
2, 6, 3, 7	0	–	1	–	PI eliminated.

Fig. 3.17

- Apply same process to the resultant column of Table of Fig. 3.16 and continue until no further elimination of literals. This is shown in Table of Fig. 3.17 above.
- All terms which remain unchecked are the PIs. However note that the minterms combination (0, 2) and (8, 10) form the same combination (0, 2, 8, 10) as the combination (0, 8) and (2, 10). The

order in which these combinations are placed does not prove any effect. Moreover, as we know that $x + x = x$, thus, we can eliminate one of these combinations.

The same occur with combination (2, 3) and (6, 7).

- Now we prepare a PI chart to determine EPIs as follows shown in Table of Fig. 3.18.

<i>Prime Implicants</i>	<i>Minterms</i>								
	0	2	3	6	7	8	10	12	13
(8, 12)						×		×	
(12, 13) *								×	×
(0, 2, 8, 10) *	×	×				×	×		
(2, 3, 6, 7) *		×	×	×	×				
	✓		✓	✓	✓		✓		✓

Fig. 3.18

-
- (a) All the PIs are represented in rows and each minterm of the function in a column.
 - (b) Crosses are placed in each row to show the composition of minterms that make PIs.
 - (c) The column that contains just a single cross, the PI corresponding to the row in which the cross appear is essential. Prime implicant. A tick mark is put against each column which has only one cross mark. A star (*) mark is placed against each. EPI.

➤ All the minterms have been covered by EPIs.

Finally, the sum of all the EPIs gives the function in its minimal SOP form

<i>EPIs.</i>	<i>Binary representation</i>				<i>Variable Representation</i>
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	
12, 13	1	1	0	–	ABC'
0, 2, 8, 10	–	0	–	0	$B'D'$
2, 3, 6, 7	0	–	1	–	$A'C$

Therefore, $F = ABC' + B'D' + A'C.$

If don't care conditions are given, they are also used to find the prime implicants, but it is not compulsory to include them in the final simplified expression.

