Logical Expression Minimization

The goal of logic expression minimization is to find an equivalent of an original logic expression that has fewer variables per term, has fewer terms and needs less logic to implement. There are three main manual methods used for logic expression minimization; algebraic minimization, Karnaugh Map minimization and Quine-McCluskey (tabular) minimization

Algebraic minimization

The algebraic minimization process is the application of the switching algebra postulates, laws, and theorems to transform the original expression. It is hard to recognize when a particular law can be applied and difficult to know if resulting expression is truly minimal. The incorrect implementation or dropped variables etc can easy lead to a mistake.

The following are two examples of the algebraic minimization process by exploiting the adjacency theorem. Look for two terms that are identical except for one variable in the following expression

$$A \cdot B \cdot C \cdot \overline{D} \; + \; A \cdot B \cdot C \cdot D$$

Application removes one term and one variable from the remaining term

$$A \cdot B \cdot C \cdot \overline{D} \; + \; A \cdot B \cdot C \cdot D \; = \; A \cdot B \cdot C$$
$$(A \cdot B \cdot C) \cdot \overline{D} \; + \; (A \cdot B \cdot C) \cdot D \; = \; A \cdot B \cdot C$$
$$(A \cdot B \cdot C) \cdot \left(\overline{D} + D\right) \; = \; (A \cdot B \cdot C) \cdot 1 \; = \; A \cdot B \cdot C$$

In the following example one can look for the adjacency

$$F = \overline{A_3} A_2 \overline{A_1} A_0 + \overline{A_3} A_2 A_1 \overline{A_0} + \overline{A_3} A_2 A_1 A_0 + A_3 \overline{A_2} \; A \; \overline{A_0} + A_3 \overline{A_2} \; \overline{A_1} A_0$$

The first and third term differ only A_1 and $\overline{A}$

The third and fourth term differ only A_0 and A_0

The second and third term differ only A_0 and A_0

Duplicate 3rd. term and rearrange

$$F = \overline{A_3} A_2 \overline{A_1} A_0 + \overline{A_3} A_2 A_1 A_0 + \overline{A_3} A_2 A_1 \overline{A_0} + \overline{A_3} A_2 A_1 A_0 + A_3 \overline{A_2} \; \overline{A_1} \overline{A_0} + A_3 \overline{A_2} \; \overline{A_1} A_0$$

Apply adjacency on term pairs

$$F = \overline{A_3}A_2A_0 + \overline{A_3}A_2A_1 + A_3\overline{A_2}\ \overline{A_1}$$

Karnaugh Map (or K-map) minimization

The Karnaugh map provides a systematic method for simplifying a Boolean expression or a truth table function. The K map can produce the simplest SOP or POS expression possible. K-map procedure is actually an application of adjacency and guarantees a minimal expression. It is easy to use, visual, fast and familiarity with Boolean laws is not required.

The K map is a table consisting of $N = 2^n$ cells, where n is the number of input variables. Assuming the input variable are A and B then the K map illustrating the four possible variable combinations is shown.

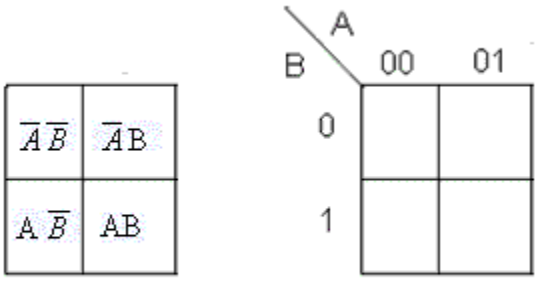


Figure 5: Two variable K map

Similarly three variable and four variable K-maps can be constructed as shown below

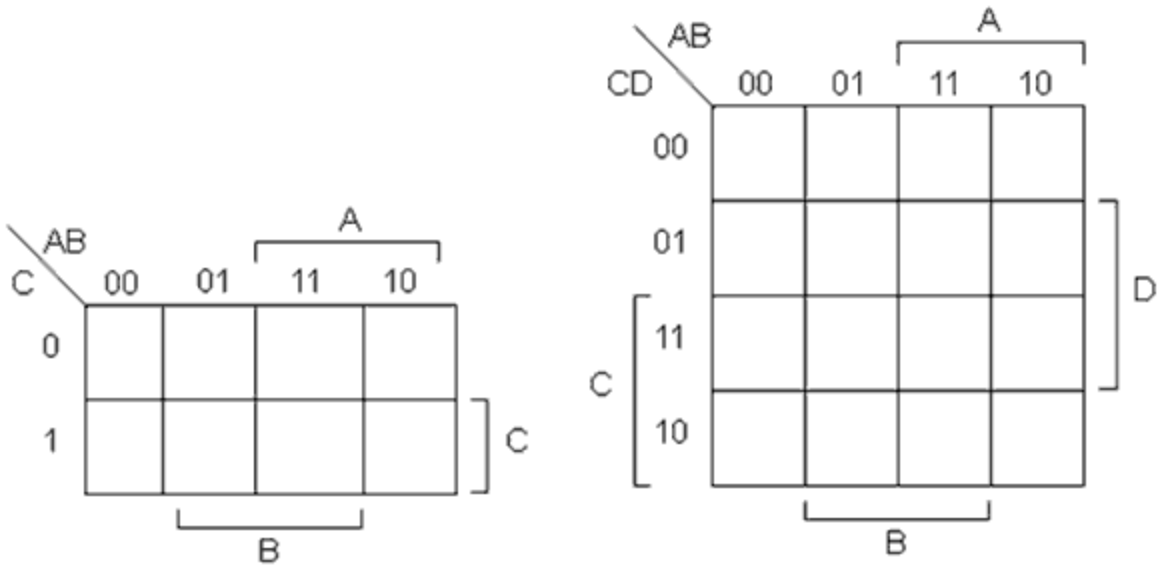


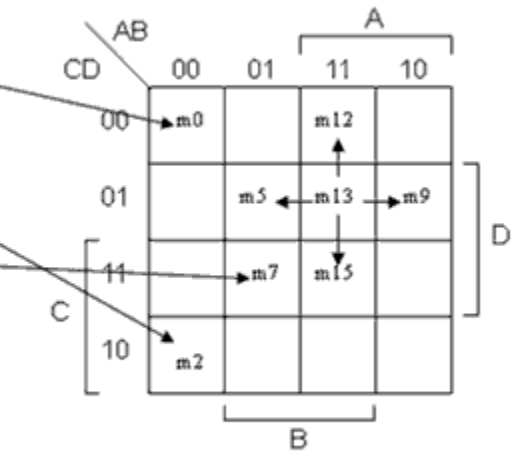
Figure 5: Three variable and four variable K maps

For a SOP expression each cell represents one particular combination of the variables in product form. The table format is such that there is a single variable change between any adjacent cells. This is the characteristic the will determine adjacency. This method is typically applicable to limited number of variables (4 ~ 8) and for n > 5 the K map technique becomes impractical unless implemented on computer. Manual errors are possible in translation from Truth Table to K-map, or when grouping of cells not done correctly.

Basic K-map is a 2-D rectangular array of cells, each K-map represents one bit column of output and each cell contains one bit of output function. The arrangement of cells in array facilitates recognition of adjacent terms and adjacent terms differ in one variable value; equivalent to difference of one bit of input row values, e.g. m6 (110) and m7 (111). The standard Truth Table ordering does not show adjacency. One uses gray code for row order however, it is still hard to see all possible adjacencies. For any cell in 2-D array, there are four direct neighbors (top, bottom, left, right). The 2-D array can therefore show adjacencies of up to four variables. One should not forget that cells are adjacent top to bottom and side to side. The number of TT rows must match number of K-map cells. Watch out for ordering of 10 and 11 rows and columns.

To simplify a SOP for of a Boolean expression using a K map, first identify all the input combinations that produce an output of logic level 1 and place them in their appropriate K map cell. Consequently, all other cells must contain zero (0). Second, group the adjacent cells that contain 1 in a manner that maximizes the size of the groups but also minimizes the total number of groups. All 1's in the output must be included in a group even if the group is only one cell. Third, as each SOP term represents an AND expression, each (AND) grouping is written with only the input variables that are common to the group. Finally, the simplified expression is formed by ORing each of the (AND) groups.

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

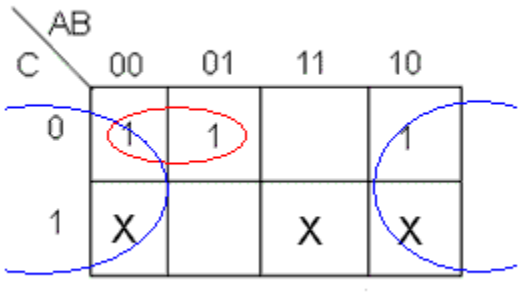


When the input combinations are irrelevant or cannot occur, the output states are in the Truth table and the K map are filled with an X and are referred to as don't care states . The don't cares can work to our advantage during minimization; we can assign either 0 or 1 as needed. When simplifying K maps with don't care states, the contents of the undefined cells (1 or 0) are chosen according to preference. The aim is to enlarge group sizes thereby eliminating as many input variables from the simplified expression as possible. Only those X's that assist in simplifying the function should be included in the groupings. No additional X's should be added that would result in additional terms in the expression. To illustrate let us consider the function specified by Table 8 and its corresponding K map shown in Fig. 8. Note that the two groupings determine that the simplified expression is expressed as $F = C + \overline{A}\,\overline{B}$

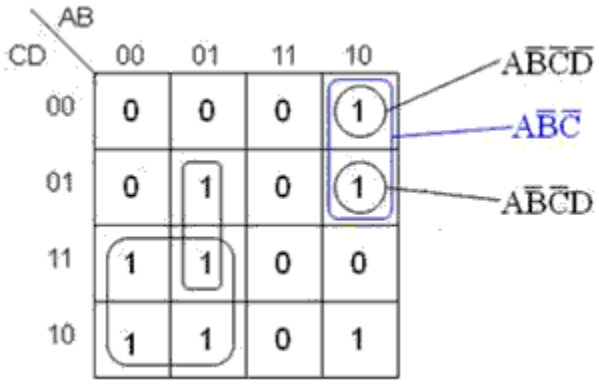
Table 8: Truth Table of the Function $F = C + \overline{A}\,\overline{B}$

Input			Ouput
A	**B**	**C**	**F**
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	×
1	1	0	×

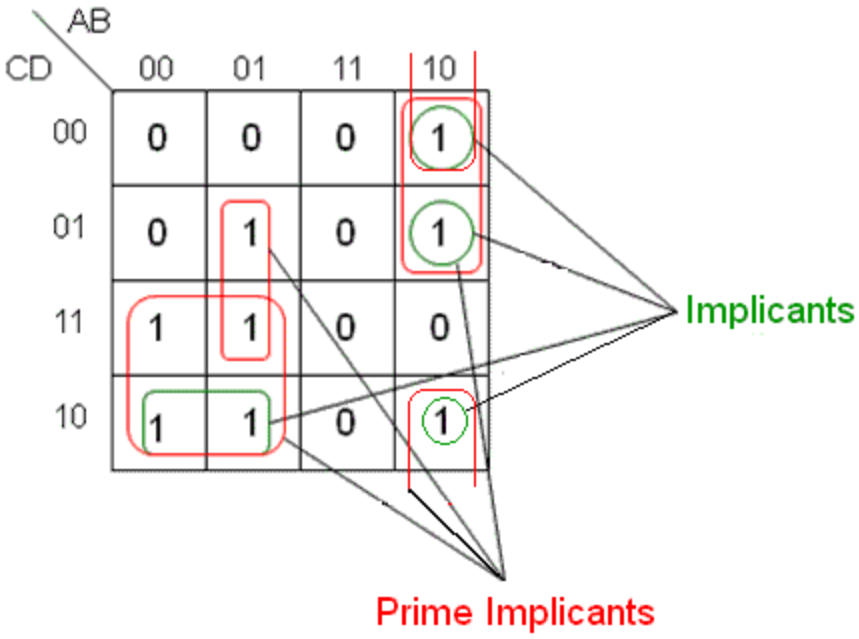
1	1	1	×



If two cells have the same value and are next to each other, the terms are adjacent. This adjacency is shown by enclosing them. Groups can have common cells. Group size is a power of 2 and groups are rectangular. You can group 0s or 1s. If '1's are grouped, the expression will be a product term and '0's are grouped the expression will be a sum term. It is important to note when a variable values change as you go cell to cell. This determines how the term expression is formed by the following table. The bottom most row right side cell is having a value "1". It can be represented by the expression $A\bar{B}C\bar{D}$. Similarly, expressions for individual as well as grouped "1"s are shown in the figure.



How the minimum expression of a function is determined using a Karnaugh Map? The concept of prime implicants can be used to determine the minimum solution. Single cells or groups that could be part of a larger group are known as implicants and a group that is as large as possible is a prime implicant. Single cells can be prime implicants is they cannot be grouped with any other cell. In the following Karnaugh map, the implicants and prime implicants are marked separately. The term $A\bar{B}C\bar{D}$ is not a prime implicant because it can be combined with $\bar{A}\bar{B}C\bar{D}$ or $A\bar{B}\bar{C}\bar{D}$.



The minimum SOP expression for a function consists of some (but not necessarily all) of the prime implicants of a function. In other words, a SOP expression containing a term, which is not a prime implicant, cannot be the minimum. This is true because if a nonprime term were present, the expression could be simplified by combining the nonprime term with additional minterms. Any set of implicants that encloses (covers) all values is "sufficient"; i.e. the associated logical expression represents the desired function. For example, all minterms or maxterms are sufficient. However, the smallest set of prime implicants that covers all values forms a minimal expression for the desired function. There may be more than one minimal set.

5 Variable K Maps

A five variable K Map can be constructed using two 4 variable maps side-by-side. The groups spanning both maps occupy the same place in both maps.

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

A = 0

DE \ BC	00	01	11	10
00	16	20	28	24
01	17	21	29	25
11	19	23	31	27
10	18	22	30	26

A = 1

DE \ BC	00	01	11	10
00	0	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

A = 0

DE \ BC	00	01	11	10
00	0	0	0	1
01	1	1	1	0
11	1	1	1	0
10	0	0	0	0

A = 1

The Fig. is a map of $f(A,B,C,D,E) = \sum m\,(2, 5, 7, 8, 10, 13, 15, 17, 19, 21, 23, 24, 29, 31)$

When checking for adjacencies, each term should be checked against the five possible adjacent squares.