IAT-2

1	State the Commutative Law, Associative Law, Minterm, Maxterm, Literal with suitable examples.	[10]

Definition: A mathematical system consisting of a set of elements B, two binary operations $(+)$ and $(\cdot)$, an equality sign $(=)$ to indicate equivalence of expressions (i.e., the expression on one side of the equality sign can be substituted for the expression on the other side), and parentheses to indicate the ordering of the operations is called a *Boolean algebra* if and only if the following postulates hold:

P1. The operations $(+)$ and $(\cdot)$ are closed; i.e., for all $x, y \in B$

(a) $x + y \in B$

(b) $x \cdot y \in B$

P2. There exist identity elements in B, denoted by 0 and 1, relative to the operations $(+)$ and $(\cdot)$, respectively; i.e., for every $x \in B$

(a) $0 + x = x + 0 = x$

(b) $1 \cdot x = x \cdot 1 = x$

P3. The operations $(+)$ and $(\cdot)$ are commutative; i.e., for all $x, y \in B$

(a) $x + y = y + x$

(b) $x \cdot y = y \cdot x$

P4. Each operation $(+)$ and $(\cdot)$ is distributive over the other; i.e., for all $x, y, z \in B$

(a) $x + (y \cdot z) = (x + y) \cdot (x + z)$

(b) $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$

P5. For every element x in B there exists an element $\bar{x}$ in B, called the *complement* of x, such that

(a) $x + \bar{x} = 1$

(b) $x \cdot \bar{x} = 0$

P6. There exist at least two elements $x, y \in B$ such that $x \neq y$.

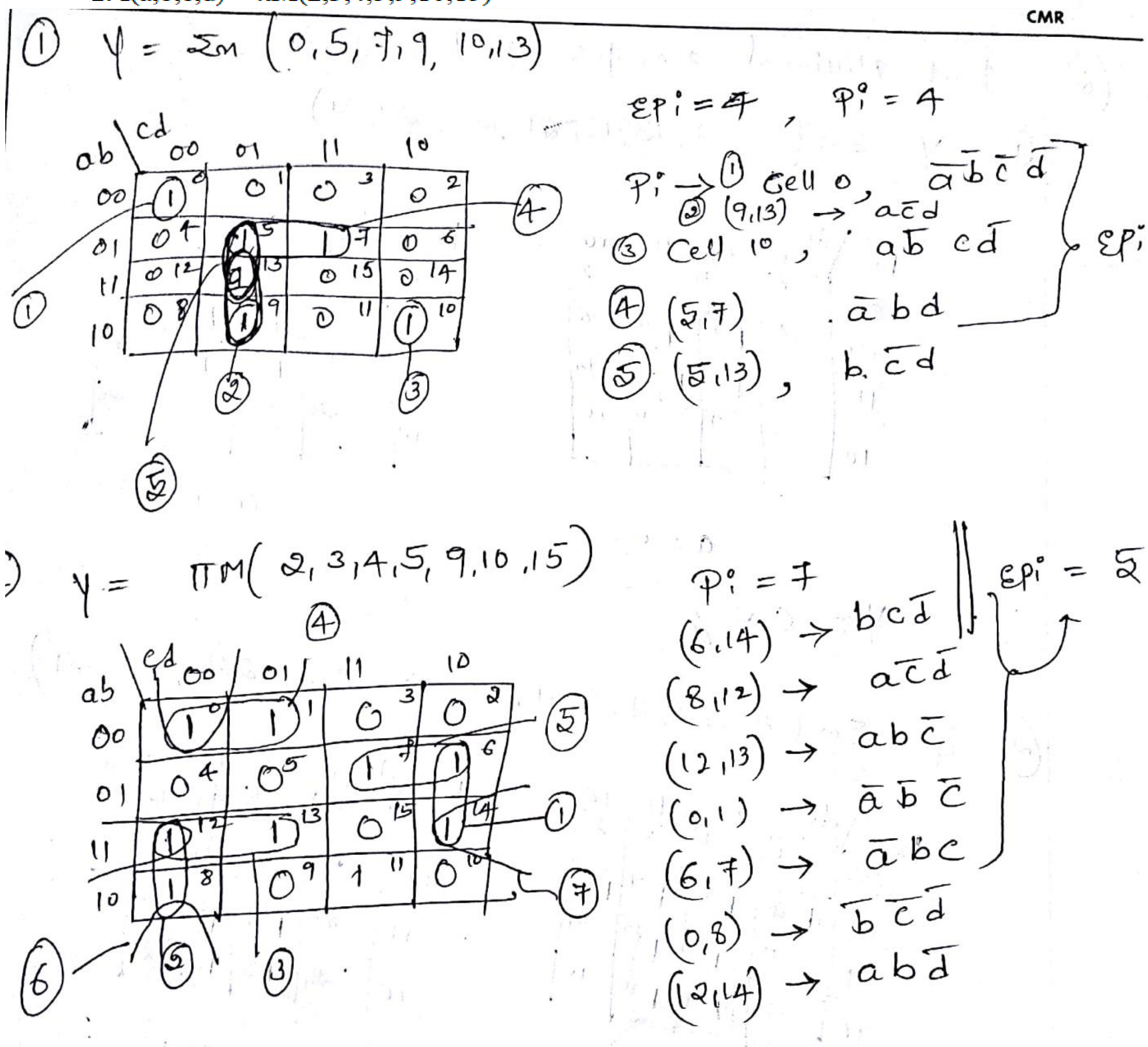
Minterm is a product of all the literals (with or without complement).
Example if we have two boolean variables X and Y then X.(~Y) is a minterm

Maxterm is a sum of all the literals (with or without complement).
Example if we have two boolean variables X and Y then X + (~Y) is a maxterm
A boolean variable and its complement are called literals.
Example Boolean variable A and its complement ~A are literals.

2 Identify all prime implicants and essential prime implicants of following function using k-map. [10]

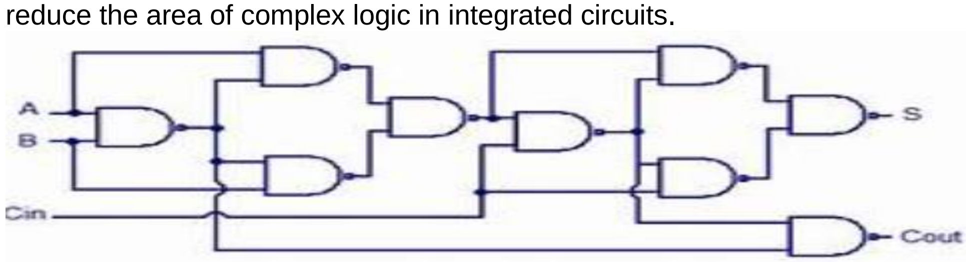
1.f(a,b,c,d) = ∑m(0,5,7,9,10,13)

2. f(a,b,c,d) = πM(2,3,4,5,9,10,15)



3 Why do we need minimization? Implement Full adder using basic and NAND gates. [10]

The problem with having a complicated circuit (i.e. one with many elements, such as logic gates) is that each element takes up physical space in its implementation and costs time and money to produce in itself. Circuit minimization may be one form of logic optimization used to reduce the area of complex logic in integrated circuits.



4 Find minimal sum for following Boolean function using Quine-McClusky method: [10]

$f(a,b,c,d) = \sum m(7,9,12,13,14,15) + d.c(4,11)$

Step 1: **Represent each minterms in its 1/0 notation.**

No.	*Minterm*	1/0 *notation*	*Index*
7	$\bar{a}bcd$	0 1 1 1	3
9	$a\bar{b}\bar{c}d$	1 0 0 1	2
12	$ab\bar{c}\bar{d}$	1 1 0 0	2
13	$ab\bar{c}d$	1 1 0 1	3
14	$abc\bar{d}$	1 1 1 0	3
15	$abcd$	1 1 1 1	4
4	$\bar{a}b\bar{c}\bar{d}$	0 1 0 0	1
11	$a\bar{b}cd$	1 0 1 1	3

Step 2: **List the minterms in increasing order of their index.**

	a	*b*	*c*	*d*	
4	0	1	0	0	index 1
9	1	0	0	1	index 2
12	1	1	0	0	
7	0	1	1	1	index 3
11	1	0	1	1	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	index 4

Fig. 2.9 **Steps 2 and 3 of Quine– McCluskey algorithm**

All further steps are shown in Fig. 2.10. The minterms combining at various iterations are shown with an offset in the '✓' marks

	a	*b*	*c*	*d*			
4	0	1	0	0	✓		
9	1	0	0	1		✓	
12	1	1	0	0	✓	✓	
7	0	1	1	1			✓
11	1	0	1	1		✓	✓
13	1	1	0	1		✓	✓
14	1	1	1	0		✓	✓
15	1	1	1	1			✓

	a	*b*	*c*	*d*	
(4, 12)	–	1	0	0	
(9, 11)	1	0	–	1	✓
(9, 13)	1	–	0	1	✓
(12, 13)	1	1	0	–	✓
(12, 14)	1	1	–	0	✓
(7, 15)	–	1	1	1	
(11, 15)	1	–	1	1	✓
(13, 15)	1	1	–	1	✓
(14, 15)	1	1	1	–	✓

	a	*b*	*c*	*d*
(9, 11) (13, 15)	1	–	–	1
(12, 13) (14, 15)	1	1	–	–

Fig. 2.10 **The Quine–McCluskey algorithm**

The terms without a ✓ besides them constitute the prime implicants. The prime implicants are

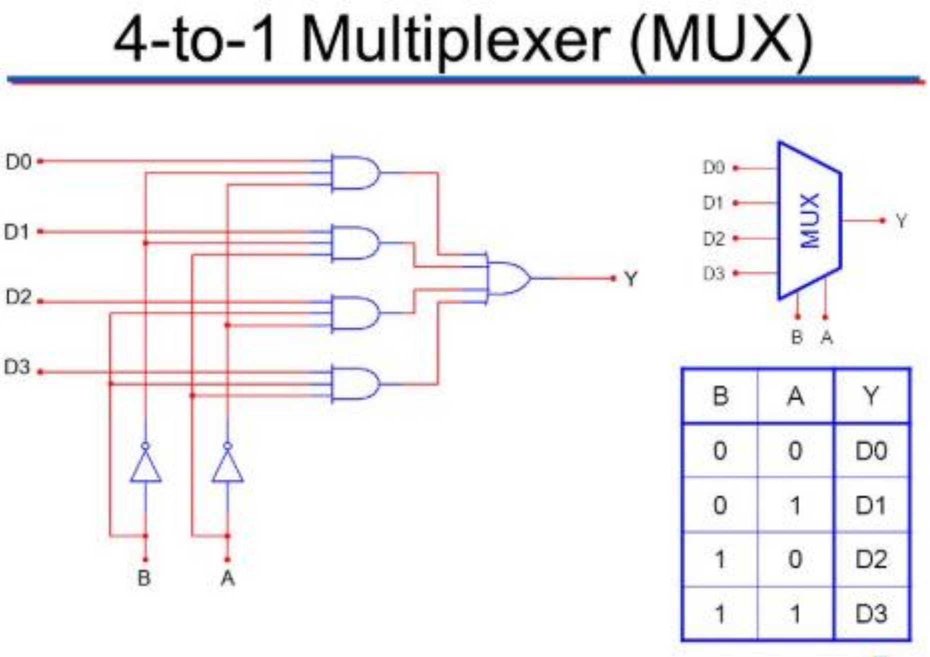
$$b\bar{c}\bar{d},\ bcd,\ ad \text{ and } ab$$

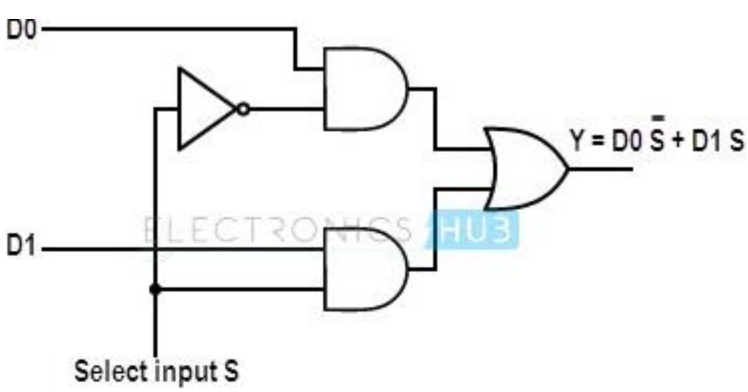
5 Define combinational circuits. State the difference between combinational circuits and sequential circuits with examples. Define multiplexer. Draw the internal diagram of 4:1 Mux and 2:1 mux. [10]

In digital circuit theory, **combinational logic** (sometimes also referred to as **time-independent logic**) is a type of digital logic which is implemented by Boolean circuits, where the output is a pure function of the present input only. This is in contrast to sequential logic, in which the output depends not only on the present input but also on the history of the input. In other words, sequential logic has *memory* while combinational logic does not.

Combinational Logic Circuits	Sequential Logic Circuits
Output is a function of the present inputs (Time Independent Logic).	Output is a function of clock, present inputs and the previous states of the system.
Do not have the ability to store data (state).	Have memory to store the present states that is sent as control input (enable) for the next operation.
It does not require any feedback. It simply outputs the input according to the logic designed.	It involves feedback from output to input that is stored in the memory for the next operation.
Used mainly for Arithmetic and Boolean operations.	Used for storing data (and hence used in RAM).
Logic gates are the elementary building blocks.	Flip flops (binary storage device) are the elementary building unit.
Independent of clock and hence does not require triggering to operate.	Clocked (Triggered for operation with electronic pulses).
Example: Adder [1+0=1; Dependency only on present inputs i.e., 1 and 0].	Example: Counter [Previous O/P +1=Current O/P; Dependency on present input as well as previous state].

A multiplexer (MUX) is a device allowing one or more low-speed analog or digital input signals to be selected, combined and transmitted at a higher speed on a single shared medium or within a single shared device. Thus, several signals may share a single device or transmission conductor such as a copper wire or fiber optic cable. A MUX functions as a multiple-input, single-output switch.

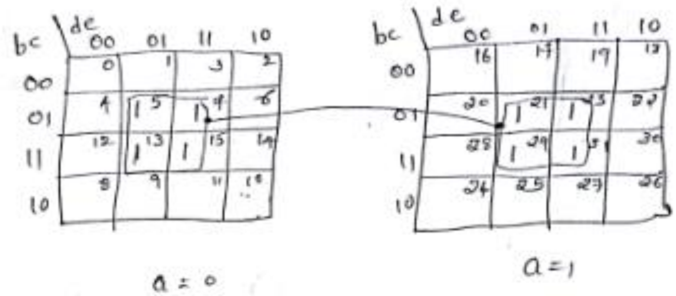




6. Find minimal sum for following Boolean function : [10]

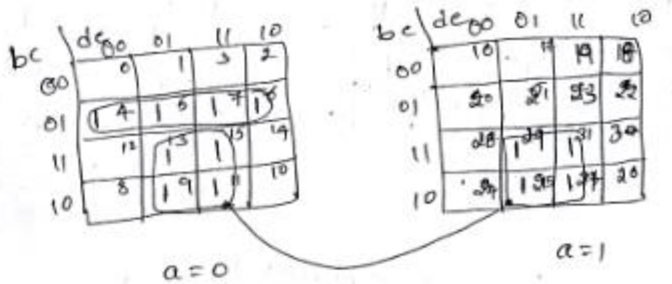
1. $f(a,b,c,d,2) = \sum m(5,7,13,15,21,23,29,31)$
2. $f(a,b,c,d,2) = \sum m(4,5,6,7,9,11,13,15,25,27,29,31)$

① $Y = \Sigma m(5, 7, 13, 15, 21, 23, 29, 31)$



$Y = ce$ //.

(ii) $Y = \Sigma m(4, 5, 6, 7, 9, 11, 13, 15, 25, 27, 29, 31)$



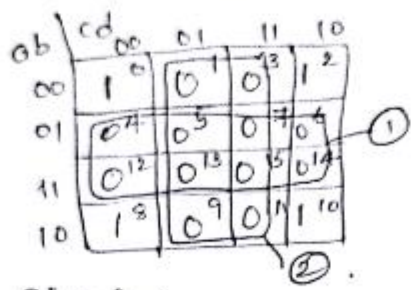
$Y = be + \bar{a}\bar{b}c$ //.

7. Find the prime implicates and essential prime implicates using K-Map: [10]

f(a,b,c,d) = ∑m (0,2,8,10)

f(a,b,c) = ∑m(0,1,2,3,4,6)

① $Y = \Sigma m(0, 2, 8, 10)$ ②



$EPi = Pi = 2$ //.

Cell (4, 5, 6, 7, 12, 13, 14, 15) → $\bar{b}$

Cell (1, 3, 5, 7, 13, 15, 9, 11) → $\bar{d}$

$Y = \Sigma m(0, 1, 2, 3, 4, 6)$

ab \ bc	00	01	11	10
00	1 [0]	1 [1]	1 [3]	1 [2]
01	1 [4]	0 [5]	0 [7]	1 [6]

$Pi = EPi = 1$

$(5, 7) \rightarrow (a + \bar{c})$