Chapter-2: Boolean Functions & Reduce Forms

Learning Objective

At the end of the chapter students will

- Understand how logic relates to computing problems
- Handle simple Boolean function in SOP and POS form
- Apply rules to simplify / expand Boolean terms / functions
- Simplify the Boolean function using K map and algebraic method
- Establish the correspondence between Karnaugh maps, truth table and logical expressions.

Till now, we were using the available boolean function, now let's learn how to make a boolean function. In a boolean function, a binary variable can appear in any form, normal (a) or complemented (a'). Now consider a boolean expression formed on two variables a & b using AND operator. Since each variable may appear in any form, there are four possible ways of combining these two variables viz. a'b', ab', a'b, ab.

Similarly if we combine them using OR operator, again there are four possible ways viz. a+b, a'+b, a+b', a'+b'

For now we will concentrate on the terms formed using AND operator only. Each of them is called a minterm or Standard Product Term. A minterm, is obtained from AND term, with each variable being complemented, if the corresponding bit of binary number (in the truth table) is 0 and normal (i.e. non complemented) if the bit is 1. These terms are designated using **m**.

Following table provide minterm for 3 variables:

a	b	c	Minterm	Designation
0	0	0	a'b'c'	m_0
0	0	1	a'b'c	m_1
0	1	0	a'bc'	m_2
0	1	1	a'bc	m_3
1	0	0	ab'c'	m_4
1	0	1	ab'c	m_5
1	1	0	abc'	m_6
1	1	1	abc	m_7

For four variables, there will be 16 minterms designated by m_0 to m_{15}.

Similarly, OR terms are called maxterms or Standard Sum Term. A maxterm is obtained from OR operator, variable being complemented if the corresponding bit of binary number (in the truth table) is 1 and normal if the bit is 0. These terms are designated by M

We already have maxterms for 2 variables, maxterms for 3 variables will be:

a	b	c	maxterm	designation
0	0	0	a+b+c	M_0
0	0	1	a+b+c'	M_1
0	1	0	a+b'+c	M_2
0	1	1	a+b'+c'	M_3
1	0	0	a'+b+c	M_4
1	0	1	a'+b+c'	M_5
1	1	0	a'+b'+c	M_6
1	1	1	a'+b'+c	M_7

Maxterm for four variables can be obtained in similar manner and they are designated as M_0 to M_{15}. If we put both maxterms and minterms together, we will have following table

a	b	c	minterm	maxterm
0	0	0	a'b'c'	a+b+c
0	0	1	a'b'c	a+b+c'
0	1	0	a'bc'	a+b'+c
0	1	1	a'bc	a+b'+c'
1	0	0	ab'c'	a'+b+c
1	0	1	ab'c	a'+b+c'
1	1	0	abc'	a'+b'+c
1	1	1	abc	a'+b'+c'

By now you might have noted that each maxterm is complement of its corresponding minterm.

For n variables, there will be 2^n, minterms (i.e. Standard Product Term) denoted as m_i, $0 < i < 2^n - 1$, and 2^n maxterms (i.e. Standard Sum Term), their complement, denoted as M_i, $0 < i < 2^n - 1$.

Now let's use this information for algebraic representation of boolean function from the truth table. This is done by forwarding a minterm for each combination of variables which produces a 1 in the function, and then by ORing these minterms.

For eg.

sentation of the ab

F_3	F_4
0	0
1	0
0	1
1	1
1	0
1	0
1	1
0	1

OR

$m_3 + m_4 + m_6$

$F_3(x,y,z) = x'y'z + x'yz + xy'z' + xy'z + xyz'$

OR

$m_1 + m_3 + m_4 + m_5 + m_6$

$F_4(x,y,z) = x'yz' + x'yz + xyz' + xyz$

OR

$m_2 + m_3 + m_6 + m_7$

All of them will result into value 1.

What if we have to represent the function in complement form i.e. for its value 0. Simple, we will OR the minterms, from the truth table, for which function value is 0. So we have,

$F_1' = x'y'z' + x'yz' + x'yz + xy'z$ (the remaining minterms)

On evaluating it for complement, we get

$F_1 = (x+y+z).(x+y'+z).(x+y'+z').(x'+y+z')$

What we get is the **Products of Sum form (POS) of the function F_1.**

POS form of the expression gives value 0, and we work on the complements. POS is formed by ANDi **maxterms.**

POS also can be represented in many ways. One way we have seen, others are

i) $F_1 = M_0.M_2.M_3.M_5$

Instead of using maxterms, we can use designation of it.

ii) $F_1 = \prod(0, 2, 3, 5)$

This one is mathematical representation of the above expression.

Let's look both, SOP and POS expression for same function

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

OR

$m_3 + m_4 + m_6$

$F_3(x,y,z) = x'y'z + x'yz + xy'z' + xy'z + xyz'$

OR

$m_1 + m_3 + m_4 + m_5 + m_6$

$F_4(x,y,z) = x'yz' + x'yz + xyz' + xyz$

OR

$m_2 + m_3 + m_6 + m_7$

All of them will result into value 1.

What if we have to represent the function in complement form i.e. for its value 0. Simple, we will OR the minterms, from the truth table, for which function value is 0. So we have,

$F_1' = x'y'z' + x'yz' + x'yz + xy'z$ (the remaining minterms)

On evaluating it for complement, we get

$F_1 = (x+y+z).(x+y'+z).(x+y'+z').(x'+y+z')$

What we get is the **Products of Sum form (POS) of the function F_1.**

POS form of the expression gives value 0, and we work on the complements. POS is formed by ANDing **maxterms.**

POS also can be represented in many ways. One way we have seen, others are

i) $F_1 = M_0.M_2.M_3.M_5$

Instead of using maxterms, we can use designation of it.

ii) $F_1 = \prod(0, 2, 3, 5)$

This one is mathematical representation of the above expression.

Let's look both, SOP and POS expression for same function

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

SOP expression for F will be

$x'y'z + x'yz'$ or $\Sigma(1,2)$

POS equivalent of it will be found using complement of minterms with value zero, we know this

$(F')' = (x'y'z' + x'yz + xy'z' + xy'z + xyz' + xyz)'$

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

OR

$= \Pi(0,3,4,5,6,7)$

Conversion from one form to another is easy. Short cut for it is :

i) Interchange the symbol (OR to AND / Σ to Π)

ii) List those terms, which are missing in original.

Expressions which were written till now, are in Canonical form. In this form,every term of the function should have all the literals. When, minterms or maxterms from truth table, are used for representation, the function is in canonical form only, as terms contains all variables (may be in any form). Since we can use either minterms or maxterms to form expression, we can have **Canonical SOP** expression or **Canonical POS** expressions.

Till now, we were given truth table and we were representing Boolean Function in algebraic form. Vice versa of it, i.e. for a given Boolean function in algebraic representation, we can represent it in Truth table. This was taken up in details in previous chapter.

What we have worked on was Canonical Boolean functions. There is another form of representation, also - both for SOP & POS. It is known as **reduced/simplified form.**

Simplified expressions are usually more efficient and effective when implemented in practice. Also simpler expressions are easier to understand, and less prone to errors. So the Canonical expressions are simplified/minimized to obtain a reduced expression (Standard form). This expression is then used for implementation of circuits in computers, *we will learn it in next chapter.*

Boolean function may be reduced in many ways, but we will be taking up the following two ways of reduction in this course.

1. Algebraic Manipulations
2. Karnaugh Map.

Algebraic reduction of Boolean function/expression: in this reduction, we apply certain algebraic axioms and theorems to reduce the number of terms or number of literals in the expression. In this way of reduction, there are no fixed rules that can be used to minimize the given expression. It is left to individual's ability to apply axioms or theorems for minimizing the expression.

Example 1:

F (a, b)	= (a+b)' + ab'	
	= a'b'+ab'	by De Morgan's theorem
	= (a'+a)b'	by Distribution
	= 1.b'	by Complement
	= b'	by Identity

Example 2:

F (a, b)	= ab+ ab'+ a'b	
	= a(b+b') + a'b	by Distribution
	= a.1 + a'b	by complement
	= a + a'b	by Identity
	= a+b	by absorption

Algebraic transformation, can always be used to produce optimal reduced form, but does not guarantee the reduced form. Also, it is not necessary to reach to same transformation, when worked on by different group of people. However, there are other methods using which, we will always reach to optimal and same solution every time.

Before moving ahead with other methods of reduction lets wait, to see how vice versa will be done, conversion of reduced from to Canonical form? Yes you guessed right, using theorems and axioms. Lets do it for 2nd example

$F(a,b) = a + b$	Reduced expression
$= a.1 + b.1$	by Identity
$= a.(b + b') + b.1$	by Complement
$= a.b + a.b' + b.1$	by Distribution
$= a.b + a.b' + 1.b$	by Commuting
$= a.b + a.b' + (a + a').b$	by Complement
$= a.b + a.b' + a.b + a'.b$	by Distribution
$= a.b + a.b' + a'.b$	by Indempotancy

Now let's learn other way of reduction.

Karnaugh Map (K-map):

A K-map is nothing more than a special form of truth table representation useful for reducing logic functions into minimal Boolean expressions. Karnaugh map was developed by Maurice Karnaugh, at Bell Labs in 1953. The map reduces boolean expression more quickly and easily, as compared to algebraic reduction.

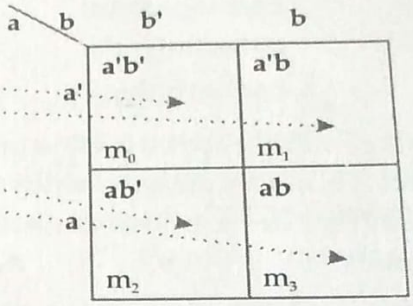
Using K-map for reduction is 4 step process

1. *Drawing K-map*
2. *Filling it, i.e. representing the boolean expression in map*
3. *Grouping the terms for reducing purpose*
4. *Reading the reduced expression from map*

1. Drawing K-map

A K-map is a diagram, made up of squares, where each square represents one minterm. So in a two variable map there will be 4 squares and for three variables, map will contain 8 squares. K-map diagram for two variables - a,b, will look like:

a	b	minterm
0	0	a'b'
0	1	a'b
1	0	ab'
1	1	ab



For three variables - a, b,c, we can draw it as:

bc / a	b'c'	b'c	bc	bc'
a'	m_0	m_1	m_3	m_2
a	m_4	m_5	m_7	m_6

OR

ab / c	c'	c
a'b'	m_0	m_1
a'b	m_2	m_3
ab	m_6	m_7
ab'	m_4	m_5

r four variables - a, b, c, d

cd / ab	c'd'	c'd	cd	cd'
a'b'	m_0	m_1	m_3	m_2
a'b	m_4	m_5	m_7	m_6
ab	m_{12}	m_{13}	m_{15}	m_{14}
ab'	m_8	m_9	m_{11}	m_{10}

OR

ab / cd	a'b'	a'b	ab	ab'
c'd'	m_0	m_4	m_{12}	m_8
c'd	m_1	m_5	m_{13}	m_9
cd	m_3	m_7	m_{15}	m_{11}
cd'	m_2	m_6	m_{14}	m_{10}

The cells of the map are marked by the minterms as explained in first map.

For eg.

If you look at the square assigned m_4 (minterm), it corresponds to binary no. 0100 of the truth table.

Now to understand, why the squares have been marked, as they are?

If you look at any two adjacent cell in the map, they differ only by one variable, which is normal in one cell and complemented in other or vice-versa. For eg. From m_4 to m_5, only a changes its state from complemented to normal. And we know, using Boolean axiom ($a'+a=1$) that we can eliminate the variable from the resultant reduced term.

2. Filling the K-map

Once a K-map is drawn, next is to fill the map i.e. **mark the function in the map.** For representation, the function should be in canonical SOP form. If we have the function represented in truth table, then it will be in canonical form only. Otherwise, we know how to convert it. We will mark 1 in all those cells of K-map, for which there is a minterm in the Boolean function/expression.

3. Grouping the minterms

Next is grouping the 1s in the map. We group 1s of adjacent cells.

Following are the rules for grouping. Remember our goal should be to maximize the size of group (i.e. no. of 1s included in the group) and to minimize the number of groups.

- A group must contain 1, 2, 4, 8, 16, ... cells, i.e. in the power of 2.

1	1
0	0

Correct

0	0	0	0
0	1	1	1

Incorrect

- Cells which are grouped, should be adjacent cells, i.e. horizontal or vertical, not diagonal.

0	1
1	1

Correct

0	1
1	0

Incorrect

- Groups should not include any cell containing zero

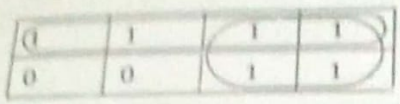
0	0
1	1

Correct

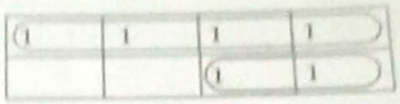
0	0
1	0

Incorrect

- Each group should be as large as possible

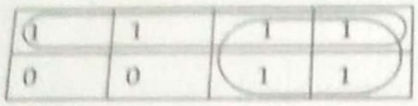


Correct

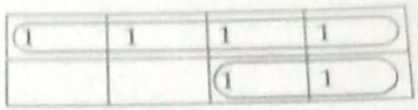


Incorrect

- Groups may overlap

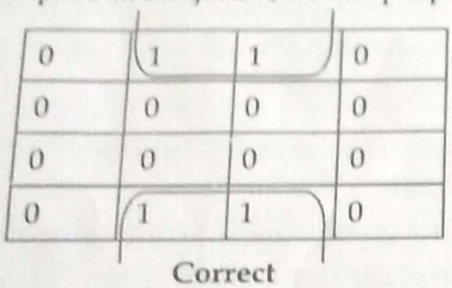


Correct

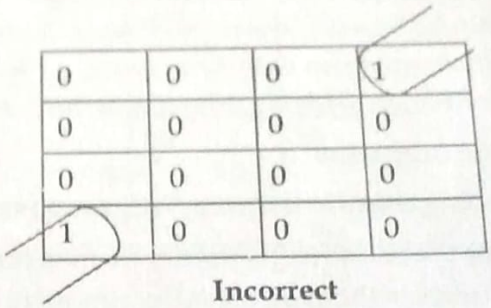


Incorrect

- Groups may wrap around the table. The left most cell(s) in a row may be grouped with rightmost cell(s), and the top cell(s) in a column may be grouped with the bottom cells(s). Cells in corners of the map are also adjacent, for that purpose



Correct



Incorrect

- There should be as few groups as possible, as long as this does not contradict any of the previous rule. Once the grouping is done, last step is

Reading the reduced expression from K-map

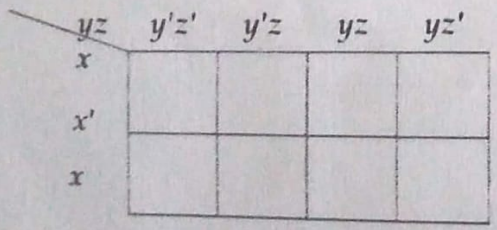
- Each group corresponds to one product term, from K-map
- The term is determined by finding the common literals in that group, i.e. the variables which change their state (from normal to complement or vice-versa) are to be omitted in resultant product term.

What we have to do is, find the variable(s) listed on top and / or side, which are the same for entire group and ignore variable(s) which are not same in the group.

Using the above strategy, write the result. Lets look at some examples

Simplify the Boolean Function F(x,y,z) = x'yz + xy'z' + xyz + xyz'

Step 1: Drawing the K-map



Step 2: Marking of function in map

x \ yz	y'z'	y'z	yz	yz'
x'	0	0	1	0
x	1	0	1	1

Step 3: Grouping

x \ yz	y'z'	y'z	yz	yz'
x	0	0	1	0
x'			1	
x	1	0	1	1

Step 4: Reading reduced expression from map

For Group 1, variable **x** is changing its state and for Group 2, variable **y** is changing its state. So x in 1st group and y in 2nd group will be omitted. The resultant expression will contain yz, because of 1st group and xz' for 2nd. As this is SOP expression so both the terms will be joined through OR operator. Resultant reduced function will be

$F = yz + xz'$

Example:

$F(w,x,y,z) = \Sigma(5,7,13,15)$

wx \ yz	y'z'	y'z	yz	yz'
w'x'	0 0	0 1	0 3	0 2
w'x	0 4	1 5	1 7	0 6
wx	0 12	1 13	1 15	0 14
wx'	0 8	0 9	0 11	0 10

$F(w,x,y,z) = xz$

Now let's take an example where the function is represented using Truth table, instead of algebraic representation

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Using minterm designation we will represent the function in K map.

x \ yz	y'z'	y'z	yz	yz'
x'	0	1	0	0
x	0	1	1	1

$f(x,y,z) = y'z + xy$

So far we were working with SOP expression/function. However, it is possible to reduce POS function also. Although we know k-map naturally reduces the expression is SOP form. In POS reduction, the approach is much the same except for, instead of minterms, we work with maxterms. We know a maxterm results into zero.

Now if we look at drawing of k-map it will be following for three variable

A \ BC	B+C	B+C'	B'+C'	B'+C
A	M_0	M_1	M_3	M_2
A'	M_4	M_5	M_7	M_6

OR

AB \ C	C	C'
A+B	M_0	M_1
A+B'	M_2	M_3
A'+B'	M_6	M_7
A'+B	M_4	M_5

Similarly a four variable map may be created.

Next is representing the Boolean expression/function in K- map. Representation remains same except, now we represent 0 instead of 1.

For reduction, 0's from the k-map will be grouped in same fashion as 1's were grouped, and reduced expression will be represented using maxterms.

Example:

$f(w, x, y, z) = \Pi(10, 11, 14, 15)$

Using maxterm numbers, we will represent the function, i.e. 0 will be represented in map. For reduced POS expression 0's will be grouped

wx \ yz	y'z'	y'z	yz	yz'
w'x'	1	1	1	1
w'x	1	1	1	1
wx	1	1	0	0
wx'	1	1	0	0

$f(w, x, y, z) = w'+y'$

f (a, b, c) = a'b'c + abc' +abc is an Sop expression, whose POS reduced form is denied.

a \ bc	b+c	b+c'	b'+c'	b'+c
a	0	0	0	1
a'	0	0	1	1

As we are working for POS, 0's will be grouped and every group will result into sumterm.

f(a,b,c) = (a+c').b

A'+B'