CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test 1 – November 2019

Sub:	Computer Organization					Code:	18MCA15
Date:	*19-11-2019*	*Duration:*	*90 mins*	*Max Marks:*	*50*	*Sem:* I A	*Branch:* *MCA*

Answer any five of the following **5 x 10 = 50 Marks**

Q1 Convert the following:

(i) $(67.6875)_{10} = (?)_2$ (ii) $(75.62)8 = (?)_2$

iii) $(10110001101011)_2 = (?)_{16}$

iv) $(B65F)_{16} = (?)_{10}$ v) $(306.D)_{16} = (?)_2$

Sol :

$(67.6875)_{10} = (?)_2$ = 1000011.1011

$(75.62)_8 = (111101.11001)_2$

$(B65F)_{16} = (11 \times 16^3) + (6 \times 16^2) + (5 \times 16^1) + (15 \times 16^0) = (46687)_{10}$

$(306.D)_{16} =$ 1100000110.1101

(2) Perform the following operation

i) Using 9's & 10's complement subtract 1234 from 4567

Sol.

10's complement, it is relatively easy to find out the 10's complement after finding out the 9's complement of that number. We have to add 1 with the **9's complement** of any number to obtain the desired 10's complement of that number. Or if we want to find out the 10's complement directly, we can do it by following the following formula, (10^n – number), where n = number of digits in the number. An example is given below to illustrate the concept of obtaining 10's complement.

A = 4567

B = 1234

We need to find out A – B

9's complement of B

8765
Adding 9's complement of B with A

```
 8765
 4567
-----
13332
```

Adding the carry with the result we get
3333
Now the answer is – 3333
NB: If there is no carry the answer will be – (9's complement of the answer)
Subtraction by 10's complement
Again we will show the procedure by an example
Taking the same data
A = 215
B = 155
10's complement of B = 845
Adding 10's complement of B to A

```
  845
 215 (+)
-------
 1060
```

In this case the carry is omitted
The answer is 60
Taking the other example
A = 4567
B = 1234
10's complement of B = 8766
Adding 10's complement of B with A

```
 8766
 4567(+)
--------
13333
```

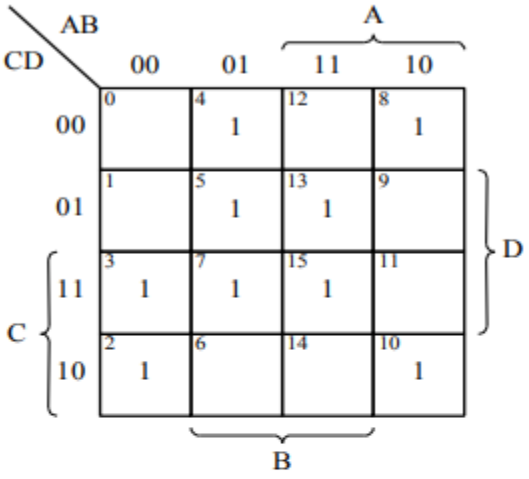
To get the answer the carry is ignored
So, the answer is – 3333

Q3. Using K-map simplify the Boolean Function

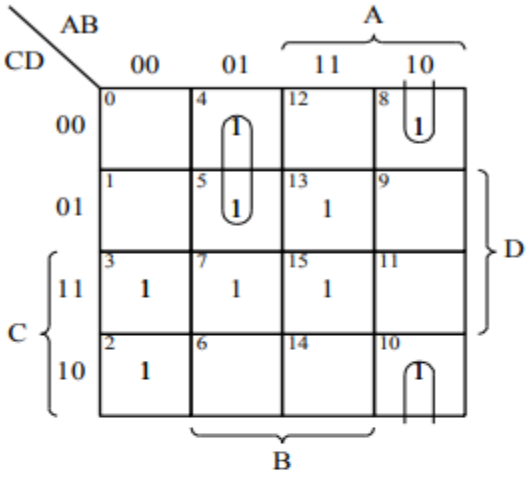
i) F(**A, B, C, D**) = ∑(2,3 4,5,7, 8, 10, 13, 15)

Sol.

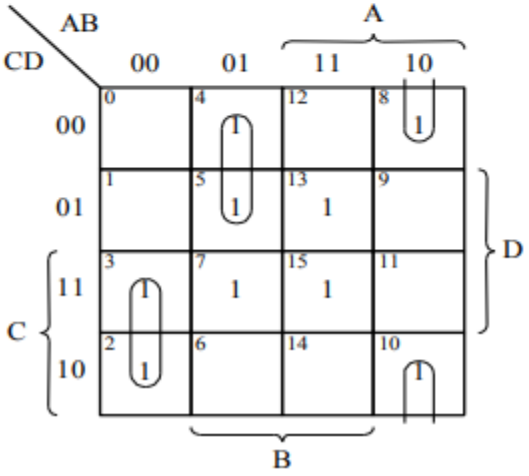
$$f(A,B,C,D) = \sum m(2,3,4,5,7,8,10,13,15)$$

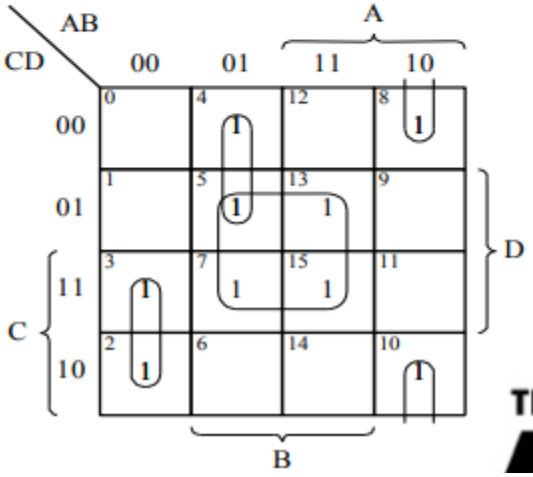


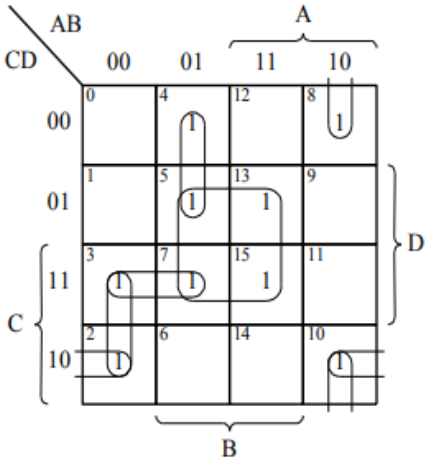
(a)



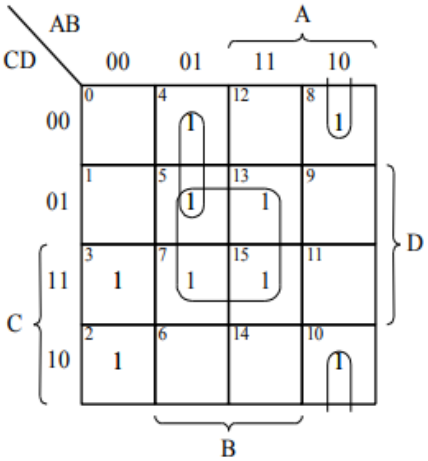
(b)



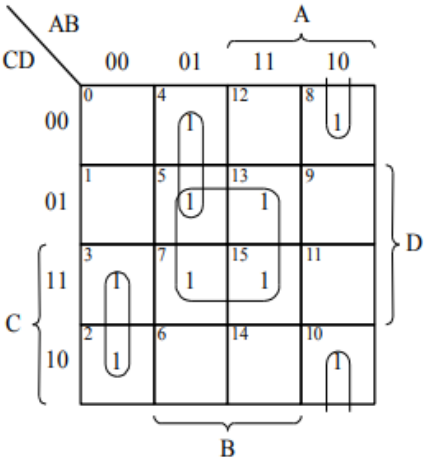




(a)



(b)



(c)

Q4. State the following Boolean laws- [closure, associate law, communicative law, identity law, inverse, distributive law

Sol :

As well as the logic symbols "0" and "1" being used to represent a digital input or output, we can also use them as constants for a permanently "Open" or "Closed" circuit or contact respectively.

A set of rules or Laws of Boolean Algebra expressions have been invented to help reduce the number of logic gates needed to perform a particular logic operation resulting in a list of functions or theorems known commonly as the **Laws of Boolean Algebra**.

Boolean Algebra is the mathematics we use to analyse digital gates and circuits. We can use these "Laws of Boolean" to both reduce and simplify a complex Boolean expression in an attempt to reduce the number of logic gates required. *Boolean Algebra* is therefore a system of mathematics based on logic that has its own set of rules or laws which are used to define and reduce Boolean expressions.

The variables used in **Boolean Algebra** only have one of two possible values, a logic "0" and a logic "1" but an expression can have an infinite number of variables all labelled individually to represent inputs to the expression,

Some basic laws for Boolean Algebra

1) **A . 0 = 0 where A can be either 0 or 1.**
2) **A . 1 = A where A can be either 0 or 1.**
3) **A . A = A where A can be either 0 or 1.**
4) **A . $\bar{A}$ = 0 where A can be either 0 or 1.**
5) **A + 0 = A where A can be either 0 or 1.**
6) **A + 1 = 1 where A can be either 0 or 1.**
7) **Inverse law:**
   A + A' = 1
   A.A'= 0
8) **Identity law:**
   A.1 = A
   A + 0 = A
9) **Cumulative Law**
10) A + B = B + A where A and B can be either 0 or 1.
    A . B = B . A where A and B can be either 0 or 1.
    According to Cumulative Law, the order of OR operations and AND operations conducted on the variables makes no differences.
11) **Associative Laws**

This law is for several variables, where the OR operation of the variables result is same though the grouping of the variables. This law is quite same in case of AND operators.

$$(A + B) + C = A + (B + C)$$
$$(A . B) . C = A . (B . C)$$

12) **Distributive Laws**

This law is composed of two operators, AND and OR.

$$A . (B + C) = A . B + A . C$$

Let us show one use of this law to prove the expression $A + B.C = (A + B).(A + C)$

Proof:

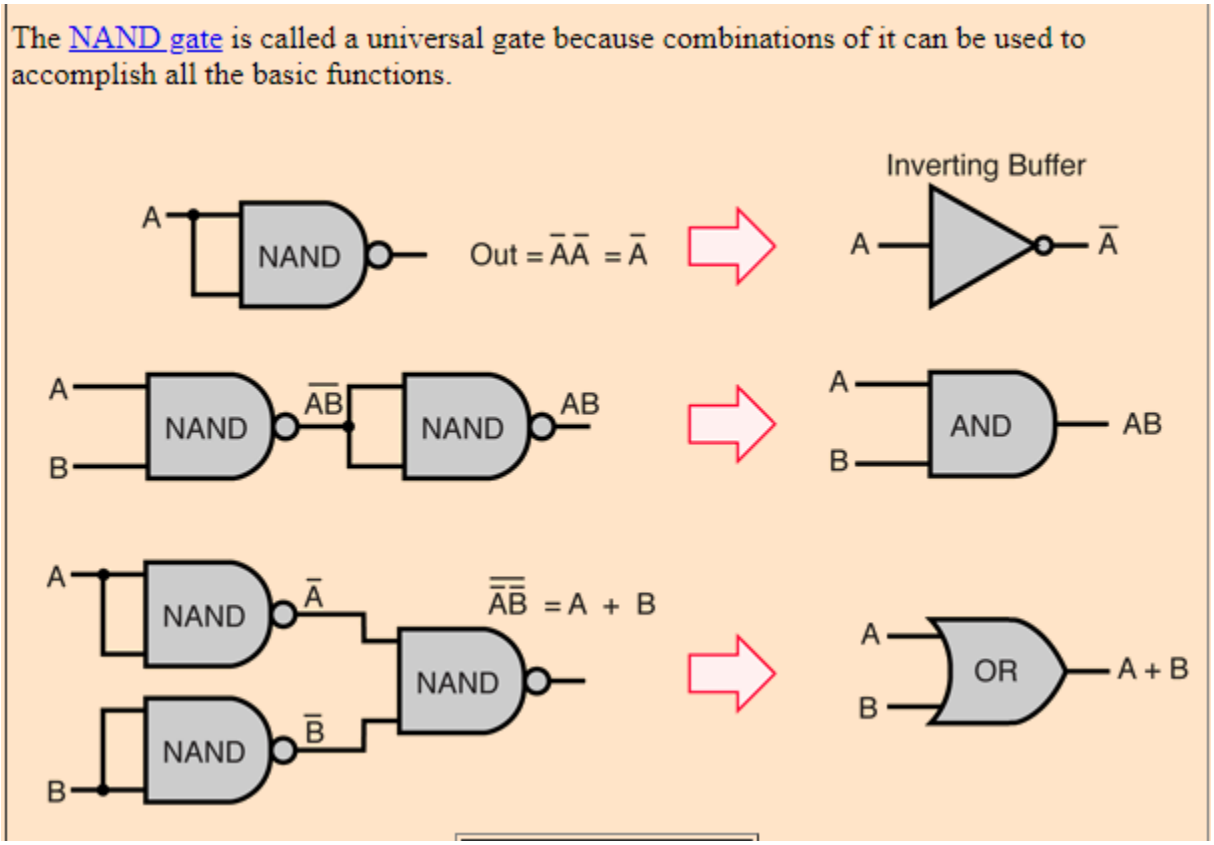
$$A + B.C = A.1 + B.C \ [Since,\ A.1 = A]$$
$$= A.(1 + B) + B.C \ [Since,\ B + 1 = 1]$$
$$= A.1 + AB + BC$$
$$= A.(1 + C) + AB + BC \ [Since,\ A.A = A.1 = A]$$
$$= A.(A + C) + B.(A + C)$$
$$= (A + B).(A + C)$$

Q6. Justify why NAND and NOR gates are known as universal gates. Derive all basic gates using NAND gates

Sol.

NAND and NOR are called universal gates because all the other gates like and, or, not and xnor can be derived from it.

Although and, or and not are basic gates but each of their functionality can be derived using nand and not as well . One more important aspect because of this AOI logic can be converted to either Nand or Nor logic.



Q 7 State and Prove the De Morgan's Theorem using Logic Circuit & Truth Table

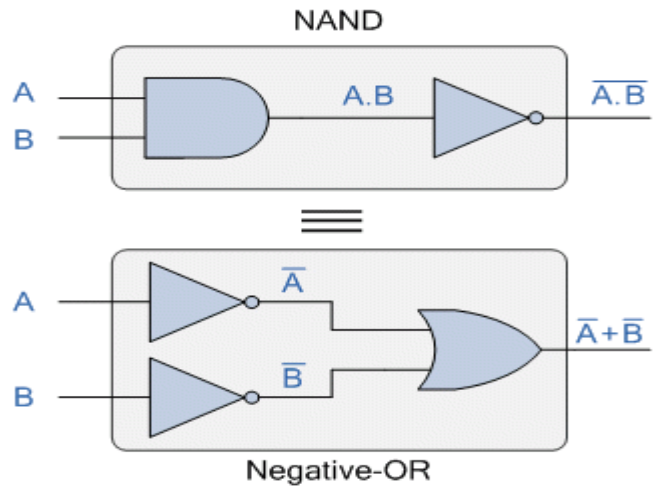
Sol.

DeMorgan's Theorems are basically two sets of rules or laws developed from the Boolean expressions for AND, OR and NOT using two input variables, A and B. These two rules or theorems allow the input variables to be negated and converted from one form of a Boolean function into an opposite form.

1) First Theorem

DeMorgan's first theorem states that two (or more) variables NOR´ed together is the same as the two variables inverted (Complement) and AND´ed, while the second theorem states that two (or more) variables NAND´ed together is the same as the two terms inverted (Complement) and OR´ed. That is replace all the OR operators with AND operators, or all the AND operators with an OR operators.

$$\overline{A.B} = \overline{A}+\overline{B}$$

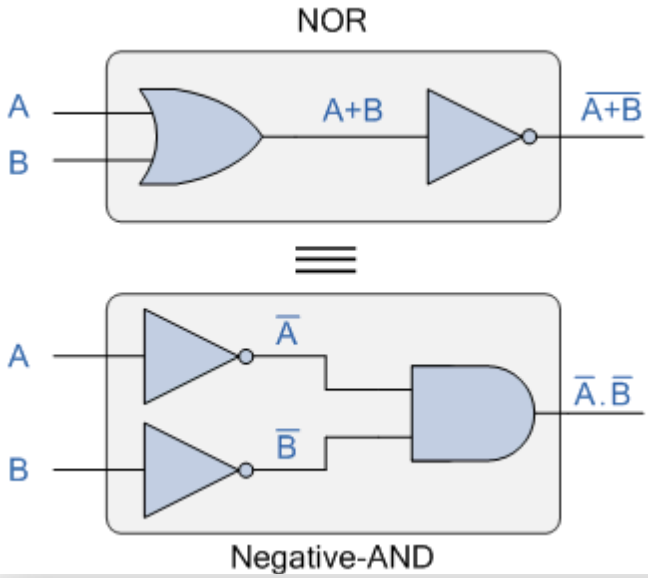


Inputs		Truth Table Outputs For Each Term				
B	A	A.B	$\overline{A.B}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	0	**1**	1	1	**1**
0	1	0	**1**	0	1	**1**
1	0	0	**1**	1	0	**1**
1	1	1	**0**	0	0	**0**

2) Second Theorem

DeMorgan's Second theorem proves that when two (or more) input variables are OR'ed and negated, they are equivalent to the AND of the complements of the individual variables. Thus the equivalent of the NOR function and is a negative-AND function proving that $\overline{A+B} = \overline{A}.\overline{B}$ and again we can show this using the following truth table.

$$\overline{A+B} = \overline{A}.\overline{B}$$



Negative-AND

Inputs		Truth Table Outputs For Each Term				
B	A	A+B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	0	**1**	1	1	**1**
0	1	1	**0**	0	1	**0**
1	0	1	**0**	1	0	**0**
1	1	1	**0**	0	0	**0**

Q8. a) Explain about sum of product and product of sum simplification with example?

b) Explain about Maxterms and Minterms with an example.

Sol.

Sum Of Product (SOP)

Sum of Product is the abbreviated form of **SOP**. Sum of product form is a form of expression in Boolean algebra in which different product terms of inputs are being summed together. This product is not arithmetical multiply but it is Boolean logical AND and the Sum is Boolean logical OR.

To understand better about SOP, we need to know about min term.

Min Term

Minterm means the term that is true for a minimum number of combination of inputs. That is true for only one combination of inputs.

Since AND gate also gives True only when all of its inputs are true so we can say min terms are AND of input combinations like in the table given below.

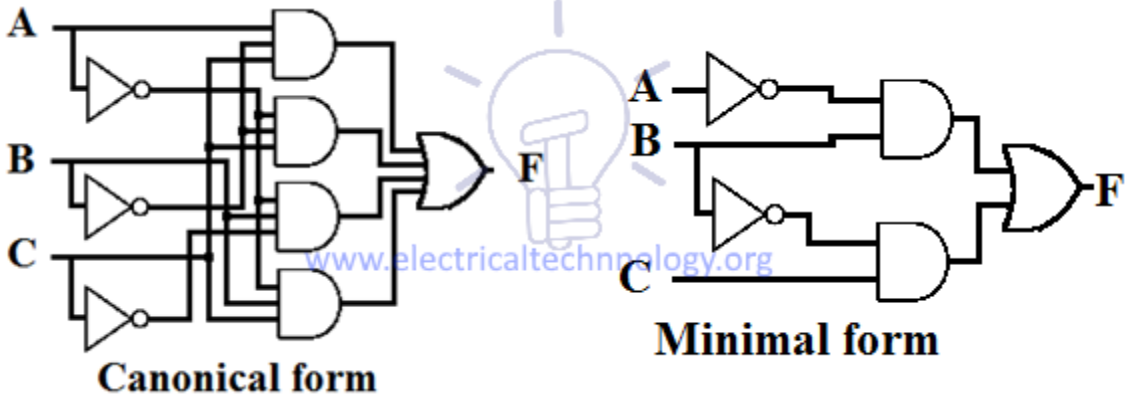
A	B	C	Min term
0	0	0	$m_0 = \bar{A}\,\bar{B}\,\bar{C}$
0	0	1	$m_1 = \bar{A}\,\bar{B}\,C$
0	1	0	$m_2 = \bar{A}\,B\,\bar{C}$
0	1	1	$m_3 = \bar{A}\,B\,C$
1	0	0	$m_4 = A\,\bar{B}\,\bar{C}$
1	0	1	$m_5 = A\,\bar{B}\,C$
1	1	0	$m_6 = A\,B\,\bar{C}$
1	1	1	$m_7 = A\,B\,C$

SOP expression implements 2 level AND-OR design in which the **1st** level gate is AND gate following the **2nd** level gate which is OR gate. Schematic design of SOP expression needs a group array of AND gates & one OR gate.

Every SOP expression has somewhat same designing i.e. all the inputs goes through AND gate and then the output of these AND gates flow through an OR gate as shown in the figure given below.

The number of inputs and the number of AND gates depend upon the expression one is implementing.

Example of designs of canonical and minimal SOP expression for a function is given below.



Max Term

Maxterm means the term or expression that is true for a maximum number of input combinations or that is false for only one combination of inputs.

Since OR gate also gives false for only one input combination. So Maxterm is OR of either complemented or non-complemented inputs.

3 inputs have 8 different combinations so it will have 8 maxterms. Maxterms are denoted by capital M and decimal combination number In the subscript as shown in the table given above.

In maxterm, each input is complemented because Maxterm gives '0' only when the mentioned combination is applied and Maxterm is complement of minterm.

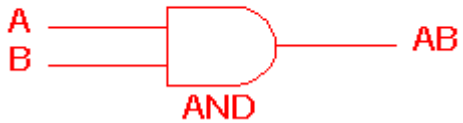
A	B	C	Max term
0	0	0	$M_0 = A + B + C$
0	0	1	$M_1 = A + B + \bar{C}$
0	1	0	$M_2 = A + \bar{B} + C$
0	1	1	$M_3 = A + \bar{B} + \bar{C}$
1	0	0	$M_4 = \bar{A} + B + C$
1	0	1	$M_5 = \bar{A} + B + \bar{C}$
1	1	0	$M_6 = \bar{A} + \bar{B} + C$
1	1	1	$M_7 = \bar{A} + \bar{B} + \bar{C}$

Q9. Describe all logic gates with symbol, truth table and logic expression?

Sol :

Digital systems are said to be constructed by using logic gates. These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates. The basic operations are described below with the aid of truth tables.

AND gate :
The AND gate is an electronic circuit that gives a **high** output (1) only if all its inputs are high. A dot (.) is used to show the AND operation i.e. A.B. Bear in mind that this dot is sometimes omitted i.e. AB



2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

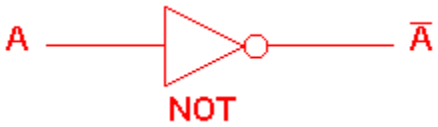
OR gate :

The OR gate is an electronic circuit that gives a high output (1) if **one or more** of its inputs are high. A plus (+) is used to show the OR operation

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT gate

The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is A, the inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs. The diagrams below show two ways that the NAND logic gate can be configured to produce a NOT gate. It can also be done using NOR logic gates in the same way.



NOT gate	
A	$\overline{A}$
0	1
1	0

NAND gate

Q10. Write Briefly explain K-Map and its advantages?

Sol.

implification of Boolean expressions is an important step while designing any digital system. **Karnaugh Maps** or **K-maps** is one among such simplification technique, introduced by Maurice Karnaugh in 1953, which is graphical in nature. This method of minimizing the logical expressions is most suitable when the number of variables involved is less than or equal to four. This is because, K-map employs the use of two-dimensional tables to simplify the expressions, whose size increases at a very high rate with the increase in the number of variables. This fact is further established by Figure 1 which shows the K-maps for two, three and four variables in order.

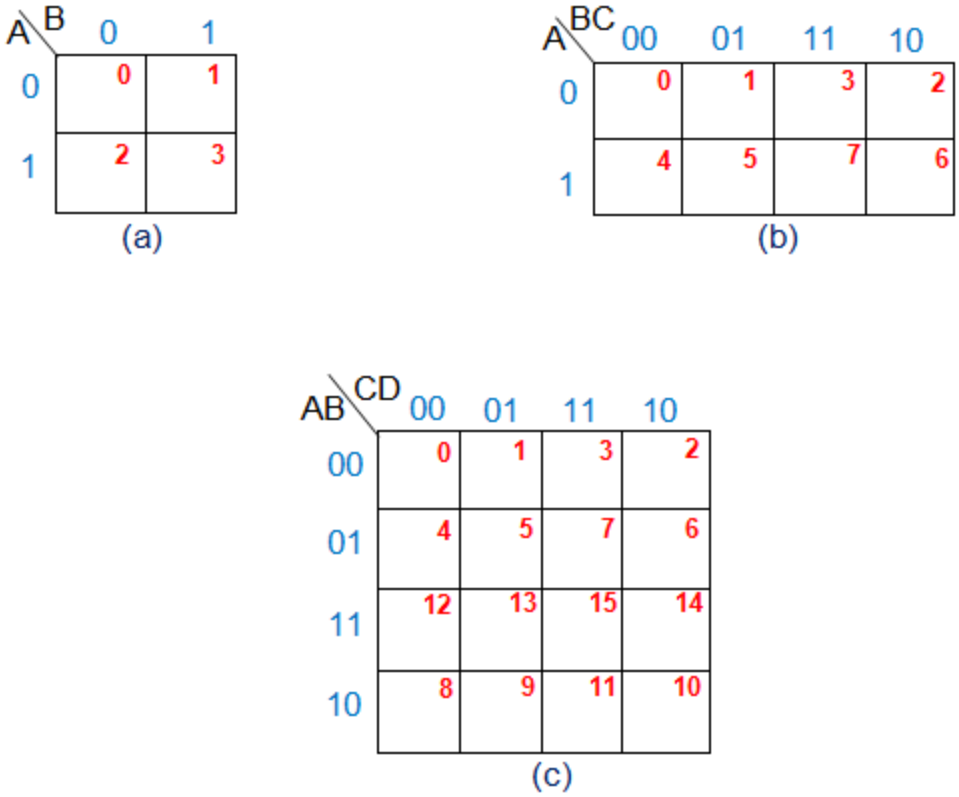


Figure 1 Karnaugh Maps for (a) Two Variables (b) Three Variables (c) Four Variables

From the figure, it is evident that the number of cells in the K-map is a function of number of inputs. In general, if there are n inputs, then the corresponding K-map has to be of 2n cells. For example, if the number of input variables is 2, then we have to consider a K-map with 4 (=22) cells, while if there are 3 input variables, then we require a 8 (=23) cell K-map, and similarly for 4 inputs one gets 16 (=24) cell K-map and so on.

All the K-maps irrespective of their size, are seen to possess a generalized structure (Figure 1). Every K-map has a set of input variables represented at its top left-corner (black alphabets). These are the input variables which are involved in the logical expression which needs to be simplified. The value(s) of these variable(s) is/are shown in binary along their respective sides (combination of zeros and ones shown in blue). Here, it is seen that the binary patterns of any two adjacent cells differ by just a single bit. This kind of encoding scheme is referred to as gray code and is employed in order to ease the process of grouping which inturn minimizes the logical expression.

Further these binary sequences are seen to assign a definite input bit pattern for each K-map cell, whose decimal equivalent is shown in red numbers inside each of them. For example, third cell of the first row in Figure 1b corresponds to the input bit pattern ABC = 011 which is represented by its decimal equivalent 3.

K-map simplification procedure is initiated by entering the values of the output variable (either for sum-of-products, SOP or for product-of-sums, POS) in appropriate K-map cells. Then one has to group the maximum number of 'ones' (incase of SOP) or 'zeros' (incase of POS). These groups should necessarily be in powers of 2 and should be carried on in descending order only. For instance, if there are 8 cells in the K-map, then, first, try grouping for 8 (=23), then for 4 (=22), next for 2 (=21) and lastly consider the isolated terms. After this, each group is expressed interms of the combination of input variables which correspond to the common binary values along the associated rows and columns. Finally, these are used to express the output of the logical expression.

Advantages of Karnaugh Map

Advantages of K-map

1. K-map simplification does not demand for the knowledge of Boolean algebraic theorems.
2. Usually it requires less number of steps when compared to algebraic minimization technique.

Disadvantages of Karnaugh Map

Disadvantages of K-map

1. Complexity of K-map simplification process increases with the increase in the number of variables
2. The minimum expression obtained might not be unique