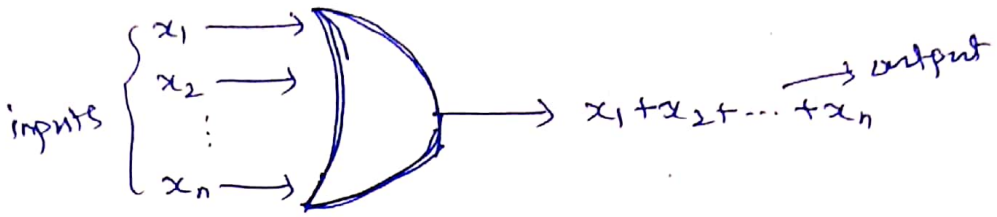
Logic gates in Boolean Algebra

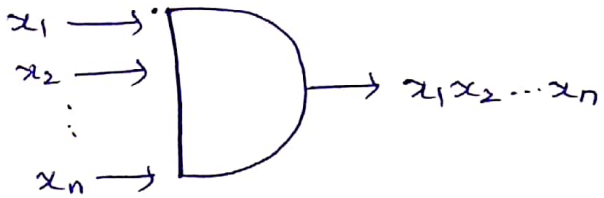
Consider the three basic types of gates that are used to construct combinational circuits:

(i) OR gate :-



This gate receives two or more inputs (Boolean Variables) and produces an output equal to the Boolean Sum of the values of the input variables.

(ii) AND gate



This gate receives two or more inputs and produces an output equal to their Boolean product.

(iii) NOT gate

This gate accepts only one input (value of one Boolean variable) and produces the complement of this value as the output.

Combination of Gates

Combinational circuits are formed by interconnecting the basic gates. One method is to indicate the inputs separately for each gate (Fig a). The other method is to use branchings that indicate all the gates that use a given input (Fig b).

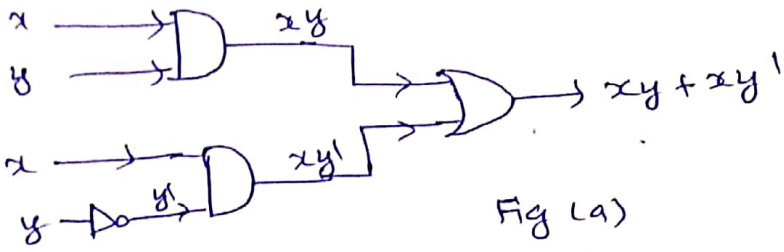


Fig (a)

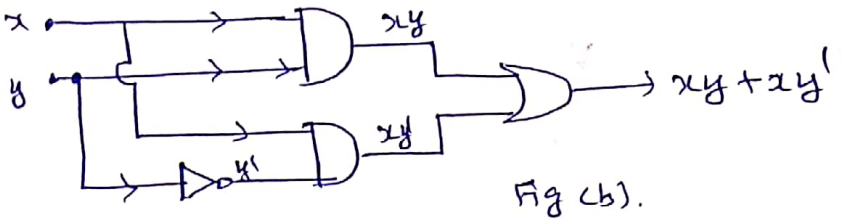
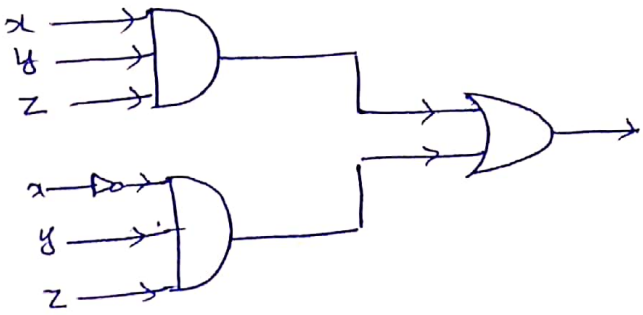


Fig (b).

Example ① Find the output of the network given in Fig (i) and design a simpler network having the same output.

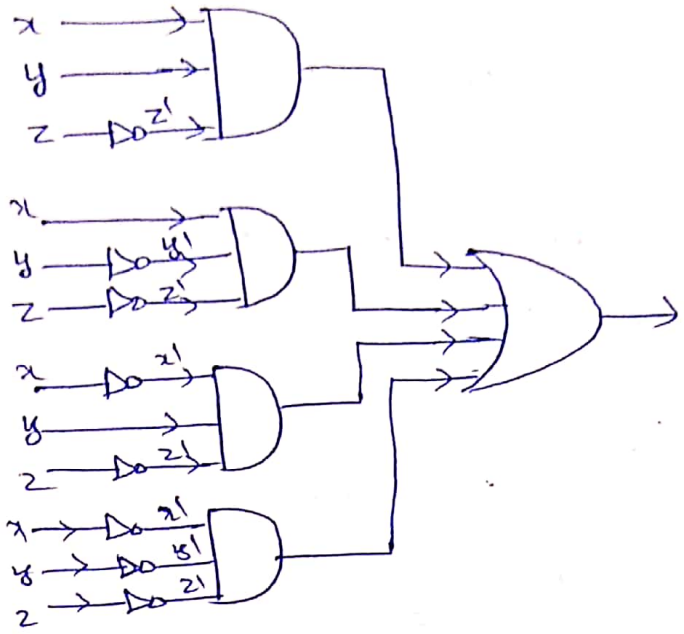


Soln:- The output of the upper AND gate is xyz. The output of the NOT before the lower AND gate is x' and so output of the lower AND gate is $x'yz$.

$$\therefore \quad xyz + x'yz = (x + x')yz$$
$$= 1(yz)$$
$$= yz$$

$\therefore$ Simple network is

② Find the output of the network given below and design a simpler network having the same output.



Sol:- ∴ The outputs of the AND gates are

xyz', $xy'z'$, $x'yz'$, $x'y'z'$

the output of the OR gate is

$$xyz' + xy'z' + x'yz' + x'y'z'$$

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

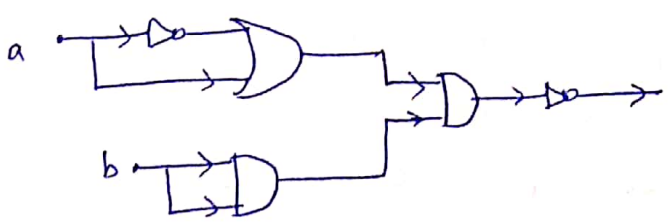
$$= xz' + x'z'$$

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

$$= z'$$

∴ The simpler gate is

③ H.W



Karnaugh Map Method for Minimization of Boolean function

Karnaugh Map (or) K-Map

Karnaugh map method is a graphical method for Simplifying Boolean expression involving Six (or) fewer Variables that are expressed in the sum of product forms and that represent combinational circuits.

A Karnaugh map is a diagram consisting of Squares. If the Boolean algebra expression contains n-variables, the corresponding K-map have 2^n squares, each of which represents a minterm. A '1' is placed in the square representing a minterm if it is present in the given expression. A '0' is placed in the square that corresponds to the minterm not present in the expression.

The Simplified Boolean expression that represents the output is then obtained by combining or grouping adjacent squares that contain 1.

To identify the adjacent cells (squares) in the K-Map for grouping, the following points may be borne in mind:

(1) The number of cells in a group must be a Power of 2. ie., 2, 4, 8, 16, etc..

(2) A cell containing 1 may be included in any number of groups.

(3) To minimise the expression to the maximum possible extent, largest possible groups must be preferred. Viz., a group of two cells should not be considered, if these cells can be included in a group of four cells and so on.

(4) Adjacent cells exist not only within the interior of the K-map, but also at the extremes of each column and each row viz. the top cell in any column is adjacent to the bottom cell in the same column. The left most cell in any row is adjacent to the rightmost cell in that row.

K-Map for 1-Variable (x)

2^1 = 2 squares

0	1
x'	x

2-Variables

2^2 = 4 squares

	y'	y
x'	$x'y'$	$x'y$
x	xy'	xy

x \ y	0	1
0	$x'y'$	$x'y$
1	xy'	xy

3 - Variables

	$x'y'$	$x'y$	xy	xy'
z'	$x'y'z'$	$x'yz'$	xyz'	$xy'z'$
z	$x'y'z$	$x'yz$	xyz	$xy'z$

z \ xy	00	01	11	10
0	$x'y'z'$	$x'yz'$	xyz'	$xy'z'$
1	$x'y'z$	$x'yz$	xyz	$xy'z$

4-Variables $2^4 = 16$ cells

	$y'z'$ (00)	$y'z$ (01)	yz (11)	yz' (10)
$w'x'$ (00)	$w'y'z'x'$	$w'x'yz$	$w'x'yz$	$w'x'yz'$
$w'x$ (01)	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
wx (11)	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
wx' (10)	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

Procedure for minimisation of Boolean expression using K-maps

1) K-map is first constructed by placing 1's in those squares corresponding to the minterms present in the expression and 0's in the other squares.

2) All those 1's that cannot be combined with any other 1's are identified and looped.

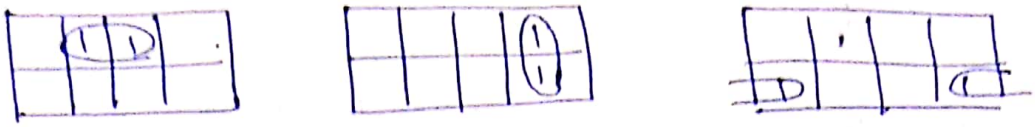
3) All those 1's that combine in a loop of two but do not make a loop of four are looped.

4) All those 1's that combine in a loop of four but do not make a loop of eight are looped.

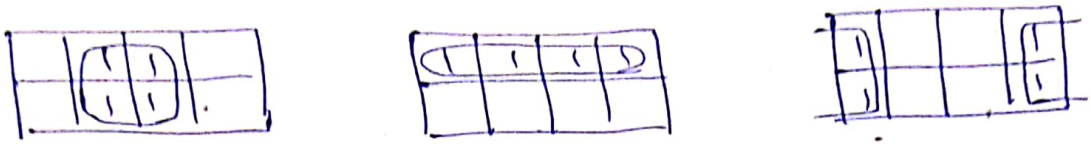
5) The process is stopped when all the 1's have been covered.

6) The simplified expression is the sum of all the terms corresponding to various loops.

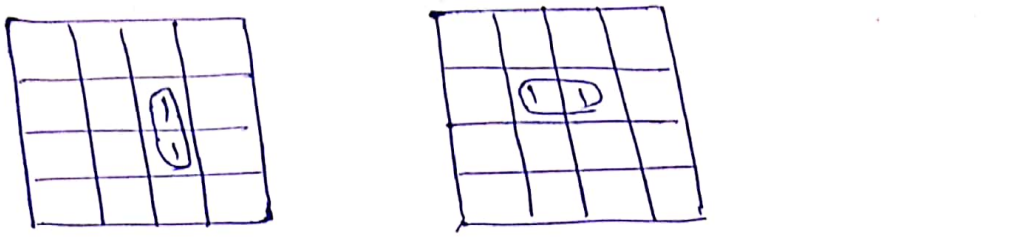
All possible forms of basic loops of 2 cells for 3 variables

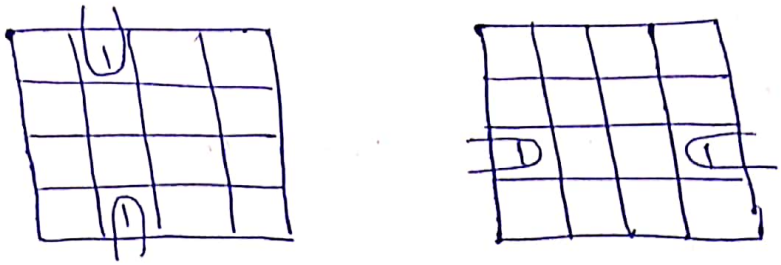


All possible forms of basic loops of 4 cells for 3 variables

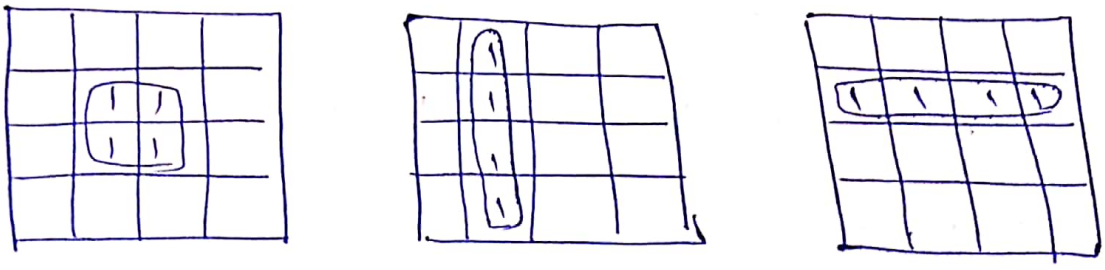


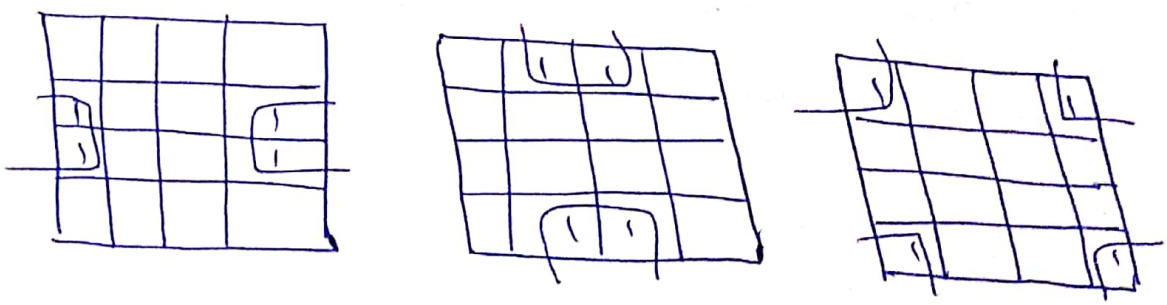
All possible forms of 2 cells loops for 4 variables





All possible forms of 4 cells loops for 4 variables





Example

① Simply the Boolean expression $f(x,y) = xy' + xy$ using the K-map.

Soln:-

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

K-map for 2-Variables

	y'	y
x'	$x'y'$	$x'y$
x	xy'	xy

x \ y	0	1
0	0	0
1	1	1

$\rightarrow x$

$\therefore \quad xy' + xy = x$

② Simplify the Boolean expression $f(x,y,z) = xyz' + xyz$.

Soln:-

	$x'y'$	$x'y$	xy	xy'
z'	$x'y'z'$	$x'yz'$	xyz'	$xy'z'$
z	$x'y'z$	$x'yz$	xyz	$xy'z$

z \ xy	00	01	11	10
0	0	0	1	0
1	0	0	1	0

$\therefore \quad xyz' + xyz = xy.$

③ $f(a,b,c) = ab'c' + abc' + abc + ab'c + a'b'c$

Soln:-

	$b'c'$	$b'c$	bc	bc'
a'	$a'b'c'$	$a'b'c$	$a'bc$	$a'bc'$
a	$ab'c'$	$ab'c$	abc	abc'

	00	01	11	10
0	0	1	0	0
1	1	1	1	1

$\therefore \quad f(a,b,c) = a + b'c$ //