Chapter-3

Simplification of Boolean Functions

Different forms of Boolean Algebra

Sum of Product (SOP)

This is an expression in which each term is a product term and all the product terms are summed together.

a) Minimal SOP

An expression in which each product term consist of minimum numbers of variables.

E.g. XY+X’Y+XY’

b) Expanded SOP

An expression in which each product term consist of maximum numbers of variables.

E.g. XYZ+X’YZ+WXYZ

Different forms of Boolean Algebra

Product of Sum (POS)

This is an expression in which several sum terms are multiplied together.

a) **Minimal POS**

In this an expression in which there are minimum number of sum terms.

E.g. $(X+Y)(X'+Y)(X+Y')$

b) **Expanded POS**

In this an expression in which there are maximum number of variables.

E.g. $(X+Y+Z)\cdot(X'+Y'+Z')\cdot(W+X+Y+Z)$

Minterms and Maxterms

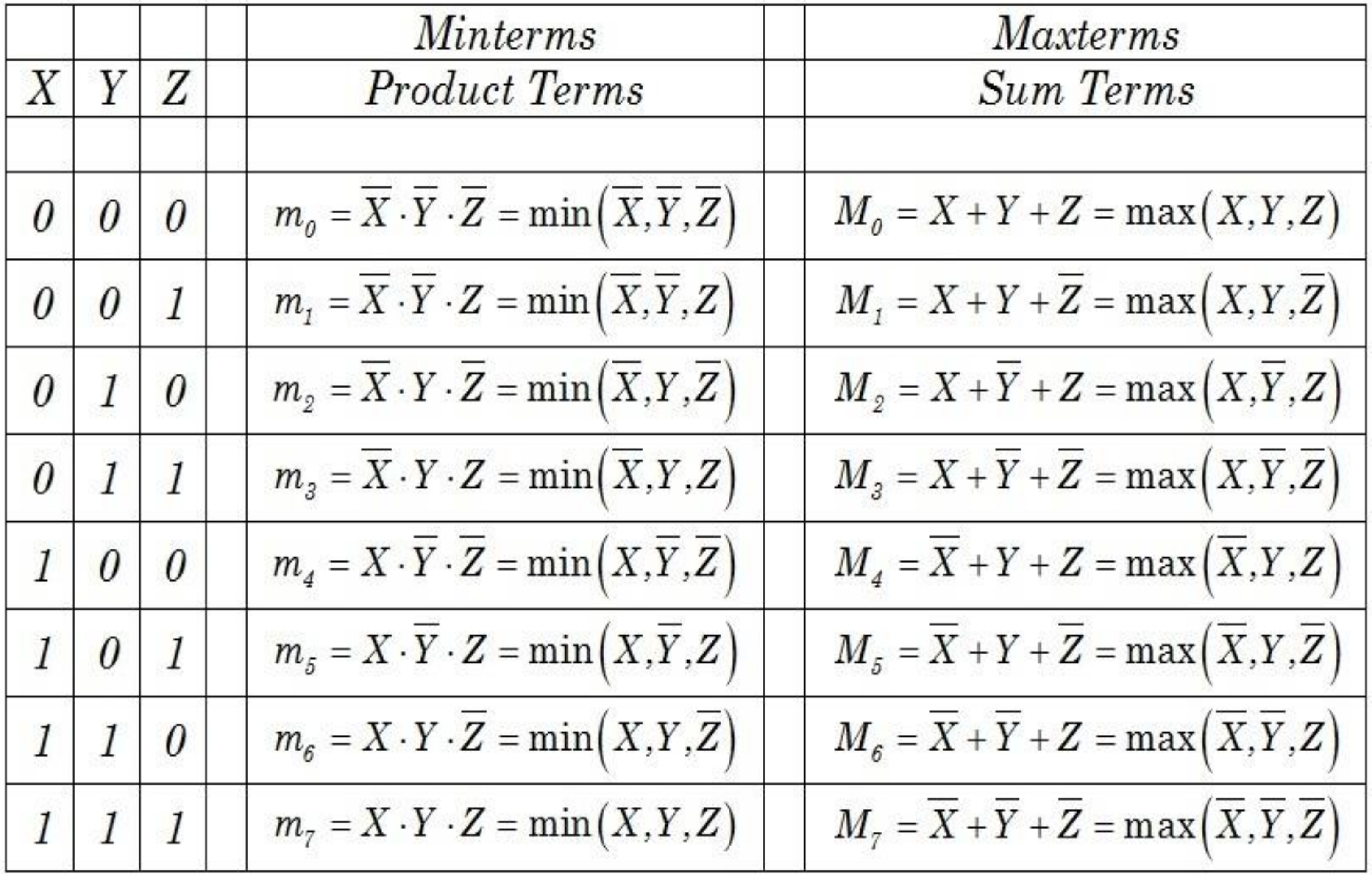
Minterms or standard product:

- Each row of a truth table can be associated with a minterm, which is a product (AND) of all variables in the function, in direct or complemented form.
- A function with n variables has 2^n minterms.
- A minterm has the property that it is equal to 1 on exactly one row of the truth table.

Maxterms or standard sums:

- Each row of a truth table is also associated with a maxterm, which is a sum (OR) of all the variables in the function, in direct or complemented form.
- A function with n variables has 2^n maxterms
- A maxterm has the property that it is equal to 0 on exactly one row of the truth table.

Minterms and Maxterms



				Minterms		*Maxterms*
X	*Y*	*Z*		*Product Terms*		*Sum Terms*
0	0	0		$m_0 = \overline{X} \cdot \overline{Y} \cdot \overline{Z} = \min\left(\overline{X},\overline{Y},\overline{Z}\right)$		$M_0 = X + Y + Z = \max\left(X,Y,Z\right)$
0	0	1		$m_1 = \overline{X} \cdot \overline{Y} \cdot Z = \min\left(\overline{X},\overline{Y},Z\right)$		$M_1 = X + Y + \overline{Z} = \max\left(X,Y,\overline{Z}\right)$
0	1	0		$m_2 = \overline{X} \cdot Y \cdot \overline{Z} = \min\left(\overline{X},Y,\overline{Z}\right)$		$M_2 = X + \overline{Y} + Z = \max\left(X,\overline{Y},Z\right)$
0	1	1		$m_3 = \overline{X} \cdot Y \cdot Z = \min\left(\overline{X},Y,Z\right)$		$M_3 = X + \overline{Y} + \overline{Z} = \max\left(X,\overline{Y},\overline{Z}\right)$
1	0	0		$m_4 = X \cdot \overline{Y} \cdot \overline{Z} = \min\left(X,\overline{Y},\overline{Z}\right)$		$M_4 = \overline{X} + Y + Z = \max\left(\overline{X},Y,Z\right)$
1	0	1		$m_5 = X \cdot \overline{Y} \cdot Z = \min\left(X,\overline{Y},Z\right)$		$M_5 = \overline{X} + Y + \overline{Z} = \max\left(\overline{X},Y,\overline{Z}\right)$
1	1	0		$m_6 = X \cdot Y \cdot \overline{Z} = \min\left(X,Y,\overline{Z}\right)$		$M_6 = \overline{X} + \overline{Y} + Z = \max\left(\overline{X},\overline{Y},Z\right)$
1	1	1		$m_7 = X \cdot Y \cdot Z = \min\left(X,Y,Z\right)$		$M_7 = \overline{X} + \overline{Y} + \overline{Z} = \max\left(\overline{X},\overline{Y},\overline{Z}\right)$

Expressing Boolean function by sum of minterms

A Boolean function may be expressed algebraically from a given truth table or from the given expression in the form of minterms:

Example:

Express the Boolean function $F = x + yz$ as a sum of minterms.

$F = x + y\,z = x + (y\,z)$

$= x(y+y')(z+z') + (x+x')yz$

$= x\,y\,z + x\,y\,z' + x\,y'\,z + x\,y'\,z' + x\,y\,z + x'\,y\,z$

$= m7 + m6 + m5 + m4 + m3$

$= \Sigma(3, 4, 5, 6, 7)$

Expressing Boolean function by sum of minterms

Express $F = (x + y z)'$ as a sum of minterms.

$= (x + y z)' = (x + (y z))'$

$= x' (y' + z')$

$= (x' y') + (x' z')$

$= x' y' (z + z') + x' (y + y') z'$

$= x' y' z + x' y' z' + x' y z'$

$= m1 + m0 + m2$

$= \Sigma(0, 1, 2)$

Expressing Boolean function by product of maxterm

Question: Express the Boolean function $F = xy + x'z$ in a product of maxterm form.

Solution:

First, con vert the function into OR terms using the distributive law:

$F = xy + x'z = (xy + x')(xy + z)$

$= (x + x')(y + x')(x + z)(y + z)$

$= (x' + y)(x + z)(y + z)$

The function has three variables: x, y, and z. Each OR term is missing one variable; therefore:

$x' + y = x' + y + zz' = (x' + y + z)(x' + y + z')$

$x + z = x + z + yy' = (x + y + z)(x + y' + z)$

$y + z = y + z + xx' = (x + y + z)(x' + y + z)$

Combing all maxterms and removing repeated terms:

$F = (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z')$

$= M_0 \; M_2 \; M_4 \; M_5$

Shorthand notation:

$F(x, y, z) = \prod(0, 2, 4, 5)$

Expressing Boolean function as a sum of minterms and product of maxterms

To express the Boolean function as a sum of minterms, expand the Boolean function into a sum of products. Each term is then inspected to see if it contains all the variables. If it misses one or more variables, it is ANDed with an expression such as $x + x'$, where x is one of the missing variables.

To express the Boolean function as a product of maxterms, expand the Boolean function into a product of sums. This may be done by using the distributive law, $x + yz = (x + y)(x+ z)$. Then any missing variable x in each OR term is ORed with xx'.

More Example

Express the Boolean function $F = A + B'C$ in a sum of minterms.

Express the Boolean function $F = x + y'z$ in a product of maxterm form.

Express the Boolean function $F = AB + A\ C + B\ C$ in a sum of minterms.

Canonical and Standard forms

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in canonical form.

Any expression in canonical form, every variable appears in every term.

The two canonical forms of Boolean algebra are basic forms that one obtain from reading a function from the truth table.

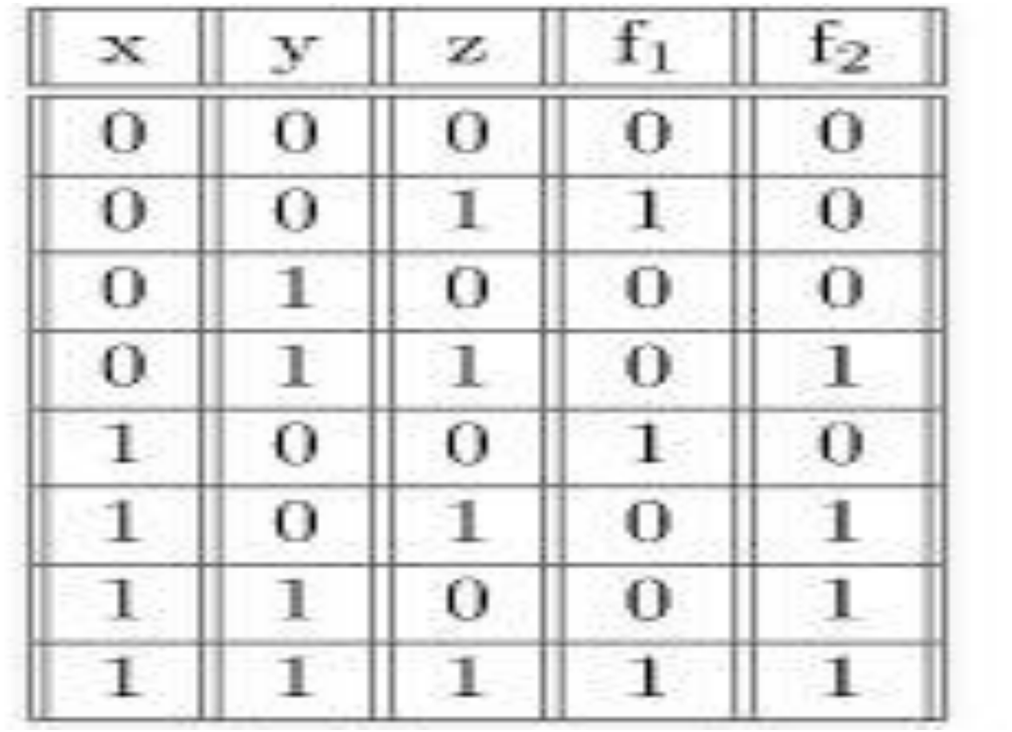
Example:

F=A'BC+ABC'(sop)

F=(A+B+C)*(A'+B'+C)   (pos)

Another way to express Boolean functions is in standard form. In this configuration, the terms that form the function may contain one, two or any number of literal.

Express the following function in Canonical SoP and PoS form



x	y	z	f_1	f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Conversion between canonical forms

General Procedure: To convert from one canonical form to another, interchange the symbols and list those numbers missing from the original form. In order to find the missing terms, one must realize that the total number of minterms or maxterms is 2^n (numbered as 0 to 2^n-1), where n is the number of binary variables in the function.

For example: Consider the function, $F(A, B, \mathrm{C}) = (1, 4, 5, 6, 7)$

Its complement can be expressed as: $F' (A, B, \mathrm{C}) = (0, 2, 3) = m0 + m2 + m3$

Consider a function, F = xy + x'z. First, derive the truth table of the function and find the complement.

Classwork

Question: Design a logic circuit having 3 inputs, A, B, C will have its output HIGH only when a majority of the inputs are HIGH.

Introduction to K-Map (Karnaugh Map)

In many digital circuits and practical problems we need to find expression with minimum variables. We can minimize Boolean expressions of 3, 4 variables very easily using K-map without using any Boolean algebra theorems. K-map can take two forms Sum of Product (SOP) and Product of Sum (POS) according to the need of problem. K-map is table like representation but it gives more information than TRUTH TABLE. We fill grid of K-map with 0’s and 1’s then solve it by making groups.

Using a K-map, expressions with two to four variables are easily minimized. Expressions with five to six variables are more difficult but achievable.

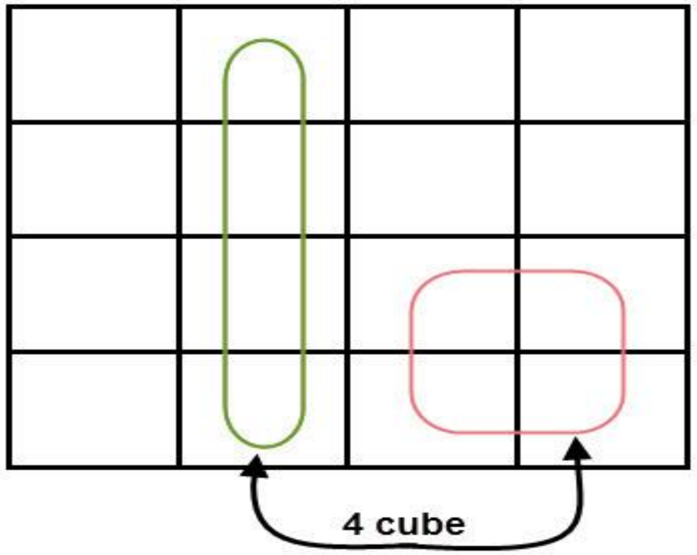
Steps to solve expression using K-map

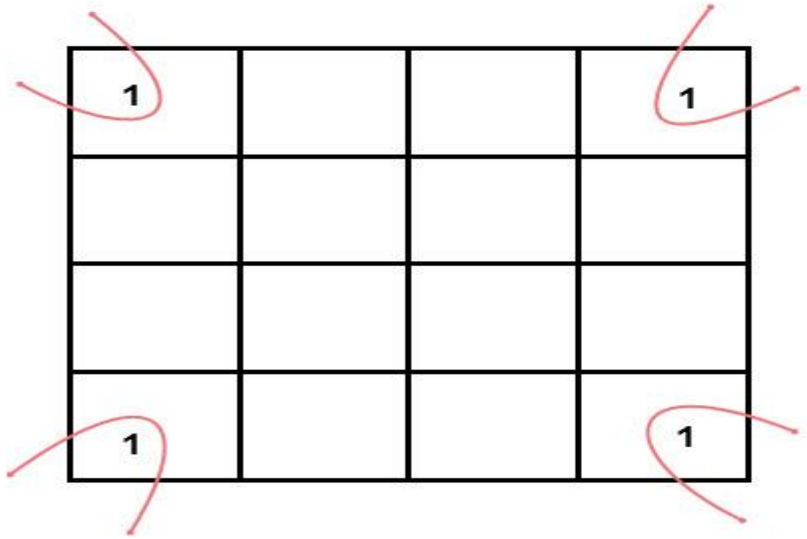
- Select K-map according to the number of variables.
- Identify minterms or maxterms as given in problem.
- For SOP put 1’s in blocks of K-map respective to the minterms (0’s elsewhere).
- For POS put 0’s in blocks of K-map respective to the maxterms(1’s elsewhere).
- Make rectangular groups containing total terms in power of two like 2,4,8 .. and try to cover as many elements as you can in one group.
- From the groups made in step 5 find the product terms and sum them up for SOP form.

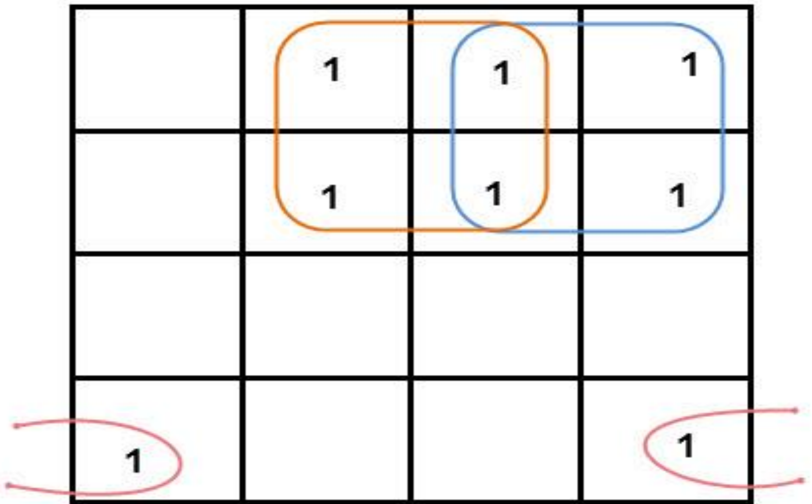
Grouping of K-map variables

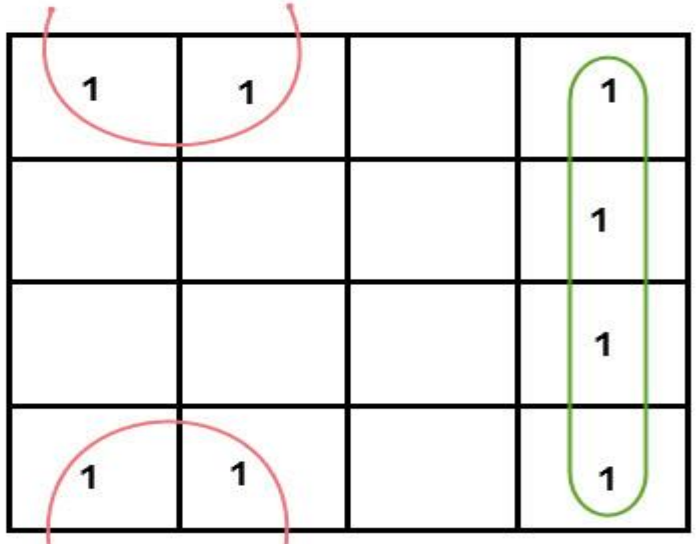
- There are some rules to follow while we are grouping the variables in K-maps. They are
- The square that contains ‘1’ should be taken in simplifying, at least once.
- The square that contains ‘1’ can be considered as many times as the grouping is possible with it.
- A group should be the as large as possible.
- Groups can be horizontal or vertical. Grouping of variables in diagonal manner is not allowed.
- if the square containing ‘1’ has no possibility to be placed in a group, then it should be added to the final expression.
- Groups can overlap.
- The number of squares in a group must be equal to powers of 2, such as 1, 2, 4, 8 etc.
- Groups can wrap around. As the K-map is considered as spherical or folded, the squares at the corners (which are at the end of the column or row) should be considered as they adjacent squares.
- The grouping of K-map variables can be done in many ways, so the obtained simplified equation need not to be unique always.
- The Boolean equation must be in canonical form, in order to draw a K-map.

Grouping of K-map variables









Practice of 2,3,4 variables K-map

1. F = ∑ (m0, m1, m2)
2. F (A,B,C) = ∑ (m0, m1, m2, m4, m5, m6)
3. F(A,B,C,D) = ∑(m0, m1, m2, m4, m5, m6, m8, m9, m12, m13, m14)
4. F (X,Y,Z,W) = ∑ (1,5,3,9,11,13,15)
5. F (A,B,C) = ∑ (0,1,5,6,7)
6. F (A,B,C,D) = ∑ (0,2,3,5,6,7,8,10,11,14,15)
7. F (A,B,C,D) = ∑ (0,2,3,7,11,13,14,15)
8. F (A,B,C,D) = ∑ (4,6,7,8,10,12,14,15)
9. F =A’B’C’+A’B’C+A’BC+A’BC’
10. F=A’B’C+A’BC+AB’C+ABC
11. F=A’B’C’+A’B’C+A’BC+A’BC’+ABC+ABC’
12. F=A’B’C’+A’BC’+AB’C’+A’BC’
13. F=A’B’C’D’+A’B’CD’+AB’C’D’+ABCD
14. F=A’B’C’D’+A’B’C’D+A’BCD+ABCD+ABCD’+AB’C’D’+AB’CD’

Don’t care Conditions

The logical sum of the minterms associated with a Boolean function specifies the conditions under which the function is equal to 1. The function is equal to 0 for the rest of the min terms. This assumes that all the combinations of the values for the variables of the function are valid. In practice, there are some applications where the function is not specified for certain combinations of the variables.

Example: four-bit binary code for the decimal digits has six combinations that are not used and consequently are considered as unspecified. In most applications, we simply don't care what value is assumed by the function for the unspecified minterms. For this reason, it is customary to call the unspecified minterms of a function don't-care conditions. These don't-care conditions can be used on a map to provide further simplification of the Boolean expression.

Don’t care Conditions

Don't-care minterm is a combination of variables whose logical value is not specified. To distinguish the don't-care condition from 1's and 0's, an X is used. Thus, an X inside a square in the map indicates that we don't care whether the value of 0 or 1 is assigned to F for the particular min term.

When choosing adjacent squares to simplify the function in a map, the don't-care minterms may be assumed to be either 0 or 1. When simplifying the function, we can choose to include each don't-care minterm with either the 1's or the 0's, depending on which combination gives the simplest expression.

Question: Simplify the Boolean function

$F(w, x, y, z)$ = (1, 3, 7, 11, 15)

that has the don't-care conditions

$d(w, x, y, z)$ = (0, 2, 5)

Solve the following

1. $F(X,Y,Z)= \sum(1,3,7)$ and $d(X,Y,Z)= \sum(0,4)$
2. $F(W,Y,Z)= \sum(0,2,5)$ and $d(W,Y,Z)= \sum(1,3)$
3. $F(W,X,Y,Z)= \sum(1,3,7,9,13,15)$ and $d(W,X,Y,Z)= \sum(0,4,6,14)$
4. $F(w,x,y,z)=\sum(0,4,9,1,12,13,15)$ and $d(w,x,y,z)= \sum(1,6,7,14)$
5. $F(p,q,r,s)=\sum(0,4,5,1,11,14,15)$ and $d(w,x,y,z)= \sum(2,3,7,8,9,13)$
6. F=AB’+ABD’ AND d=A’B’C+A’BD[HINTS: first make sop by multiplying missing terms like(C+C’)]
7. F=BC+AB’D AND d=A’B’C+A’BD

POS form of K-map

The 1's placed in the squares of the map represent the minterms of the function. The minterms not included in the function belong to the complement of the function. From this, we see that the complement of a function is represented in the map by the squares not marked by 1's. If we mark the empty squares with 0's and combine them into valid rectangles, we obtain an optimized expression of the complement of the function (F’). We then take the complement of F to obtain the function F as a product of sums

Simplify the following Boolean function in SOP and POS form

1. $F(W,X,Y,Z)= \sum(1,3,7,9,13,15)$ and $d(W,X,Y,Z)= \sum(0,4,6,14)$
2. $F(A,B,C,D)=\prod(0,1,2,4,5,7,10,15)$
3. $F(W,X,Y,Z)=\prod(0,4,5,7,8,9,13,15)$
4. $F=(A+B+C+D').(A'+B'+C'+D).(A'+B'+C+D')$
5. $F(W,X,Y,Z)=\prod(0,4,5,7,8,9)$ AND $D= \prod(1,2,3)$
6. $F=\prod(1,3,7,11,15,9,4)$ AND $D= \prod(0,2,14,12)$

NAND and NOR implementation

Digital circuits are more frequently constructed with NAND or NOR gates than with AND and OR gates.

NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families. The procedure for two-level implementation is presented in this section.

NAND and NOR conversions (from AND, OR and NOT implemented Boolean functions)

Because of the prominence of NAND and NOR gates in the design of digital circuits, rules and procedures have been developed for the conversion from Boolean functions given in terms of AND, OR, and NOT into equivalent NAND and NOR logic diagrams. To facilitate the conversion to NAND and NOR logic, there are two other graphic symbols for these gates.

(a) NAND gate

Two equivalent symbols for the NAND gate are shown in diagram below:

[CHECK universal gate implementation]

NAND implementation

The implementation of a Boolean function with NAND gates requires that the function be simplified in the sum of products form. To see the relationship between a sum of products expression and its equivalent NAND implementation, consider the logic diagrams of Fig below. All three diagrams are equivalent and implement the function: $F=AB + CD + E$

The rule for obtaining the NAND logic diagram from a Boolean function is as follows

Simplify the function and express it in sum of products.

Draw a NAND gate for each product term of the function that has at least two literals. The inputs to each NAND gate are the literals of the term. This constitutes a group of first-level gates.

Draw a single NAND gate (using the AND-invert or invert-OR graphic symbol) in the second level, with inputs coming from outputs of first-level gates.

A term with a single literal requires an inverter in the first level or may be complemented and applied as an input to the second-level NAND gate.

NAND implementation

Firstly make circuit diagram using AND ,NOT and OR gate

Use bubble in the output of AND gate and in the input of OR gate

Use NOT gate in place of bubble inserted places

Cancel the double NOT gate and Double NAND gate circuit if exist any

Convert into the equivalent circuit.

Example: F=AB + CD + E

NOR implementation

Firstly make circuit diagram using AND ,NOT and OR gate

Use bubble in the output of NOR gate and in the input of AND gate

Use NOT gate in place of bubble inserted places

Cancel the double NOT gate and Double NAND gate circuit if exist any

Convert into the equivalent circuit.

Example: F = (A + B) (C + D)E

Exercises

Q. **Draw the circuit diagram for the following function using NAND gates only**

F = A + (B' + C) (D' + BE ‘)

F= AB+(B’C+D)(AC’+B)

F= AB+CD

F=(X’+Y).(Y+Z’X)

Q. Draw the circuit diagram for the following function using NOR gates only

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

F=X+(Y+Z).(X’Y’)

F=X’(Y+Z)+(YZ+X’Y).(Y+Z)+XY’