



**ANNAMACHARYA INSTITUTE OF TECHNOLOGY & SCIENCES
(AUTONOMOUS)**

UTUKUR, C. K. DINNE (V & M), KADAPA, YSR DIST.

Approved by AICTE, New Delhi & Affiliation to JNTUA, Anantapuramu.

Accredited by NAAC with 'A' Grade, Bangalore.

**DEPARTMENT OF
ELECTRONICS AND COMMUNICATION ENGINEERING**



**DIGITAL CIRCUITS & DESIGN
(23HPC0404)**

COURSE MATERIAL

B. TECH II YEAR I SEM

II Year B.Tech. – I Semester

L	T	P	C
3	0	0	3

(23HPC0404) DIGITAL CIRCUITS DESIGN**Course Objectives:**

- Understand the properties of Boolean algebra, logic operations, and minimization of Boolean functions.
- Analyze combinational and analyze sequential logic circuits.
- Understand the concepts of FSM and compare various Programmable logic devices.
- Model combinational and sequential circuits using HDLs.

Course Outcomes: After completing the course, the student should be able to:

- Understand the properties of Boolean algebra, logic operations, concepts of FSM (L2)
- Apply techniques for minimization of Boolean functions (L3)
- Analyze combinational and Sequential logic circuits. (L4)
- Compare various Programmable logic devices. (L4)
- Design and Model combinational and sequential circuits using HDLs. (L5, L6)

UNIT I Boolean algebra, logic operations, and minimization of Boolean functions

Review of Number Systems and Codes, Representation of unsigned and signed integers, Floating Point representation of real numbers, Laws of Boolean Algebra, Theorems of Boolean Algebra, Realization of functions using logic gates, Canonical forms of Boolean Functions, Minimization of Functions using Karnaugh Maps.

UNIT II Combinational Logic Circuits

Combinational circuits, Design with basic logic gates, design procedure, adders, subtractors, 4-bit binary adder/ subtractor circuit, BCD adder, carry look-ahead adder, binary multiplier, magnitude comparator, data selectors, priority encoders, decoders, multiplexers, demultiplexers.

UNIT III Hardware Description Language

Introduction to Verilog - structural specification of logic circuits, behavioral specification of logic circuits, hierarchical Verilog Code, Verilog for combinational circuits - conditional operator, if-else statement, case statement, for loop using sequential circuits with CAD tools.

UNIT IV Sequential Logic Circuits

Basic architectural distinction between combinational and sequential circuits, Design procedure, latches, flip-flops, truth tables and excitation tables, timing and triggering consideration, conversion of flip-flops, design of counters, ripple counters, synchronous counters, ring counter, Johnson counter, registers, shift registers, universal shift register.

UNIT V Finite State Machines and Programmable Logic Devices

Types of FSM, capabilities and limitations of FSM, state assignment, realization of FSM using flip-flops, Mealy to Moore conversion and vice-versa, reduction of state tables using

partition technique, Design of sequence detector. Types of PLD's: PROM, PAL, PLA, basic structure of CPLD and FPGA, advantages of FPGAs.

Textbooks:

1. M. Morris Mano, "Digital Design", 3rd Edition, PHI. (Unit I to IV)
2. Stephen Brown and Zvonko Vranesic, "Fundamentals of Digital Logic with Verilog Design", 3rd Edition, McGraw-Hill (Unit V)

Reference Books:

1. Charles H. Roth, Jr, "Fundamentals of Logic Design", 4th Edition, Jaico Publishers.
2. Zvi Kohavi and Niraj K. Jha, "Switching and Finite Automata Theory, 3rd Edition, Cambridge University Press, 2010.
3. Samir Palnitkar, "Verilog HDL: A Guide to Digital Design and Synthesis", 2nd Edition, Prentice Hall PTR.
4. D.P. Leach, A.P. Malvino, "Digital Principles and Applications", TMH, 7th Edition.

①

UNIT-I

Boolean Algebra

①

Introduction:-

We use decimal digits 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9 to represent any decimal number where mathematical operations like addition, subtraction, multiplication and division operated on them. But in computers (or) digital circuits other types of numbers such as binary, octal and hexadecimal numbers are also used for processing. Each number comes under its own category called as 'Number system' and is coded as per the Rules of corresponding Number system.

Number system:-

A Number system is defined as an ordered collection of digits (symbols) on which arithmetic operations like addition, subtraction etc. are performed based on certain Rules.

* Number systems are broadly classified into 2 types, They are

1. positional number system
2. Non positional number system

Non-positional number system:-

- * Roman number system is an example of non-positional number system.
- * Letters are used to represent some numerical digits as shown below

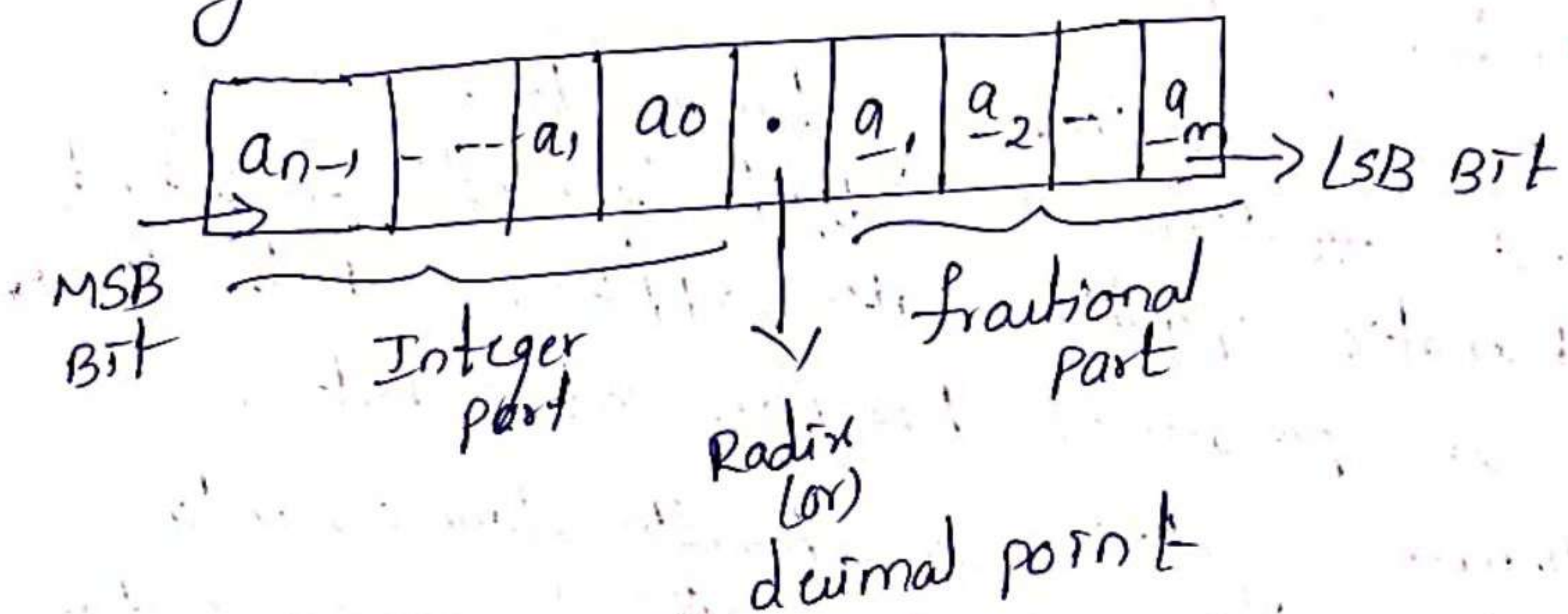
$$V=5, X=10, L=50, C=100, D=500$$

- * Combinations of these letters are used to represent various numbers such as

$$8 = \text{VIII}, 12 = \text{XII}, 25 = \text{XXV}$$

positional number system:-

- * A number is made up with a collection of digits and it has two parts. They are integer part and fractional part. Which are separated by a decimal point



- ② * Number systems are categorized into 4 types ②
1. Binary number system
 2. Decimal number system.
 3. Octal number system.
 4. Hexa decimal number system.

Decimal number system:-

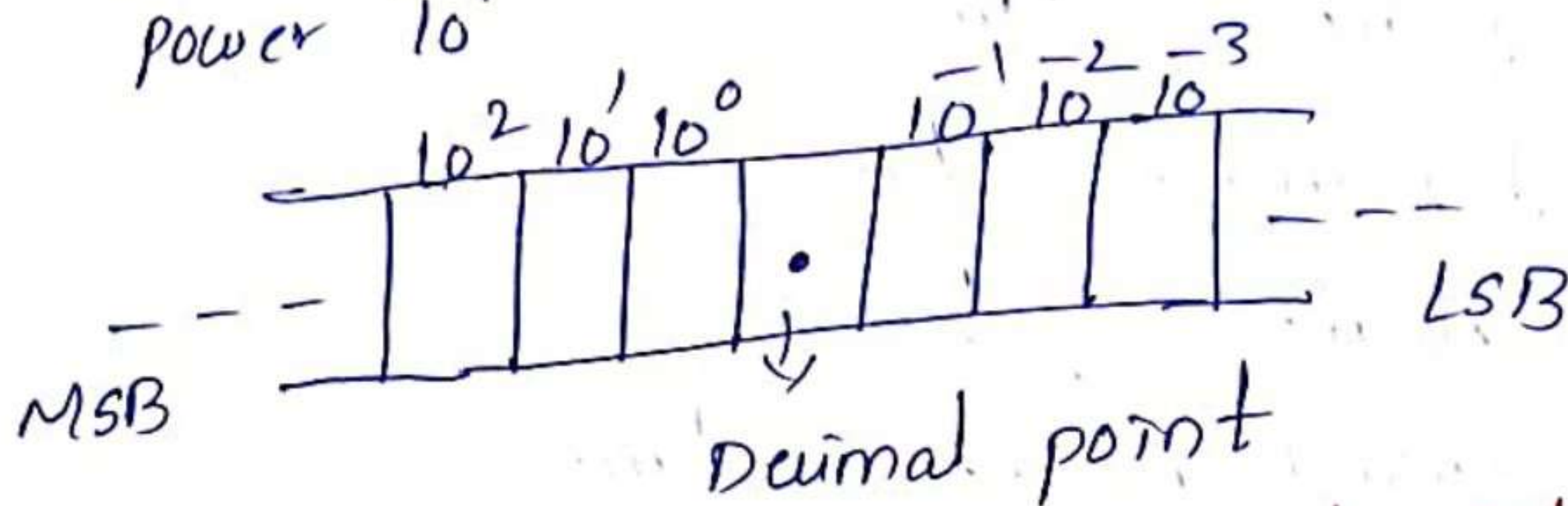
- * In decimal number system, we can express any decimal number in units, 10's, 100's and 1000's and so on.
- * The number system with Base (or) Radix 10 is known as the Decimal number system.
- * As the Base is 10 the system uses ten different symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Eg:- $7628 \Rightarrow 7 \times 1000 + 6 \times 100 + 2 \times 10 + 8 \times 1$

$\downarrow \quad \quad \downarrow$
MSB LSB $\Rightarrow 7628$

- * The left most bit digit which has the greatest weight is called the "MSB bit" [Most significant bit] and Right most digit which has the least weight called the [Least significant digit] "LSB bit".

* Decimal digit and its weight expressed as a power 10



Eg:- Represent the following numbers into power of 10

a) 1234.5 b) 7085

$$a) (1234.5) = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1} \\ = (1234.5)_{10}$$

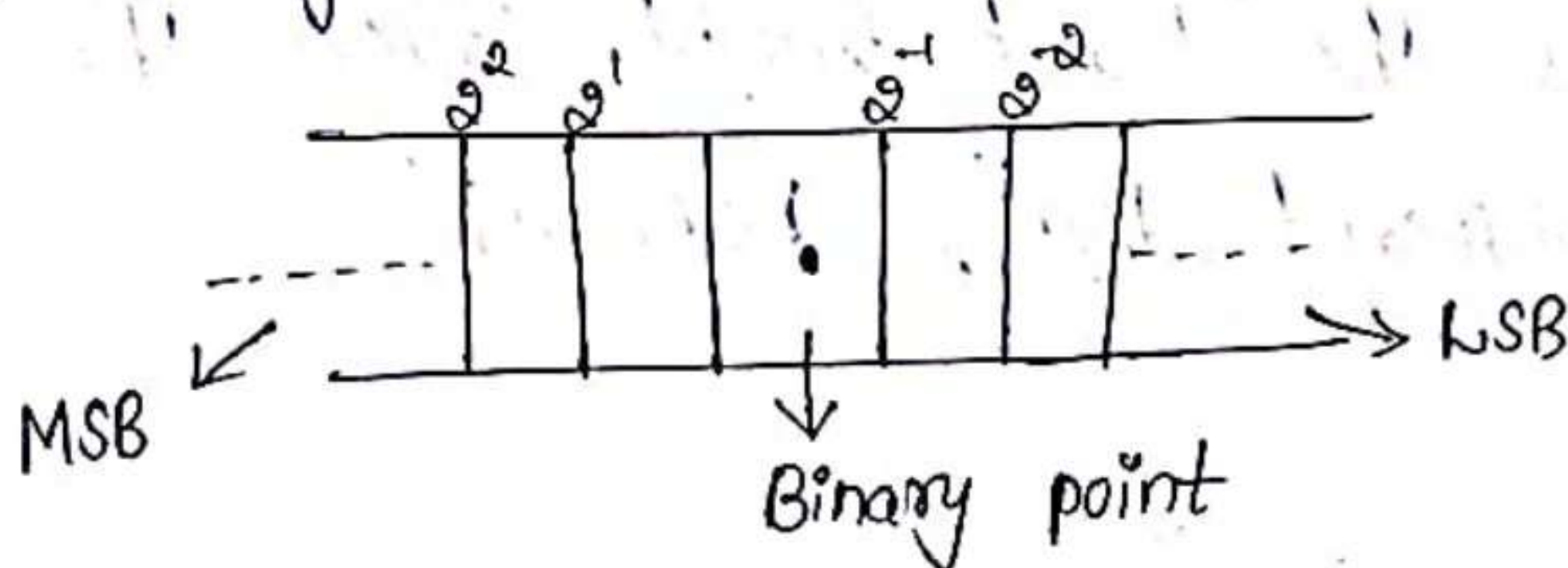
$$b) (7085) = 7 \times 10^3 + 0 \times 10^2 + 8 \times 10^1 + 5 \times 10^0 \\ = (7085)_{10}$$

Binary Number System:-

1) The number system base or radix "2" is known as Binary number system.

2) Binary means 2. Thus binary number system uses 2 different symbols (0, 1).

3) Any binary number can be expressed as the combination of these two symbols ("0" & "1").



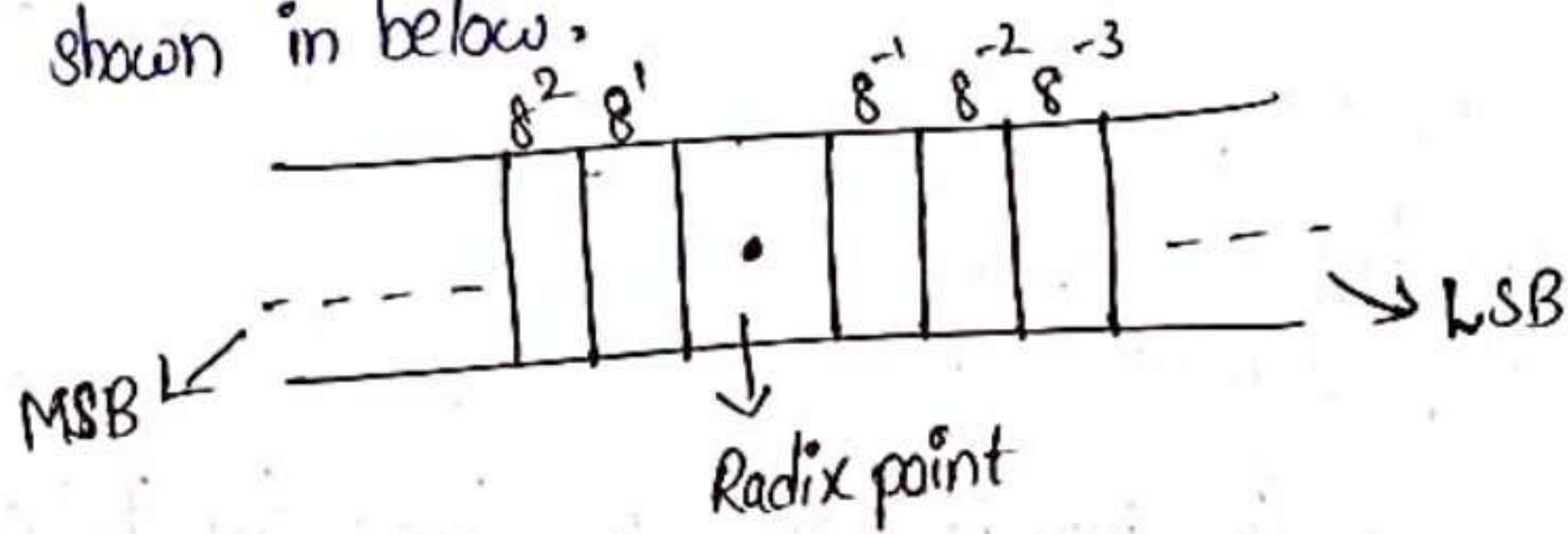
Octal number system:-

(3)

- 1) The number system which base or Radix is "8" is known as octal number system.
- 2) This system uses 8 different symbols (0, 1, 2, 3, 4, 5, 6, 7) are used to represent number.

Eg:- 1705, 256 etc.

- 3) In Octal number system weight is expressed as power of "8" as shown in below.

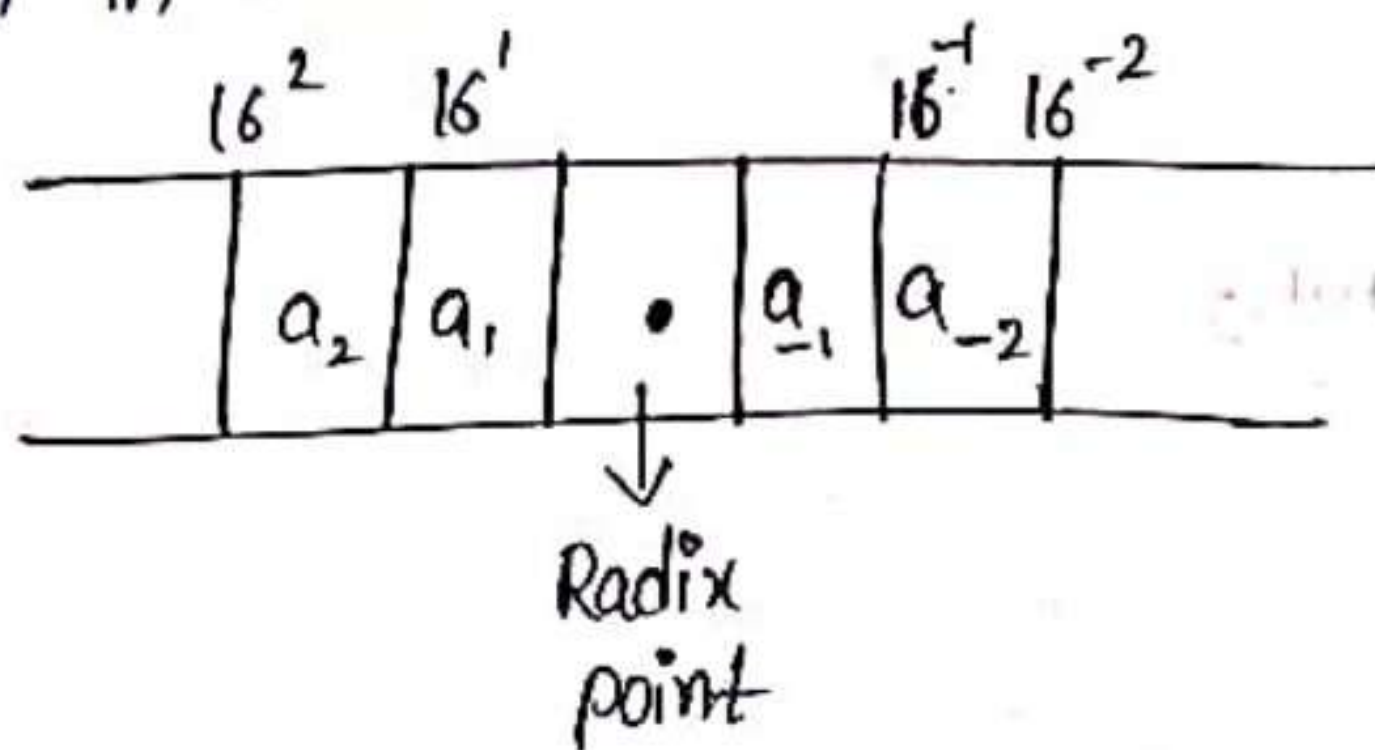


Hexa decimal number System:-

- 1) The number system with base or Radix is 16. is known as the hexa decimal number system.
- 2) Hexa decimal number system uses 10 numerals (0, 1, 2, 3, 4, 5, 6, 7, 8 & 9 and 10-A, 11-B, 12-C, 13-D, 14-E, 15-F).

Eg:- 2A3, B7 etc.

- 3) In hexadecimal number system weight is expressed as power of "16" as shown in below.



*** Comparison of Number System & Radix & digits.

<u>S.No</u>	<u>Number Systems</u>	<u>Radix</u>	<u>digits</u>
1.	Binary	2	0, 1
2.	Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
3.	Octal	8	0, 1, 2, 3, 4, 5, 6, 7
4.	Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A-10, B-11, C-12, D-13, E-14, F-15.

Number Conversions :-

1) Decimal to other base conversions:-

a) Decimal to Binary Conversion:-

(i) $(27)_{10}$

$$\begin{array}{r|l}
 2 & 27 \\
 \hline
 2 & 13 \rightarrow 1 \\
 2 & 6 \rightarrow 1 \\
 2 & 3 \rightarrow 0 \\
 2 & 1 \rightarrow 1 \\
 & 0 \rightarrow 1
 \end{array}
 \uparrow$$

$(11011)_2$

* for integer part we have to taken from bottom to top.

(ii) $(27.315)_{10}$ $(27)_{10} = (11011)_2$

$$\begin{array}{l}
 0.315 \times 2 = 0.63 \rightarrow 0 \\
 0.63 \times 2 = 1.26 \rightarrow 1 \\
 0.26 \times 2 = 0.52 \rightarrow 0 \\
 0.52 \times 2 = 1.04 \rightarrow 1
 \end{array}
 \downarrow$$

$(0101)_2$

$(27.315)_{10} = (11011.0101)_2$

* for fractional part we have to taken from top to bottom.

(iii) $(19)_{10}$

$$\begin{array}{r|l}
 2 & 19 \\
 \hline
 2 & 9 \rightarrow 1 \\
 2 & 4 \rightarrow 1 \\
 2 & 2 \rightarrow 0 \\
 & 1 \rightarrow 1
 \end{array}$$

$$(10011)_2$$

4

(iv) $(19.325)_{10}$

$$(19)_{10} = (10011)_2$$

$$0.325 \times 2 = 0.65 \rightarrow 0$$

$$0.65 \times 2 = 1.3 \rightarrow 1$$

$$0.3 \times 2 = 0.6 \rightarrow 0$$

$$0.6 \times 2 = 1.2 \rightarrow 1$$

$$(0101)_2$$

$$(19.325)_{10} = (10011.0101)_2$$

Home work

a) $(21)_{10}$

$$\begin{array}{r|l}
 2 & 21 \\
 \hline
 2 & 10 \rightarrow 1 \\
 2 & 5 \rightarrow 0 \\
 2 & 2 \rightarrow 1 \\
 & 1 \rightarrow 0
 \end{array}$$

$$(10101)_2$$

b) $(0.35)_{10}$

$$0.35 \times 2 = 0.7 \rightarrow 0$$

$$0.7 \times 2 = 1.4 \rightarrow 1$$

$$0.4 \times 2 = 0.8 \rightarrow 0$$

$$0.8 \times 2 = 1.6 \rightarrow 1$$

$$(0101)_2$$

$$(0.35)_{10} = (0.0101)_2$$

c) $(23)_{10}$

$$\begin{array}{r|l}
 2 & 23 \\
 \hline
 2 & 11 \rightarrow 1 \\
 2 & 5 \rightarrow 1 \\
 2 & 2 \rightarrow 1 \\
 & 1 \rightarrow 0
 \end{array}$$

$$(10111)_2$$

d) $(38.15)_{10}$

$$\begin{array}{r|l}
 2 & 38 \\
 \hline
 2 & 19 \rightarrow 0 \\
 2 & 9 \rightarrow 1 \\
 2 & 4 \rightarrow 1 \\
 2 & 2 \rightarrow 0 \\
 & 1 \rightarrow 0
 \end{array}$$

$$(100110)_2$$

$$0.15 \times 2 = 0.3 \rightarrow 0$$

$$0.3 \times 2 = 0.6 \rightarrow 0$$

$$0.6 \times 2 = 1.2 \rightarrow 1$$

$$0.2 \times 2 = 0.4 \rightarrow 0$$

$$(0010)_2$$

$$(38.15)_{10} = (100110.0010)_2$$

e) $(0.875)_{10}$

$$0.875 \times 2 = 1.75 \rightarrow 1$$

$$0.75 \times 2 = 1.5 \rightarrow 1$$

$$0.5 \times 2 = 1.0 \rightarrow 1$$

$$0.0 \times 2 = 0 \rightarrow 0$$

$$(0.875)_{10} = (111)_2$$

f) $(13.65625)_{10}$

$$\begin{array}{r|l}
 2 & 13 \\
 \hline
 2 & 6 \rightarrow 1 \\
 2 & 3 \rightarrow 0 \\
 2 & 1 \rightarrow 1 \\
 & 0 \rightarrow 1
 \end{array}$$

$$(01101)_2$$

$$0.65625 \times 2 = 1.3125 \rightarrow 1$$

$$0.3125 \times 2 = 0.625 \rightarrow 0$$

$$0.625 \times 2 = 1.25 \rightarrow 1$$

$$0.25 \times 2 = 0.5 \rightarrow 0$$

$$(1010)_2$$

$$(13.65625)_{10} = (01101.1010)_2$$

Decimal to Octal Conversion:-

(i) $(275)_{10}$

$$\begin{array}{r|l}
 8 & 275 \\
 \hline
 8 & 34 \rightarrow 3 \\
 8 & 4 \rightarrow 2 \\
 & 0 \rightarrow 4
 \end{array}$$

$$(275)_{10} = (423)_8$$

(2) $(275.4)_{10} = (423.3146)_8$

$$(275)_{10} = (423)_8$$

$$0.4 \times 8 = 3.2 \rightarrow 3$$

$$0.2 \times 8 = 1.6 \rightarrow 1$$

$$0.6 \times 8 = 4.8 \rightarrow 4$$

$$0.4 \times 8 = 3.2 \rightarrow 3$$

$$(3146)_8$$

Homework.

a) $(974)_{10}$

$$\begin{array}{r|l} 8 & 974 \\ \hline & 121 \rightarrow 6 \\ 8 & 15 \rightarrow 1 \\ 8 & 1 \rightarrow 7 \\ 8 & 0 \rightarrow 1 \end{array} \uparrow$$

$$(974)_{10} = (1716)_8$$

b) $(0.35)_{10}$

$$\begin{array}{l} 0.35 \times 8 = 2.8 \rightarrow 2 \\ 0.8 \times 8 = 6.4 \rightarrow 6 \\ 0.4 \times 8 = 3.2 \rightarrow 3 \\ 0.2 \times 8 = 1.6 \rightarrow 1 \end{array} \downarrow$$

$$(0.35)_{10} = (0.2631)_8$$

c) $(974.35)_{10}$

$$(974)_{10} = (1716)_8$$

$$\begin{array}{l} 0.35 \times 8 = 2.8 \rightarrow 2 \\ 0.8 \times 8 = 6.4 \rightarrow 6 \\ 0.4 \times 8 = 3.2 \rightarrow 3 \\ 0.2 \times 8 = 1.6 \rightarrow 1 \end{array} \downarrow$$

$$(974.35)_{10} = (1716.2631)_8$$

d) $(247.54)_{10}$

$$\begin{array}{r|l} 8 & 247 \\ \hline & 30 \rightarrow 7 \\ 8 & 3 \rightarrow 6 \\ 8 & 0 \rightarrow 3 \end{array} \uparrow$$

$$(247)_{10} = (367)_8$$

$$\begin{array}{l} 0.54 \times 8 = 4.32 \rightarrow 4 \\ 0.32 \times 8 = 2.56 \rightarrow 2 \\ 0.56 \times 8 = 4.48 \rightarrow 4 \\ 0.48 \times 8 = 3.84 \rightarrow 3 \end{array} \downarrow$$

$$(247.54)_{10} = (367.4243)_8$$

$$(0.54)_{10} = (4243)_8$$

Decimal to Hexadecimal Conversion:-

(i) $(382)_{10}$

$$\begin{array}{r|l} 16 & 382 \\ \hline & 23 \rightarrow 14 \\ 16 & 1 \rightarrow 7 \\ 16 & 0 \rightarrow 1 \end{array} \uparrow$$

$$(382)_{10} = (17E)_{16}$$

(ii) $(382.525)_{10}$

$$(382)_{10} = (17E)_{16}$$

$$\begin{array}{l} 0.525 \times 16 = 8.4 \rightarrow 8 \\ 0.4 \times 16 = 6.4 \rightarrow 6 \end{array} \downarrow$$

$$(0.525)_{10} = (86)_{16}$$

$$(382.525)_{10} = (17E.86)_{16}$$

Binary to other base Conversion:-

Binary to Decimal Conversion:-

(i) $(11010)_2$

$$\begin{array}{cccccc} 1 & 1 & 0 & 1 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

$$= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$= 16 + 8 + 2 = (26)_{10}$$

(ii) $(0.101)_2$

$$\begin{array}{ccc} 0.101 \\ \downarrow & \downarrow & \downarrow \\ 2^{-1} & 2^{-2} & 2^{-3} \end{array}$$

$$= 1 \times \frac{1}{2} + 0 + 1 \times \frac{1}{8} \Rightarrow \frac{8+2}{16} = \frac{10}{16}$$

$$= (0.625)_{10}$$

(iii) $(10011.101)_2$

(5)

(iv) $(11011)_2$

$$\begin{array}{ccccccc}
 1 & 0 & 0 & 1 & 1 & . & 1 & 0 & 1 \\
 \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\
 2^4 & 2^3 & 2^2 & 2^1 & 2^0 & & 2^{-1} & 2^{-2} & 2^{-3}
 \end{array}$$

$$= 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

$$= 16 + 2 + 1 + \frac{1}{2} + \frac{1}{8}$$

$$= (19.625)_{10}$$

$$\begin{array}{ccccc}
 1 & 1 & 0 & 1 & 1 \\
 \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
 2^4 & 2^3 & 2^2 & 2^1 & 2^0
 \end{array}$$

$$= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$= 16 + 8 + 2 + 1$$

$$= (27)_{10}$$

(v) $(1101.11)_2$

$$\begin{array}{ccccccc}
 1 & 1 & 0 & 1 & . & 1 & 1 \\
 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}
 \end{array}$$

$$8 + 4 + 1 + \frac{1}{2} + \frac{1}{4}$$

$$(13.75)_{10}$$

Binary to Octal Conversion:-

(i) $(111011)_2$

$$\begin{array}{cccccc}
 1 & 1 & 1 & 0 & 1 & 1 \\
 \leftarrow & & & & & \\
 \text{group 'n to 3 bit}
 \end{array}$$

$$\begin{array}{ccc}
 1 & 1 & 1 & 0 & 1 & 1 \\
 \downarrow & & \downarrow & & \downarrow & \\
 7 & & 3 & & &
 \end{array}$$

$$(73)_8$$

(ii) $(111011.10)_2$

$$\begin{array}{ccccccc}
 1 & 1 & 1 & 0 & 1 & 1 & . & 1 & 0 & 0 \\
 \leftarrow & \leftarrow & & & & & & \downarrow & & \\
 7 & 3 & & & & & & 4 & &
 \end{array}$$

$$(73.4)_8$$

<u>Binary</u>	$\rightarrow$	<u>Octal</u>
000	$\rightarrow$	0
001	$\rightarrow$	1
010	$\rightarrow$	2
011	$\rightarrow$	3
100	$\rightarrow$	4
101	$\rightarrow$	5
110	$\rightarrow$	6
111	$\rightarrow$	7

(iii) $(10011.101)_2$

$$\begin{array}{ccccccc}
 0 & 1 & 0 & 0 & 1 & 1 & . & 1 & 0 & 1 \\
 \downarrow & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \\
 (2 & 3 & . & 5) & & & & & &
 \end{array}$$

$$(23.5)_8$$

(iv) $(101.10100110)_2$

$$\begin{array}{ccccccc}
 1 & 0 & 1 & . & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
 \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \\
 (5 & . & 5 & & 1 & & 4) & & & & &
 \end{array}$$

$$(5.514)_8$$

Binary to Hexadecimal Conversion:-

(i) $(111110011)_2$

group into 4 bits

$$\begin{array}{ccc} 0001 & 1111 & 0011 \\ \downarrow & \downarrow & \downarrow \\ 1 & F & 3 \end{array} \Rightarrow (1F3)_{16}$$

(ii) $(1011001001.1000001)_2$

$$\begin{array}{ccccc} 0010 & 1100 & 1001 & 1000 & 0100 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 2 & C & 9 & 8 & 4 \end{array}$$

$(2C9.84)_{16}$

<u>Binary</u>	$\longrightarrow$	<u>Hexadecimal</u>
0000	$\longrightarrow$	0
0001	$\longrightarrow$	1
0010	$\longrightarrow$	2
0011	$\longrightarrow$	3
0100	$\longrightarrow$	4
0101	$\longrightarrow$	5
0110	$\longrightarrow$	6
0111	$\longrightarrow$	7
1000	$\longrightarrow$	8
1001	$\longrightarrow$	9
1010	$\longrightarrow$	A
1011	$\longrightarrow$	B
1100	$\longrightarrow$	C
1101	$\longrightarrow$	D
1110	$\longrightarrow$	E
1111	$\longrightarrow$	F

Octal to other base conversion:-

Octal to Binary conversion

(i) $(57)_8$

$$\begin{array}{cc} 5 & 7 \\ \downarrow & \downarrow \\ 101 & 111 \end{array}$$

$(101111)_2$

(ii) $(72.45)_8$

$$\begin{array}{ccccc} 7 & 2 & . & 4 & 5 \\ \downarrow & \downarrow & & \downarrow & \downarrow \\ 111 & 010 & & 100 & 101 \end{array}$$

$(111010100101)_2$

(iii) $(765.031)_8$

$$\begin{array}{c} 765.031 \\ (111110101.000011001)_2 \end{array}$$

(iv) $(725.63)_8$

$$\begin{array}{c} (725.63)_8 \\ (111010101.110011)_2 \end{array}$$

Octal to Decimal conversion:

(i) $(127)_8$

$$\begin{array}{ccc} 1 & 2 & 7 \\ \downarrow & \downarrow & \downarrow \\ 8^2 & 8^1 & 8^0 \end{array}$$

$$1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0$$

$$64 + 16 + 7$$

$(87)_{10}$

(ii) $(26.24)_8$

$$2 \times 8^1 + 6 \times 8^0 + 2 \times 8^{-1} + 4 \times 8^{-2}$$

$$16 + 6 + 2 \times \frac{1}{8} + 4 \times \frac{1}{64}$$

$$22 + \frac{1}{4} + \frac{1}{16}$$

$(22.3125)_{10}$

iii) (567)₈

$$\begin{array}{ccc} 5 & 6 & 7 \\ \downarrow & \downarrow & \downarrow \\ 8^2 & 8^1 & 8^0 \end{array}$$

$$5 \times 8^2 + 6 \times 8^1 + 7 \times 8^0$$

$$320 + 48 + 7$$

$$(375)_{10}$$

iv) (22.74)₈

$$\begin{array}{ccc} 2 & 2 & . & 7 & 4 \\ \downarrow & \downarrow & & \downarrow & \downarrow \\ 8^1 & 8^0 & & 8^1 & 8^{-2} \end{array}$$

$$2 \times 8^1 + 2 \times 8^0 + 7 \times 8^{-1} + 4 \times 8^{-2}$$

$$16 + 2 + 7 \times \frac{1}{8} + 4 \times \frac{1}{64}$$

$$18 + \frac{7}{8} + \frac{1}{16}$$

$$= (18.890625)_{10}$$

(6)

Octal to Hexadecimal conversion

i) (1267.34)₈

$$\begin{array}{cccccc} 1 & 2 & 6 & 7 & . & 3 & 4 \\ \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow \\ (001 & 010 & 110 & 111 & 011 & 100)_2 \end{array}$$

Group into 4 bits

$$\begin{array}{cccccc} 0010 & 1011 & 0111 & . & 0111 & 0000 \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow \\ 2 & 11 & 7 & & 7 & 0 \end{array}$$

$$(2B7.7)_{16}$$

ii) (615)₈

$$\begin{array}{ccc} 6 & 1 & 5 \\ \downarrow & \downarrow & \downarrow \\ (110 & 001 & 101)_2 \end{array}$$

Group into 4 bits

$$\begin{array}{ccc} 0001 & 1000 & 1101 \\ \downarrow & \downarrow & \downarrow \\ 1 & 8 & D \end{array}$$

$$(18D)_{16}$$

iii) (407)₈

$$\begin{array}{ccc} 4 & 0 & 7 \\ \downarrow & \downarrow & \downarrow \\ (100 & 000 & 011)_2 \end{array}$$

Group into 4 bits

$$\begin{array}{ccc} 0001 & 0000 & 0111 \\ \downarrow & \downarrow & \downarrow \\ 1 & 0 & 7 \end{array}$$

$$(107)_{16}$$

iv) (72)₈

$$\begin{array}{cc} 7 & 2 \\ \downarrow & \downarrow \\ (111 & 010)_2 \end{array}$$

Group into 4 bits

$$\begin{array}{cc} 0011 & 1010 \\ \downarrow & \downarrow \\ 3 & A \end{array}$$

$$(3A)_{16}$$

Hexadecimal to other base conversion:

i) Hexadecimal to decimal conversion

i) (5D.98)₁₆

$$\begin{array}{cccc} 5 & D & . & 9 & 8 \\ \downarrow & \downarrow & & \downarrow & \downarrow \\ 16^1 & 16^0 & & 16^{-1} & 16^{-2} \end{array}$$

$$= 5 \times 16^1 + 13 \times 16^0 + 9 \times 16^{-1} + 8 \times 16^{-2}$$

$$= (93.59375)_{10}$$

ii) (F5.E)₁₆

$$\begin{array}{ccc} F & 5 & . & E \\ \downarrow & \downarrow & & \downarrow \\ 16^1 & 16^0 & & 16^{-1} \end{array}$$

$$15 \times 16^1 + 5 \times 16^0 + 14 \times 16^{-1}$$

$$= (245.875)_{10}$$

$$(iii) (2A.6)_{16}$$

$$\begin{array}{ccc} 2 & A & 6 \\ \downarrow & \downarrow & \downarrow \\ 16^1 & 16^0 & 16^{-1} \end{array}$$

$$2 \times 16^1 + 10 \times 16^0 + 6 \times 16^{-1}$$

$$(42.375)_{10}$$

$$(iv) (1F.B)_{16}$$

$$\begin{array}{ccc} 1 & F & B \\ \downarrow & \downarrow & \downarrow \\ 16^1 & 16^0 & 16^{-1} \end{array}$$

$$1 \times 16^1 + 15 \times 16^0 + 11 \times 16^{-1}$$

$$(31.687)_{10}$$

$$(v) (0.A2)_{16}$$

$$\begin{array}{ccc} 0 & A & 2 \\ \downarrow & \downarrow & \downarrow \\ 16^0 & 16^{-1} & 16^{-2} \end{array}$$

$$0 \times 16^0 + A \times 16^{-1} + 2 \times 16^{-2}$$

$$(0.6328)_{10}$$

Hexadecimal to Binary conversion:

$$(i) (B65F)_{16}$$

$$\begin{array}{cccc} B & 6 & 5 & F \\ \downarrow & \downarrow & \downarrow & \downarrow \end{array}$$

$$(1011 \ 0110 \ 0101 \ 1111)_2$$

$$(ii) (2BD)_{16}$$

$$\begin{array}{ccc} 2 & B & D \\ \downarrow & \downarrow & \downarrow \end{array}$$

$$(0010 \ 1011 \ 1101)_2$$

$$(iii) (5A3.CE2)_{16}$$

$$(010110100011 \cdot 110011100011)_2$$

$$(iv) (93.A)_{16}$$

$$(v) (3FA)_{16}$$

$$(10010011 \cdot 1010)_2$$

$$(001111111101)_2$$

Hexadecimal to Octal conversion:

$$(i) (FACE.25)_{16}$$

$$\begin{array}{cccccc} F & A & C & E & 2 & 5 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{array}$$

$$(1111 \ 1010 \ 1100 \ 1110 \ 0010 \ 0101)_2$$

Group into 3 bits

$$\begin{array}{cccccc} 001 & 111 & 101 & 011 & 001 & 110 & 001 & 001 & 010 \\ \hline (1 & 7 & 5 & 3 & 1 & 6 & 1 & 1 & 2)_8 \end{array}$$

$$(ii) (B4)_{16}$$

$$\begin{array}{cc} B & 4 \\ \downarrow & \downarrow \end{array}$$

$$(1011 \ 0100)_2$$

Group into 3 bits

$$\begin{array}{ccc} 010 & 110 & 100 \\ \downarrow & \downarrow & \downarrow \\ (2 & 6 & 4)_8 \end{array}$$

$$(iii) (0.BF85)_{16}$$

$$\begin{array}{cccc} B & F & 8 & 5 \\ \downarrow & \downarrow & \downarrow & \downarrow \end{array}$$

$$(0.1011 \ 1111 \ 1000 \ 0101)_2$$

Group into 3 bits

$$0.101 \ 111 \ 111 \ 000 \ 010 \ 0100$$

$$(0.577024)_8$$

$$(iv) (2DA)_{16}$$

$$\begin{array}{ccc} 2 & 0 & A \\ \downarrow & \downarrow & \downarrow \end{array}$$

$$(0010 \ 0000 \ 1010)_2$$

Group into 3 bits

$$001 \ 000 \ 001 \ 010$$

$$(1012)_8$$

BCD codes [Binary coded decimal] (7) (3)

Def:- BCD code is a numeric weighted code where it represents each decimal digit (0, 1, 2, 3, 4, 5, 6, 7, 8 & 9) with a four-bit binary number.

- * It is a way of representing letters, characters, or numbers using group of 0's & 1's.
- * The Binary codes can be classified as 2 types.
 - i) Numeric codes
 - ii) Alpha numeric codes

Numeric codes:-

Numeric codes are codes which represent numeric information that means only numbers has a series of 0's & 1's.

- * 8421, XS-3 code, Gray code are numeric codes.
- * Numeric codes used to represent the decimal digits are called BCD codes [Binary coded decimal]
- Eg:- 8421, 2421, 5211 are BCD codes

Alphanumeric codes:-

Alphanumeric codes are used to represent numbers, alphabets and special symbols. These are classified into 2 types ASCII code & EBCDIC code.

i) Error detecting codes are used to identify errors occurred in data during data transmission where parity codes are used for error detection.

ii) Error correcting codes are used to deduce correct code word from erroneous code word, where hamming code is used for error correction.

Eg:- A, @, ||

* classification of binary codes:-

1. Numeric codes

a) Weighted codes

* +ve Weighted code

8421, 2421, 5211

* -ve Weighted code

84-2-1

b) Non-weighted codes

* EX-3 code

* Gray code

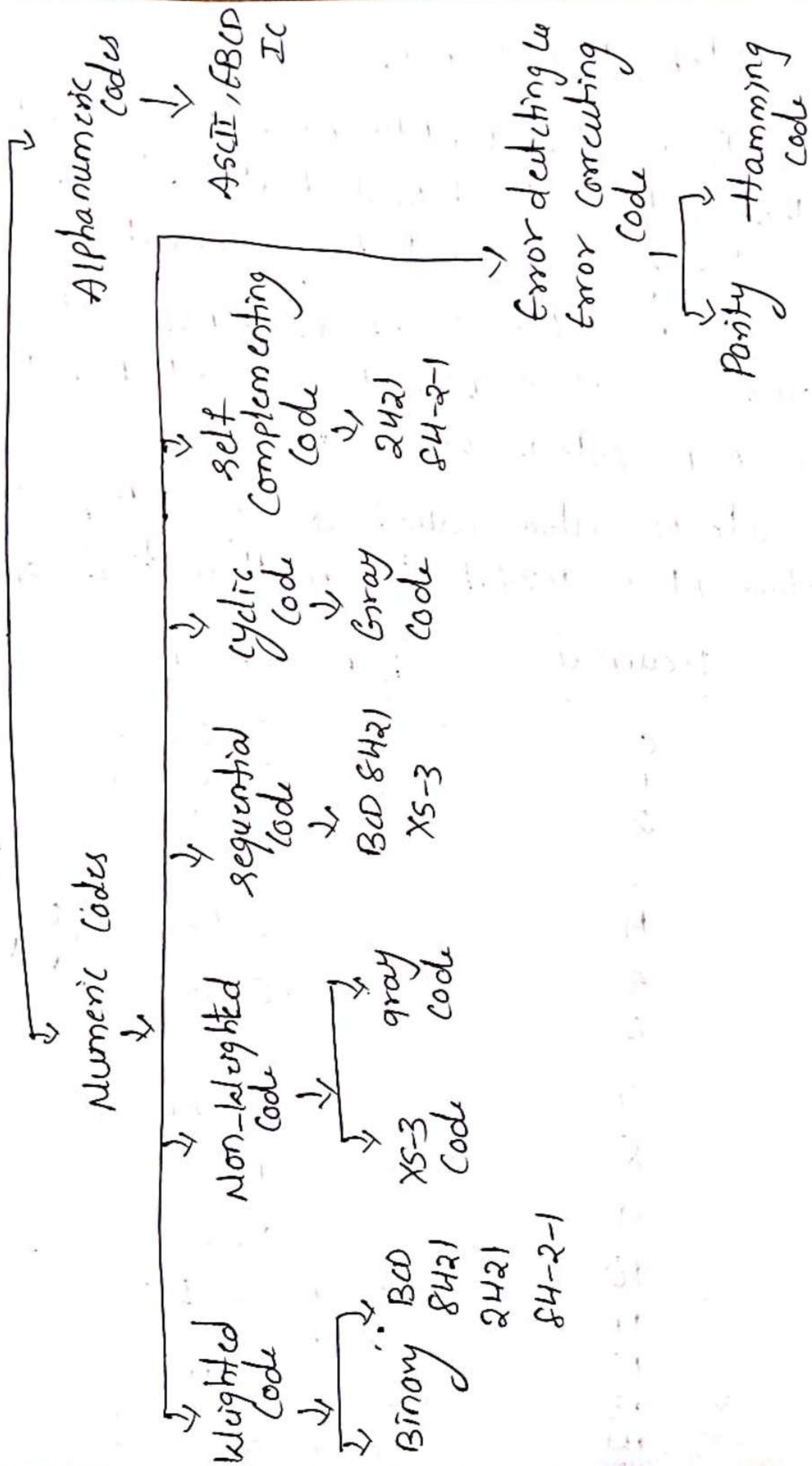
2. Alphanumeric code :- 1. ASCII code

2. EBCDIC code

3. Error detecting codes:- 1. parity codes

4. Error correcting codes:- 1. Hamming code

Binary Codes



Binary Coded decimal [BCD]:-

- * BCD code is the widely used binary code. In this code each decimal digit 0-9 is represented by a 4-bit binary number.
- * BCD code is also called as 8421 code because $2^3, 2^2, 2^1, 2^0$ weights are attached to it. hence BCD code is weighted code.
- * BCD code is also called a sequential code. therefore it is useful for mathematical operations.

<u>Decimal</u> <u>2 2</u>	<u>BCD</u> <u>8421</u>
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
11	0001 0001
12	0001 0010
13	0001 0011
14	0001 0100
15	0001 0101

{ 0-9
BCD codes }

Advantages:-

(9)

- * It is an easy mode of conversion from decimal to binary
- * It is faster and efficient system.

Applications:-

- * It is mainly used in digital display systems.
- * digital calculators, digital clocks, digital Voltmeters.

Examples:-

1. Convert decimal number into BCD code

a) $(23)_{10}$

b) $(56.79)_{10}$

A) $(00100011)_{BCD}$

A) $(01010110.01111001)_{BCD}$

2. Convert hexadecimal to BCD

a) $(5C)_{16}$

b)

A) $(5C)_{16} \rightarrow (C)_{16} \rightarrow (C)_{10} \rightarrow (BCD)$

$$\begin{array}{l} (512)_{16} \Rightarrow 12 \times 16^0 + 5 \times 16^1 = (92)_{10} = (10010010)_{BCD} \\ \downarrow \downarrow \\ 16^1 \quad 16^0 \end{array}$$

3. BCD to decimal conversion

a) 10000001

b) 10010100

$$\frac{1000}{8} \frac{0001}{1} \Rightarrow (81)_{10}$$

$$\frac{1001}{9} \frac{0100}{4} \Rightarrow (94)_{10}$$

Self Complementing codes:-

i) 2421 code:-

- * 2421 is a numeric five weighted BCD code.
- * In 2421, each decimal digit (0 to 9) is represented individually using a four-bit code word.
- * one can perform arithmetic operations using 2421 codes

Decimal Number

2421 Code

0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

Eg:- Convert the 2421 BCD codes to decimal numbers

a) 10

b) 97

or the following

c) 567.89

A) 10
↓ ↓
0001 0000

=> (00010000) 2421 BCD

b) 97
↓ ↓
1111 1101

(1111 1101) 2421 BCD

Eg 2:- find the decimal numbers on the following (b)
2421 BCD codes

a) 00010000 b) 11111101

Ans:- a) 00010000

$\downarrow \quad \downarrow$
1 0 $\rightarrow (10)_{10}$

b) 11111101

$\downarrow \quad \downarrow$
9 7 $\rightarrow (97)_{10}$

84-2-1 Complementing Code $\Rightarrow$ 84-2-1 Code:-

* 84-2-1 is a numeric negative weighted BCD code.

* In 84-2-1, each decimal digit (0 to 9) is represented individually using a four-bit code word.

* The positional weights 8, 4, -2 and -1 assigned from left to right.

* 84-2-1 code contains positive weights as well as negative weights.

* 84-2-1 BCD codes are given in below

(10)

<u>Decimal Number</u>		<u>84-2-1 Code</u>
0	→	0000
1	→	0111
2	→	0110
3	→	0101
4	→	0100
5	→	1011
6	→	1010
7	→	1001
8	→	1000
9	→	1111

Examples:- find the 84-2-1 BCD codes of the following decimal numbers

a) 10 b) 97 c) 567.89

A) 1 0
 ↓ ↓
 0111 0000

⇒ (0111 0000)
 (84-2-1 BCD)

b) 9 7
 ↓ ↓
 1111 1001

⇒ (1111 1001)
 (84-2-1 BCD)

2. find the decimal numbers of the following 84-2-1 BCD codes

a) 01110000

b) 11111001

A)

0111 0000

$$\begin{array}{cc} \downarrow & \downarrow \\ 1 & 0 \end{array}$$
 $\Rightarrow (10)_{10}$

(11)

b) 1111 1001

$$\begin{array}{cc} 1111 & 1001 \\ \hline 9 & 7 \end{array}$$
 $\rightarrow (97)_{10}$

(7)

Excess-3 Code:-

def:- Excess-3 Code is a modified form of a BCD number. The excess-3 code can be derived from the natural BCD code by adding 3 to each coded number.

* This is an un-weighted code and also a sequential code.

* If we see the difference b/w any two binary codes then it is equal to one. It is also a self complementing code.

Decimal
= 2²8421
BCD Code
= 2Excess-3 code
by adding 0011
2

0	0000 + 0011 $\rightarrow$	0011
1	0001 + 0011 $\rightarrow$	0100
2	0010 + 0011 $\rightarrow$	0101
3	0011 + 0011 $\rightarrow$	0110
4	0100 + 0011 $\rightarrow$	0111
5	0101 + 0011 $\rightarrow$	1000
6	0110 + 0011 $\rightarrow$	1001
7	0111 + 0011 $\rightarrow$	1010
8	1000 + 0011 $\rightarrow$	1011
9	1001 + 0011 $\rightarrow$	1100

Eg:- Convert Decimal to XS-3 code.

i) $(23)_{10}$ ii) $(29)_{10}$

A): Decimal number $\rightarrow 29$
BCD number $\rightarrow 0010 \ 1001$

$$\begin{array}{r} 29 \\ + 33 \\ \hline 512 \end{array}$$

Add 3 to BCD number $\rightarrow 0010 \ 1001$

$$\begin{array}{r} 0011 0011 \\ 1 11 \\ \hline 0101 \ 1100 \\ \downarrow \downarrow \\ 5 12 \end{array}$$

$\therefore$ XS-3 code of decimal number is
 $(0101 \ 1100)_{XS-3 \text{ code}}$

ii) $(23)_{10}$

A) $(23)_{10} \rightarrow ()_{BCD} + 0011 \rightarrow XS-3 \text{ code}$

$$\begin{array}{r} 2 3 \\ \downarrow \downarrow \\ 0010 0011 \end{array}$$

Add 3 to $\Rightarrow 0010 \ 0011$

$$\begin{array}{r} 0011 0011 \\ 1 11 \\ \hline 0101 \ 0110 \\ \downarrow \downarrow \\ 5 6 \end{array}$$

$$\begin{array}{r} 23 \\ + 33 \\ \hline 56 \end{array}$$

$\therefore$ XS-3 code is $\Rightarrow 01010110$

Convert hexadecimal to XS-3 code

(12)

(8)

1) $(5C)_{16}$

$$(5C)_{16} \Rightarrow (C)_{16} \Rightarrow (C)_{10} \Rightarrow (B)_{BCD} + 0011 = \text{XS-3 code}$$

$$\Rightarrow \begin{array}{cc} 5 & C \\ \downarrow & \downarrow \\ 5 & 12 \end{array}$$

$$16^1 \ 16^0 \Rightarrow 5 \times 16^1 + 12 \times 16^0 = (92)_{10}$$

$$\Rightarrow \begin{array}{cc} 1001 & 0010 \\ 0011 & 0011 \\ \hline 1100 & 0101 \end{array} \Rightarrow (11000101)_{\text{XS-3 code}}$$

$\downarrow \quad \downarrow$
12 5

Convert octal to XS-3 code

2) $(734)_8$

A) $(734)_8 \Rightarrow (C)_8 \rightarrow (C)_{10} \rightarrow (C)_{BCD} + 0011 = \text{XS-3 code}$

$$\begin{array}{ccc} 7 & 3 & 4 \\ 8^2 & 8^1 & 8^0 \end{array}$$

$$\Rightarrow 7 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 = (476)_{10}$$

$$\Rightarrow \begin{array}{ccc} 4 & 7 & 6 \\ \downarrow & \downarrow & \downarrow \\ 0100 & 0111 & 0110 \\ 0011 & 0011 & 0011 \\ \hline 0111 & 1010 & 1001 \end{array} \Rightarrow (011110101001)_{\text{XS-3 code}}$$

Gray code:-

- * The code which exhibit only a single bit change from one code to the next number is known as Graycode.
- * Graycode is also a Reflected binary code and unweighted code.
- * The errors may be recovery by using Graycode.
- * Gray code is also called unit distance code and minimum error code and cyclic code.
- * The successive value differ in only '1' bit

Decimal = 2	Binary = 2	Graycode = 2
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Eg:- Binary to Graycode Conversion

(13)

(9)

1) $(100110)_2$

A) 1 0 0 1 1 0

$\Rightarrow 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0$

$\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$

(1 1 0 1 0 1) Graycode

2) Convert Gray to Binary

A) (110101) Gray

1 1 0 1 0 1
 $\downarrow \oplus \downarrow \oplus \downarrow \oplus \downarrow \oplus$

1 0 0 1 1 0 $\Rightarrow$ (100110) binary

Error Detecting and Error Correcting codes:

* When digital data (or) information is transmitted from one system to the other and unwanted electrical disturbance called noise gets added to each.

* The noise can force an error in the digital information that means '0' may change to '1' (or) '1' to '0'.

* To detect and correct such errors we can use some special force which passes the capacity to detect and correct the error. Such codes called as error detecting and error correcting codes.

* Error correcting codes we use special code that is called hamming code.

Binary number Representation:-

1) Binary numbers can be represented in 2 ways.

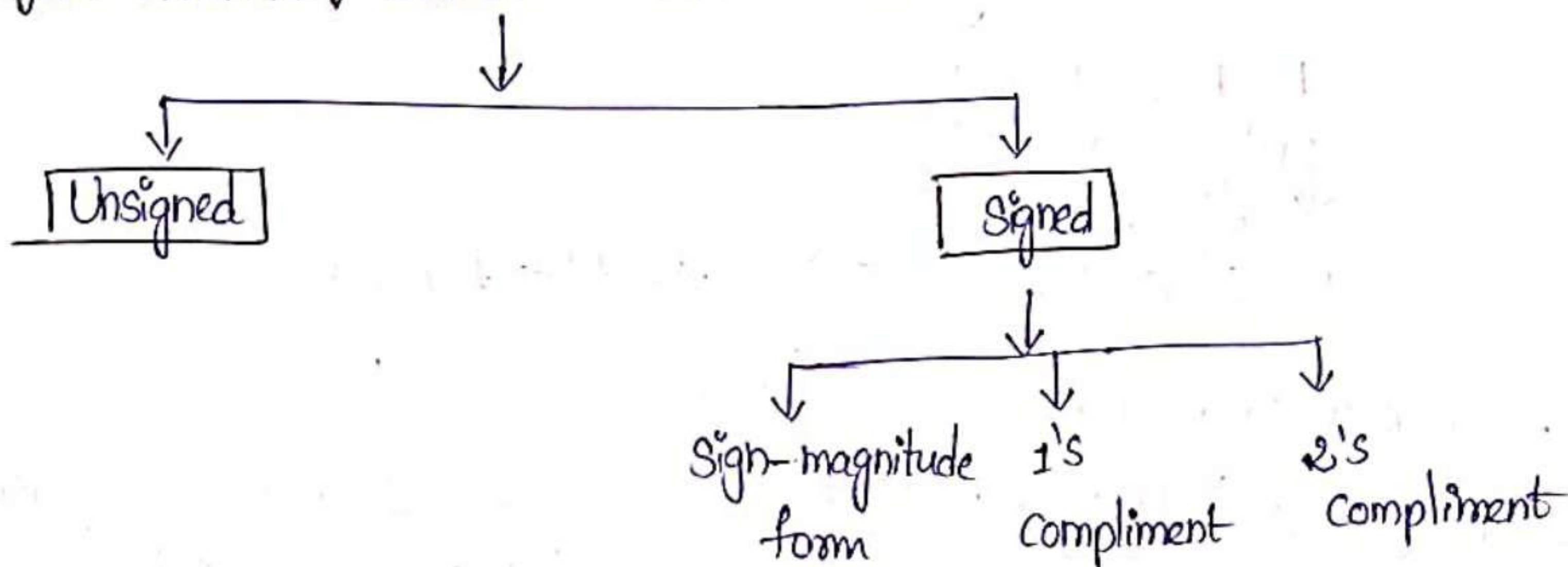
(i) Signed binary numbers

(ii) Unsigned binary numbers.

2) Unsigned binary numbers does not have sign bit.

3) Whereas signed binary numbers uses sign bit as well as these can be difference between +ve and -ve numbers.

Types of Binary number Representation:-



Unsigned binary number Representation:-

1) Unsigned means there is no sign. Then the default sign is +ve.

2) Unsigned binary numbers are always useful for representing +ve numbers.

3) Unsigned binary numbers does not have any sign bit it contains only magnitude (It means number (or) bit)

4) Let us assume $n=3$ bits.

5) If $n=3$ we represent (0 to 7).

6) The range of unsigned binary number representation is 0 to $2^n - 1$

7) If $n=4$ the range is $2^n - 1 = 2^4 - 1 = 0$ to 15

(14)

8) If $n=8$ the range is $2^n - 1 = 2^8 - 1 = 0$ to 255

8) The range is represented in unsigned binary numbers is

0 to $2^n - 1$

9) The main drawback of unsigned binary number is it is only used for +ve numbers. Not for -ve numbers.

Signed binary number Representation:-

1) We will have a sign before the binary number.

2) It may be a +ve number (0) -ve number.

3) By using sign representation we can represent both +ve as well as -ve numbers.

4) Signed binary numbers is made up of 2 parts.

(i) Sign bit

(ii) Magnitude bit

5) The signed binary numbers are represent in 3 ways.

(i) Sign - magnitude

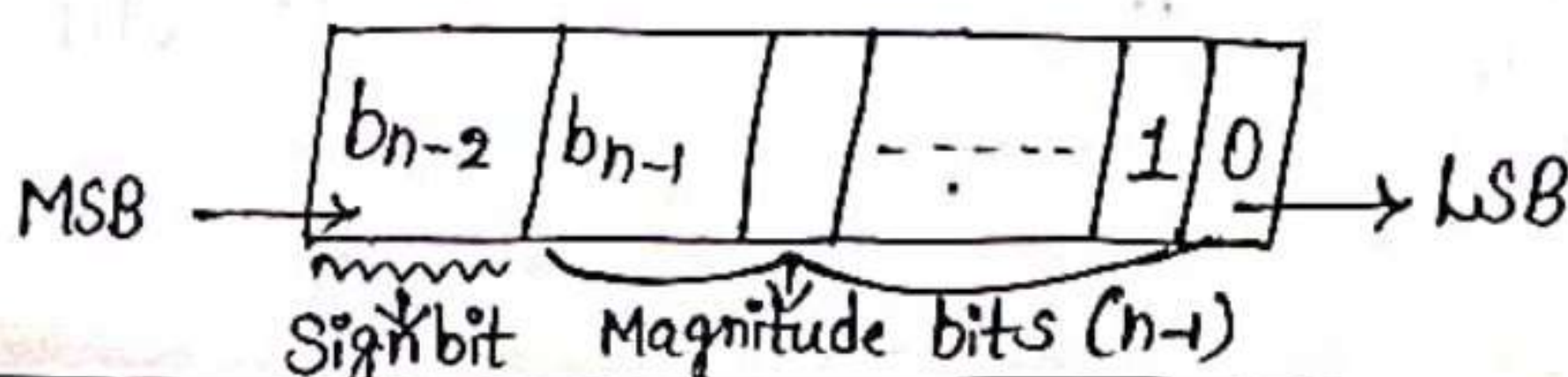
(ii) 1's Complement.

(iii) 2's Complement.

1) Sign - magnitude representation:-

1) Let us take a binary number.

2) The 1st bit is represented as sign bit.



3) In sign-magnitude form an addition bit called the sign bit is placed in front of the number.

4) If the sign bit is '0' the number is +ve.

5) If the sign bit is '1' the number is -ve.

6) The remaining $n-1$ bits represent the magnitude bit.

Example:-

Represent the following in Signed representation.

a) 10

binary number = $\underset{\substack{\downarrow \\ \text{Sign bit}}}{0} \underset{\substack{\downarrow \\ \text{Magnitude bit}}}{1010}$

b) -10

binary number = $\underset{\substack{\downarrow \\ \text{Sign bit}}}{1} \underset{\substack{\downarrow \\ \text{Magnitude bit}}}{1010}$

c) 10 in 8 bits

binary number = $\underset{\substack{\downarrow \\ \text{Sign bit}}}{0} \overset{2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0}{\underset{\substack{\downarrow \\ \text{Magnitude bit}}}{00001010}}$

d) 3 in '8' bits

binary number = $\underset{\substack{\downarrow \\ \text{Sign bit}}}{0} \overset{\text{Magnitude bit}}{\underbrace{00000011}}$

e) -3 in '8' bits

binary number = $\underset{\substack{\downarrow \\ \text{Sign bit}}}{1} \overset{\text{Magnitude bit}}{\underbrace{00000011}}$

Represent the following in Signed binary number representation

a) 00001111

Sign = 0,

Magnitude = 0001111

00001111 = +15

b) 10001111

Sign = 1

Magnitude = 0001111

10001111 = -15

c) 00000111

Sign = 0

Magnitude = 0000111

00000111 = +7

d) 10000111

Sign = 1

Magnitude = 0000111

10000111 = -7

* Therefore, in an 'n' bit sign magnitude binary number, the minimum value that can be represented is $-(2^{n-1})$ and maximum value that can be represented is (2^{n-1}) . (15)

$$[-(2^{n-1}) \text{ to } (2^{n-1})]$$

1's Complement form:-

- 1) 1's complement representation is mostly used for representing signed binary number. (both +ve & -ve numbers).
- 2) 1's complement of a binary number is obtained by replacing all 0's with 1's and 1's with 0's, including the sign bit.
- 3) If a +ve number is complemented then resulted value is a -ve number with equal magnitude.
- 4) If a -ve number is complemented then rep resulted value is a +ve number with equal magnitude.

Example:-

Find the 1's complement of Signed binary numbers.

- a) 00001111 $\rightarrow$ 11110000
- b) 10101010 $\rightarrow$ 01010101
- c) 1101.01 $\rightarrow$ 0010.10
- d) 1001.1001 $\rightarrow$ 0110.0110.

Decimal numbers into 1's complement.

a) +4 and -4

+4 binary number = 0100 (sign bit, magnitude bit)

-4 binary number = 1011

b) +15 and -15

+15 = 01111

-15 = 10000

2's Complement Representation:-

1) The 2's complement of any binary number is determined by adding 1 to the 1's complement of that number.

$$\therefore \text{2's complement} = \text{1's complement} + 1$$

Example :-

1) 1101

binary number = 1101

1's complement = 0010

2's complement = $\begin{array}{r} +1 \\ \hline 0011 \end{array}$

2) 20

binary number = 10100

1's complement = 01011

2's complement = $\begin{array}{r} +1 \\ \hline 01100 \end{array}$

Represent a) -23 b) -8 in sign magnitude, 1's complement and 2's complement.

a) -23

binary number = 11011

1's complement = 10100

2's complement = $\begin{array}{r} +1 \\ \hline 101001 \end{array}$

b) -8

binary number = 11000

1's complement = 10111

2's complement = $\begin{array}{r} +1 \\ \hline 11000 \end{array}$

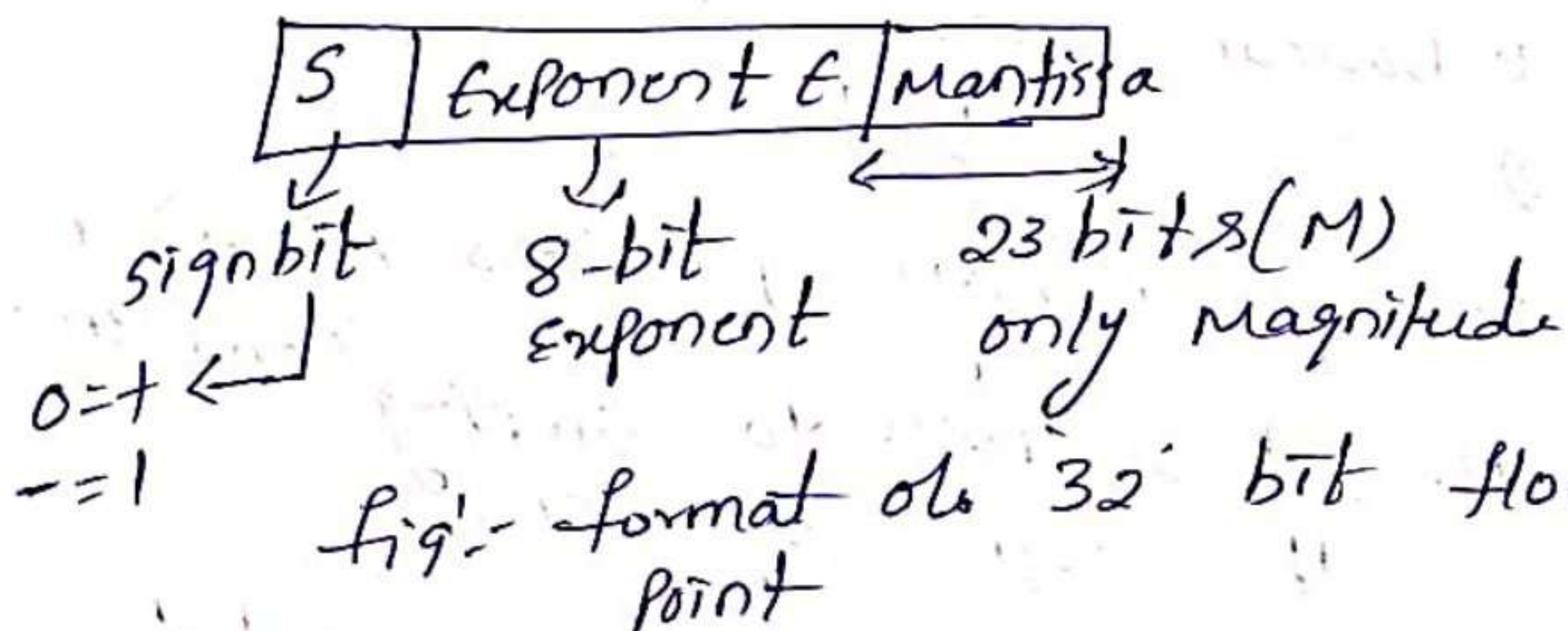
Format of floating point Representation of (16) Real Numbers:-

* Floating point is always interpreted to represent a number in the following form

$$\boxed{\therefore \text{Significant} * \text{Base}^{\text{Exponent}}}$$

* To represent very small (tiny) number or very large number floating point representation is used.

* The floating point representation of a number has two parts. The first part represents a signed, fixed point number called the "Mantissa". The second part designates the position of the decimal (or) binary point and is called the "Exponent".



Introduction to Boolean Algebra:-

- * Boolean Algebra is also known as switching Algebra.
- * Boolean Algebra is a set of rules, laws or theorems by which all logical operations can be mathematically expressed.
- * This algebra was first introduced by George Boole an English mathematician.
- * It is a basic tool to determine logical operations of digital electronic systems.

postulates of Boolean Algebra:- (or) Laws of Boolean Algebra:-

An Algebraic structure defined by a set of elements with two binary operations

1. Boolean sum 2.

2. " product

3. one unary operations $\bar{A}$ (or) A'

4. set of rules to simplify the logic expressions. [Complement]

Boolean sum

$$0+0=0$$

$$0+1=1$$

$$1+0=1$$

$$1+1=1$$

Boolean product

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

1. Annulment Law:-

(17)

$$A+1=1$$

$$A \cdot 0 = 0$$

2. Identity:- $A+0=A$ $A \cdot 1=A$

3. Complement:- If $A=0$, then $\bar{A}=1$

$$A+\bar{A}=1$$

$$A \cdot \bar{A}=0$$

4. Double Negation:- $\overline{\bar{A}}=A$

5. Idempotent Laws:-

$$i) A+A=A$$

$$ii) A \cdot A=A$$

$$A) (A+A) \cdot 1$$

$$(A+A)(A+\bar{A})$$

$$AA+A\bar{A}+AA+A\bar{A}$$

$$\Rightarrow A+0+A+0$$

$$\Rightarrow A+A=A$$

$$\Rightarrow A \cdot A \neq 0$$

$$\Rightarrow A \cdot A + A \cdot \bar{A}$$

$$\Rightarrow A[A+\bar{A}]$$

$$\Rightarrow A \cdot 1$$

$$\Rightarrow A$$

6. Absorption Law:-

$$i) A+AB=A$$

$$L.H.S \Rightarrow A+AB$$

$$\Rightarrow A[1+B]$$

$$\Rightarrow A$$

$$\left. \begin{array}{l} HB=1 \end{array} \right\}$$

$$ii) A(A+B)=A$$

$$L.H.S \Rightarrow A(A+B)$$

$$\Rightarrow AA+AB$$

$$\Rightarrow A+AB$$

$$\Rightarrow A[1+B]$$

$$\Rightarrow A$$

Simplify the following logic expressions using Boolean Algebra:-

i) $(A+B)(A+\bar{B})$

A) $AA + A\bar{B} + AB + B\bar{B}$

$\Rightarrow A + A\bar{B} + AB + 0$

$\Rightarrow A + A[B + \bar{B}]$

$\Rightarrow A + A \cdot 1 \quad \left\{ B + \bar{B} = 1 \right\}$

$\Rightarrow A + A$

$\Rightarrow A$

ii) $ABC + A\bar{B}C + AB\bar{C}$

A) $\Rightarrow ABC + A\bar{B}C + AB\bar{C}$

$\Rightarrow AC[B + \bar{B}] + AB\bar{C}$

$\Rightarrow AC \cdot 1 + AB\bar{C}$

$\Rightarrow A[C + B\bar{C}] \quad \left\{ C + B\bar{C} = B + C \right\}$

$\Rightarrow A[B + C]$

iii) $AB + BC + \bar{B}C = AB + C$

A) $\Rightarrow AB + BC + \bar{B}C$

$\Rightarrow AB + C[B + \bar{B}] \quad \left\{ B + \bar{B} = 1 \right\}$

$\Rightarrow AB + C$

Important properties or theorems or laws of Boolean Algebra:-

Theorems of Boolean Algebra are useful in manipulating and simplifying the Boolean algebraic expressions.

S.No.

Theorems

specific purpose (18).

1.

$$A+0=A$$

$$A+1=1$$

$$A+A=A$$

$$A+\bar{A}=1$$

OR operation

2.

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

$$A \cdot A = A$$

$$A \cdot \bar{A} = 0$$

AND operation

3.

$$\bar{\bar{A}} = A \left\{ \begin{array}{l} \text{if } A=0 \text{ then } \bar{A}=1 \\ A=1, \text{ then } \bar{A}=0 \end{array} \right\} \text{Complement operation}$$

4.

$$\overline{A+B} = \bar{A} \cdot \bar{B}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

DeMorgan's Theorem

5.

$$A+B=B+A$$

$$A \cdot B = B \cdot A$$

Commutative Law

6.

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

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

Associative Law

7.

$$A \cdot (B+C) = A \cdot B + A \cdot C$$

$$(A+B) \cdot C = A \cdot C + B \cdot C$$

distributive Law

8.

$$A+\bar{A}B=A+B$$

$$A+AB=A$$

Auxiliary Law

Applications of Boolean Algebra:-

- * It is used in the theory of probability & Information theory.
- * It is widely used in digital computers.
- * It is used in set-theory and statistics.

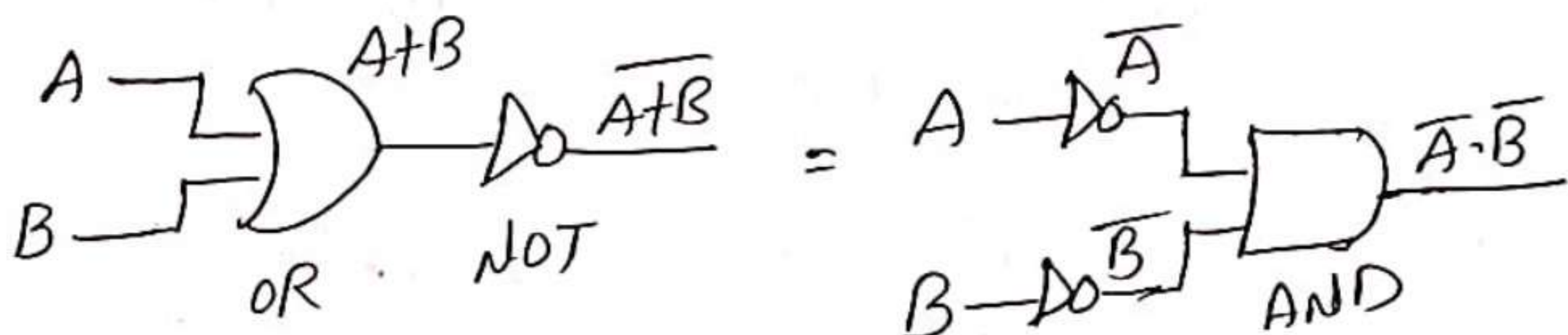
De-Morgan's theorem:-

- * A famous Mathematician De-morgan derived the two most important theorems of Boolean Algebra.

i) De-morgan's 1st theorem:-

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

- * The Complement of the sum of all the terms is equals to product of the complements of each term.



$$L.H.S = R.H.S$$

Truth table Verification:-

(19)

A	B	$A+B$	$\overline{A+B}$	=	A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	0	1		0	0	1	1	1
0	1	1	0		0	1	1	0	0
1	0	1	0		1	0	0	1	0
1	1	1	0		1	1	0	0	0

L.H.S = R.H.S

Example:-

i) $(A+B) + (\overline{A+B}) = 1$

A) $(A+B) + (\overline{A} \cdot \overline{B})$

$$(A+B+\overline{A}) \cdot (A+B+\overline{B})$$

$$(1+B) \cdot (1+A)$$

$$= 1 \cdot 1$$

$$= 1$$

$$\left. \begin{array}{l} \overline{A+B} = \overline{A} \cdot \overline{B} \\ 1+A=1 \\ A+\overline{A}=1 \end{array} \right\}$$

ii) $(A+B) \cdot (\overline{A+B}) = 0$

A) $(A+B) \cdot (\overline{A+B})$

$$\Rightarrow (A+B) \cdot (\overline{A} \cdot \overline{B})$$

$$\Rightarrow A(\overline{A} \cdot \overline{B}) + B(\overline{A} \cdot \overline{B})$$

$$= 0 + 0$$

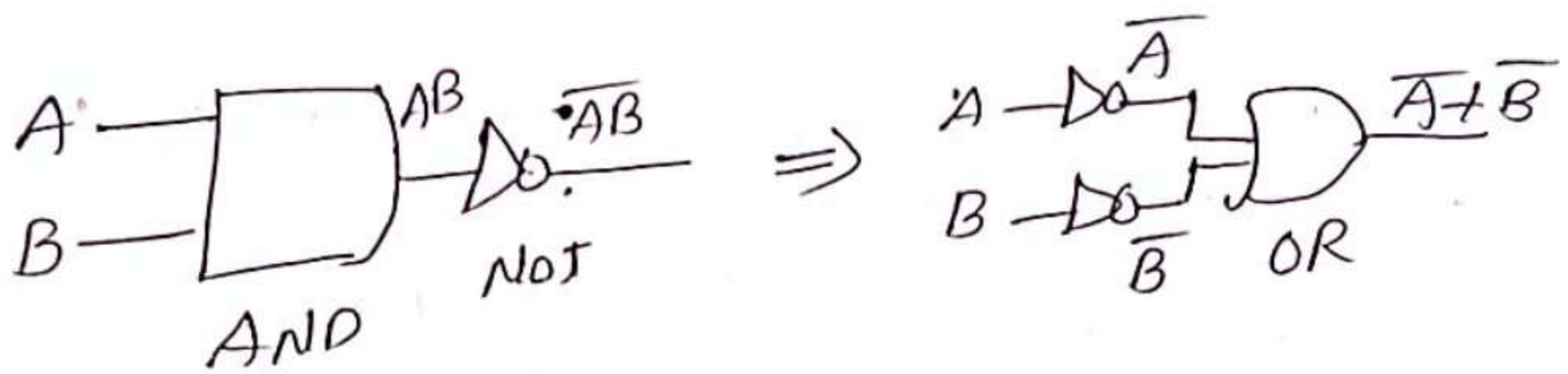
$$= 0$$

$$\left. \begin{array}{l} A \cdot \overline{A} = 0 \\ B \cdot \overline{B} = 0 \end{array} \right\}$$

ii) demorgan's 2nd theorem:-

$$\boxed{\overline{A \cdot B} = \overline{A} + \overline{B}}$$

* The complement of the product of all terms is equals to the sum of complements of each other.



Truth table:-

A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

=

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

L.H.S = R.H.S

Ex:-

1) $\overline{AB} + AB = 1$

1a) $\{ \overline{AB} = \overline{A} + \overline{B} \}$

$= (\overline{A} + \overline{B}) \cdot (AB) = 1$

$= (A + \overline{A} + \overline{B}) \cdot (B + \overline{A} + \overline{B}) = 1$

$= (1 + \overline{B}) \cdot (1 + \overline{A})$

$= 1 \cdot 1 = 1$

2) $(\overline{AB})(AB) = 0$

1a) $(\overline{A} + \overline{B})(AB) = 0$

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

$= 0$

$\{ A \cdot \overline{A} = 0 \}$

Simplification of logic expressions:-

(20)

1) $A \cdot \bar{A} C = 0$

2) $ABCD + ABD = ABD(1+C) = ABD \quad \{1+C=1\}$

3) $ABCD + A\bar{B}CD$
 $\Rightarrow ACD(B+\bar{B})$
 $\Rightarrow ACD$

4) $A(A+B) \Rightarrow AA+AB \Rightarrow A+AB$
 $\Rightarrow A(1+B)$
 $\Rightarrow A$

5) $AB + ABC + AB(D+E)$

Sol: $AB(1+C+D+E)$

$\Rightarrow AB$

$\left. \begin{array}{l} 1+C=1 \\ 1+D=1 \\ 1+E=1 \end{array} \right\}$

6) $xy + xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z}$

A) $xy(1+\bar{z}) + x\bar{y}\bar{z} + \bar{x}y\bar{z}$

$\Rightarrow xy + x\bar{y}\bar{z} + \bar{x}y\bar{z}$

$\Rightarrow xy(1+\bar{z}) + x\bar{y}\bar{z} + \bar{x}y\bar{z}$

$\{1+\bar{z}=1\}$

$\Rightarrow xy + \bar{x}y\bar{z}$

$\Rightarrow y(x+\bar{x}\bar{z})$

$\{A+\bar{A}B=A+B\}$

$\Rightarrow y(x+\bar{z})$

$$7) \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}BC$$

$$\Rightarrow \overline{A}\overline{C}[B + \overline{B}] + \overline{A}BC$$

$$\Rightarrow \overline{A}\overline{C} + \overline{A}BC$$

$$\Rightarrow \overline{A}[\overline{C} + BC]$$

$$\Rightarrow \boxed{\overline{A}[\overline{C} + B]}$$

$$\{ B + \overline{B} = 1 \}$$

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

$$8) ABC + A\overline{B}C + AB\overline{C}$$

$$\Rightarrow AC[B + \overline{B}] + AB\overline{C}$$

$$\Rightarrow AC + AB\overline{C}$$

$$\Rightarrow A[C + B\overline{C}] \Rightarrow A[B + C]$$

$$\{ C + B\overline{C} = C + B \}$$

$$9) A + C(A + \overline{A}B)$$

$$\Rightarrow A + AC + \overline{A}BC$$

$$\Rightarrow A + \overline{A}BC$$

$$\Rightarrow C[A + \overline{A}B]$$

$$\Rightarrow C[A + B]$$

$$10) \overline{A}B\overline{C}D + \overline{A}BCD + ABD$$

$$\Rightarrow \overline{A}BD[C + \overline{C}] + ABD$$

$$\Rightarrow \overline{A}BD + ABD$$

$$\Rightarrow BD[A + \overline{A}]$$

$$\Rightarrow B$$

$$\{ A + \overline{A} = 1 \}$$

$$11. A + \overline{A}B + A\overline{B} = A + B$$

$$A) \Rightarrow A + \overline{A}B + A\overline{B}$$

$$\Rightarrow A + B + A\overline{B}$$

$$\Rightarrow A[1 + \overline{B}] + B$$

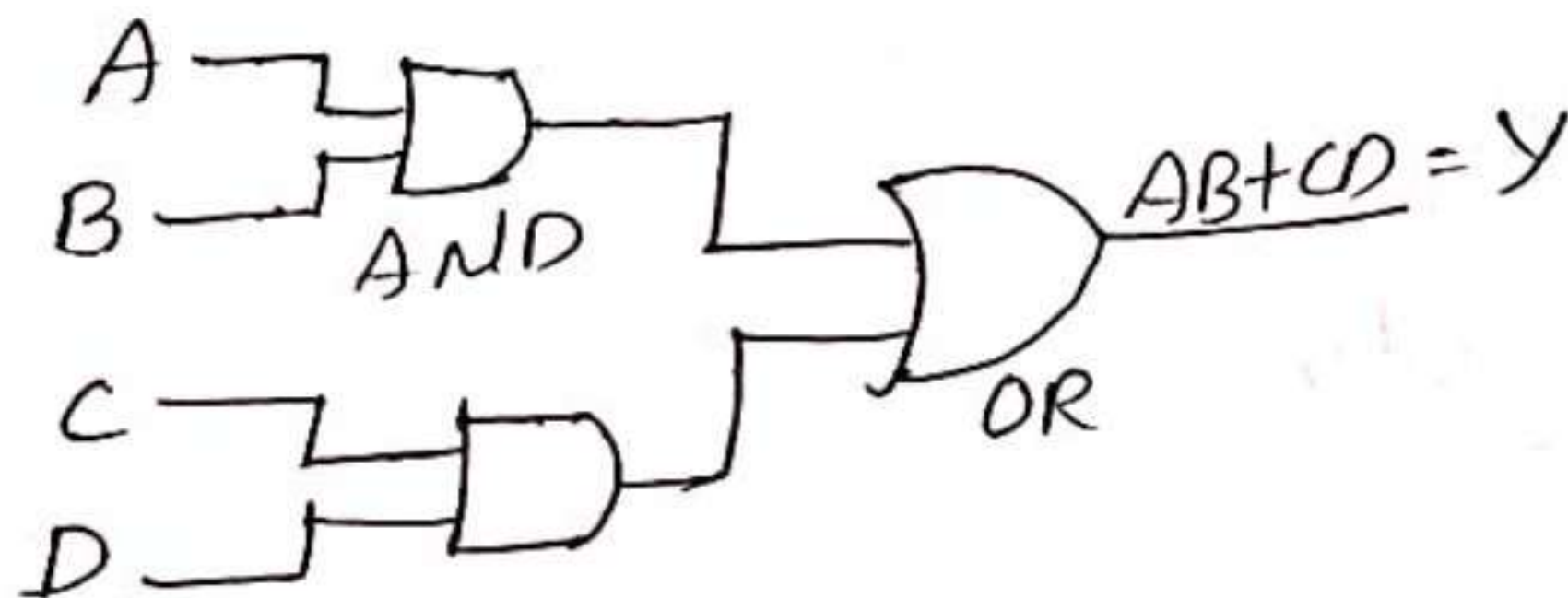
$$\Rightarrow A + B$$

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

Realizations of functions using logic gates:- (21)

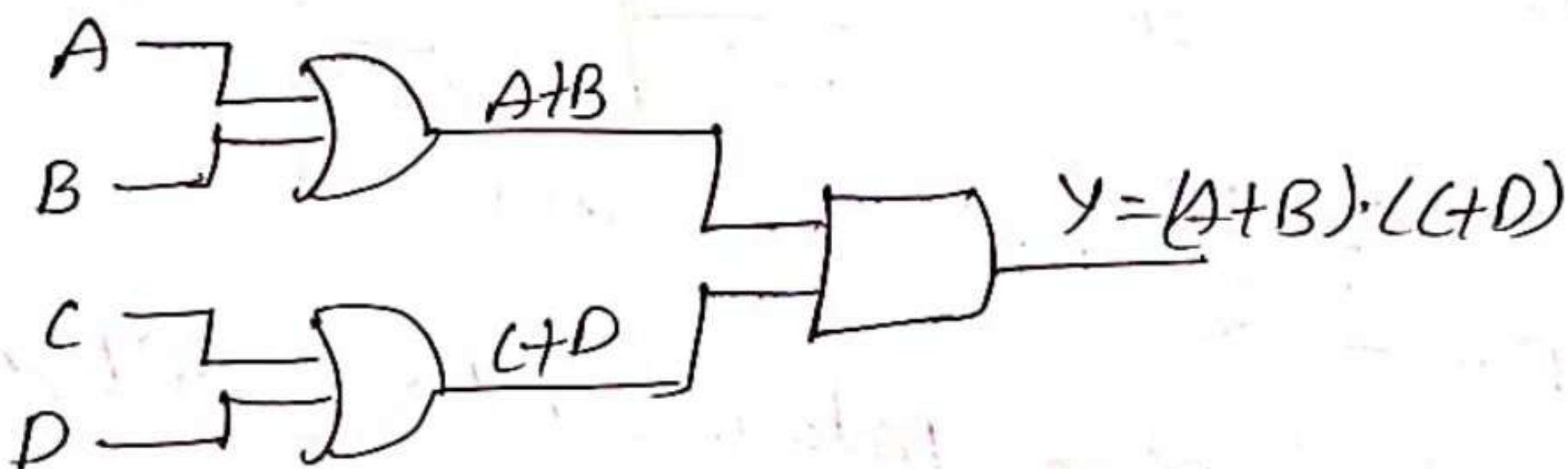
1. $Y = AB + CD$ Realize the function using logic gates

A) $Y = AB + CD$



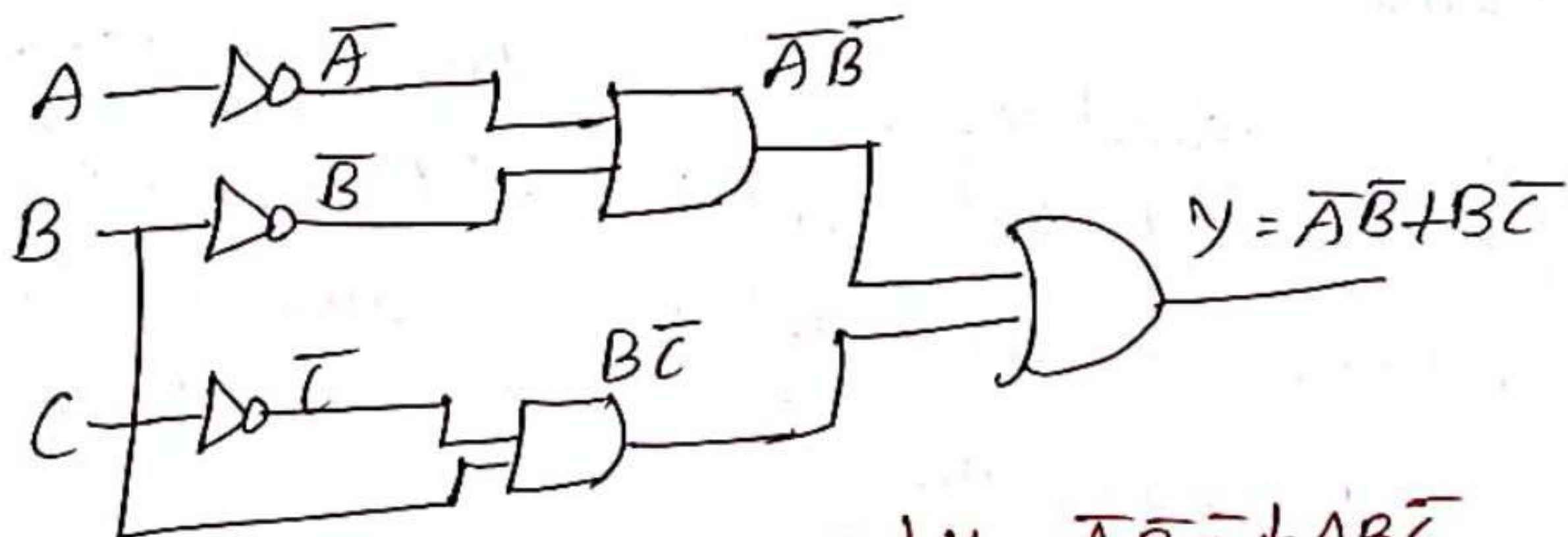
2) $(A+B) \cdot (C+D) = Y$

A)



3) $Y = \overline{A}\overline{B} + B\overline{C}$

A)



4) $Y = A + BC$

5) $Y = A + \overline{B}\overline{C} + DE\overline{F}$

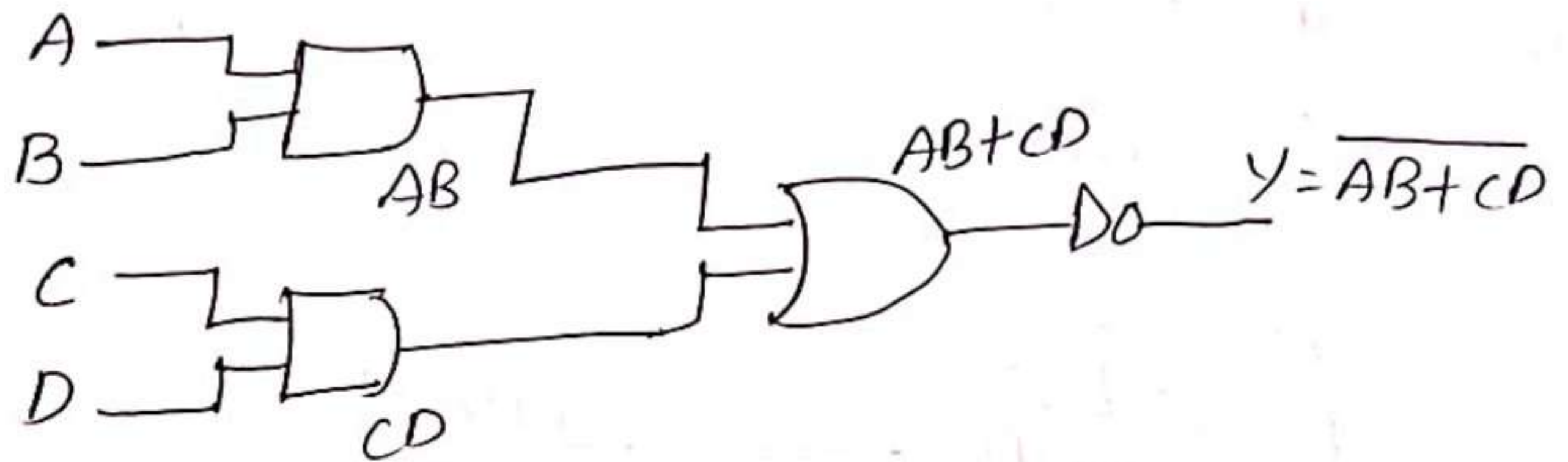
6) $Y = A(\overline{B} + \overline{C})(D + E + F)$

7) $Y = \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C}$

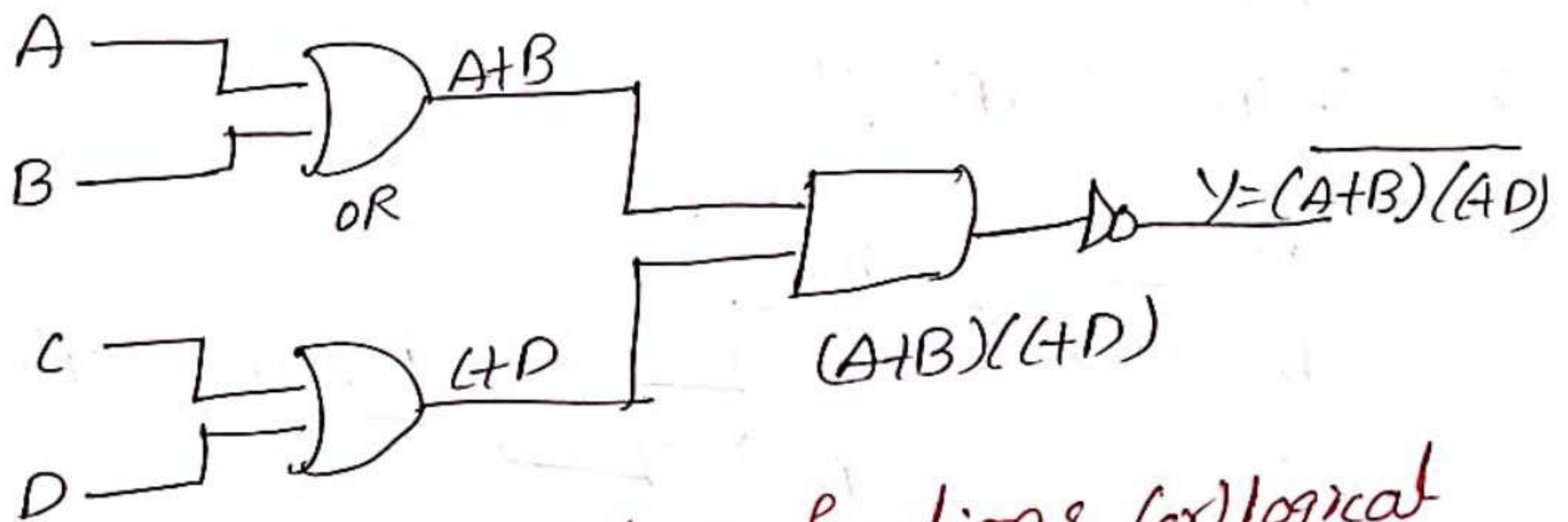
8) $Y = (A + \overline{B} + \overline{C})(A + \overline{B} + \overline{C})$

9) $Y = A\overline{B}\overline{C} + A\overline{B}\overline{C} + \overline{A}B\overline{C}$

10) AND (OR) Inverter:-



11) OR-AND Inverter:-



Canonical forms of Boolean functions (or) Logical functions

* Logical functions are generally expressed in terms of logical variables. These logical functions are represented in following terms.

1. SOP form [Sum of product]
2. POS form [Product of sum]

* SOP form:- The sum of product expression contains two (or) more AND functions ORed together. (22)

* These products are called as "Minterms".

* denoted in 'm'

Eq:- i) $f(A, B, C) = ABC + A\bar{B}C$

↓
sum

product

ii) $f(P, Q, R, S) = \bar{P}Q + QR + RS$

↓
sum

product

2. POS form:- The product of sum expressions contains two (or) more OR functions ANDed together.

* These product are called "Maxterms"

* denoted by "M"

Eq:- i) $(\bar{A} + \bar{B}C) \cdot (\bar{A} + B + C)$ ii) $(A + B + C) \cdot (\bar{A} + \bar{B} + C) \cdot (\bar{A} + B + C)$

↓
sum

product

Standard form of SOP (or) Canonical Boolean

Expressions:-

* If each product term in SOP form contains all the variables then the SOP form is known as "Standard form or Canonical SOP form".

* Similarly, If each sum term in pos-form contains all the variables then the pos-form is known as standard (or) canonical pos-form

Examples:-

1. Convert the given Boolean Expression into standard sop form.

$$\left. \begin{array}{l} \text{In SOP } A=1 \\ \bar{A}=0 \end{array} \right\}$$

i) $AB + BC + AC$

Sol:- $AB + BC + AC = AB(C + \bar{C}) + BC(A + \bar{A}) + AC(B + \bar{B})$

$$\Rightarrow ABC + AB\bar{C} + AB\bar{C} + \bar{A}BC + AB\bar{C} + A\bar{B}C$$

$$\Rightarrow ABC + \bar{A}BC + A\bar{B}C + AB\bar{C}$$

$$\Rightarrow 111 + 011 + 101 + 110$$

$$\Rightarrow m_7 + m_3 + m_5 + m_6 \Rightarrow \Sigma m(3, 5, 6, 7)$$

ii) $A + AB + ABC$

Sol:- $\Rightarrow A(B + \bar{B})(C + \bar{C}) + AB(C + \bar{C}) + ABC$

$$\Rightarrow (AB + A\bar{B})(C + \bar{C}) + AB\bar{C} + ABC$$

$$\Rightarrow ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + AB\bar{C} + ABC + ABC$$

$$\Rightarrow ABC + A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}\bar{C}$$

$$\Rightarrow 111 + 110 + 101 + 100 \Rightarrow m_7 + m_6 + m_5 + m_4$$

$$\Rightarrow \Sigma m(4, 5, 6, 7)$$

3) $A + \bar{B}$

(23)

Sol:- $A + \bar{B} \Rightarrow A[B + \bar{B}] + \bar{B}[A + \bar{A}]$

$$\Rightarrow AB + A\bar{B} + A\bar{B} + \bar{A}\bar{B}$$

$$\Rightarrow AB + A\bar{B} + \bar{A}\bar{B}$$

$$\Rightarrow 11 + 10 + 00 \Rightarrow m_3 + m_2 + m_0$$

$$\Rightarrow \Sigma m(0, 2, 3)$$

Examples:- Convert the following given Boolean expression into standard pos form.

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

$\left. \begin{array}{l} \text{In pos} \\ \text{If } A=0 \\ \bar{A}=1 \end{array} \right\}$

Sol:- $f = (A+B)(B+C)(C+A)$

$$f = (A+B+C\bar{C})(B+C+A\bar{A})(C+A+B\bar{B})$$

$$= (A+B+C)(A+B+\bar{C})(A+B+C)(\bar{A}+B+C)$$

$$(A+B+C)(A+\bar{B}+C)$$

$$= (A+B+C)(A+B+\bar{C})(\bar{A}+B+C)(A+\bar{B}+C)$$

$$= (000)(001)(100)(010)$$

$$= m_0 \cdot m_1 \cdot m_4 \cdot m_2$$

$$= \Pi M(0, 1, 2, 4)$$

2) $f = A(A+B)(A+B+C)$

Sol:- $f = (A+B\bar{B}+C\bar{C})(A+B+C\bar{C})(A+B+C)$

$$= (A+B)(A+\bar{B}+C\bar{C})(A+B+C)(A+B+\bar{C})(A+B+C)$$

$$= (A+B+C)(A+\bar{B}+\bar{C})(A+B+C)(A+B+\bar{C})(A+B+C)$$

$$= (A+B+C) (A+B+\bar{C}) (A+\bar{B}+C) (A+\bar{B}+\bar{C})$$

$$= (000) (001) (010) (011)$$

$$= M_0 \cdot M_1 \cdot M_2 \cdot M_3$$

$$\Rightarrow \Pi M(0,1,2,3)$$

H.W:- Convert the following expressions into pos terms.

$$1) f(A,B,C) = (A+B)(B+C)$$

$$2) f(A,B,C) = (A+B)(B+C)(\bar{C}+A)$$

$$3) f(x,y,z) = (x+y+z)(x+y) x$$

Example:-

1) Convert the following expression sop form into other canonical form.

$$f = \Sigma m(0,3,4,6)$$

Sol:- * Given function

$$f = \Sigma m(0,3,4,6)$$

$$f = m_0 + m_3 + m_4 + m_6$$

* The complement of the given function

$$\bar{f} = \Sigma m(1,2,5,7)$$

$$= m_1 + m_2 + m_5 + m_7$$

* By applying demorgan's theorem to the complement function.

$$\bar{\bar{f}} = f = \overline{m_1 + m_2 + m_5 + m_7}$$

$$= \bar{m}_1 \cdot \bar{m}_2 \cdot \bar{m}_5 \cdot \bar{m}_7$$

$$= M_1 \cdot M_2 \cdot M_5 \cdot M_7$$

$$\therefore f = \pi M(1, 2, 5, 7)$$

(2A)

2) Convert the following expression to the other canonical form,

$$f = \sum M(0, 1, 2, 5, 6, 7, 11) \quad \{ \sum M(0, 1) \pi M \}$$

Sol. Given function $f = \pi M(0, 1, 2, 5, 6, 7, 11)$

$$f = M_0 \cdot M_1 \cdot M_2 \cdot M_5 \cdot M_6 \cdot M_7 \cdot M_{11}$$

* The complement of the given function

$$\bar{f} = \pi M(3, 4, 8, 9, 10, 12, 13, 14, 15)$$

$$= M_3 \cdot M_4 \cdot M_8 \cdot M_9 \cdot M_{10} \cdot M_{12} \cdot M_{13} \cdot M_{14} \cdot M_{15}$$

* By applying De Morgan's theorem to the complement

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

$$\bar{f} = f = M_3 \cdot M_4 \cdot M_8 \cdot M_9 \cdot M_{10} \cdot M_{12} \cdot M_{13} \cdot M_{14} \cdot M_{15}$$

$$f = m_3 + m_4 + m_8 + m_9 + m_{10} + m_{12} + m_{13} + m_{14} + m_{15}$$

$$f = \sum m(3, 4, 8, 9, 10, 12, 13, 14, 15)$$

H.W:-

1) $f(A, B, C) = \sum(0, 1, 4, 6)$ convert into SOP to POS form.

2) $f(A, B, C) = \sum(\bar{A}\bar{B} + AC)$ find SOP & POS term.

2a) :- SOP form :- $\Sigma(\overline{A}\overline{B} + AC)$

$$\Rightarrow \overline{A}\overline{B}(C + \overline{C}) + AC(B + \overline{B})$$

$$\Rightarrow \overline{A}\overline{B}C + \overline{A}\overline{B}\overline{C} + ABC + A\overline{B}\overline{C}$$

$$\Rightarrow 001 + 000 + 111 + 101$$

$$\Rightarrow m_1 + m_0 + m_7 + m_5$$

$$\left. \begin{array}{l} \text{To SOP} \\ A = 1 \\ \overline{A} = 0 \end{array} \right\}$$

$$f \Rightarrow \Sigma m(1, 0, 7, 5) \Rightarrow \Sigma m(0, 1, 5, 7)$$

* The Complement of the given function is

$$\overline{f} = m_2 + m_3 + m_4 + m_6$$

$$\overline{f} = m_2 + m_3 + m_4 + m_6$$

* By Applying De Morgan's Theorem, The complement of $\overline{f}$ is

$$\overline{\overline{f}} = f = \overline{(m_2 + m_3 + m_4 + m_6)}$$

$$= M_2 \cdot M_3 \cdot M_4 \cdot M_6$$

$$f = \Pi M(2, 3, 4, 6)$$

3. $f(A, B, C) = \Sigma(0, 1, 4, 6)$ Convert given SOP into POS terms.

4. $f(A, B, C) = \Sigma m(0, 2, 7)$ Convert given SOP terms into POS forms.

5. $f(A, B, C, D) = \Sigma m(0, 1, 5, 6, 7, 8, 10, 11, 15)$ Convert SOP to POS

6. $f(A, B, C) = \pi(1, 2, 6)$ (convert pos to sop terms)

7. $f(A, B, C, D) = \pi(0, 1, 4, 5, 9, 14)$ (convert pos terms to sop form).

8. $(\bar{A} + \bar{B})$ expand minterms to maxterms.

9. expand $(A + B\bar{C} + AB\bar{D} + ABCD)$ to minterms to maxterms

10. expand $A(\bar{B} + A)B$ to maxterms to min terms

11. Write Boolean expression of minterms (Sop's) standard form, from the table

I/P's			O/P
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

In sop
 $\sim$
 $A = 1$
 $\bar{A} = 0$

Sol: Step 1:-

* Note down the I/P combinations for the O/P's as '1'

* Such cases, there are 001, 011, 101 & 111

step-2:-

* Write the product term for each case.

<u>A</u>	<u>B</u>	<u>C</u>	<u>product term</u>
0	0	1	$\bar{A}\bar{B}C$
0	1	1	$\bar{A}BC$
1	0	1	$A\bar{B}C$
1	1	1	ABC

step 3:- Write the 'Sop' expression.

$$\text{Sop} = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + ABC$$

$$= m_1 + m_3 + m_5 + m_7$$

$$\boxed{\text{Sop} = \sum m(1, 3, 5, 7)} \rightarrow \text{Min terms}$$

12) Write the Boolean expression, of Maxterms (Pos's), in the standard form, from the table

<u>I/P</u>	<u>O/P</u>
<u>A B C</u>	<u>Y</u>
000	1
001	0
010	0
011	1
100	1
101	1
110	0
111	0

$\left. \begin{array}{l} \text{TO POS} \\ A=0 \\ \bar{A}=1 \end{array} \right\}$

(26)

Ans:-Step 1:-

* Note down the I/P combinations for the o/p is 0.

* These are 010, 011, 110, 111

Step 2:-

* Write the sum term, each case.

I/P combinations , sum term

A	B	C
0	1	0
0	1	1
1	1	0
1	1	1

$$A + \bar{B} + C$$

$$A + \bar{B} + \bar{C}$$

$$\bar{A} + \bar{B} + C$$

$$\bar{A} + \bar{B} + \bar{C}$$

$$* \text{ POS expression } = (A + \bar{B} + C)(A + \bar{B} + \bar{C})(\bar{A} + \bar{B} + C)(\bar{A} + \bar{B} + \bar{C})$$

$$\Rightarrow M_2 \cdot M_3 \cdot M_6 \cdot M_7$$

$$\boxed{\text{POS} \Rightarrow \prod M(0) \cup M(2, 3, 6, 7)} \rightarrow \text{Max terms.}$$

2M*
K-Map:- [Karnaugh Map]

* 2nd K-Map :- [Karnaugh Map] :-

* The K-map is made up of 2^n cells. Each of the cell represents one of the 2^n possible product (or) sum terms that can be formed from n variables.

* 2-Variable map contains $\Rightarrow 2^2 = 4$ cells

* 3- " " " " $\Rightarrow 2^3 = 8$ cells

* 4- " " " " " " $\Rightarrow 2^4 = 16$ cells

* 5- " " " " " " $\Rightarrow 2^5 = 32$ cells

A \ B	0	1
0	0	1
1	2	3

2 Variable K-map

A \ B ^c	0	1	3	2
0	0	1	3	2
1	4	5	7	6

3-Variable K-Map

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

4-Variable K-Map

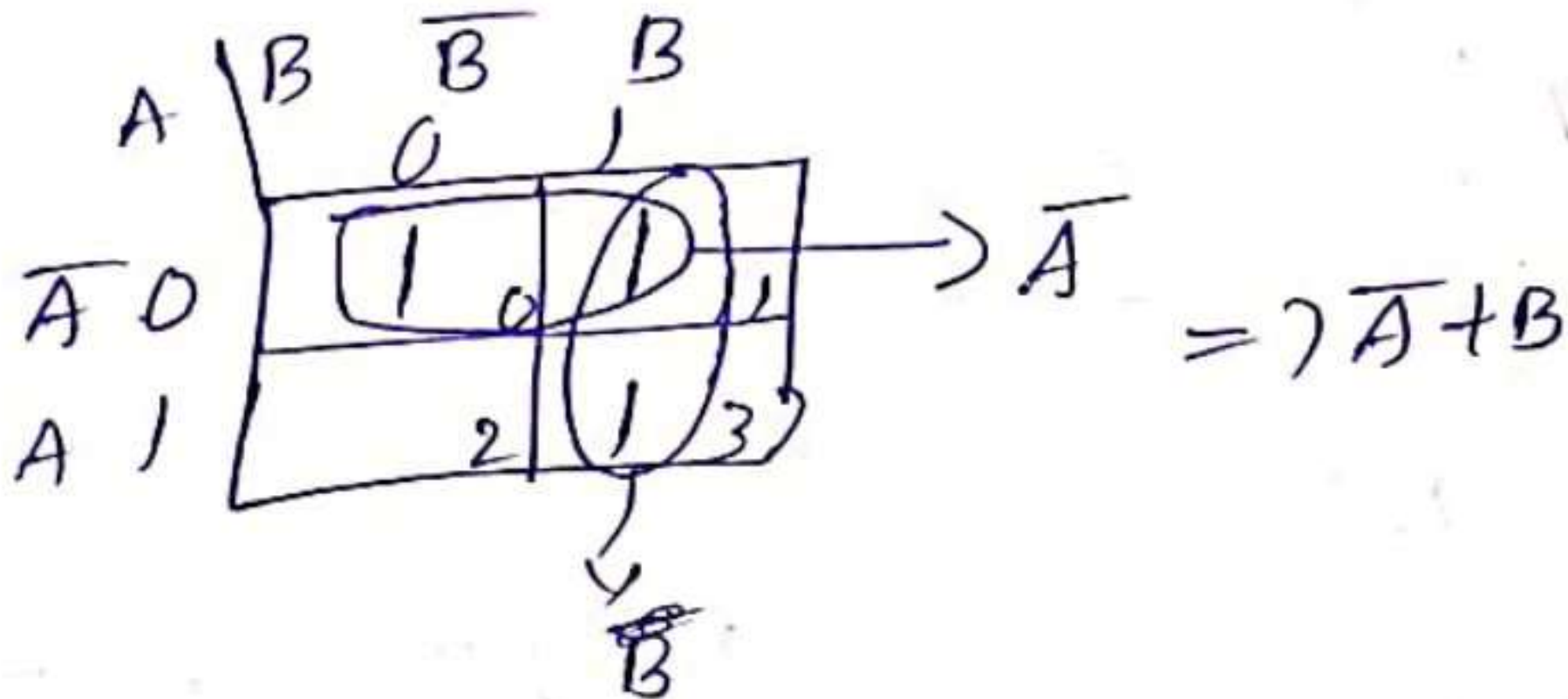
Examples:-

(27)

2-Variable K-map:-

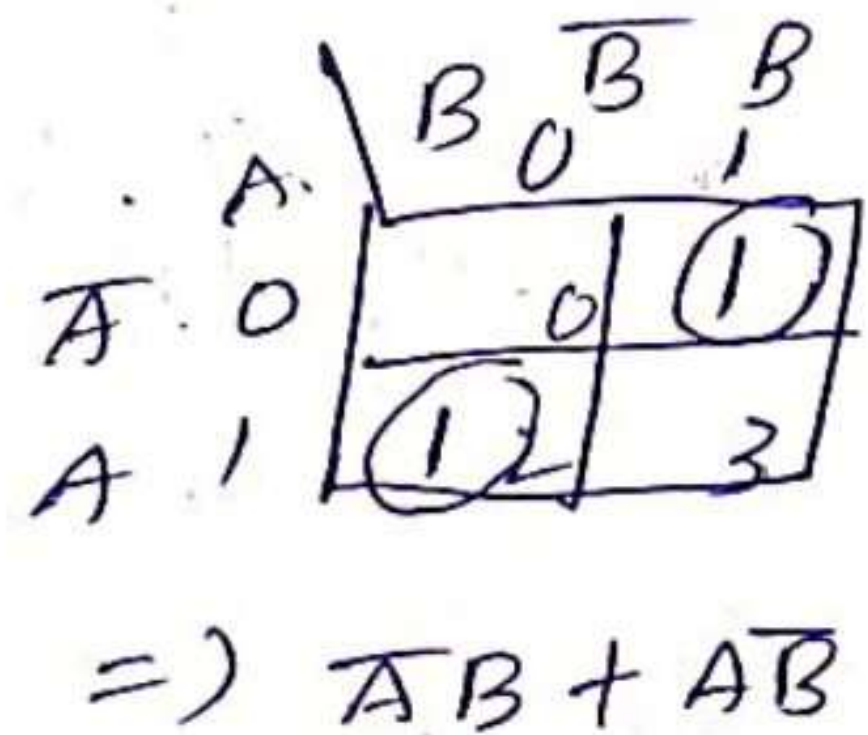
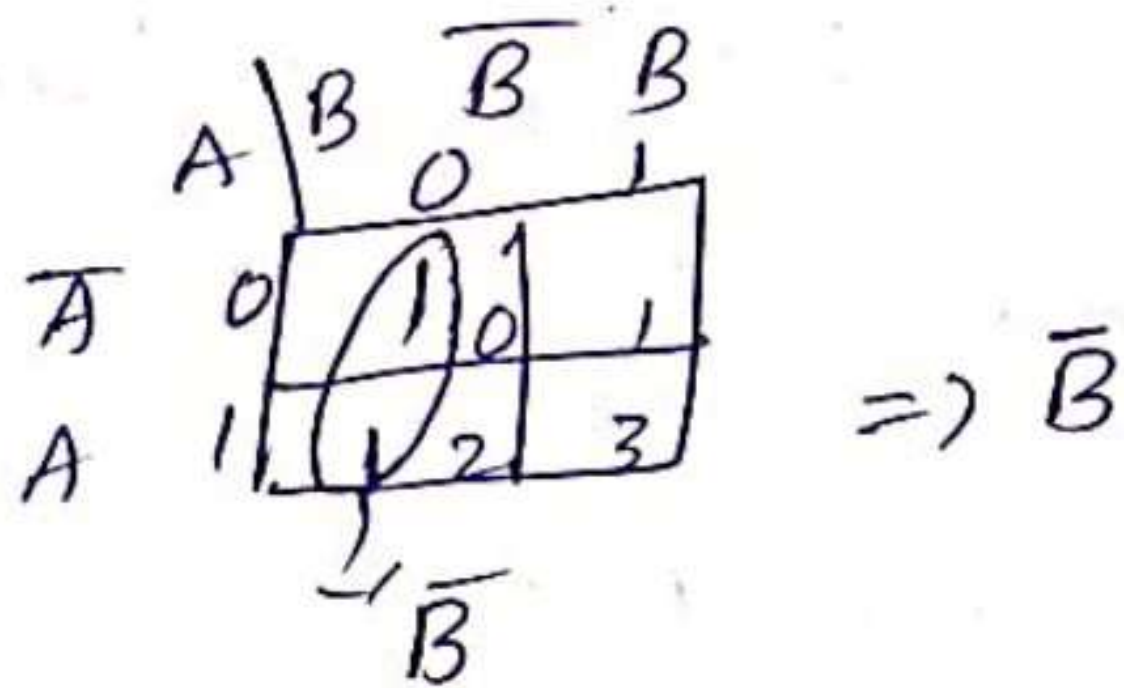
1) $f_1 = \Sigma m(0, 1, 3)$

Ans:-

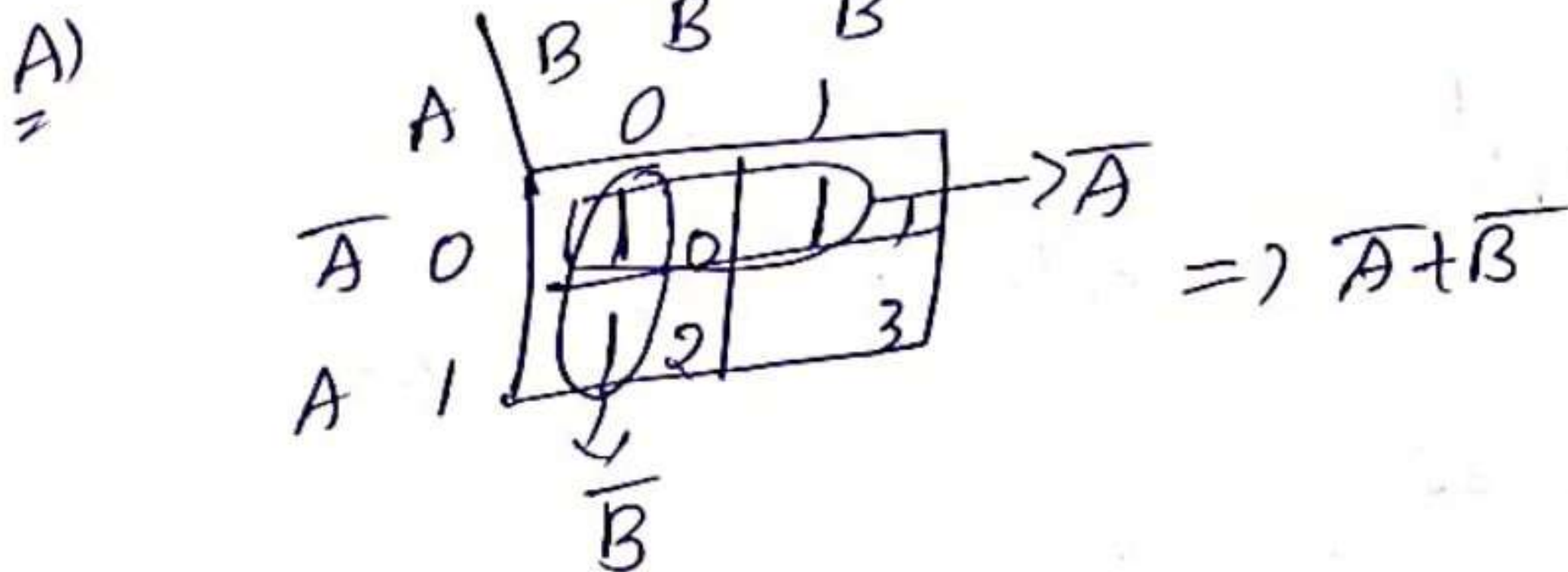


2) $A\bar{B} + \bar{A}\bar{B}$ (or) $\Sigma m(0, 2)$ 3) $A\bar{B} + \bar{A}B$ (or) $\Sigma m(1, 2)$

A)



4) $A\bar{B} + \bar{A}B + \bar{A}\bar{B}$ (or) $\Sigma m(0, 1, 2)$



3-Variable K-Map:-

1) Em (0, 1, 5)

A)

		$B\bar{C}$	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
		00	01	11	10	
$\bar{A}$	0	1	0	1	0	0
A	1	0	1	0	1	0

Groupings: $\bar{A}\bar{B}$ (cells 0, 1), $\bar{B}C$ (cells 1, 5)

$$\Rightarrow \bar{A}\bar{B} + \bar{B}C$$

2) Em (0, 1, 3, 5)

A)

		$B\bar{C}$	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
		00	01	11	10	
$\bar{A}$	0	1	0	1	0	0
A	1	0	1	0	1	0

Groupings: $\bar{A}\bar{B}$ (cells 0, 1), $\bar{B}C$ (cells 1, 5), $\bar{A}C$ (cells 0, 3)

grouping of 2 Variables Contains We can reduce the Expression.

$$\Rightarrow \bar{A}\bar{B} + \bar{B}C + \bar{A}C$$

3) Em (0, 1, 5, 7)

A)

		$B\bar{C}$	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
		00	01	11	10	
$\bar{A}$	0	1	0	1	0	0
A	1	0	1	0	1	0

Groupings: $\bar{A}\bar{B}$ (cells 0, 1), $\bar{A}C$ (cells 0, 3), AC (cells 3, 7)

$$\Rightarrow \bar{A}\bar{B} + AC$$

4) Em (0, 2, 3, 4, 5, 6)

A)

		$B\bar{C}$	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
		00	01	11	10	
$\bar{A}$	0	1	0	1	0	0
A	1	0	1	0	1	0

Groupings: $\bar{A}\bar{B}$ (cells 0, 1), $\bar{A}B$ (cells 2, 3), $\bar{A}C$ (cells 0, 2), AB (cells 4, 5), $\bar{A}B$ (cells 6, 7)

$$\Rightarrow \bar{A}\bar{B} + \bar{A}B + \bar{A}C$$

5) Em (0, 1, 3, 4, 5, 7)

(28)

sol:-

A \ BC	00	01	11	10
	$\overline{B}\overline{C}$	$\overline{B}C$	$B\overline{C}$	BC
$\overline{A}$ 0	1	1	1	0
A 1	1	1	1	1

$\Rightarrow \overline{B} + \overline{C}$

6) Em (0, 1, 2, 4)

A \ BC	00	01	11	10
	$\overline{B}\overline{C}$	$\overline{B}C$	$B\overline{C}$	BC
$\overline{A}$ 0	1	1	0	1
A 1	1	0	0	0

$\Rightarrow \overline{A}\overline{B} + \overline{B}\overline{C} + \overline{A}C$

7) Em (0, 1, 2, 3)

A \ BC	00	01	11	10
	$\overline{B}\overline{C}$	$\overline{B}C$	$B\overline{C}$	BC
$\overline{A}$ 0	1	1	1	1
A 1	0	0	0	0

$\Rightarrow \overline{A}$

8) Em (0, 2, 5)

A \ BC	00	01	11	10
	$\overline{B}\overline{C}$	$\overline{B}C$	$B\overline{C}$	BC
$\overline{A}$ 0	1	0	1	0
A 1	0	1	0	1

$\Rightarrow \overline{A}\overline{C} + A\overline{B}C$

9) $\Sigma m(1, 2, 3, 4, 5, 7)$

A \ BC	$\overline{B}\overline{C}$ $\overline{B}C$ $B\overline{C}$ BC			
	00	01	11	10
$\overline{A}$ 0	0	1	1	1
A 1	1	1	1	0

$\Rightarrow \overline{A}B + A\overline{B} + C$
 $\overline{A}B$ (circled in top row)
 $A\overline{B}$ (circled in bottom row)
 C (circled in middle column)

10) $\Sigma m(0, 1, 2, 3, 4, 5, 6, 7)$

A \ BC	$\overline{B}\overline{C}$ $\overline{B}C$ $B\overline{C}$ BC			
	00	01	11	10
$\overline{A}$ 0	1	1	1	1
A 1	1	1	1	1

$\Rightarrow 1$

ii) Variable K-map

1) $\Sigma m(0, 2, 4, 5, 6, 10, 13, 14)$

eg:

AB \ CD	$\overline{C}\overline{D}$ $\overline{C}D$ $C\overline{D}$ CD			
	00	01	11	10
$\overline{A}\overline{B}$ 00	1	0	1	3
$\overline{A}B$ 01	1	4	5	7
$A\overline{B}$ 11	12	13	15	14
AB 10	8	9	11	10

$\Rightarrow \overline{D}\overline{A}$ (circled in top-left)
 $\Rightarrow \overline{C}\overline{D}$ (circled in top-right)
 $\Rightarrow B\overline{C}D$ (circled in bottom-middle)
 $\Rightarrow \overline{C}\overline{D} + \overline{D}\overline{A} + B\overline{C}D$

2) $\Sigma m(3, 4, 5, 6, 11, 14)$

A)

AB \ CD	$\overline{C}\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\overline{B}$ 00	0	1	3	2
$\overline{A}B$ 01	4	5	7	6
AB 11	12	13	15	14
$A\overline{B}$ 10	8	9	11	10

$\rightarrow \overline{B}CD$
 $\rightarrow B\overline{C}\overline{D}$
 $\Rightarrow \overline{A}B\overline{C} + \overline{B}CD + B\overline{C}\overline{D}$
 $\downarrow \overline{A}B\overline{C}$

3) $\Sigma m(0, 2, 5, 7, 8, 10, 13, 15)$

A)

AB \ CD	$\overline{C}\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\overline{B}$ 00	0	1	3	2
$\overline{A}B$ 01	4	5	7	6
AB 11	12	13	15	14
$A\overline{B}$ 10	8	9	11	10

$\rightarrow$ Adjacent pairs
 $\overline{B}\overline{D} + BD$

4) $\Sigma m(1, 3, 4, 5, 9, 11, 12, 13)$

A)

AB \ CD	$\overline{C}\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\overline{B}$ 00	0	1	3	2
$\overline{A}B$ 01	4	5	7	6
AB 11	12	13	15	14
$A\overline{B}$ 10	8	9	11	10

$\rightarrow D\overline{B}$
 $\Rightarrow \overline{B}D + B\overline{C}$
 $\rightarrow B\overline{C}$

5) $\Sigma m(3, 4, 5, 7, 9, 13, 14, 15)$

A)

AB \ CD	$\overline{C}\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\overline{B}$ 00	0	1	1	2
$\overline{A}B$ 01	1	4	1	5
AB 11	12	13	1	14
$A\overline{B}$ 10	8	9	11	10

$$\Rightarrow \overline{A}B\overline{C} + A\overline{C}D + AB\overline{C} + \overline{A}CD$$

6) $\Sigma m(2, 6, 7, 9, 13, 15)$

A)

AB \ CD	$\overline{C}\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\overline{B}$ 00	0	1	3	1
$\overline{A}B$ 01	4	5	1	6
AB 11	12	13	1	14
$A\overline{B}$ 10	8	9	11	10

$$\Rightarrow A\overline{C}D + B\overline{C}D + \overline{A}C\overline{D}$$

7) $\Sigma m(0, 2, 3, 7, 11, 13, 14, 15)$

AB \ CD	$\overline{C}\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\overline{B}$ 00	1	0	1	1
$\overline{A}B$ 01	4	5	1	6
AB 11	12	13	1	14
$A\overline{B}$ 10	8	9	1	10

$$\Rightarrow CD + \overline{A}\overline{B}\overline{D} + ABD + AB\overline{C}$$

8) Em $\{0, 2, 3, 5, 7, 8, 10, 11, 14, 15\}$

A) $\overline{B}\overline{C} + CD + \overline{A}BD + AC$

9) Em $\{0, 1, 2, 6, 8, 10, 13, 14\}$

A) $C\overline{D} + \overline{D} + \overline{A}\overline{B}\overline{C} + AB\overline{C}\overline{D}$

10) Em $\{0, 1, 2, 8, 9, 10\}$

A) $\overline{B}\overline{C} + \overline{B}\overline{D}$

11) Em $\{0, 2, 5, 6, 7, 8, 10, 13, 14, 15\}$

A) $\overline{B}\overline{D} + BD + BC$

12) Em $\{0, 1, 5, 6, 8, 9, 13, 14\}$

A) $C\overline{D} + B\overline{C}\overline{D} + \overline{B}\overline{C}$

13) Em $\{0, 1, 2, 3, 4, 6, 7, 8, 9, 11, 15\}$

A) $\overline{B}\overline{C} + \overline{A}\overline{D} + CD$

14) Em $\{1, 3, 4, 5, 6, 9, 11, 12, 13\}$

A) $1 \Rightarrow \overline{D}B + D\overline{B} + \overline{C}D$

15) Em $\{0, 4, 6, 8, 9, 10, 11, 14, 15\}$

A) $\overline{A}\overline{C}\overline{D} + B\overline{C}\overline{D} + AC + A\overline{B}$

16) Em $\{0, 1, 3, 5, 7, 8, 9, 11, 13, 15\}$

A) $D + \overline{B}\overline{C}$

16) $\Sigma m(0, 1, 2, 3, 4, 8, 9, 10, 11, 12)$

A) $\overline{C}D + B$

17) $\Sigma m(0, 1, 2, 3, 4, 5, 6, 8, 9, 12, 13, 14)$

A) $\overline{A}B + \overline{C} + B\overline{D}$

Minimize the expressions:-

(In SOP $A=1$
 $\overline{A}=0$)

1) $Y = \overline{A}\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + \overline{A}B\overline{C}$

A) $Y = 101 + 001 + 011 + 100 + 000$

$\Rightarrow (5, 1, 3, 4, 0)$

$= \Sigma m(0, 1, 3, 4, 5)$

AB \ CD		CD			
		00	01	11	10
$\overline{A}\overline{B}$	00	0	1	3	2
$\overline{A}B$	01	4	5	7	6
AB	11	12	13	15	14
$A\overline{B}$	10	8	9	11	10

A \ BC		BC			
		00	01	11	10
$\overline{A}$	0	1	0	1	3
A	1	1	4	1	5

$\overline{B} + C\overline{A}$

2) $Y = \overline{A}\overline{B}\overline{C} + \overline{B}C + \overline{A}B$

A) $000 + 01$

$Y = \overline{A}\overline{B}\overline{C} + \overline{B}C(A + \overline{A}) + \overline{A}B(C + \overline{C})$

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

$= 000 + 101 + 001 + 011 + 001$

$= \Sigma m(0, 1, 2, 3, 5)$

(31)

		$B\bar{C}$	$\bar{B}\bar{C}$	$\bar{B}C$	BC	BC
	A	00	01	11	10	
$\bar{A}$	0	1	1	1	1	2
A	1	4	5	6	7	6

$$\Rightarrow \bar{A} + \bar{B}C$$

$$3) Y = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} + A\bar{B}\bar{C}D + \bar{A}\bar{B}CD$$

$$\text{Ans: } Y = \bar{B}\bar{C} + A\bar{C}D + \bar{A}\bar{B}CD$$

Don't Care Conditions using in K-Map:-

- * The don't care is treated as denoted by d, X (or) $\emptyset$.
- * during the process of design using the as SOP map, each don't care is treated as '1' if it is helpful in map reduction, otherwise treated as '0' and left alone.
- * similarly, during the process of design using a POS map, each don't care is treated as a '0' if it is useful in map reduction, otherwise it is treated as a '1' and left alone.

K-map by don't care conditions:-

Eg:- 1) Em (1,3,4,5, 9,11,14,15) + Ed (2,6,7,8)

Sol:-

AB \ CD	00	01	11	10
$\overline{A}\overline{B}$ 00	0	1	1	X ₂
$\overline{A}B$ 01	1	1	X ₃	X ₆
AB 11			1	1
$A\overline{B}$ 10	X ₈	1	1	1

$$\Rightarrow \overline{A}B + BC + D\overline{B}$$

2) Em (1,3,7, 11,15) + Ed (0,2,5)

A)

AB \ CD	00	01	11	10
$\overline{A}\overline{B}$ 00	X ₀	1	1	X ₂
$\overline{A}B$ 01		X ₅	1	
AB 11			1	
$A\overline{B}$ 10			1	

$$\Rightarrow CD + \overline{A}\overline{B}$$

3) Em (1,5,9, 11,12,13) + Ed (3,10,14,15)

A)

AB \ CD	00	01	11	10
$\overline{A}\overline{B}$ 00	0	1	X ₃	2
$\overline{A}B$ 01		1		6
AB 11	1	1	X ₁₅	X ₁₄
$A\overline{B}$ 10		1	1	X ₁₀

$$\Rightarrow AB + D\overline{B} + \overline{C}D$$

4) $\Sigma m(0, 7, 10) + \Sigma d(2, 5, 8, 15)$

(32)

A)

AB \ CD		CD			
		$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$	00	1 X_0	0	3 X_2	2 X_2
$\bar{A}B$	01	4 X_5	5 X_5	7	6
AB	11	12	13	15 X_{15}	14
$A\bar{B}$	10	8 X_8	9	11	10

$\bar{D}\bar{B} + DAB$

examples:-

1) $f(A, B, C, D) = \Sigma m(1, 4, 7, 10, 13) + \Sigma d(5, 14, 15)$

Ans:- $f = \bar{A}B\bar{C} + \bar{A}\bar{C}D + BD + ACD$

2) $f(A, B, C, D) = \Sigma m(4, 5, 7, 12, 14, 15) + \Sigma d(3, 8, 10)$

Ans:- $f = A\bar{D} + BCD + \bar{A}B\bar{C}$

3) $f(W, X, Y, Z) = \Sigma m(1, 3, 7, 11, 15) + \Sigma d(0, 2, 5)$

Ans:- $f = \bar{W}\bar{X} + YZ$

Pos-forms:-

i) $f(a, b, c, d) = \Pi M(0, 1, 2, 3, 4, 10, 11, 15)$

Ans:- $(B + \bar{C})(A + B)(A + C + D)(\bar{A} + \bar{B} + \bar{C})$

ii) $f(W, X, Y, Z) = \Pi M(2, 3, 4, 6, 9, 11, 12, 13)$

Ans:- $(W + \bar{Y} + \bar{Z})(X + Y + \bar{Z})(\bar{W} + Y + \bar{Z})(\bar{X} + Y + \bar{Z})$

$$c) f(A, B, C, D) = \prod M(6, 7, 8, 9) \cdot \prod d(10, 11, 12, 13, 14, 15)$$

Ans) $\overline{A}(B+\overline{C})$

$$4) \prod M(1, 8, 15) \cdot \prod d(4, 5, 9, 14)$$

Sol:-

AB \ CD	00	01	11	10
AB 00		0	3	2
$\overline{A}\overline{B}$ 01	X	4	7	6
$\overline{A}\overline{B}$ 11			15	X
$\overline{A}\overline{B}$ 10	8	X	9	10

7th POS
 $A=0$
 $\overline{A}=1$

$$\rightarrow (A + (\overline{C} + \overline{D})) (\overline{C} + \overline{A} + \overline{B})$$

$$(C + \overline{A} + B)$$

$$5) \prod M(1, 2, 3, 8, 9, 10, 11, 14) \cdot \prod d(7, 15)$$

Sol:-

AB \ CD	00	01	11	10
AB 00		0	3	2
$\overline{A}\overline{B}$ 01	4	5	X	6
$\overline{A}\overline{B}$ 11	12	13	X	14
$\overline{A}\overline{B}$ 10	8	9	11	10

$$(\overline{A} + B)(\overline{D} + B)(\overline{C} + B)$$

$$(\overline{C} + \overline{A})$$

K-map Important questions:-

(33)

1) $f(x, y, z) = \sum (0, 1, 5, 7)$

Ans:- $x\bar{y} + xz$

2) $f(A, B, C) = \sum (0, 2, 5, 7) \Rightarrow$ Ans:- $AC + \bar{A}\bar{C}$

3) $f(A, B, C) = \sum (0, 2, 3, 4, 6, 7) \Rightarrow$ Ans:- $A + B + \bar{C}$

4) simplify the following Boolean expression by using K-map.

$$f(A, B, C) = AB\bar{C} + ABC + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C}$$

Ans:- $\bar{A}\bar{B} + AB$

5) simplify the following Boolean function into minimal sop form by using K-map

i) $f(A, B, C, D) = \sum (0, 1, 2, 3, 4, 6, 8, 9, 10, 11)$

Ans:- $\bar{A}\bar{D} + \bar{B}$

ii) $f(A, B, C, D) = \sum (0, 1, 3, 7, 8, 9, 11, 15)$

Ans:- $CD + \bar{B}\bar{C}$

iii) $f(A, B, C, D) = \sum (1, 3, 6, 8, 9, 10)$

Ans:- $\bar{A}\bar{B}D + AB\bar{C} + A\bar{B}\bar{D} + \bar{A}BCD$

iv) $f(W, X, Y, Z) = \bar{W}\bar{X}\bar{Y}\bar{Z} + W\bar{X}\bar{Y}\bar{Z} + \bar{W}X\bar{Y}Z + W\bar{X}YZ$

Ans:- $\bar{W}\bar{X}\bar{Y}\bar{Z} + \bar{W}X\bar{Y}\bar{Z} + W\bar{X}\bar{Y}\bar{Z} + W\bar{X}YZ$

v) $f(A, B, C, D) = A\bar{B}\bar{C} + AC + \bar{A}CD$

Ans:- $C\bar{D} + \bar{A}\bar{B} + AC$

1. Introduction

2. Background

3. Methodology

4. Results

5. Discussion

6. Conclusion

7. References

8. Appendix

9. Bibliography

10. Glossary

11. Index

12. Acknowledgements

13. About the Author

14. Contact Information

15. Disclaimer

16. Copyright Notice

17. Privacy Policy

18. Terms of Service

UNIT-II

①

Combinational logic circuits

* Basic Combinational circuit:-

def:- A logic circuit which has no memory, no feedback from output to input and the present o/p depends on the present i/p, this logic circuit is called as "Combinational logic circuit".

* Combinational circuit designed with logic gates. It consists of i/p variables, logic gates & output variables.

* They are used for performing arithmetic operations, logical operations & data comparison and data manipulation.

* Some of the Basic Combinational circuits are Half Adder, full Adder, Half subtractor, Parallel Adder, subtractor and digital comparator.

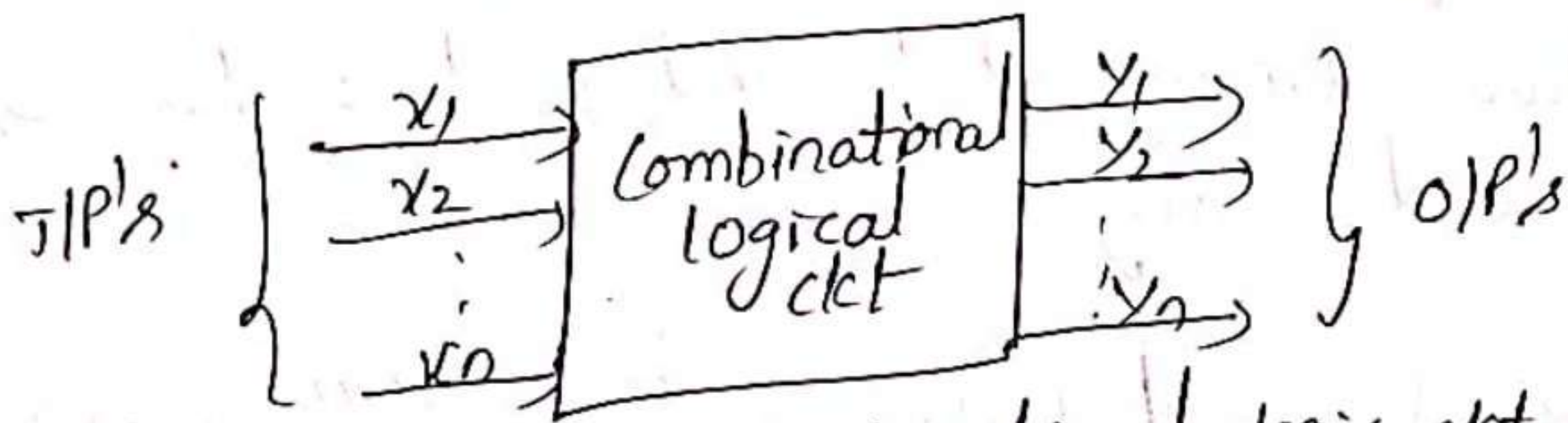


fig:- Combinational logic ckt.

Design procedure:-

* design procedure for Combinational logic circuit

1. for a given specification determine number of inputs and outputs.
2. Assign symbols to the Variables
3. derive a truth table as Required for Specifications.
4. determine a minimized boolean expression for the o/p variables by using K-map.
5. Implement the logic ckt by using basic gates (AND, OR, NOT, NAND, NOR, XOR w/ XNOR)

* Combinational units are divided into 2 types

1. Half Adder
2. Full Adder.

1. Half Adder:-

Half Adder is a logic unit that add two binary bits is known as "Half Adder". The function of Half Adder is to add two binary bits and produce two o/p's sum and Carry.

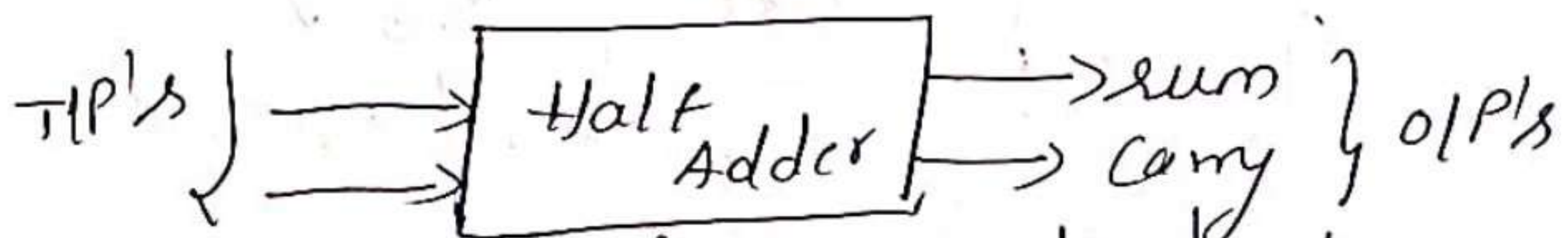


Fig:- Combinational ckt

Truth table:-

(2)

IP's		OP's	
A	B	sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Implementation of half Adder:-

* The circuit of half Adder can be implemented by using K-Map based on its truth table.

K-Map for sum:-

A \ B	$\bar{B}$	B	
	0	1	
$\bar{A}$ 0	0	1	$\rightarrow \bar{A}B$
A 1	1	0	

$y = \bar{A}B + A\bar{B}$
 $y = A \oplus B$

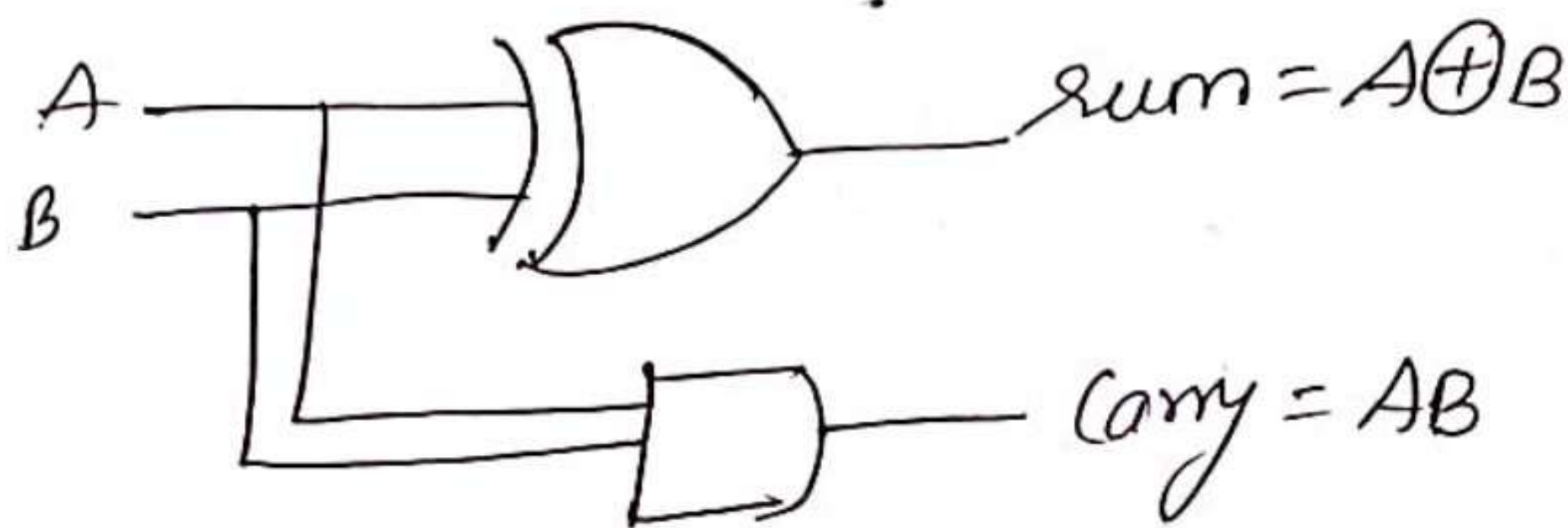
$\bar{A}B$

K-map for Carry:-

A \ B	$\bar{B}$	B	
	0	1	
$\bar{A}$ 0	0	0	$\rightarrow AB$
A 1	0	1	

$y = AB$

Logic diagram:-



Full Adder:-

- * A logic circuit that can add three bits at a time is known as "Full Adder".
- * The function of full adder is capable of adding three bits at a time and produces two o/p's sum and carry.

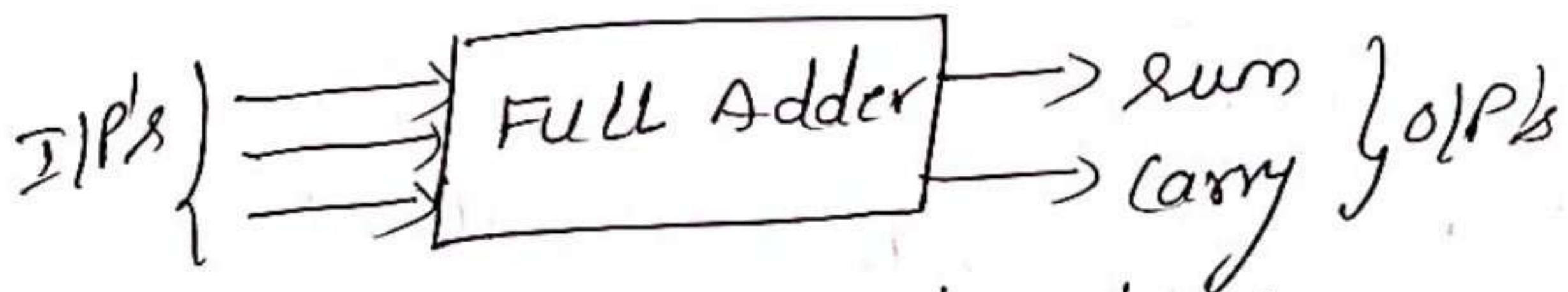


Fig:- Combinational ckt

(3)

Inputs			O/P's	
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	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

Implementation of full Adder:-

The logic circuit of full adder can be implemented by using its truth table. The o/p's sum & carry are mapped in K-map as shown in fig.

K-map for sum:-

		$\overline{A} \overline{B} C$			
		$B \overline{C}$	$\overline{B} \overline{C}$	$B C$	$\overline{B} C$
		00	01	11	10
$\overline{A}$	0		① ₁		① ₃
A	1	① ₄		① ₆	

$\rightarrow \overline{A} B \overline{C}$
 $\downarrow A \overline{B} \overline{C}$ $\rightarrow A B C$

$$\begin{aligned}
 \therefore \text{Sum} &= \overline{A} \overline{B} C + \overline{A} B \overline{C} + A B C + A \overline{B} \overline{C} \\
 &= \overline{C} (A \overline{B} + \overline{A} B) + C (A B + \overline{A} \overline{B}) \\
 &= \overline{C} (A \oplus B) + C (\overline{A \oplus B})
 \end{aligned}$$

Assume $X = A \oplus B$,

$\bar{X} = A \oplus \bar{B}$

$\therefore \text{Sum} = X\bar{C} + C\bar{X}$

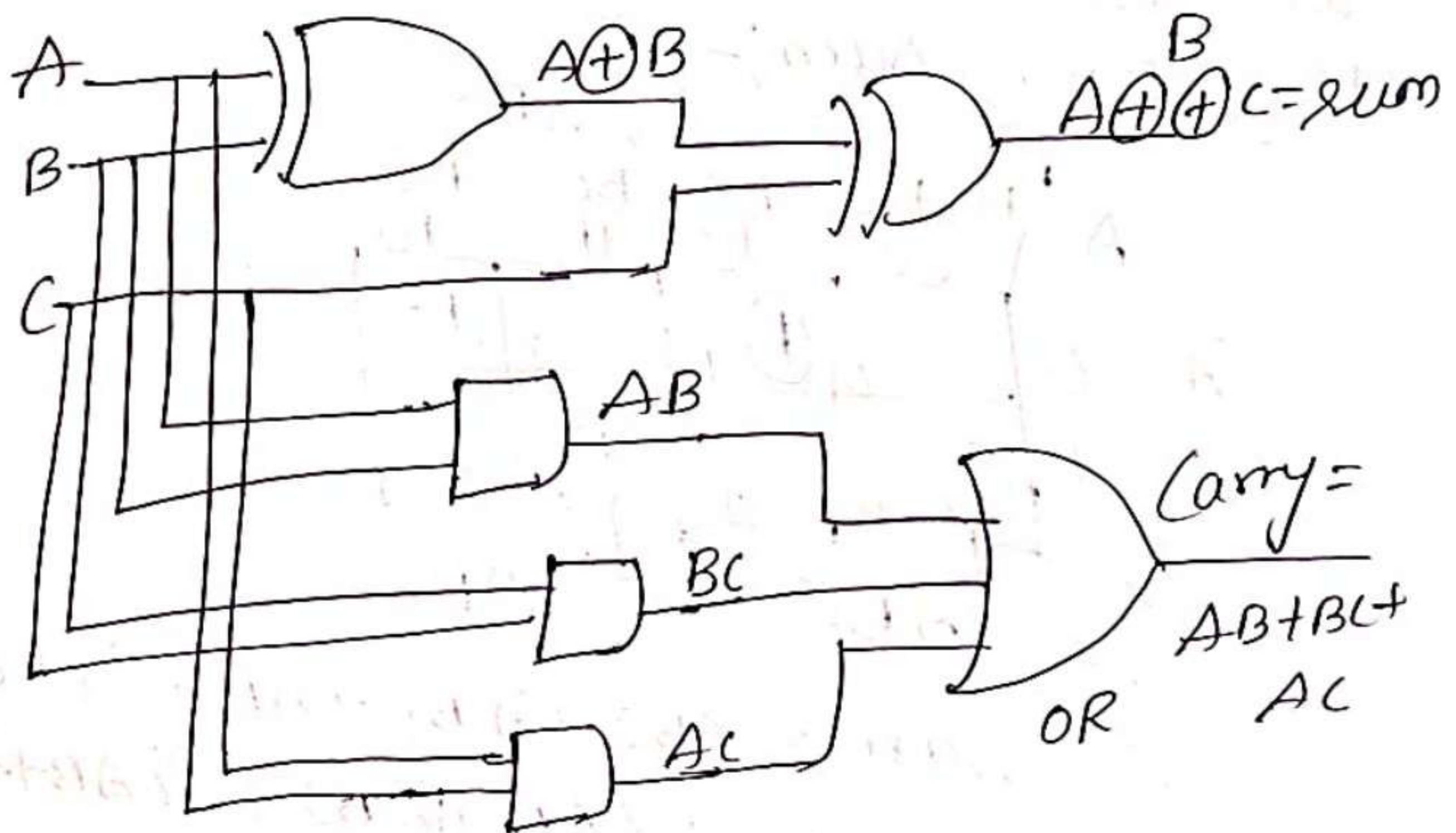
$\text{Sum} = X \oplus C = A \oplus B \oplus C$

K-map for Carry:-

A \ BC	00	01	11	10
	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$ 0	0	1	1	0
A 1	0	1	1	1

$\text{Carry} = AB + BC + AC$

* The logic diagram of Sum and Carry is shown below.



Realization of full Adder by using two half Adders (4)

* Full Adder can be constructed by using two half Adders as shown in fig.

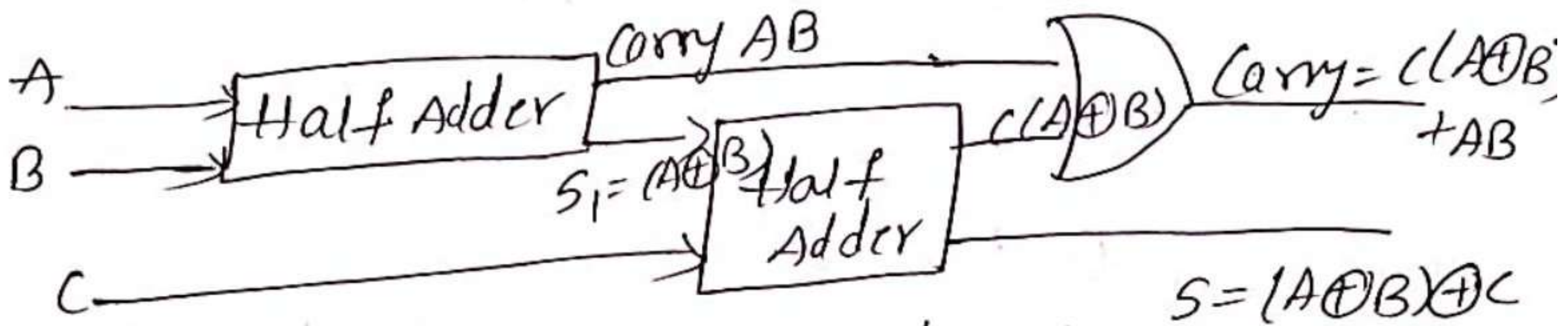
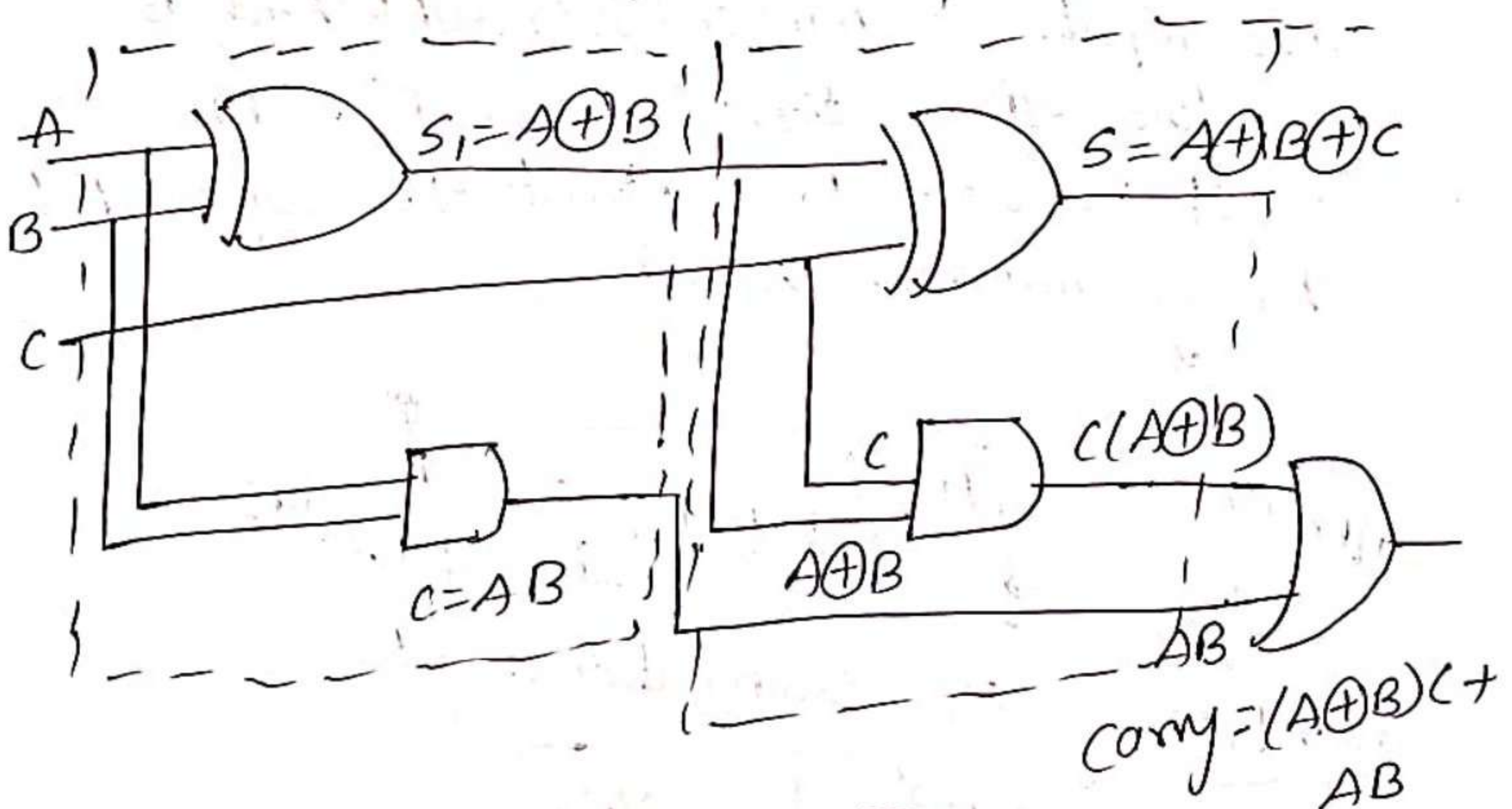


Fig:- full adder constructed from two half adders and OR gate



$$\begin{aligned}
 \therefore \text{Sum} &= \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC \\
 &= C(\bar{A}\bar{B} + AB) + \bar{C}(\bar{A}B + A\bar{B}) \\
 &= C(\overline{A \oplus B}) + \bar{C}(A \oplus B) \\
 &= \bar{X}C + X\bar{C} = X \oplus C
 \end{aligned}$$

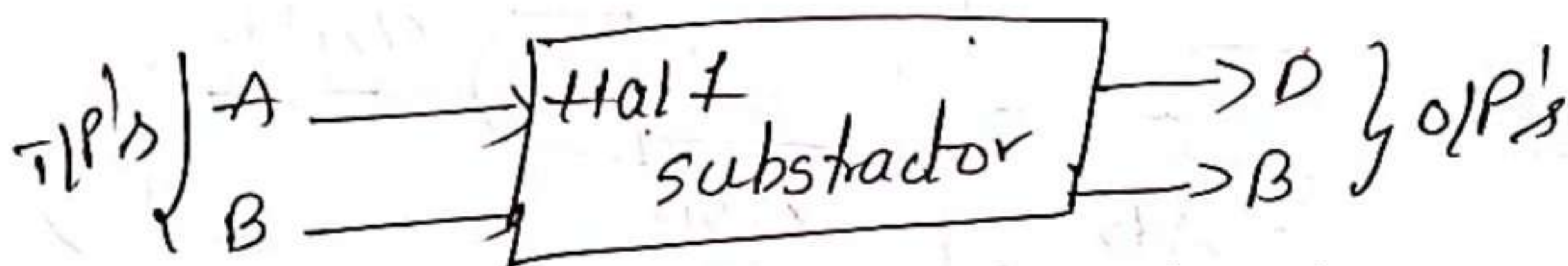
$$\therefore \text{sum} = A \oplus B \oplus C$$

$$\begin{aligned} \Rightarrow \text{Carry} &= \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC \\ &= \overline{A}BC + A\overline{B}C + AB(C + \overline{C}) \\ &= C(\overline{A}B + A\overline{B}) + AB \end{aligned}$$

$$\therefore \text{Carry} = C(A \oplus B) + AB$$

Half subtractor:-

* The half subtractor is also building block for subtracting two binary numbers. It has two inputs (minuend A or subtrahend B) and two o/p's (difference or borrow B). This circuit is used to subtract two single bit binary numbers 'A' or 'B'.



Truth table:- Fig:- Combinational ckt

I/P's		O/P's	
A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K-Map for Diff

A \ B	0	1
	$\bar{B}$	B
$\bar{A}$	0	1
A	1	0

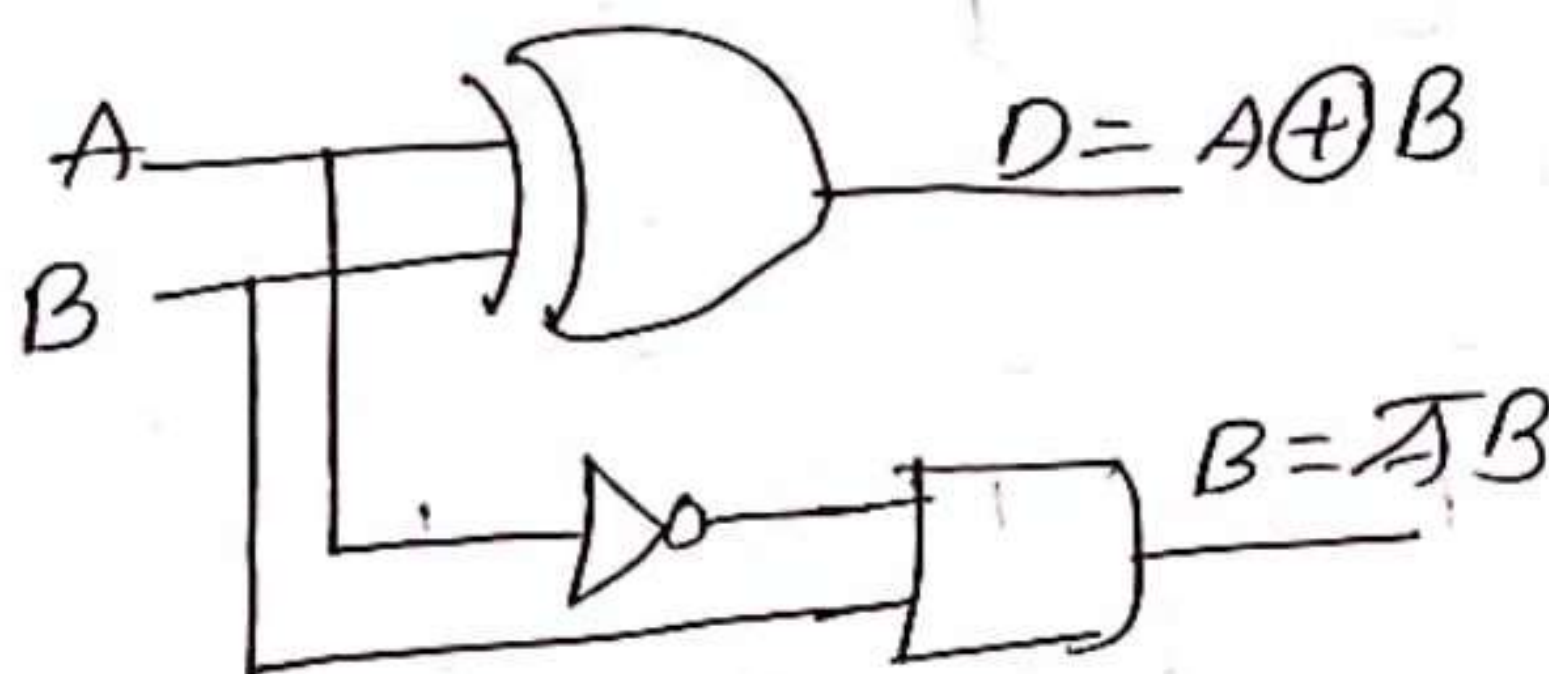
$$D = \bar{A}B + A\bar{B} = A \oplus B$$

K-Map for Diff B ⑤

A \ B	0	1
	$\bar{B}$	B
$\bar{A}$	0	1
A	1	0

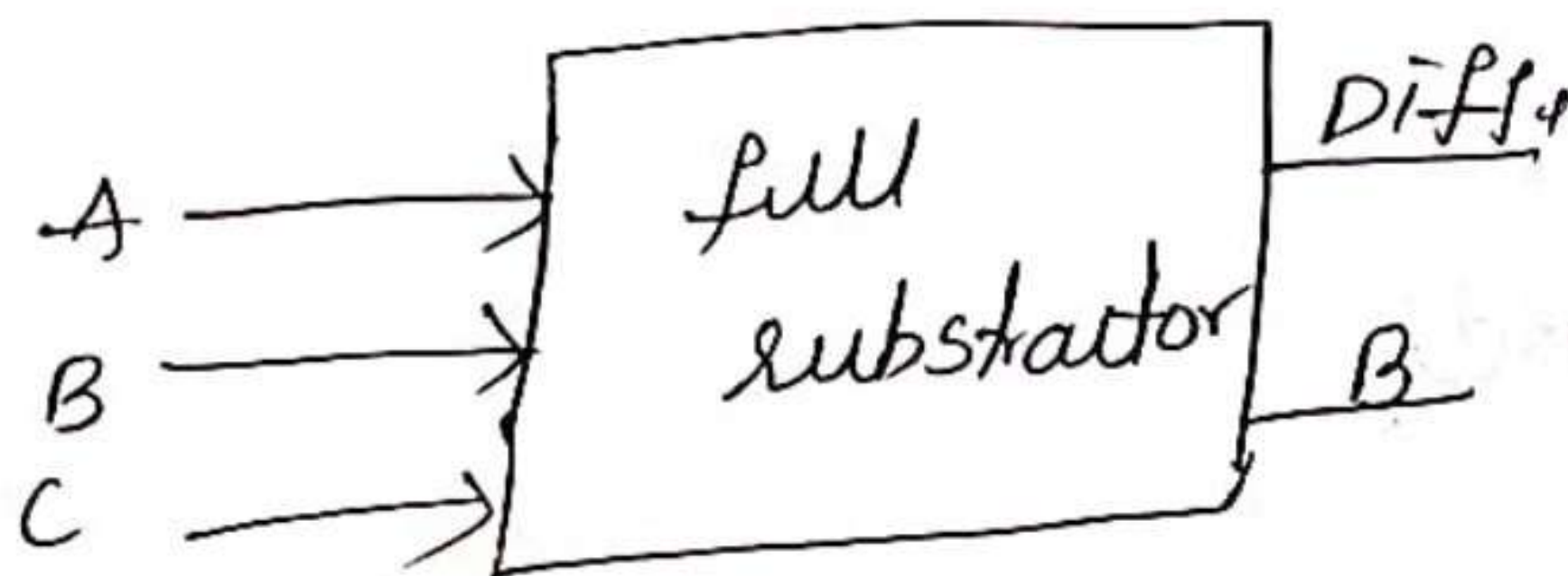
$$B = \bar{A}B$$

logic diagram:-



Full Subtractor:-

* It is a combinational ckt that performs that arithmetic subtraction of three binary numbers. It consists of 3 I/P's (A, B and C) and two o/p's (diff & Borrow)



Truth-table

Inputs			outputs	
A	B	C	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-Map for D

A \ BC	00	01	11	10
0		1	1	1
1	1	1	1	0

$$D = \overline{A}\overline{B}C + A\overline{B}C + \overline{A}BC + ABC$$

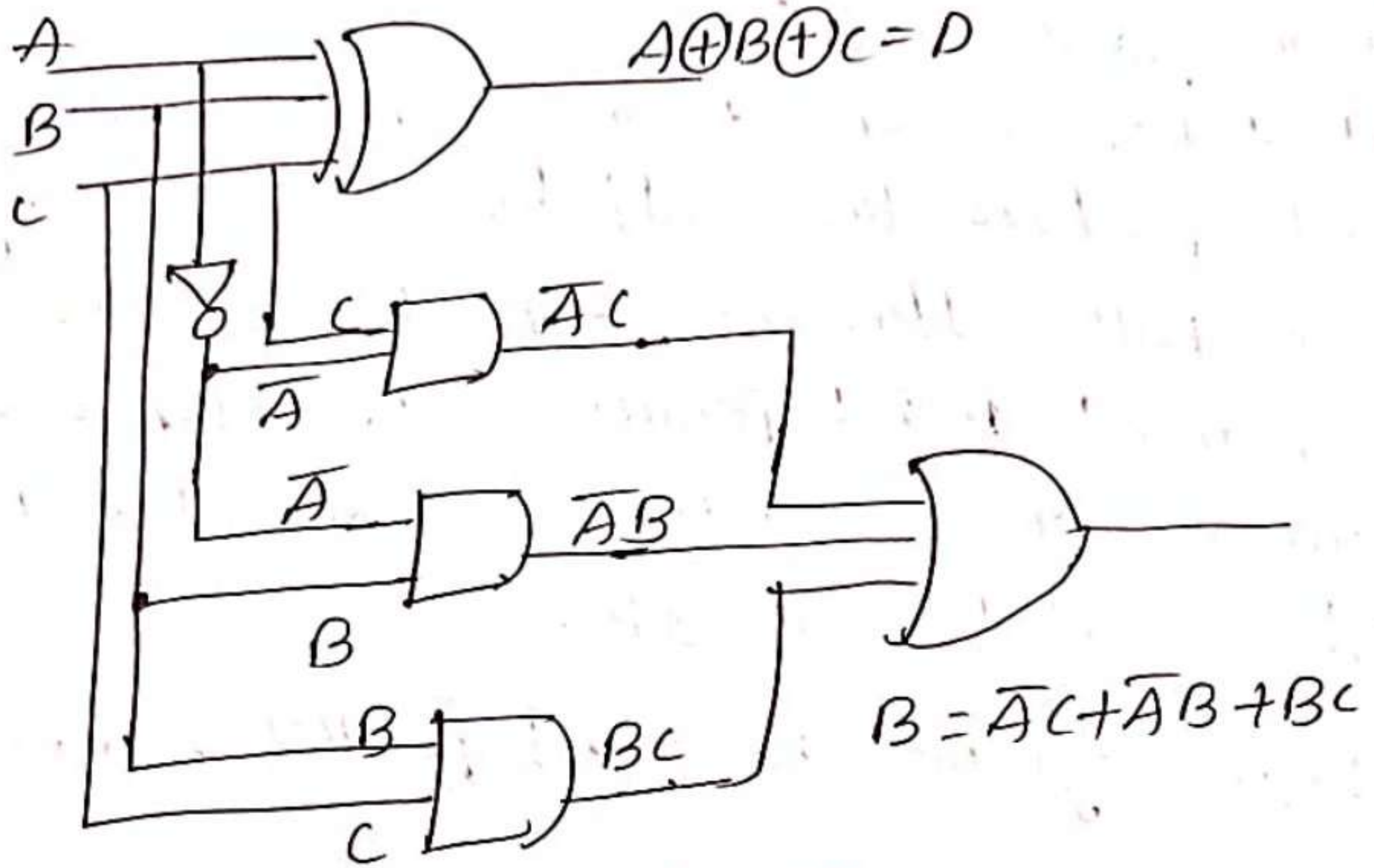
$$\therefore D = A \oplus B \oplus C$$

K-Map for B

A \ BC	00	01	11	10
0		1	1	1
1	1	1	1	0

$$B = \overline{A}C + \overline{A}B + BC$$

⑥



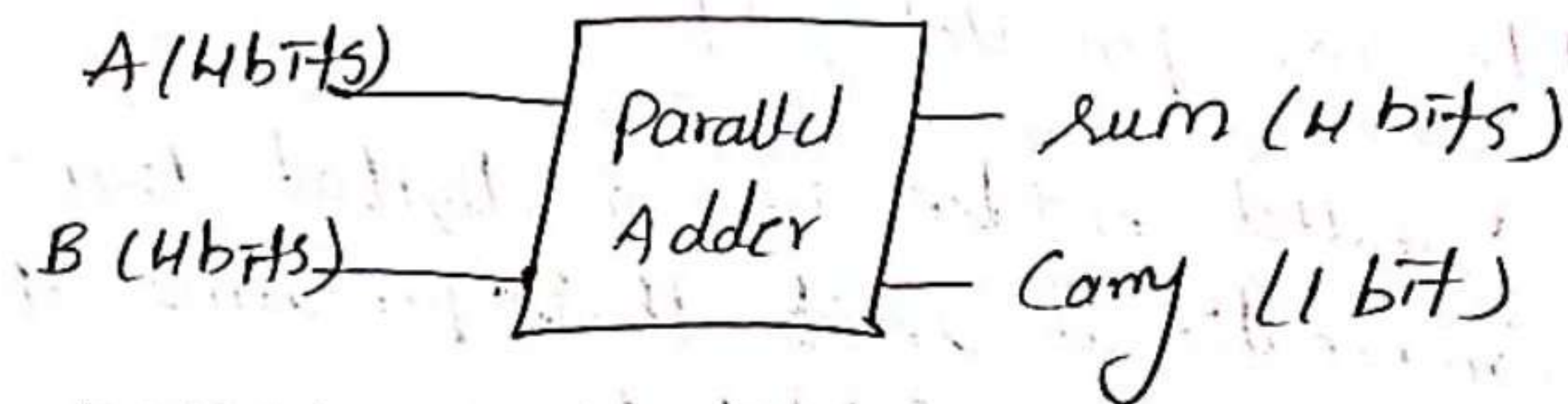
* 2M. Binary Adder (or) Parallel Adder:-

Parallel Adder is a digital logic ckt (or) combinational circuit that performs the arithmetic sum of two n -bit binary numbers and produces two outputs sum and carry. It takes two binary numbers A (having n bits), B (having n bits) as inputs and produces two o/p's sum (having n bits) and carry (1 bit).

4-bit Binary Adder (or) Parallel Adder by using Full Adder:-

* 4-bit Parallel Adder is a digital circuit that performs addition of two 4-bit binary numbers.

- * It adds two binary numbers simultaneously (or) in parallel fashion.
- * It takes A & B (having n bits) as inputs and produce two outputs sum & carry.
- * Four Full Adders are connected in cascade to implement n -bit parallel Adder. Where output carry of one full Adder is connected to input carry of next full Adder.
- * Logic symbol of n -bit parallel Adder is



- * $A = A_3 A_2 A_1 A_0$ & $B = B_3 B_2 B_1 B_0$ are two 4-bit binary numbers where A_0, B_0 are LSB bits and A_3, B_3 are MSB bits.

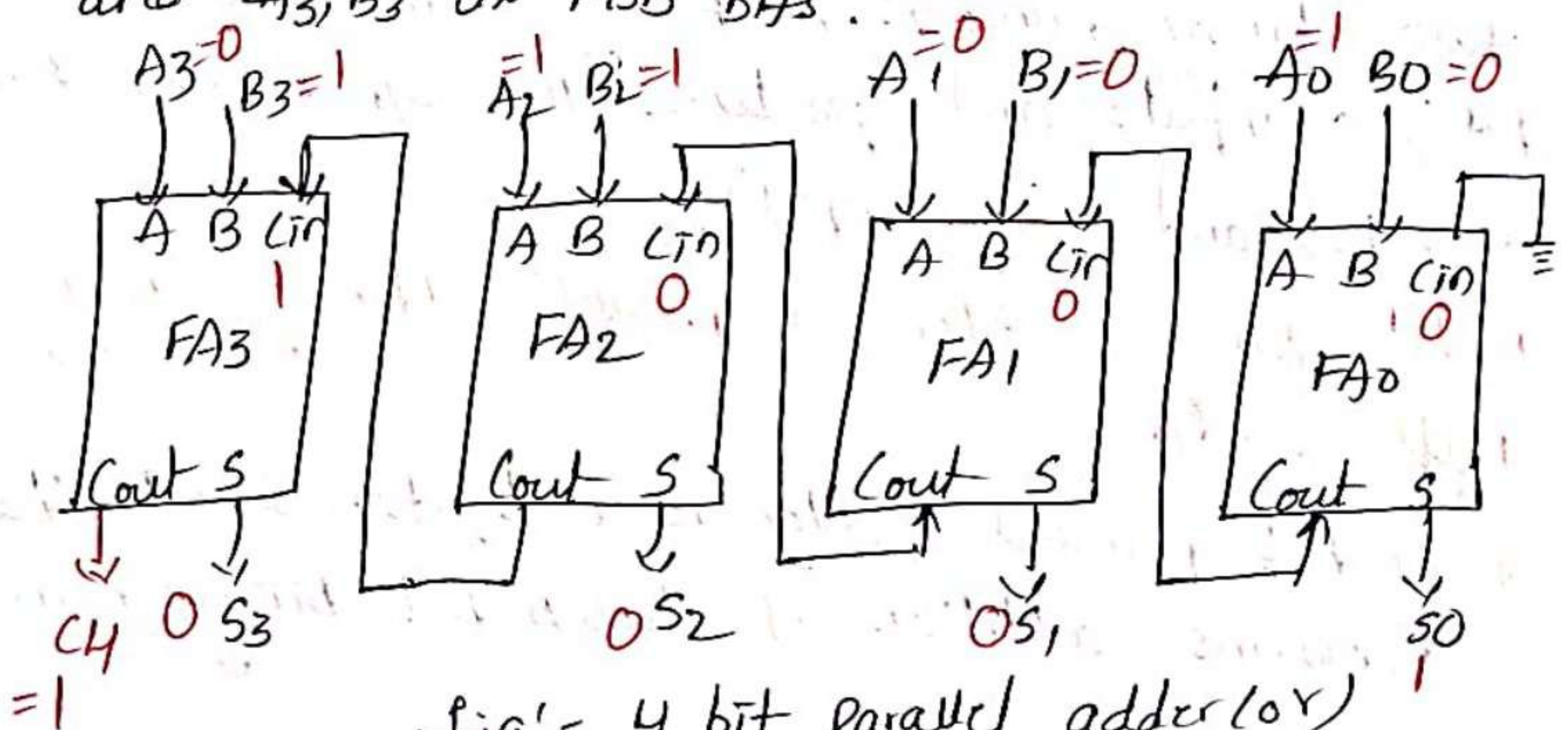


Fig:- 4 bit parallel adder (or) Binary Adder.

* Let us take example

$$A = \overset{\text{MSB}}{0} \overset{\text{LSB}}{1} 0 1 = A_3 A_2 A_1 A_0$$

$$B = 1 1 0 0 = B_3 B_2 B_1 B_0$$

$$\textcircled{1} \quad \begin{array}{r} 1 \\ 0001 \end{array}$$

$$\therefore \text{sum} = 1 (0001)$$

* The Resultant Value produced in n bits =
there is no flow.

* The Resultant Value produced in $(n+1)$ bits =
over flow.

N-Bit Binary Adder / Subtractor Unit:-

* Parallel adder in general performs addition, but it can be customized to perform subtraction also and such customized unit is called as 2's Complement parallel adder-subtractor.

* It adds (or) subtracts two binary numbers simultaneously (or) in parallel fashion.

* It takes A & B (having n bits each) as inputs and produces two o/p's sum and carry.

* circuit diagram of n -bit adder/subtractor as shown in below.

⑧

$\Rightarrow A + 2's \text{ Complement of } B$

$\Rightarrow (A - B)$

$\left\{ \begin{aligned} [1 + \bar{B}] &= 2's \\ &\text{Complement} \end{aligned} \right\}$

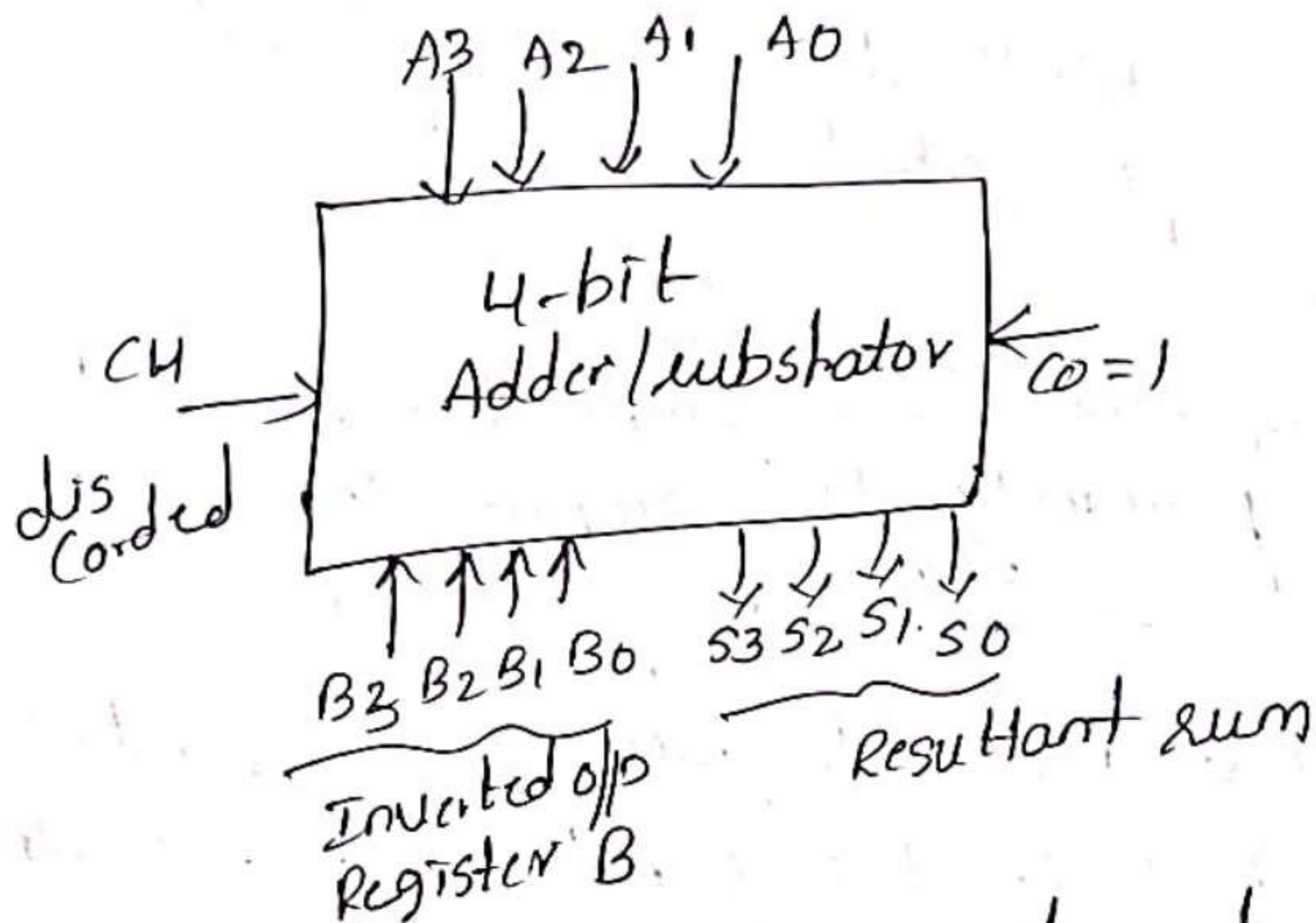


fig:- 2's Complement subtraction.

* 2nd digital Comparator (or) Magnitude Comparator:-
def:-

A Magnitude Comparator is a combinational unit that compares the magnitude of two numbers (A & B) and generates one of the following o/p's: $A = B$, $A < B$ & $A > B$.
The block diagram of single bit Magnitude Comparator is shown in fig.

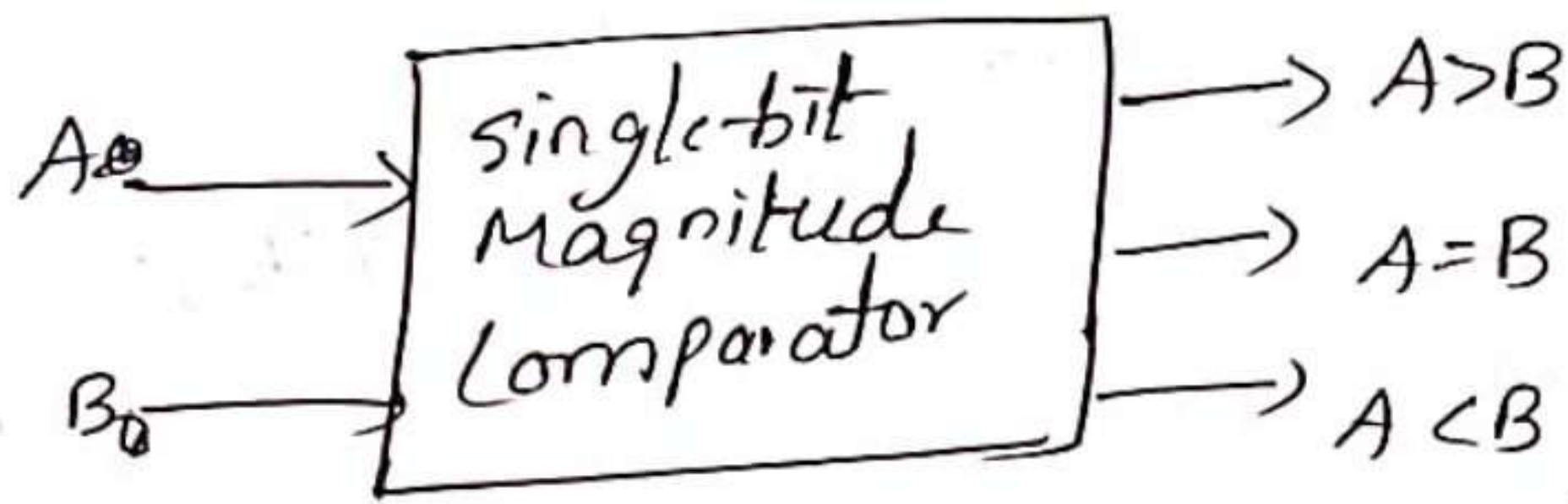


fig:- Block diagram of single bit Magnitude Comparator.

2-bit Comparator:-

- * A Comparator is a combinational circuit designed primarily to compare the relative magnitude of two binary numbers A & B.
- * functional Block diagram of Comparator which receives two 2-bit numbers as I/P and produce 3 o/p's

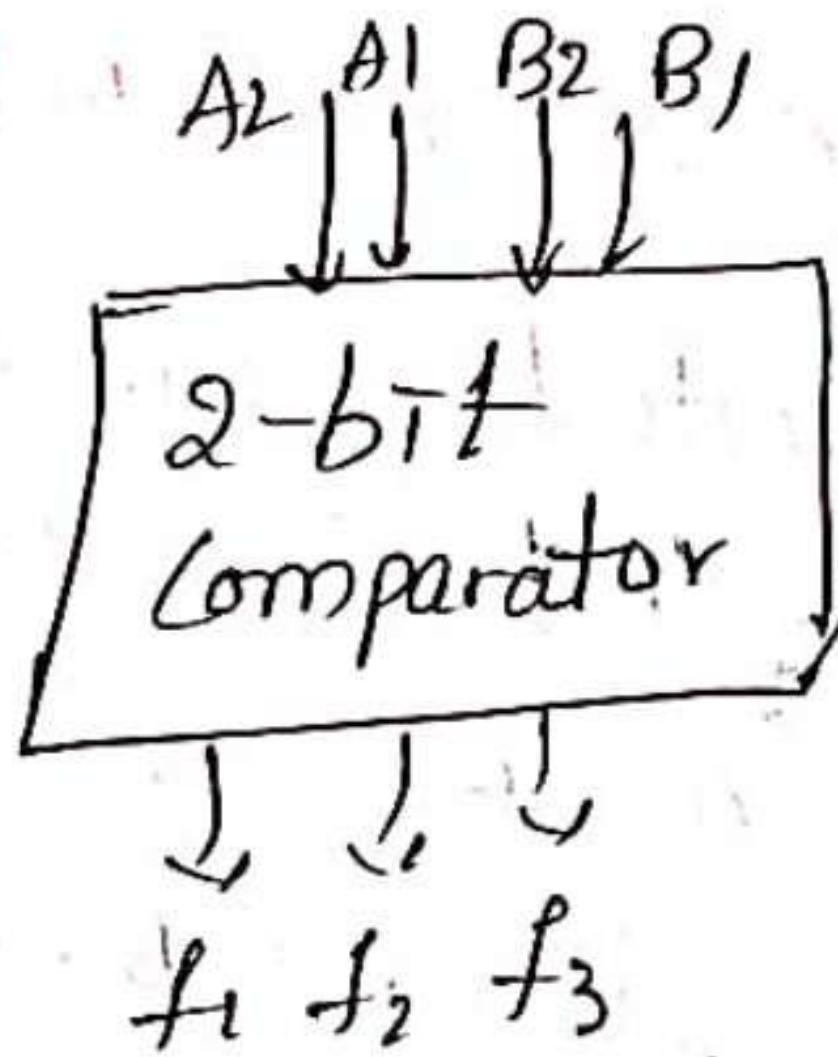


fig:- 2-bit Comparator

Truth table :-

(9)

Inputs

outputs

A ₂	A ₁	B ₂	B ₁	f ₁ A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

* The K-Maps for the three opp's and corresponding simplified expression have been shown in fig

K-Map for $A > B$

$A_2 A_1$	$B_2 B_1$	$\overline{B_2} \overline{B_1}$	$\overline{B_2} B_1$	$B_2 \overline{B_1}$	$B_2 B_1$
	00	01	11	10	
$\overline{A_2} \overline{A_1}$ 00					
$\overline{A_2} A_1$ 01	1				
$A_2 \overline{A_1}$ 11	1	1		1	
$A_2 A_1$ 10	1	1			

$$f_1 = A_2 \overline{B_2} + A_1 \overline{B_1} \overline{B_2} + A_1 A_2 \overline{B_1}$$

$A < B$

$A_2 A_1$	$B_2 \overline{B_1}$	$\overline{B_2} \overline{B_1}$	$\overline{B_2} B_1$	$B_2 B_1$	$B_2 \overline{B_1}$
	00	01	11	10	
$\overline{A_2} \overline{A_1}$ 00			1	1	1
$\overline{A_2} A_1$ 01			1	1	
$A_2 \overline{A_1}$ 11					
$A_2 A_1$ 10				1	

$$f_2 = \overline{A_1} \overline{A_2} B_1 + \overline{A_2} B_2 + \overline{A_1} \overline{B_1} B_2$$

$A = B$

$A_2 A_1$	$B_2 B_1$	$\overline{B_2} \overline{B_1}$	$\overline{B_2} B_1$	$B_2 \overline{B_1}$	$B_2 B_1$
	00	01	11	10	
$\overline{A_2} \overline{A_1}$ 00	1				
$\overline{A_2} A_1$ 01		1			
$A_2 \overline{A_1}$ 11			1		
$A_2 A_1$ 10				1	

$$\begin{aligned}
 f_3 &= \overline{A_1} \overline{A_2} \overline{B_1} \overline{B_2} + A_1 \overline{A_2} B_1 \overline{B_2} + A_1 A_2 B_1 B_2 + \overline{A_1} A_2 \overline{B_1} B_2 \\
 &= \overline{A_1} \overline{B_1} (\overline{A_2} \overline{B_2} + A_2 B_2) + A_1 B_1 (\overline{A_2} \overline{B_2} + A_2 B_2) \\
 &= (\overline{A_1} \overline{B_1} + A_1 B_1) (\overline{A_2} \overline{B_2} + A_2 B_2) \\
 &= (A_1 \odot B_1) (A_2 \odot B_2)
 \end{aligned}$$

10

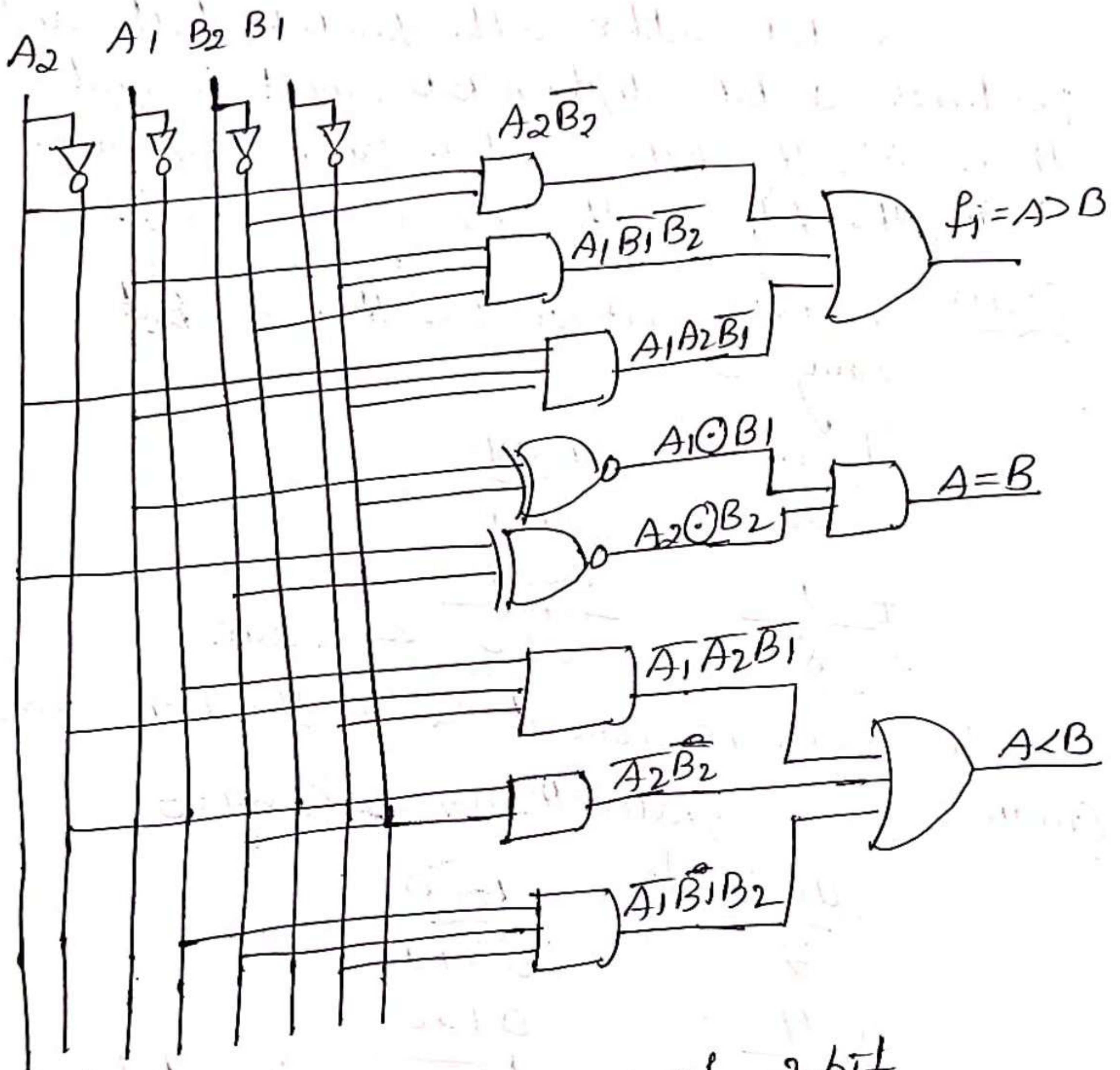


Fig. Logic diagram of 2-bit Comparator

BCD Adder:-

A BCD adder adds two BCD digits and produces a BCD digit. A BCD cannot be greater than '9'. The BCD addition can be explained with the help of the following 3 cases.

Case i:- sum equal (or) less than 9, but
Carry = 0

<u>decimal</u>	<u>BCD</u>
7	0111
+ 1	0001
<u>8</u>	<u>1000</u> ← sum

* The sum is correct in the BCD form.

Case ii:- sum greater than 9, Carry = 0

<u>decimal</u>	<u>BCD</u>
7	0111
+ 4	0100
<u>11</u>	<u>1011</u> → Invalid BCD Number

* Hence, to correct the answer, we add six (0110) to the Invalid BCD Answer.

	1011
	0110
	<u>1001</u>
0001	<u>0001</u> ← correct Answer
<u>1</u>	<u>1</u>

Caution:- Sum less than (or) Equal to 9 but (11)

Carry = 1

Decimal

$$\begin{array}{r} 9 \\ + 8 \\ \hline 17 \end{array}$$

Bcd

1001

0000

0001 → Invalid Bcd Number

Carry ←

* The result (0001 0001) = (11)₁₀ not correct. To correct the wrong result, we have to add six (0110) as shown in below.

$$\begin{array}{r} 0001 \quad 0001 \\ 0000 \quad 0110 \\ \hline 0001 \quad 0111 \end{array} \leftarrow \text{Correct Bcd Answer.}$$

1 7

Bcd Adder [Binary Coded decimal]:-

* Each decimal digit is represented by a 4-bit binary number.

Eg:- 5 → 0101

9 → 1001

10 → 0001 0000

185 → 0001 1000 0101

defn:- Bcd Adder is a combinational logic circuit that adds two Bcd numbers in parallel and produces a valid Bcd number as a output.

* BCD Adder consists of a

i) 4-bit binary adder for initial addition.

ii) logic circuit to detect whether the sum is greater than '9' $\rightarrow (1001)$

iii) 4-bit adder to add (0110) with the sum if the sum > 9 (or) carry is '1'.

Design principle:-

The logic circuit to detect the (sum > 9) can be determined by simplifying the Boolean expression of 4-bit adder o/p.

Truth table:-

sum o/p's				O/P
S ₃	S ₂	S ₁	S ₀	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1

1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(12)

K-map for o/p: [X]-

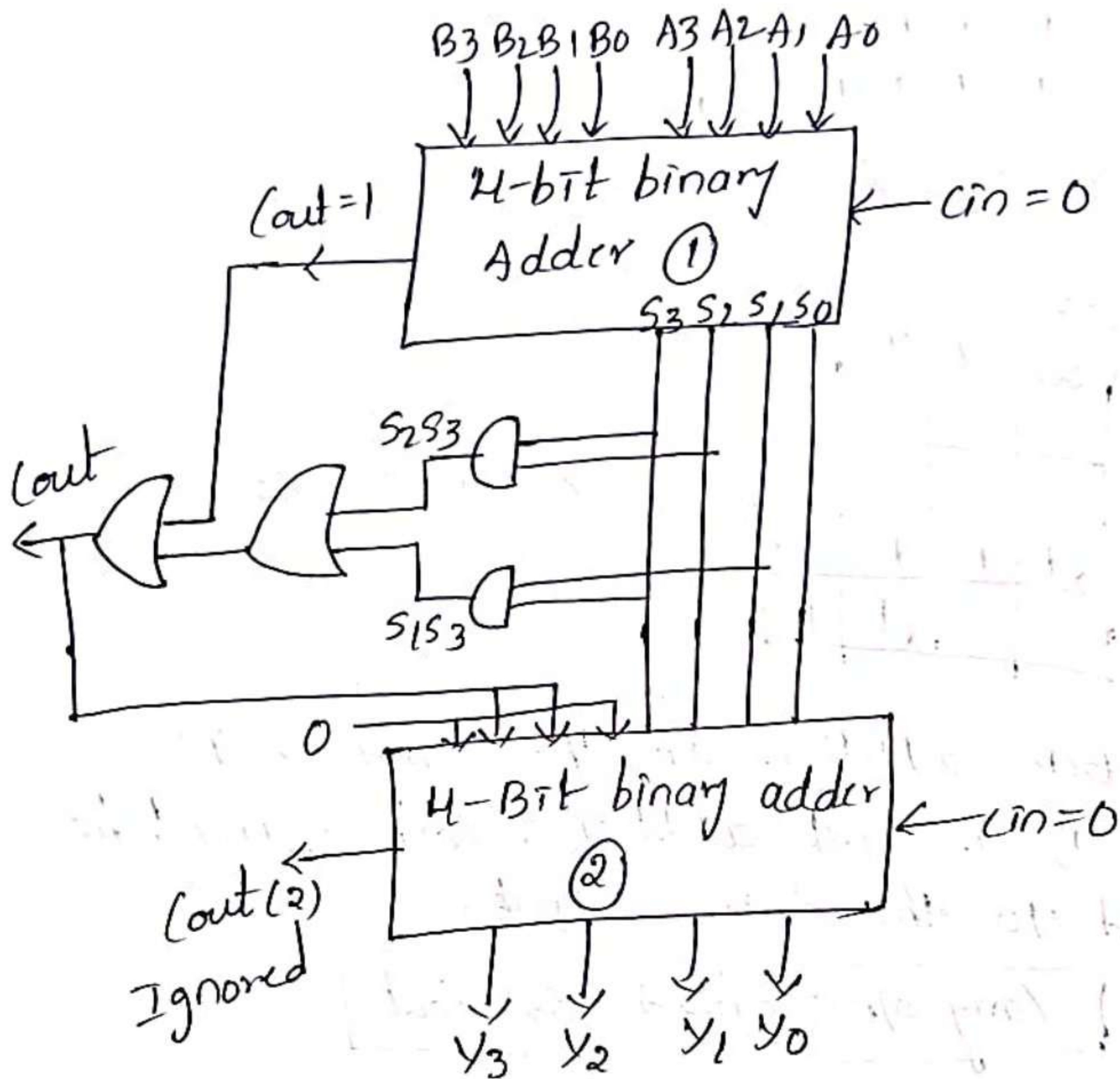
S ₂ S ₃	00	01	11	10
00				
01				
11	1	1	1	1
10			1	1

$$X = S_1S_3 + S_2S_3$$

* In BCD addition, If the carry is 1 (C_{out}=1) in first addition, (0110)₂ must be added to the sum output.

$$\boxed{\text{Carry o/p} = S_1S_3 + S_2S_3 + C_{out}}$$

Logic diagram:-



Applications:-

- * It is widely used in computers and the calculators.
- * Used in Arithmetic logic unit
- * It accepts the Binary coded form of decimal numbers.

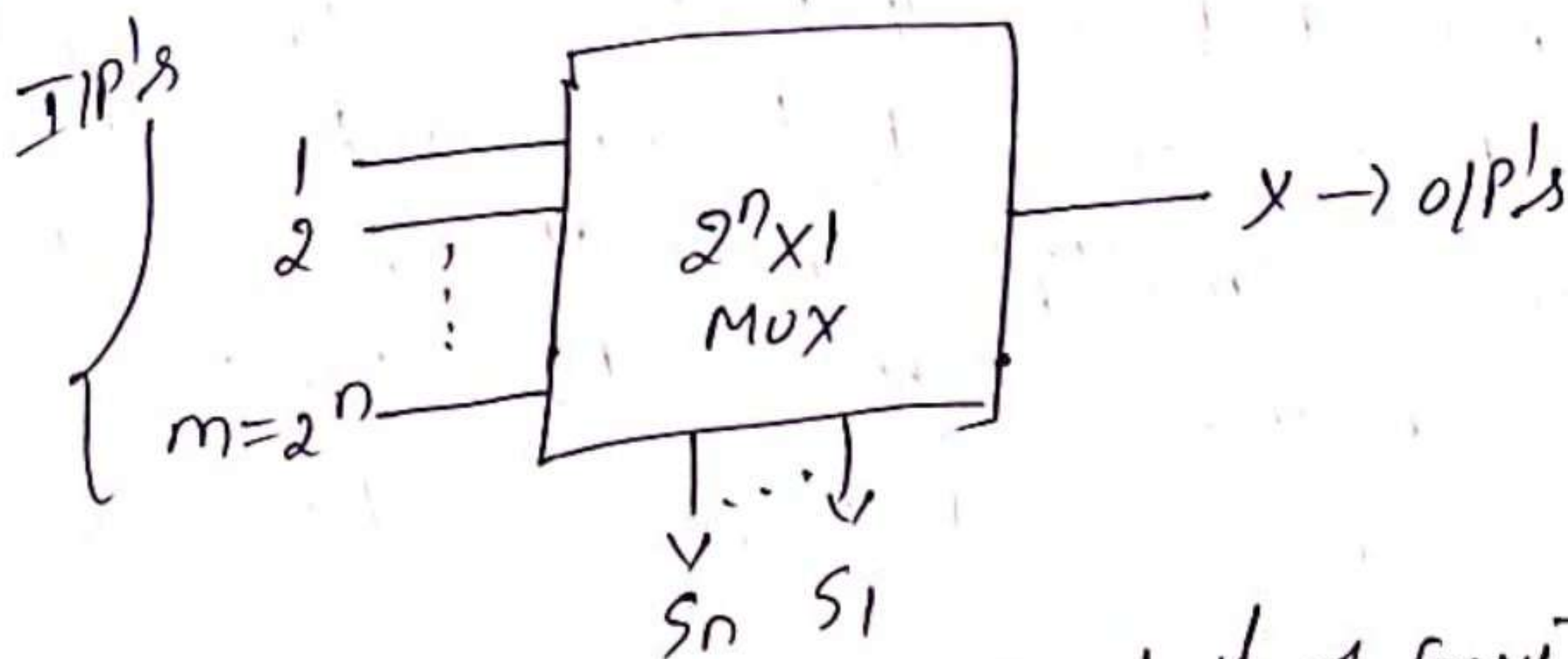


Fig.:- logic symbol of $[m \times 1]$ MUX

Classification of multiplexers:-

1. Multiplexers may be classified as under:-

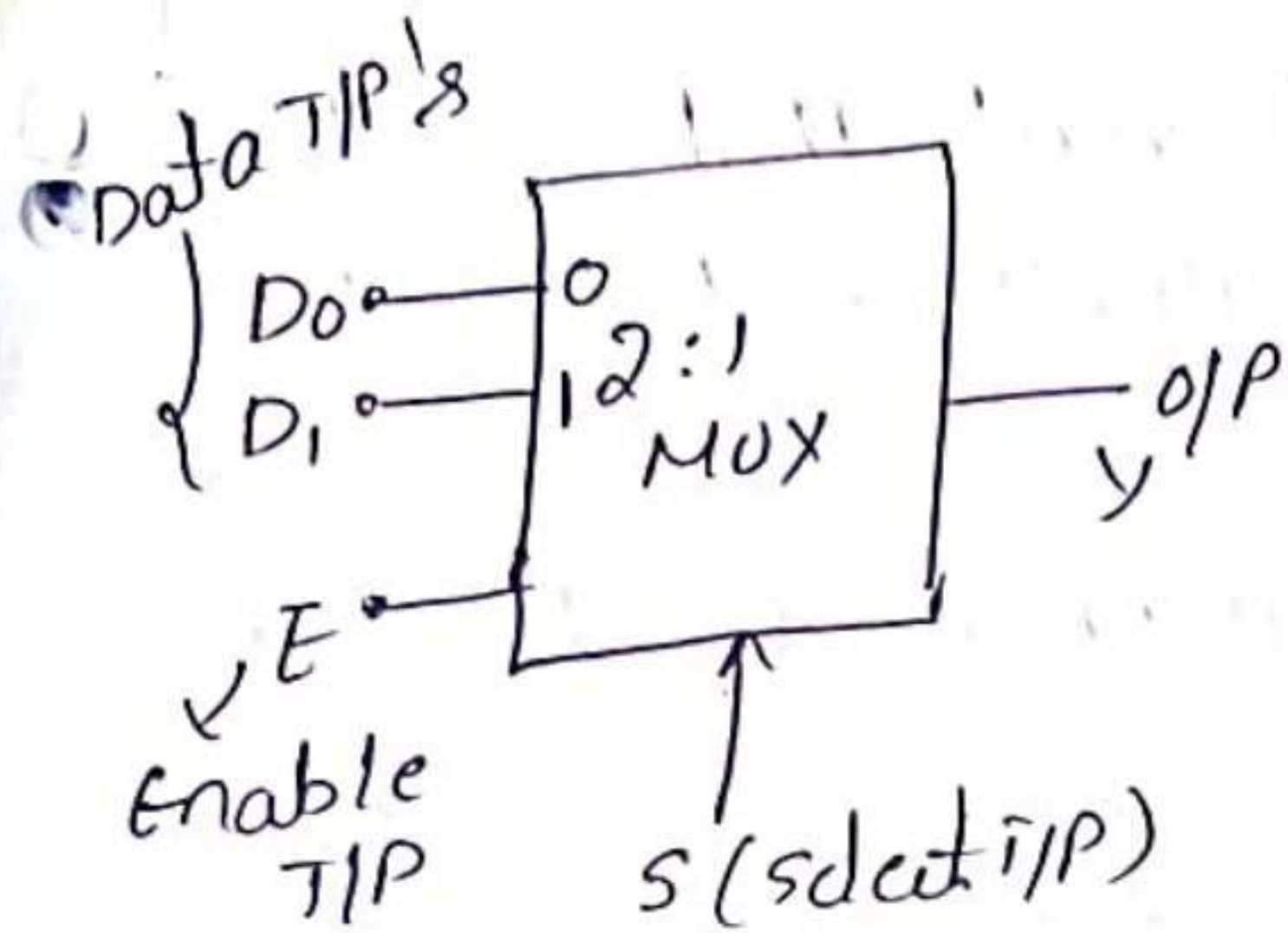
- 2:1 Multiplexer
- 4:1 Multiplexer
- 8:1 Multiplexer
- 16:1 Multiplexer

2:1 [2x1] Multiplexer:-

* The block diagram of 2x1 Multiplexer has been shown in fig. It has 2 data inputs D_0 & D_1 , one select input S , an enable I/P & one output.

* The truth table of this MUX has been shown in fig.

(2)



a) Block diagram.

E	select T/P (S)	O/P Y
0	X	0
1	0	D ₀ → $\overline{SD_0}E$
1	1	D ₁ → SD_1E

b) Truth table

Realization of a 2:1 MUX using gates:-

Enable E	selects (S)	D ₁	D ₀	O/P = Y
0	X	X	X	0
1	0	X	0	0
1	0	X	1	1 ← $\overline{SD_0}$
1	1	0	X	0
1	1	1	X	1 ← SD_1

$$Y = [\overline{SD_0} + SD_1] E$$

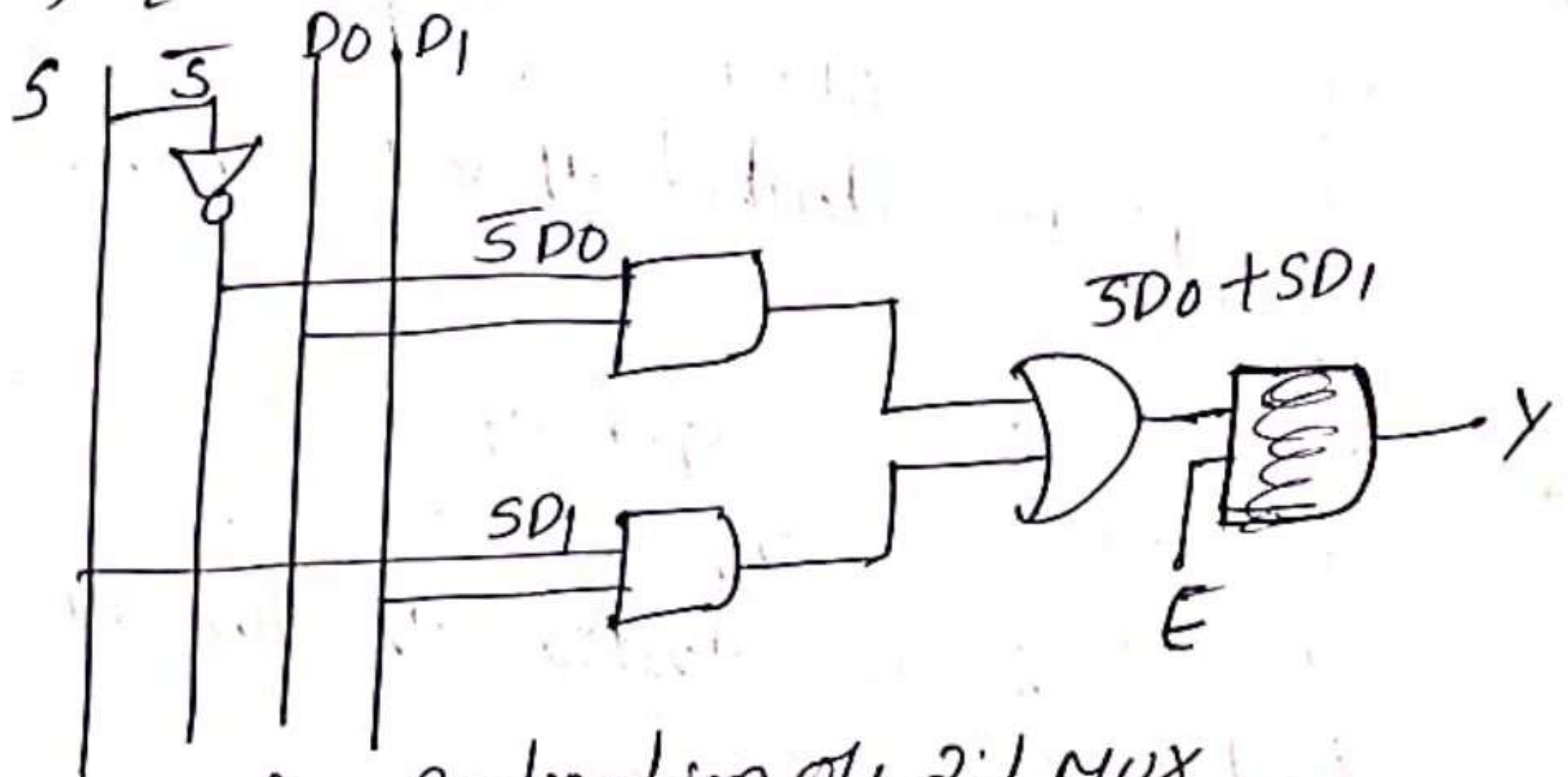
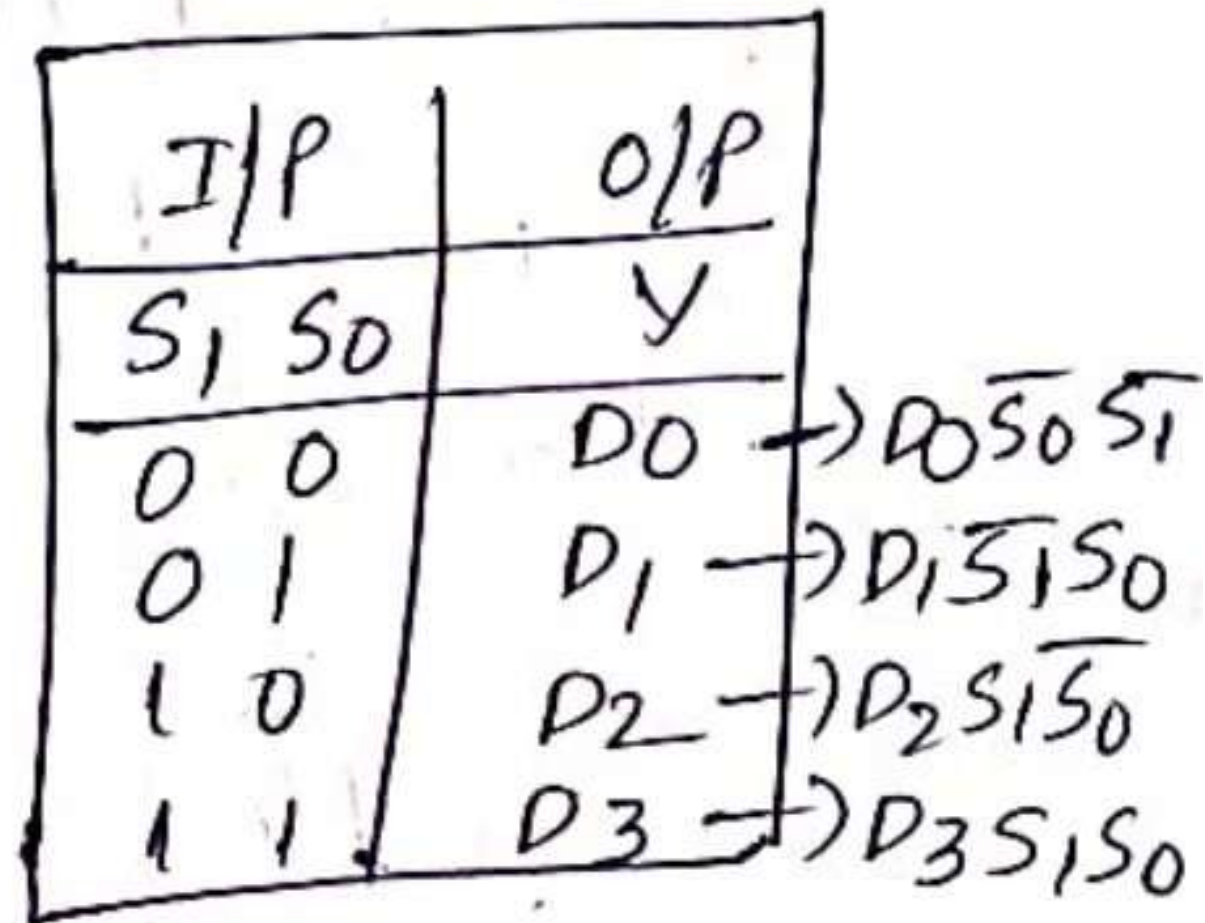


fig. Realization of 2:1 MUX

* A $N \times 1$ MUX is a digital unit that sends data from one of N I/P lines to single O/P line.

* A $N \times 1$ MUX is a digital unit that sends data from one of N I/P lines to single O/P line.

four data I/P



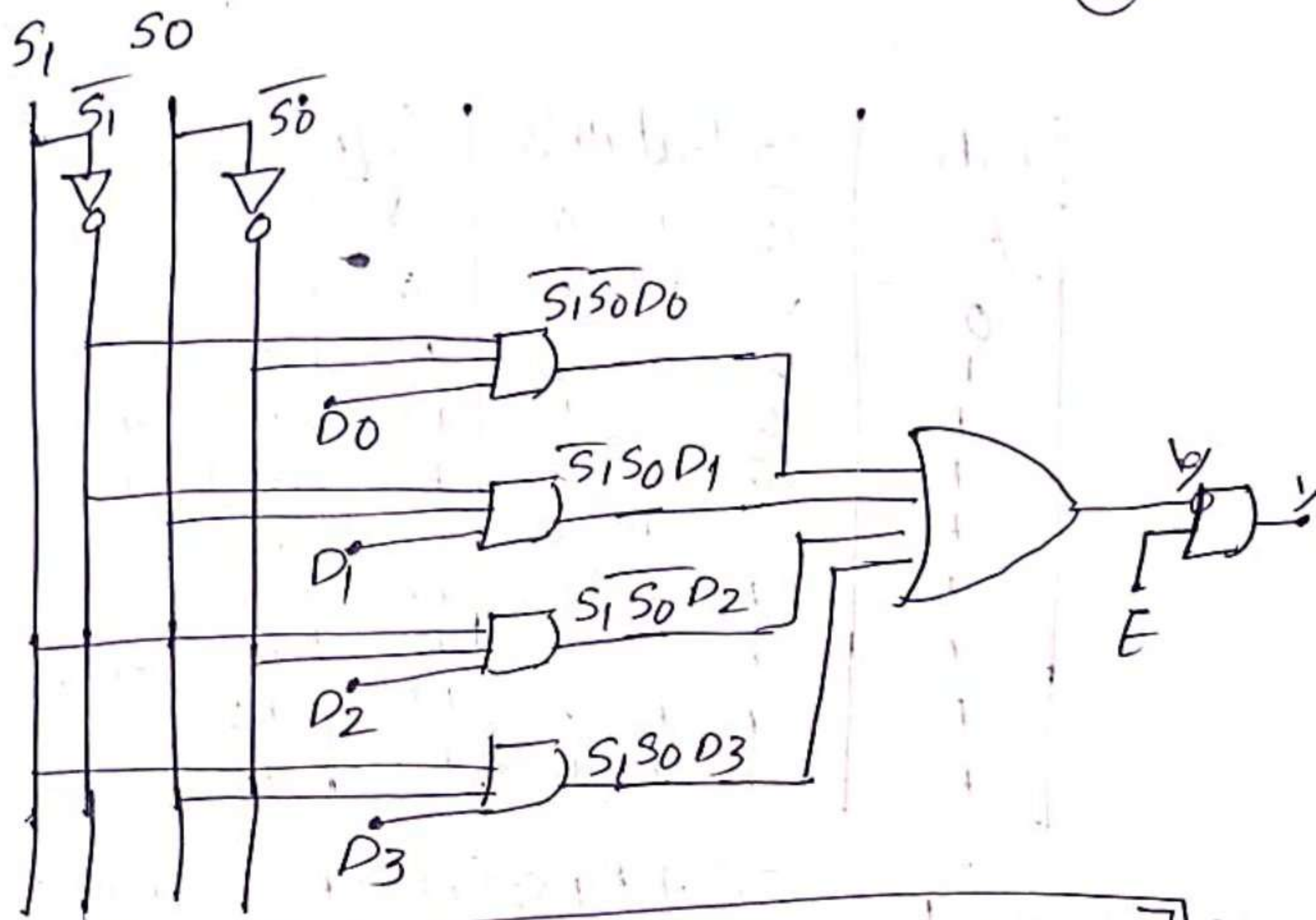
* When dilution lines $S_1=0, S_0=0$, then decimal equivalent of $00=0$, therefore, TLP line numbered as 0 is selected, thereby TLP DO is send to O/P y.

* " " " $S_1 = 0$, " $S_0 = 1$, " " "

" " " $O_1 = 1$, " " " " "

" " IS selected, then by TLP $\bar{D}_1$ is send to
o/p y' .

[illegible]

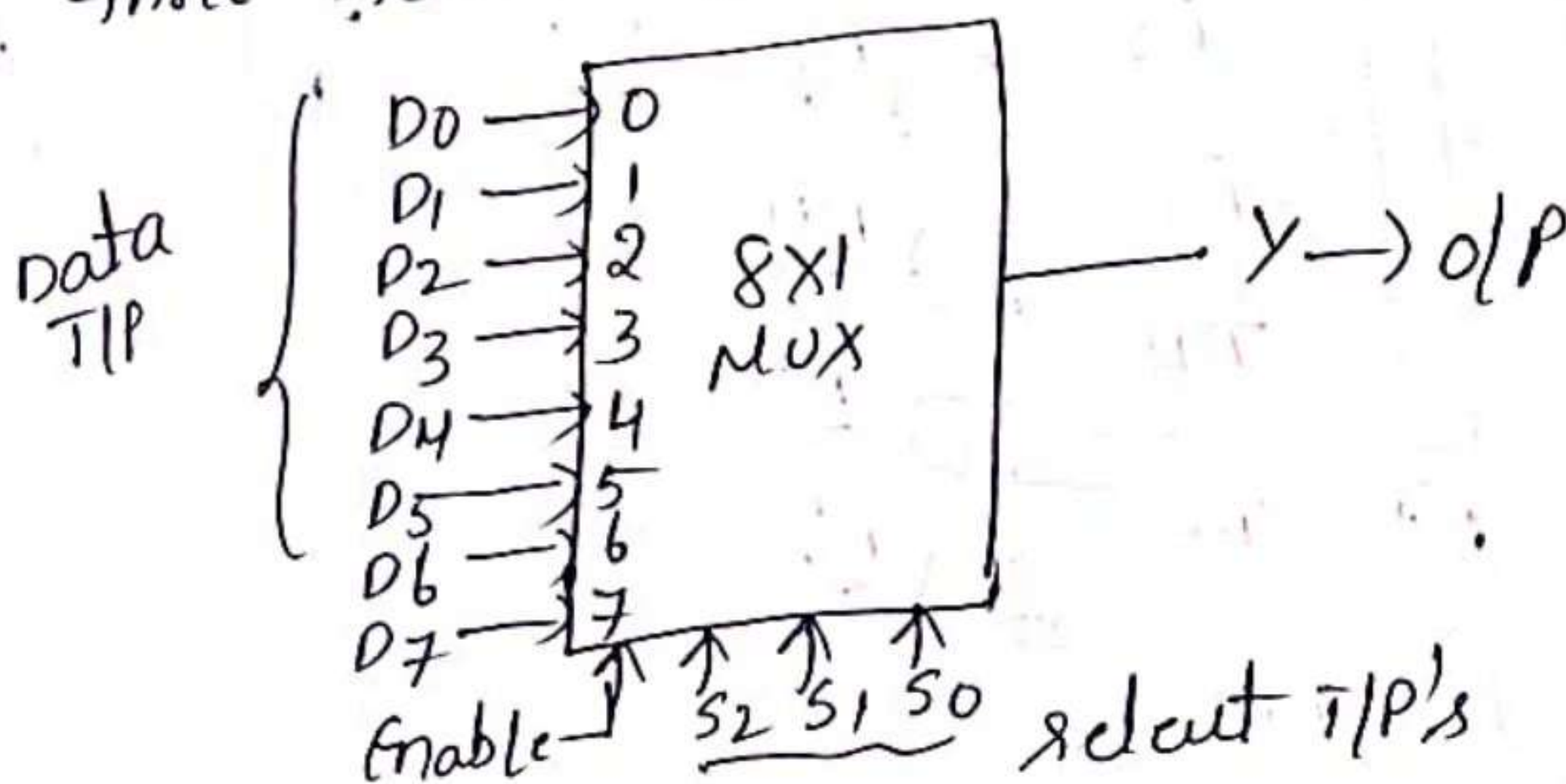


$$Y = [S_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3] E$$

8:1 (or) 8X1 MUX:-

The block diagram of an 8X1 MUX has been shown in fig and its truth table shown in fig.

* It has eight data T/P's, one Enable T/P, three select inputs & one O/P.

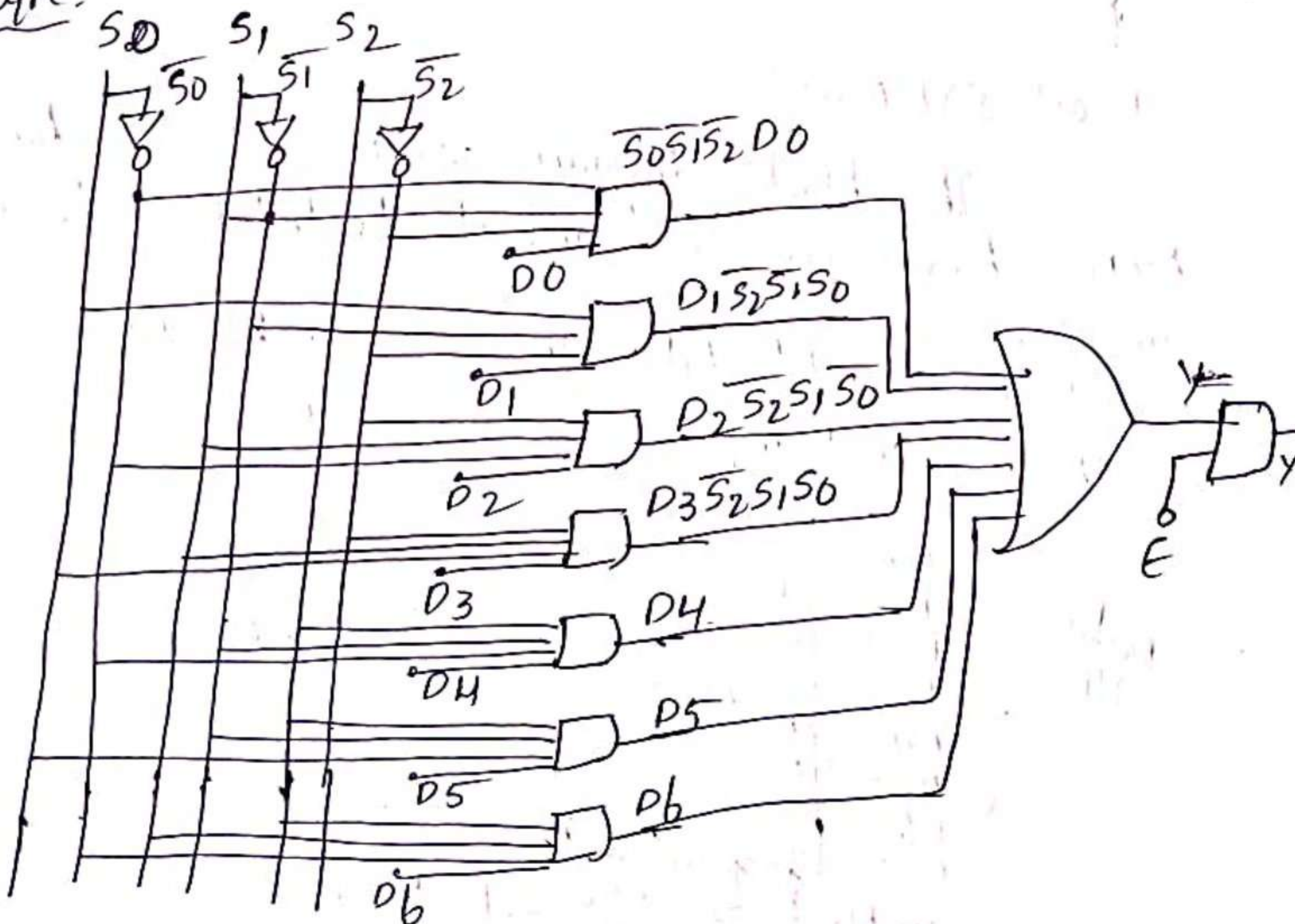


Truth table:-

Enable	Select i/p's			O/P
E	S ₂	S ₁	S ₀	Y
0	X	X	X	0
1	0	0	0	D ₀ → $D_0 \bar{S}_0 \bar{S}_1 \bar{S}_2$
1	0	0	1	D ₁ → $D_1 \bar{S}_0 \bar{S}_1 S_0$
1	0	1	0	D ₂ → $D_2 \bar{S}_2 S_1 \bar{S}_0$
1	0	1	1	D ₃ → $D_3 \bar{S}_2 S_1 S_0$
1	1	0	0	D ₄ → $D_4 S_2 \bar{S}_1 \bar{S}_0$
1	1	0	1	D ₅ → $D_5 S_2 \bar{S}_1 S_0$
1	1	1	0	D ₆ → $D_6 S_2 S_1 \bar{S}_0$
1	1	1	1	D ₇ → $D_7 S_2 S_1 S_0$

$$Y = [D_0 \bar{S}_0 \bar{S}_1 \bar{S}_2 + D_1 \bar{S}_0 \bar{S}_1 S_0 + D_2 \bar{S}_2 S_1 \bar{S}_0 + D_3 \bar{S}_2 S_1 S_0 + D_4 S_2 \bar{S}_1 \bar{S}_0 + D_5 S_2 \bar{S}_1 S_0 + D_6 S_2 S_1 \bar{S}_0 + D_7 S_2 S_1 S_0] E$$

Logic:-



16:1 MUX -

The block-diagram of 16X1 MUX has been shown in fig. It has 16 data I/P's, 4 select lines, one enable I/P, & one O/P.

* The Block diagram & Truth table of 16X1 MUX as shown in fig.

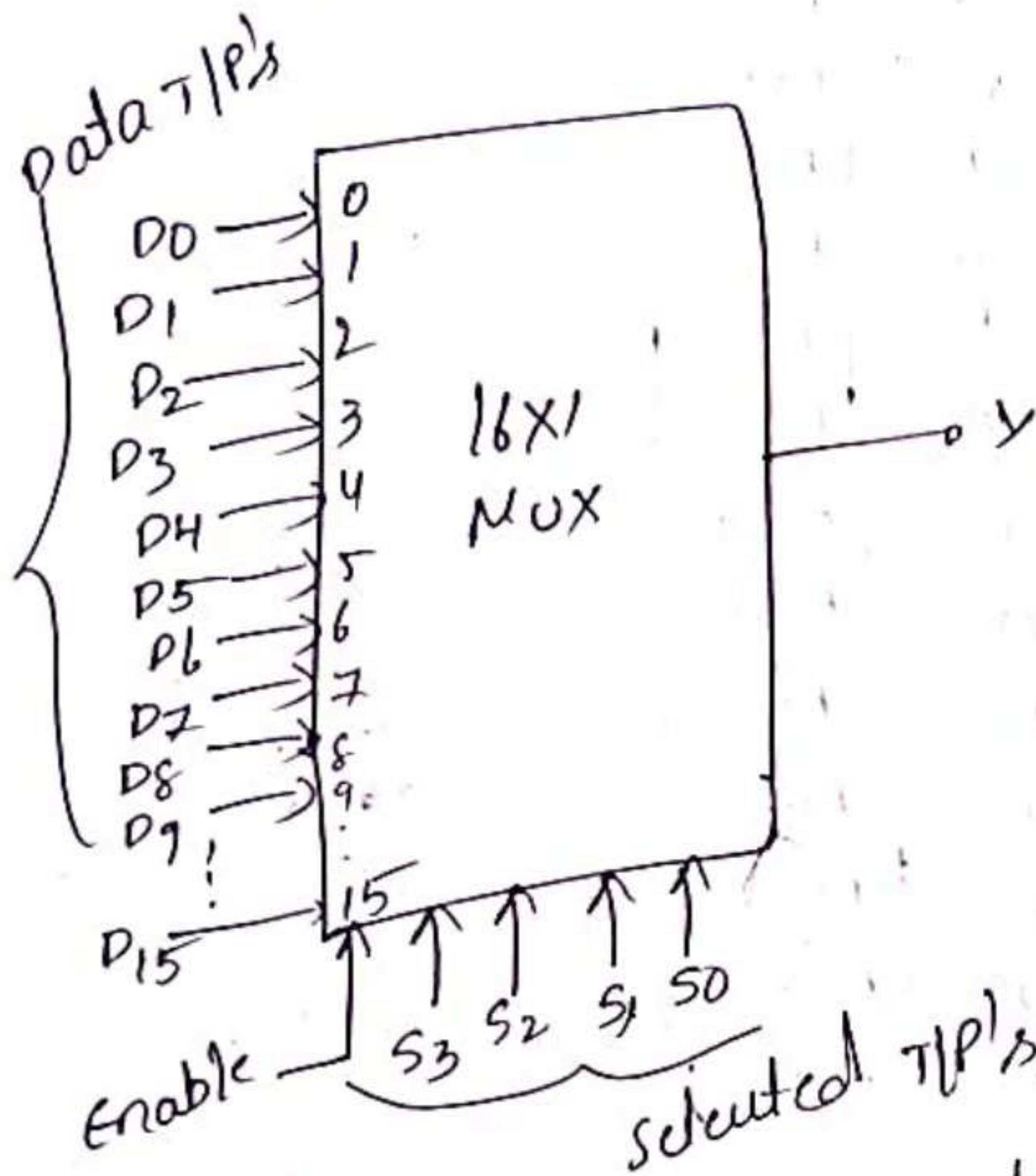


Fig. - 16X1 MUX schematic diagram.

Truth table:-

Enable	T/P's				O/P
E	S ₃	S ₂	S ₁	S ₀	Y
0	x	x	x	x	0
1	0	0	0	0	D ₀
1	0	0	0	1	D ₁
1	0	0	1	0	D ₂
1	0	0	1	1	D ₃
1	0	1	0	0	D ₄
1	0	1	0	1	D ₅
1	0	1	1	0	D ₆
1	0	1	1	1	D ₇
1	1	0	0	0	D ₈
1	1	0	0	1	D ₉
1	1	0	1	0	D ₁₀
1	1	0	1	1	D ₁₁
1	1	1	0	0	D ₁₂
1	1	1	0	1	D ₁₃
1	1	1	1	0	D ₁₄
1	1	1	1	1	D ₁₅

* Draw the logic diagram of 16x1 Mux by using above Truth table.

Applications of Multiplexer:-

- * It is used as data selector to select one out of many data T/P's
- * It is used for simplification of logic design

5

- * In the data acquisition system.
- * To designing the combinational ckt.
- * In the D/A converters.
- * To minimize the number of connections.

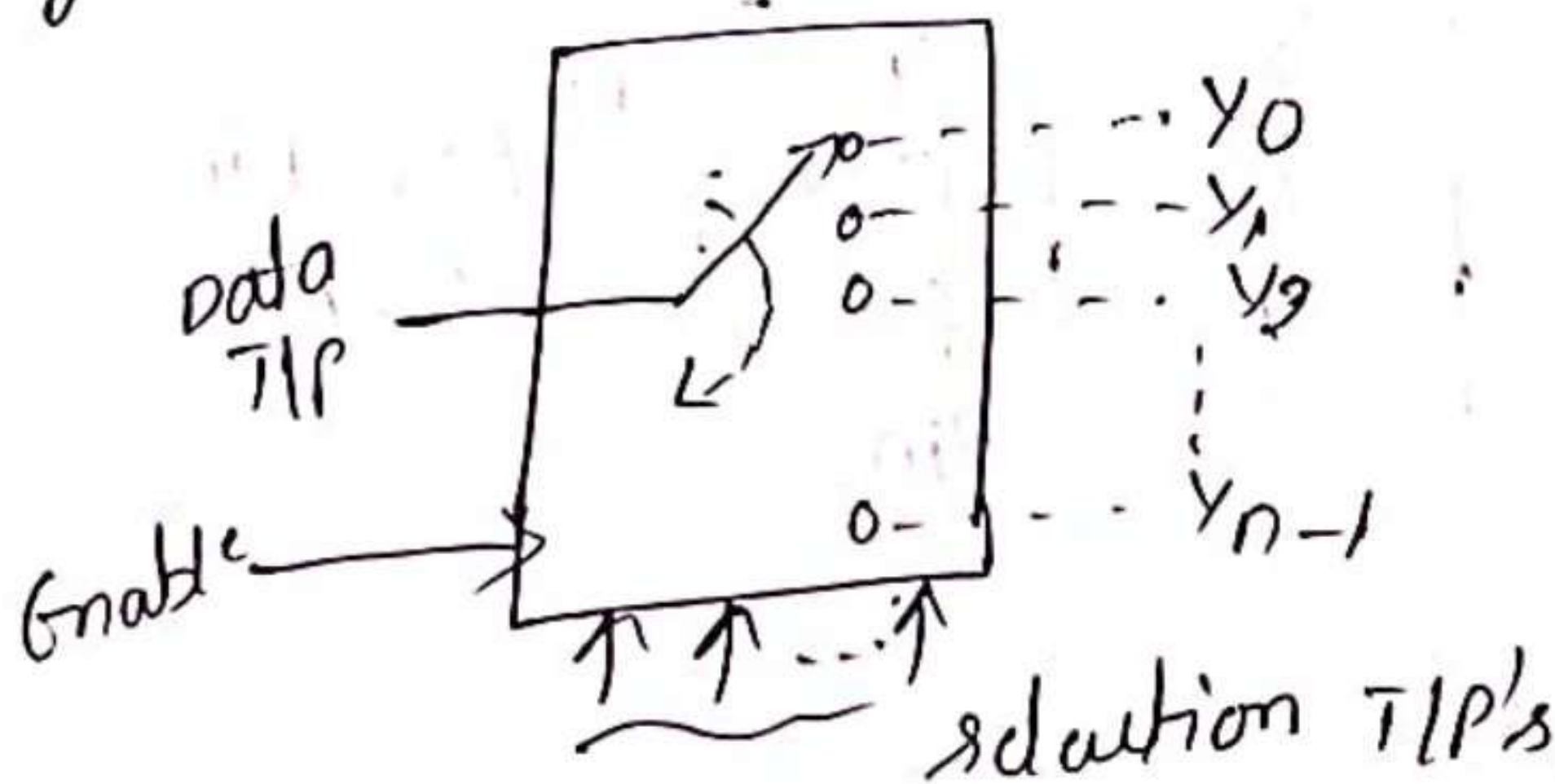
Demultiplexer:- (or) data distributor:-

The block diagram of a demultiplexer (or) decoder as shown in fig. It has only one T/P, n O/P's and m select T/P's.

- * A demultiplexer performs the reverse operation of a multiplexer, i.e. It receives one T/P & distributes it over several O/P's at a time, only one O/P line is selected by the select lines and the T/P is transmitted to the select O/P line.

[or]

def:- A demultiplexer is a digital unit (or) a combinational unit that sends data from a single T/P line to one or many (2^n) O/P lines, where the selection of an O/P line is decided by n selection lines.



Classification of demultiplexer:-

* 1:2 Demultiplexer

* 1:4 "

* 1:8 "

* 1:16 "

1X2 Demultiplexer:-

* A 1X2 demultiplexer has one data T/P, one select T/P, one Enable (E) & two O/P's, Y_0 & Y_1 . The Block diagramme of 1X2 demultiplexer as shown in fig.

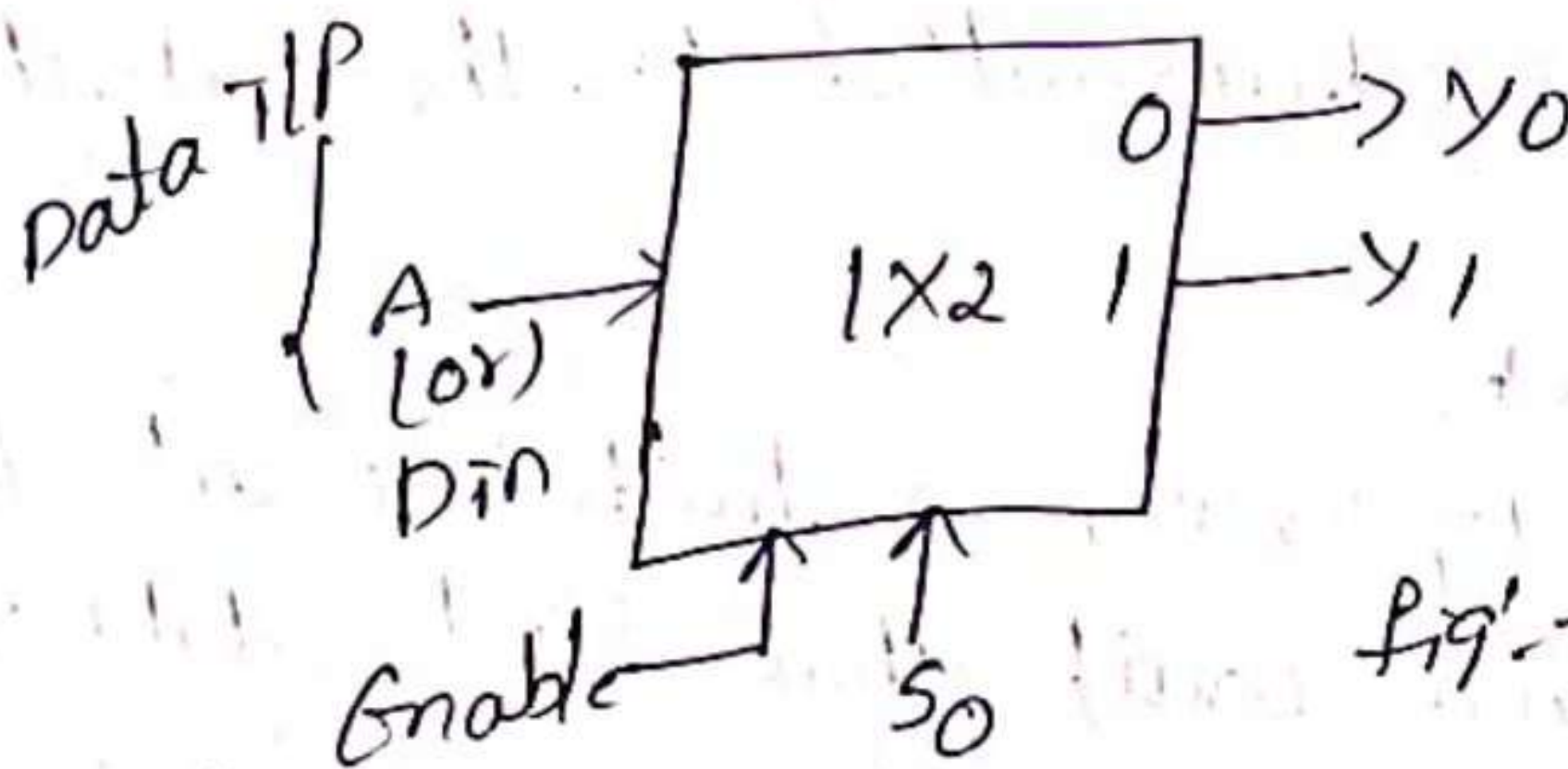


Fig:- 1X2 demux

Truth table

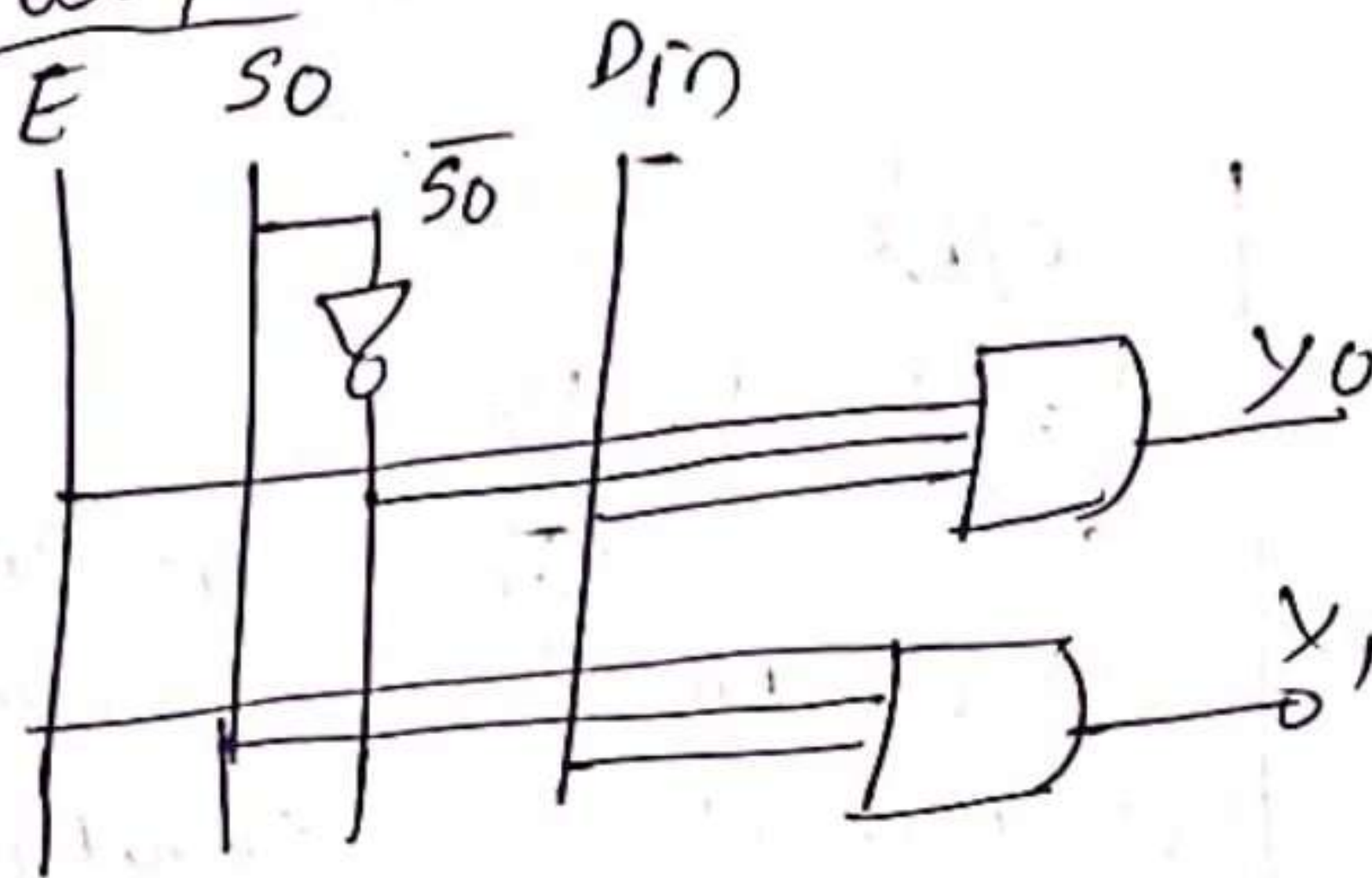
Enable E	Select S0	O/P's X ₁ Y ₀
0	X	0 0
1	0	0 A ^(or) Din
1	1	A ^(or) 0

Din

$E \bar{S}_0 \text{ Din}$
 $E S_0 \text{ Din}$

Logic diagram:-

(6)



1x4 demultiplexer:-

- * A 1x4 demux is a digital circuit that sends data from single I/P line to one of 4 O/P lines.
- * The logic symbol of 1x4 demux as shown in fig.
- * Din is the I/P line, O/P's are Y_0, Y_1, Y_2, Y_3
- * It selects one O/P out of 4 O/P's and sends I/P to the selected O/P line. S_0, S_1 are the selection lines

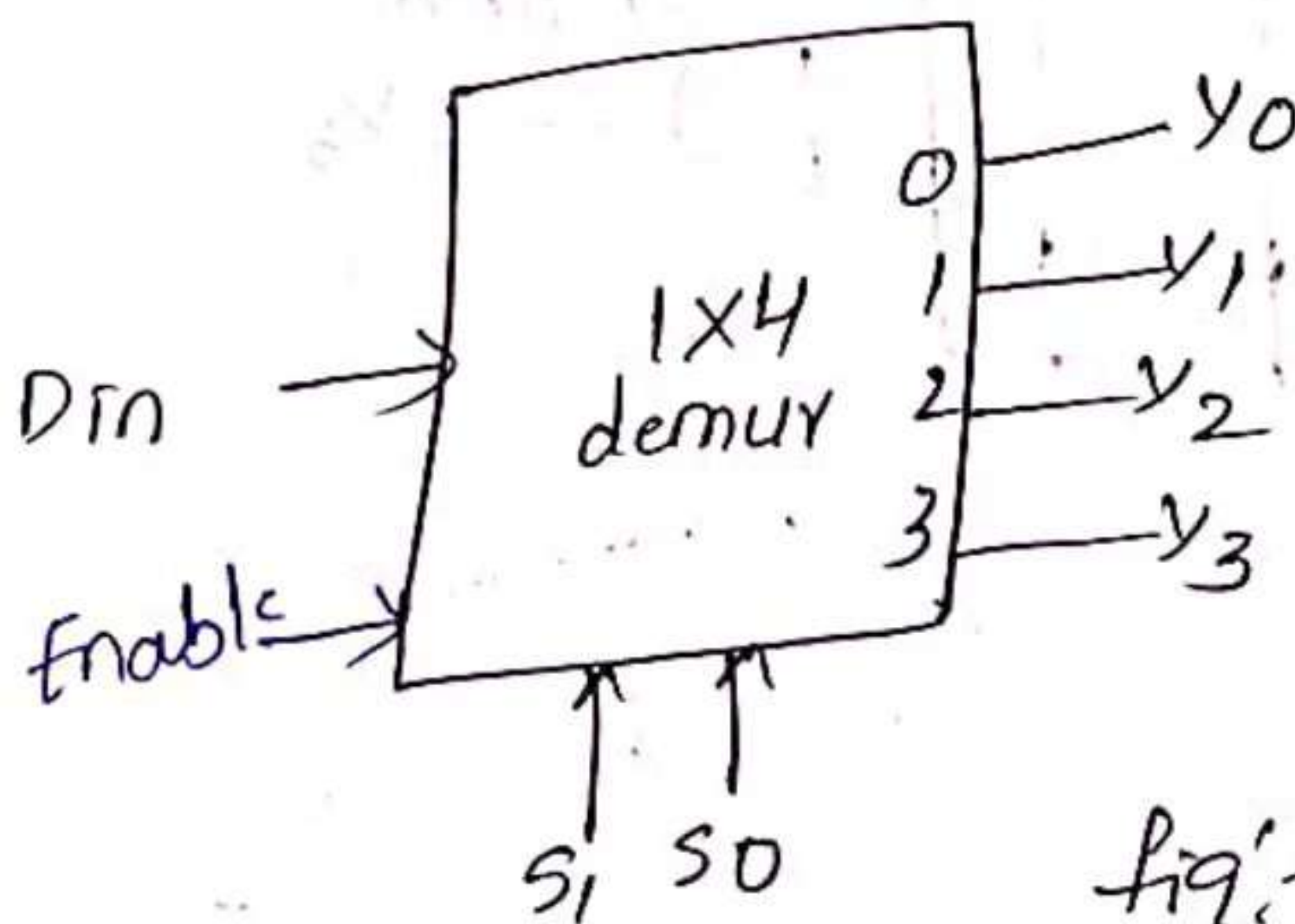
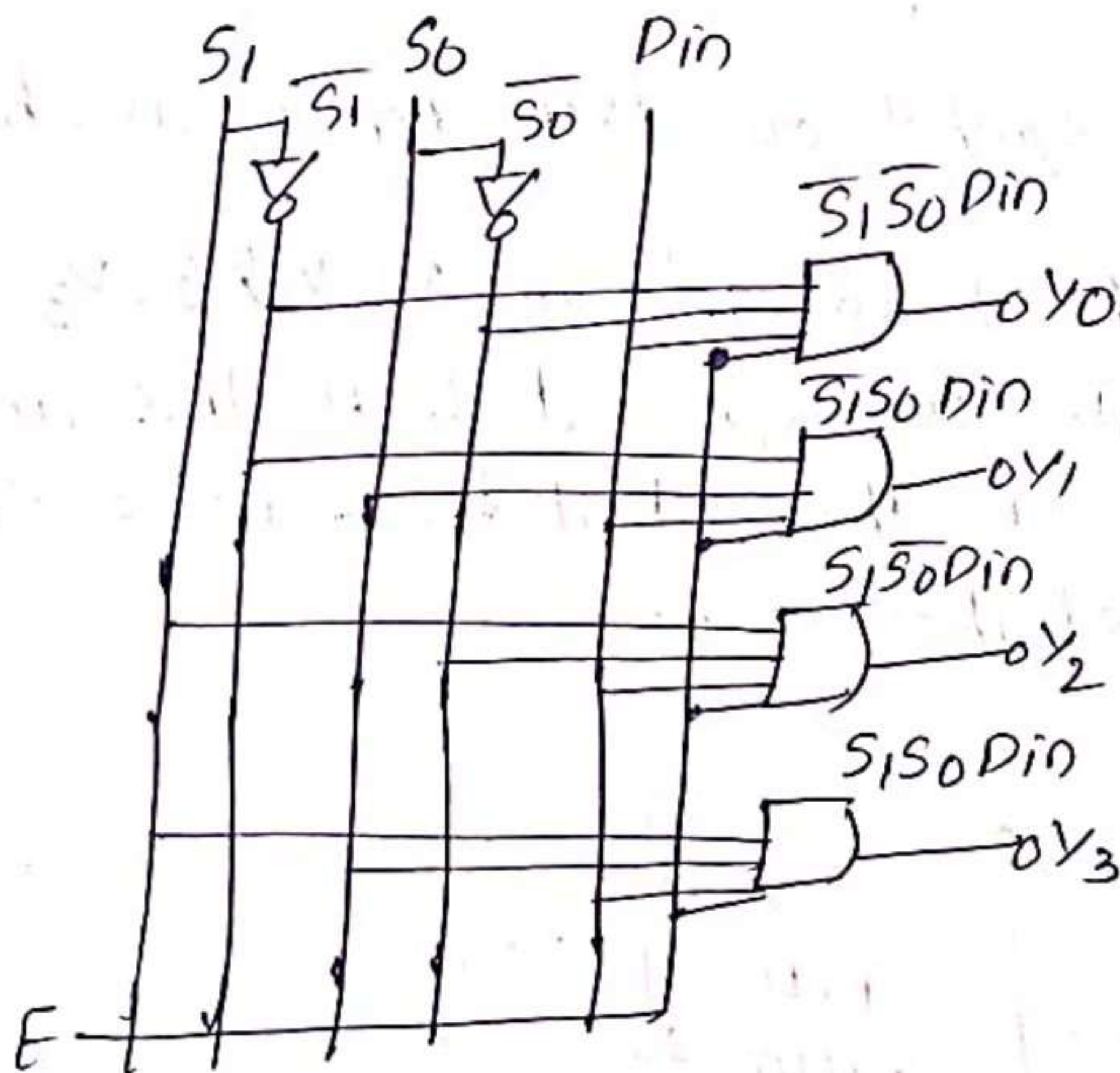


fig:- logic symbol of 1x4 demux

Truth table

I/P's		O/P's				
S_1	S_0	Y_3	Y_2	Y_1	Y_0	
0	0	0	0	0	$\overline{D_{in}}$	$\rightarrow \overline{S_1} \overline{S_0} D_{in}$
0	1	0	0	D_{in}	0	$\rightarrow \overline{S_1} S_0 D_{in}$
1	0	0	D_{in}	0	0	$\rightarrow S_1 \overline{S_0} D_{in}$
1	1	D_{in}	0	0	0	$\rightarrow S_1 S_0 D_{in}$

Logic diagram:-



1x8 demultiplexer:-

- * A 1x8 demux is digital circuit that sends data from single I/P line to one of 8 o/p lines
- * The logic symbol of 1x8 demux as shown in fig.
- * D_{in} is the I/P line and $y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7$ are the o/p line.
- * 3 selection lines are needed to select one out of 8 o/p's $8 = 2^3$
- * here s_0, s_1, s_2 are the selection lines.

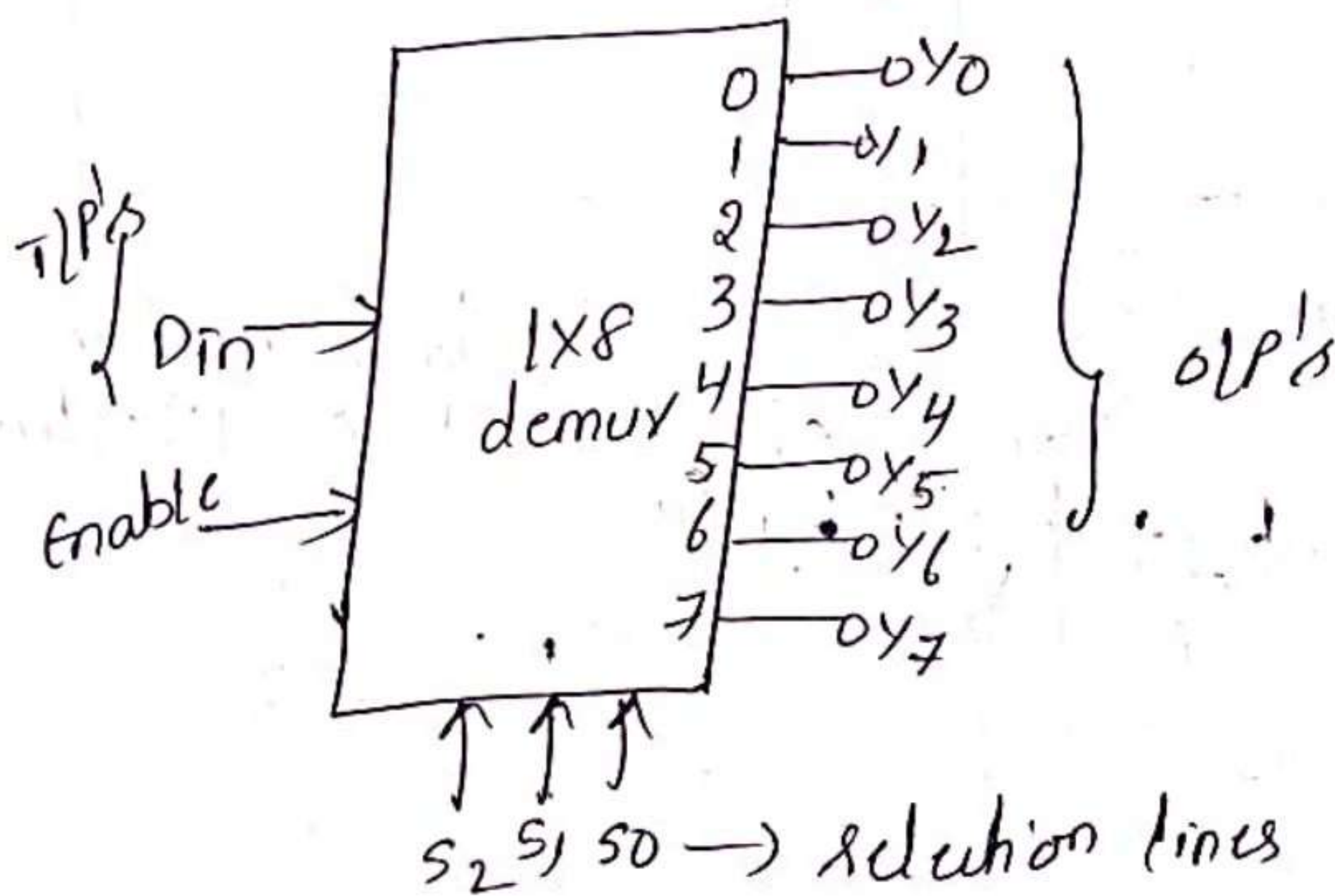


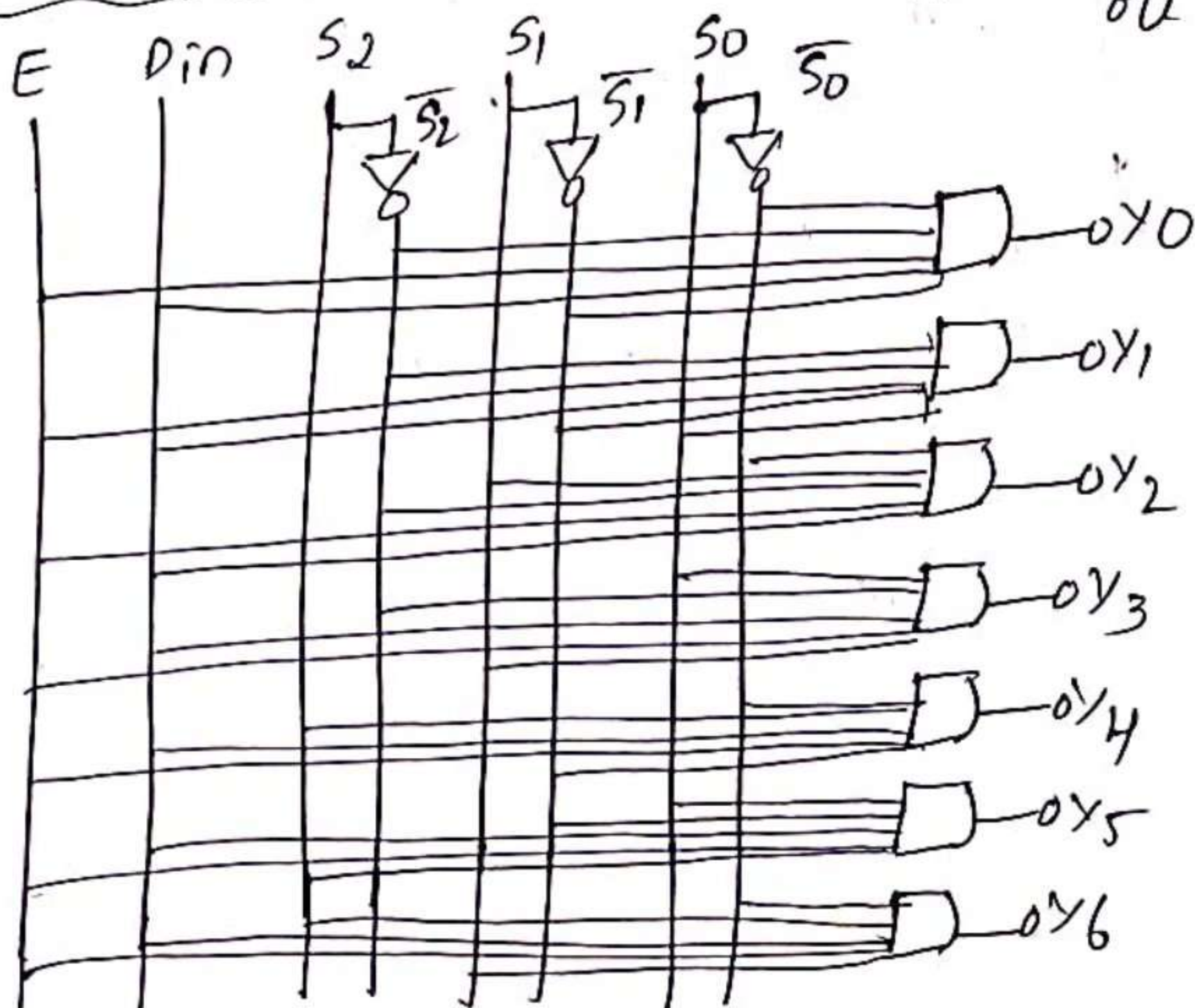
Fig. 1x8 demux

Truth Table:-

enable T/P's select				o/p's							
E	S ₂	S ₁	S ₀	Y ₇	Y ₆	Y ₅	Y ₄	Y ₃	Y ₂	Y ₁	Y ₀
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	Din
1	0	0	1	0	0	0	0	0	0	Din	0
1	0	1	0	0	0	0	0	0	Din	0	0
1	0	1	1	0	0	0	0	Din	0	0	0
1	1	0	0	0	0	0	Din	0	0	0	0
1	1	0	1	0	0	Din	0	0	0	0	0
1	1	1	0	0	Din	0	0	0	0	0	0
1	1	1	1	Din	0	0	0	0	0	0	0

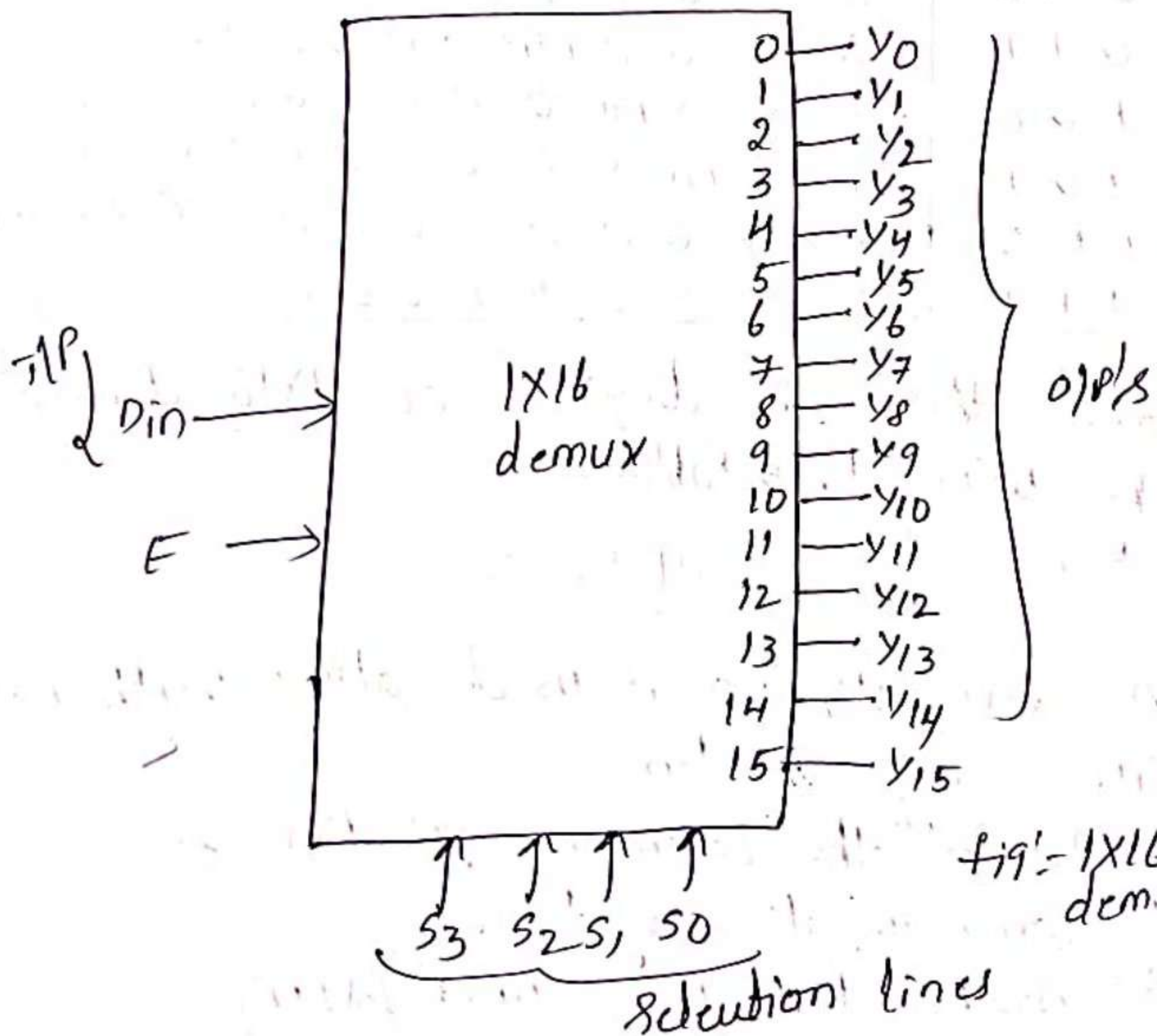
Logic diagram

fig:- logic diagram of 1x8 demux



1x16 demultiplexer:-

- * A 1x16 demux is digital circuit that sends data from single I/P line to one of 16 o/p lines
- * The logic symbol of 1x16 demux as shown in fig.
- * 'Din' is the I/P line $y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11}, y_{12}, y_{13}, y_{14}, y_{15}$ are the o/p lines.
- * here 'n' selection lines are needed to select one output of 16 o/p's $2^4 = 16$
- * here s_0, s_1, s_2, s_3 are the selection lines.



Truth table:-

selection I/P's				outputs															
S ₃	S ₂	S ₁	S ₀	Y ₁₅	Y ₁₄	Y ₁₃	Y ₁₂	Y ₁₁	Y ₁₀	Y ₉	Y ₈	Y ₇	Y ₆	Y ₅	Y ₄	Y ₃	Y ₂	Y ₁	Y ₀
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Pin
1	1	1	1	1	Pin	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* draw the logic diagram of 1X16 demux by using truth table.

Applications of Demultiplexer:-

- * A demultiplexer is used along with multiplexer in single system.
- * some of the common applications are.
 1. Communication systems
 2. Arithmetic logic unit [ALU]

3. Logic function generator

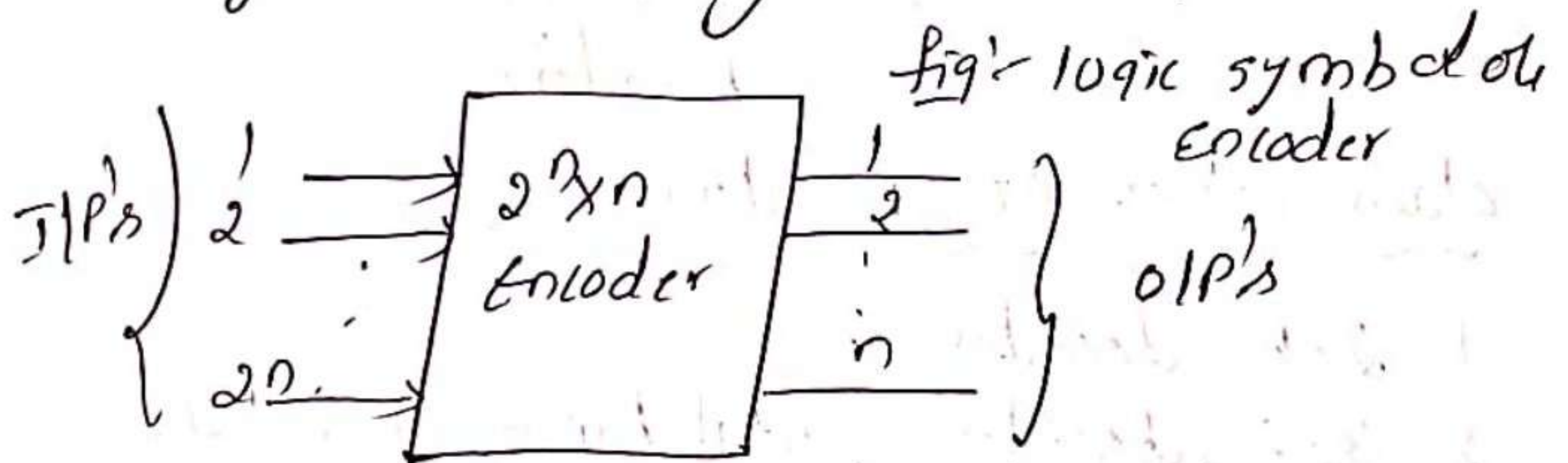
4. Line Controller.

(9)

* A de-multiplexer is used to distribute the signals received on a common channel to multiple lines.

^{2M}
* Encoder:-

Encoder is a digital circuit (or) a combinational circuit that converts data from 2^n I/P lines to n O/P lines. Where it converts data from decimal, octal and hexa-decimal number system to binary number system.



* An encoder has multiple I/P's and multiple O/P's as shown in fig.

* The I/P to encoder may be decimal, octal (or) hexa decimal number but the O/P is binary number.

* 2M Decoder :-

Decoder is a digital unit (or) a combinational circuit that converts data from n i/p lines to 2^n o/p lines, where it converts data from binary number system to decimal, octal or hexa decimal number systems.

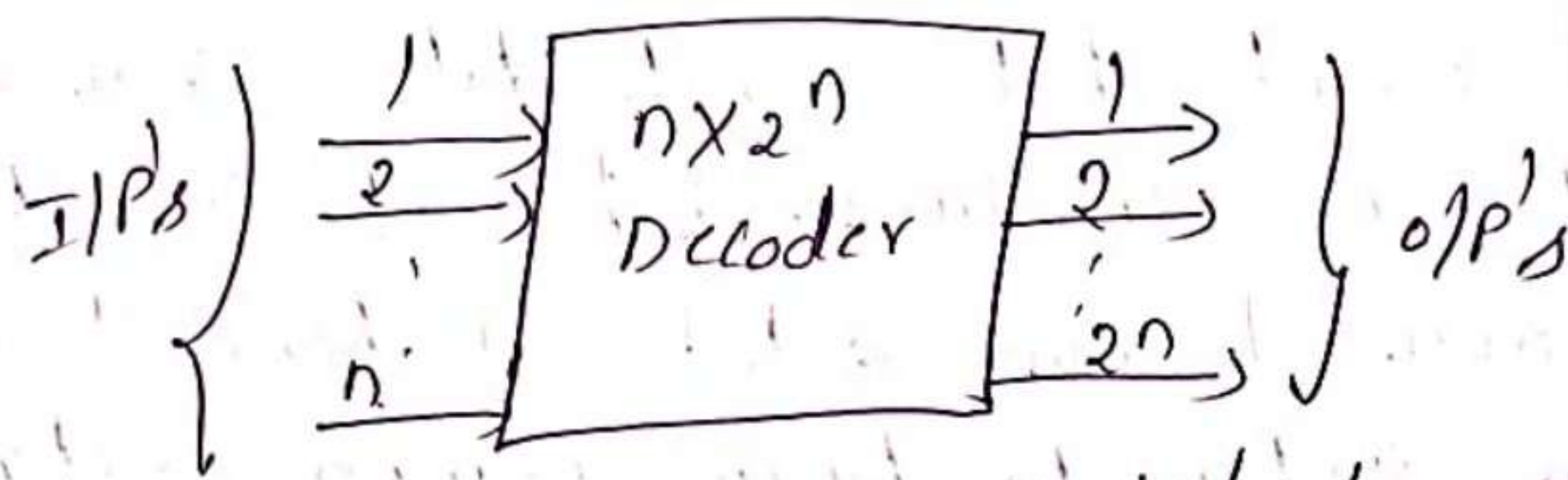


Fig. - logic symbol of Decoder.

Classification of Decoders :-

1. 2×4 decoder
2. 3×8 decoder [octal binary to octal]
3. 4×10 decoder [binary to decimal]
4. 4×16 decoder [binary to hexa decimal decoder]

1. 2x4 decoder:-

- * The Block diagram of 2x4 line decoder has shown in Below.
- * A 2 to 4 line decoder also termed as 2x4 (or) 2:4 decoder.
- * A 2x4 decoder has 2 i/p's in 4 o/p's
- * A logic symbol of 2x4 decoder where, A & B are i/p's in o/p's are y_0, y_1, y_2 & y_3
- * The truth table of 2x4 decoder as shown in below

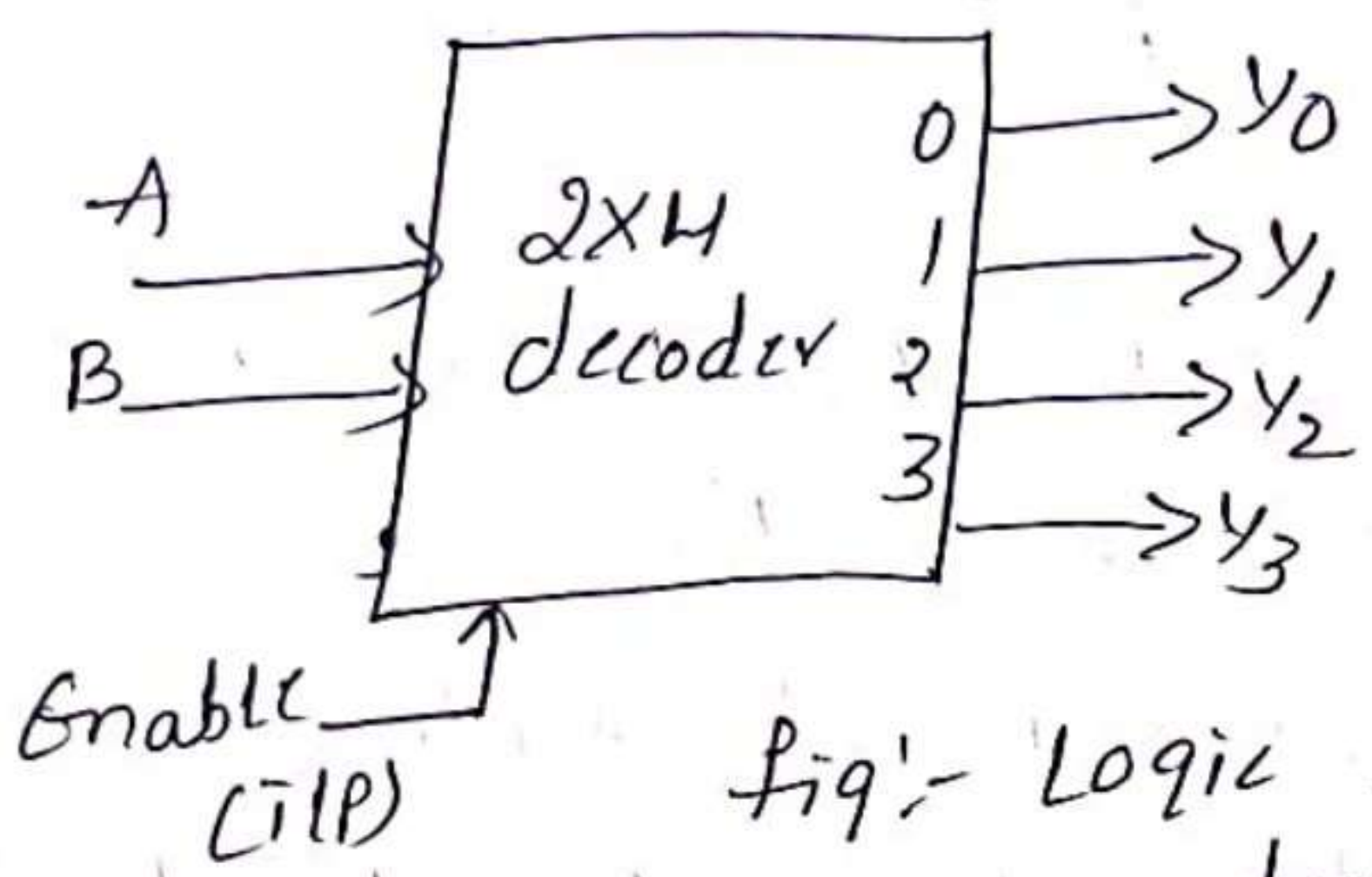


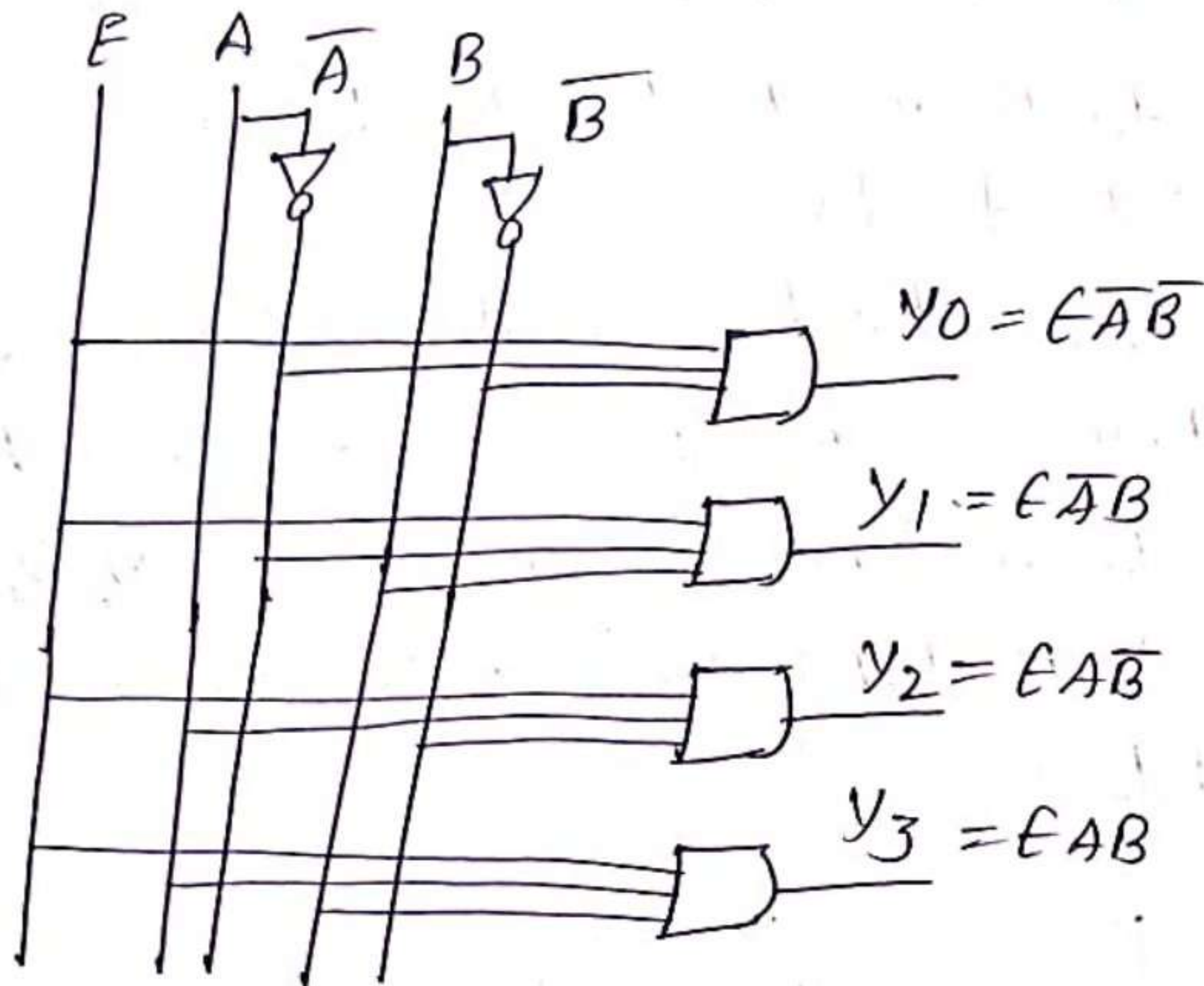
Fig:- Logic symbol of 2x4 decoder

Truth table:-

Enable E	I/P's A B		O/P's y_0 y_1 y_2 y_3				
0	X	X	0	0	0	0	
1	0	0	1	0	0	0	$\rightarrow E \bar{A} \bar{B}$
1	0	1	0	1	0	0	$\rightarrow E \bar{A} B$
1	1	0	0	0	1	0	$\rightarrow E A \bar{B}$
1	1	1	0	0	0	1	$\rightarrow E A B$

$$Y_0 = E \bar{A} \bar{B}, Y_1 = E \bar{A} B, Y_2 = E A \bar{B}, Y_3 = E A B$$

Logic diagram:-



2. 3 to 8 line decoder:-

- * A 3 to 8 line decoder also termed as 3x8 decoder (or) 3:8 decoder.
- * A 3x8 decoder has 3 I/P's & 8 O/P's
- * It is also called as Binary to octal decoder.
- * The logic symbol of 3x8 decoder, where A, B, & C are the inputs and outputs are $Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6$ & Y_7 .
- * The truth table of 3x8 decoder as shown in below.

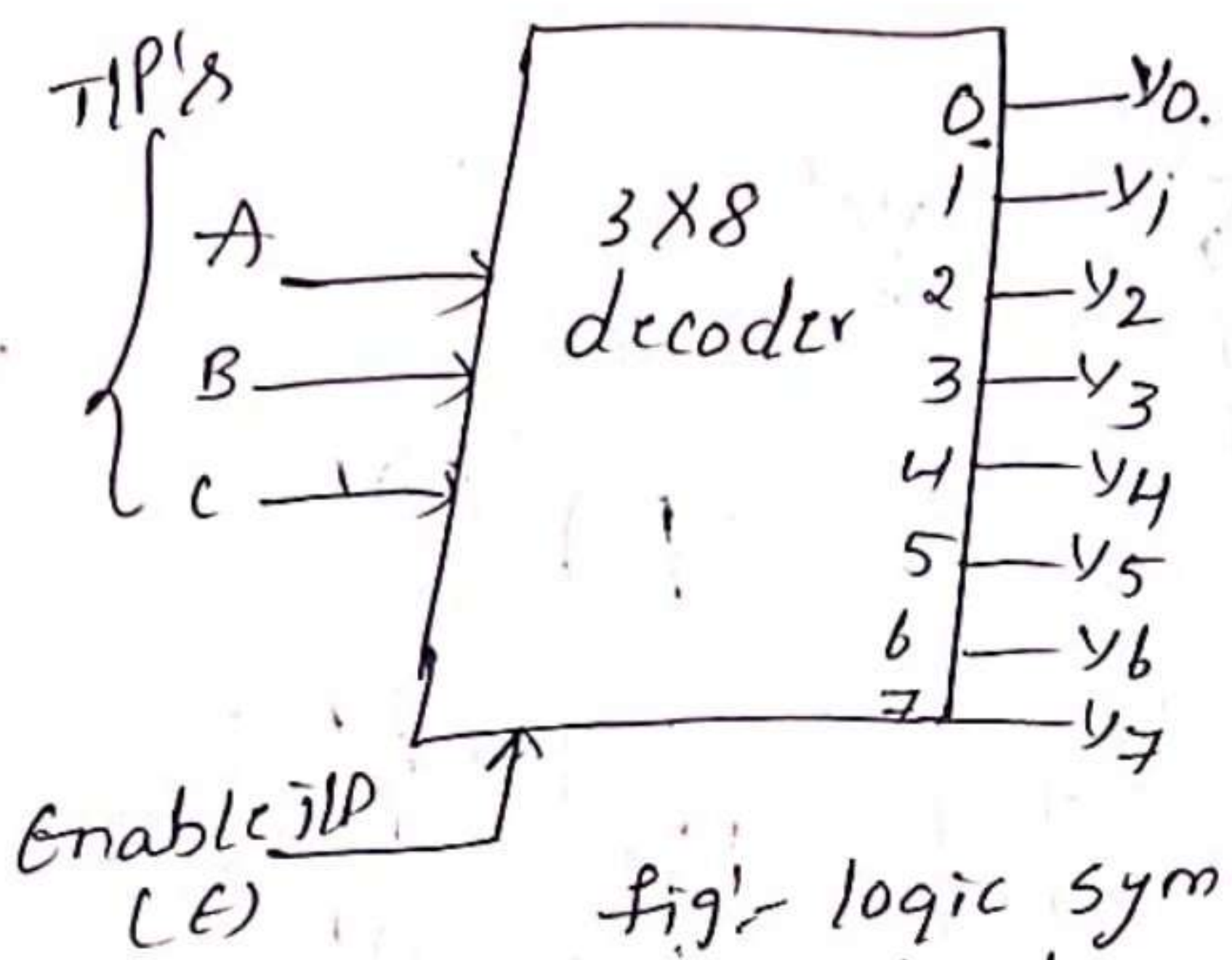
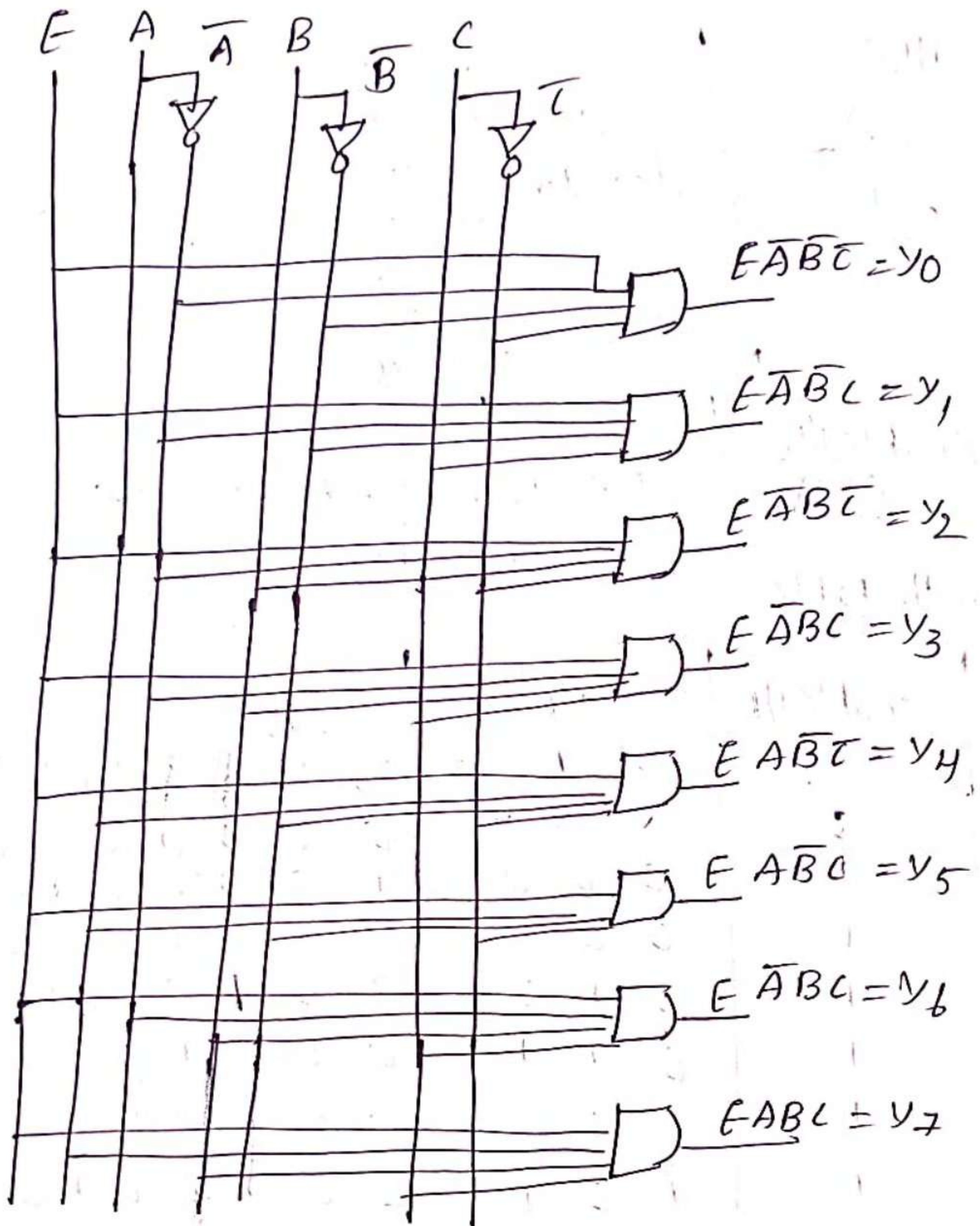


fig:- logic symbol of 3x8 decoder.

Truth table:-

Enable I/P E	I/P's A B C			O/P's y ₀ y ₁ y ₂ y ₃ y ₄ y ₅ y ₆ y ₇								
0	X	X	X	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0	0
1	1	1	0	0	0	0	0	0	0	1	0	0
1	1	1	1	0	0	0	0	0	0	0	0	1

$$\begin{aligned}
 y_0 &= E \bar{A} \bar{B} \bar{C}, y_1 = E \bar{A} \bar{B} C, y_2 = E \bar{A} B \bar{C} \\
 y_3 &= E \bar{A} B C, y_4 = E A \bar{B} \bar{C}, y_5 = E A \bar{B} C \\
 y_6 &= E A B \bar{C}, y_7 = E A B C
 \end{aligned}$$



3. 4x10 decoder:-

(12)

- * A 4-to 10 line decoder also termed as 4x10 decoder (or) 4:10 decoder.
- * A 4x10 decoder has four I/P's w 10 o/P's
- * It is also called as BCD-to Decimal decoder.
- * The logic symbol of 4x10 decoder, Where the I/P's are A, B, C & D, w the o/P's are $y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9$.
- * The truth table of 4x10 decoder as shown in below.

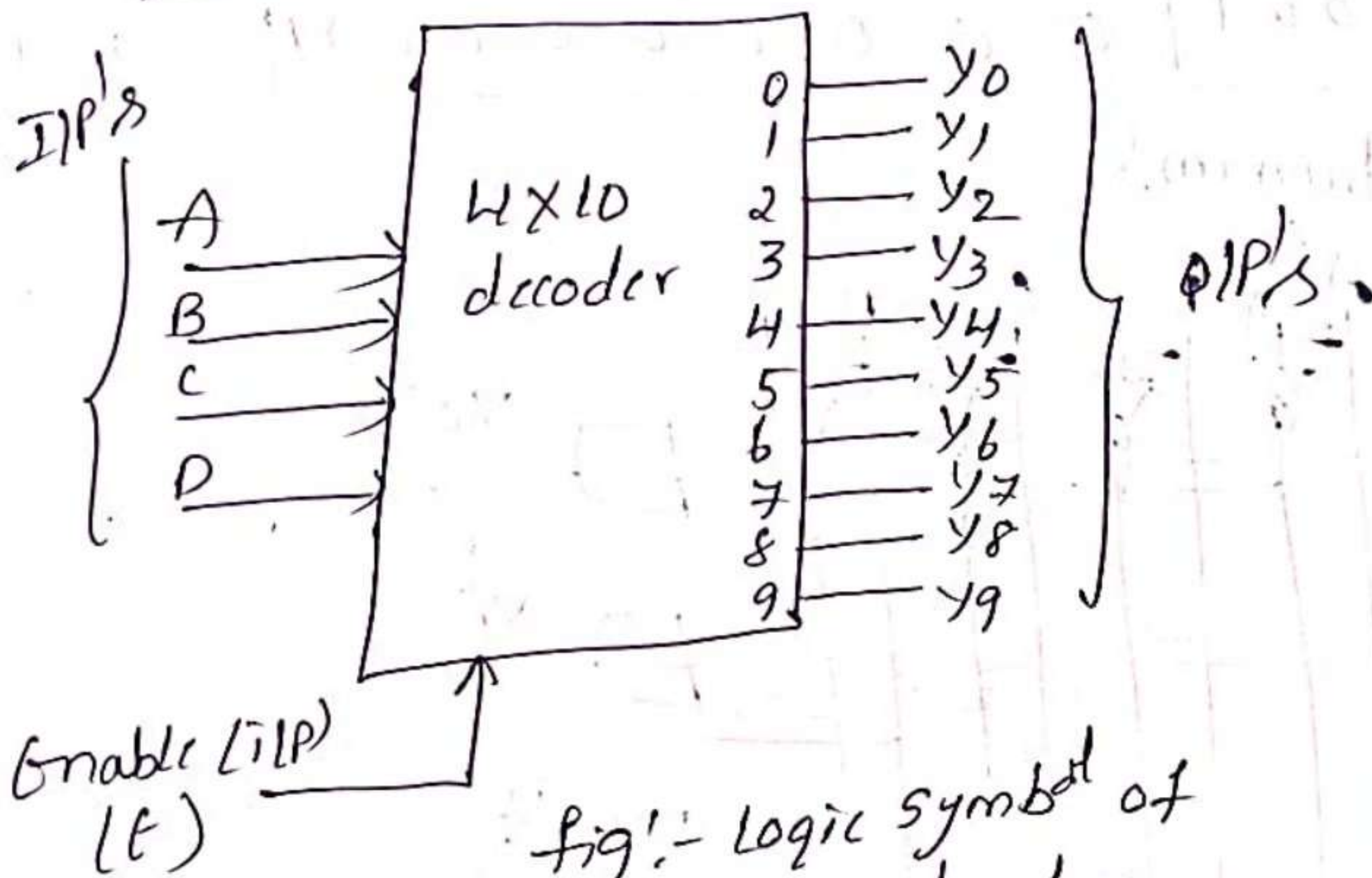


Fig:- Logic symbol of 4x10 decoder.

Truth table:-

I/P's					O/P's									
E	A	B	C	D	Y ₀	Y ₁	Y ₂	Y ₃	Y ₄	Y ₅	Y ₆	Y ₇	Y ₈	Y ₉
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	1	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	1	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	1

$$\rightarrow \bar{A}\bar{B}\bar{C}\bar{D} = Y_0$$

$$\rightarrow \bar{A}\bar{B}\bar{C}D = Y_1$$

$$\rightarrow \bar{A}\bar{B}C\bar{D} = Y_2$$

$$\rightarrow \bar{A}\bar{B}CD = Y_3$$

$$\rightarrow \bar{A}B\bar{C}\bar{D} = Y_4$$

$$\rightarrow \bar{A}B\bar{C}D = Y_5$$

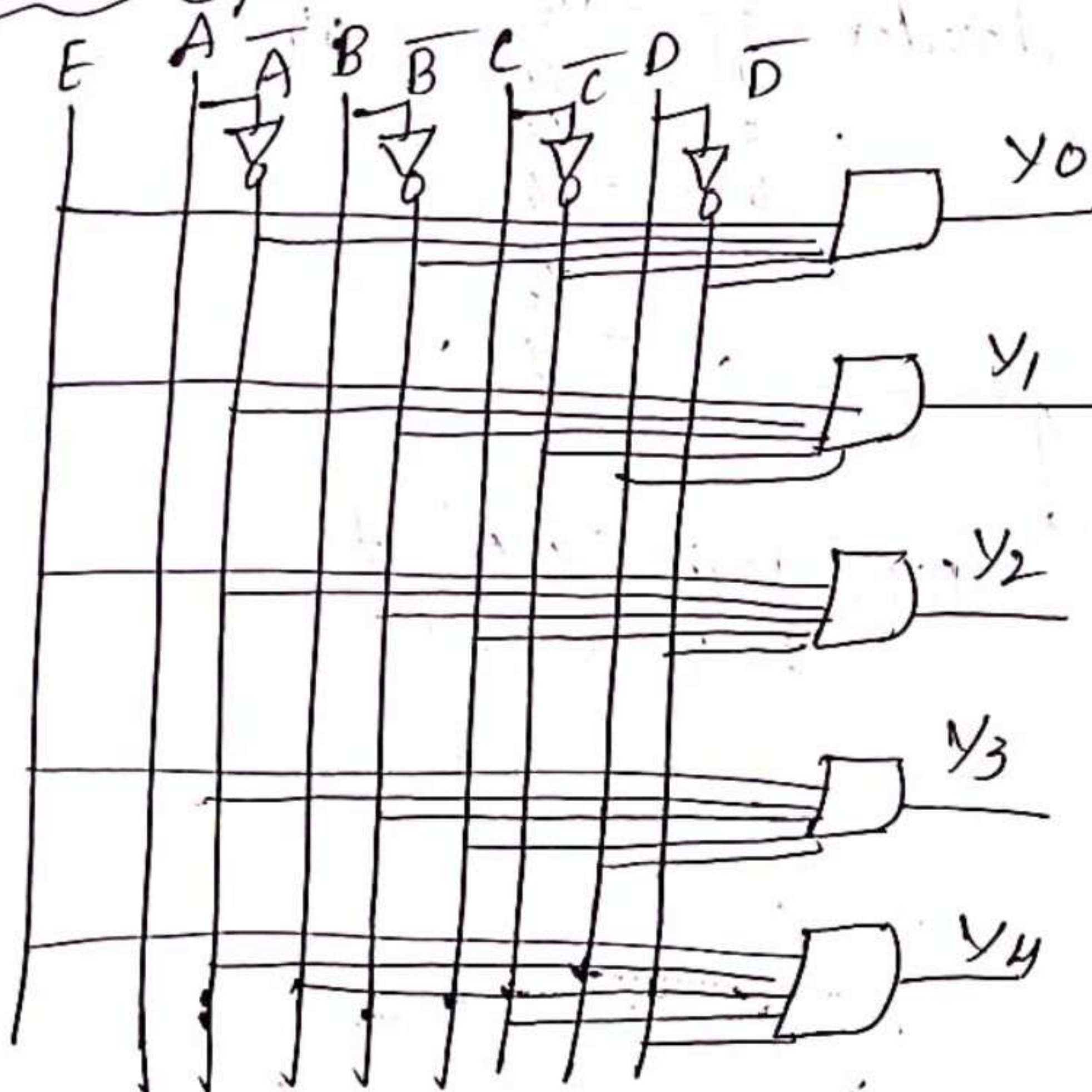
$$\rightarrow \bar{A}B C\bar{D} = Y_6$$

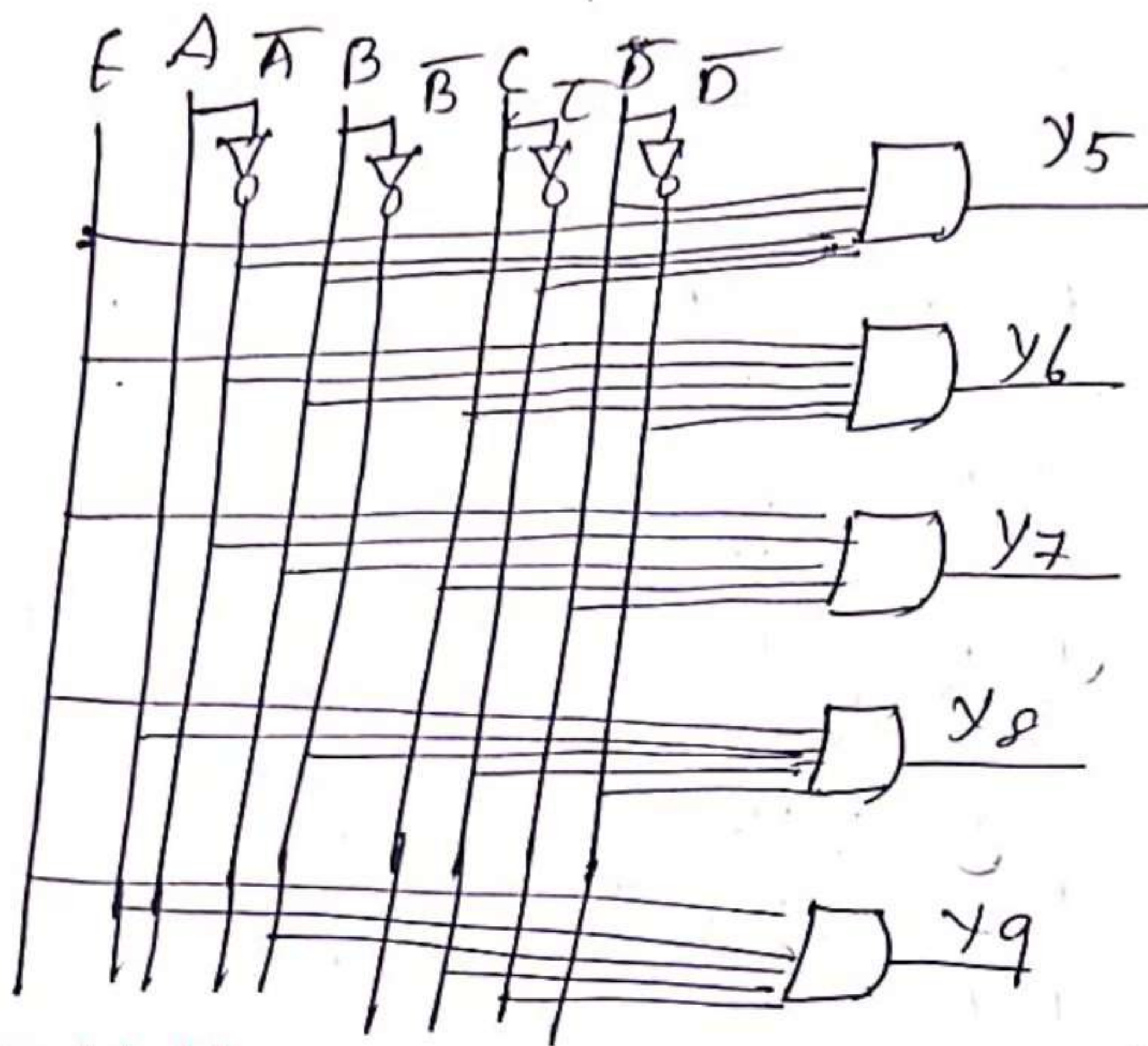
$$\rightarrow \bar{A}B CD = Y_7$$

$$\rightarrow A\bar{B}\bar{C}\bar{D} = Y_8$$

$$\rightarrow A\bar{B}\bar{C}D = Y_9$$

Logic diagram:-





(13)

Home Work:-

1) Explain 4x16 decoder?

Construct 3x8 decoder by using ^{two} 2x4 decoders:-

* The Connection diagram for a 3x8 decoder using 2x4 decoder as shown in figure.

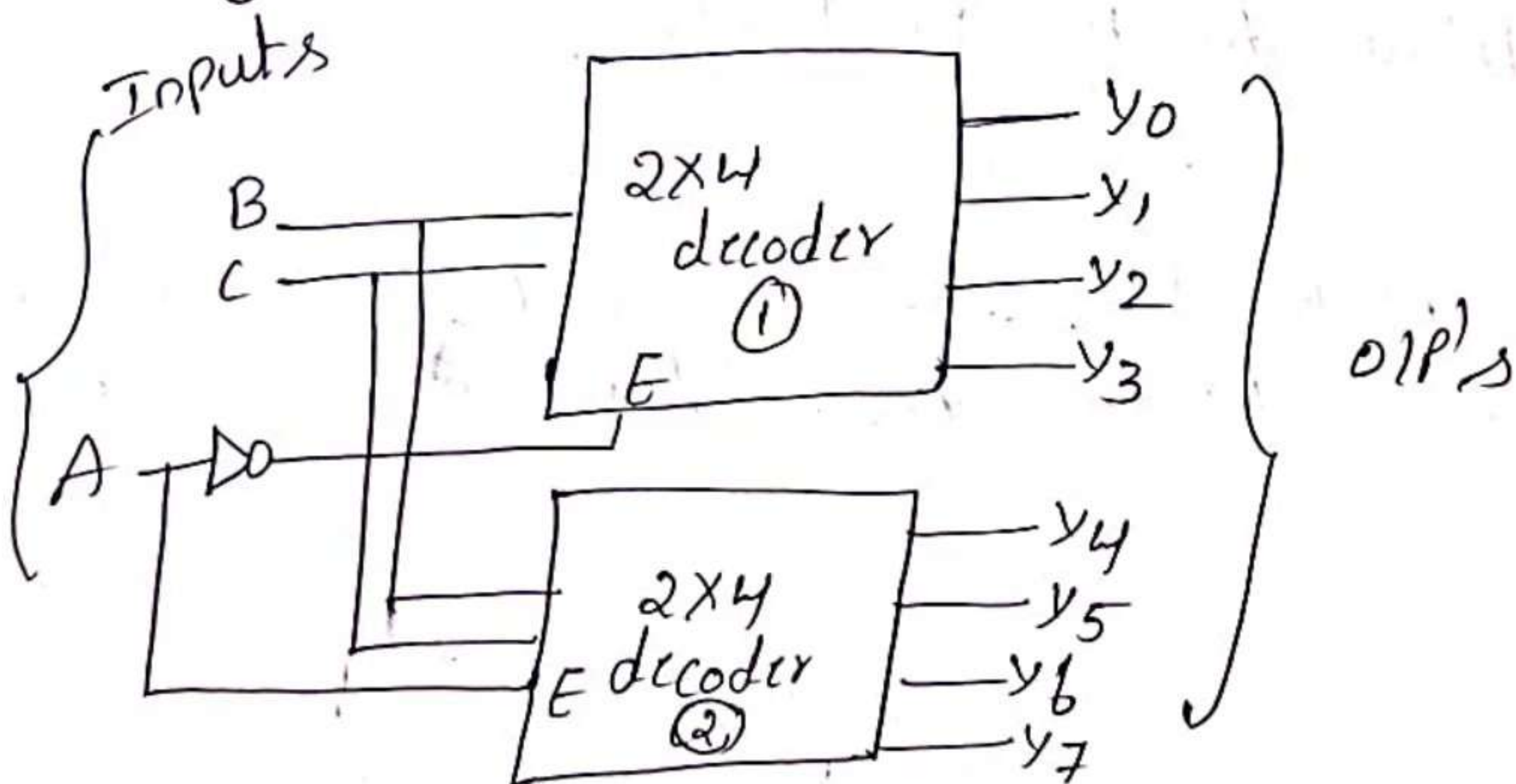


Fig:- 3x8 decoder using 2x4 decoder.

Truth table:-

	A	B	C	O/P
A=0	0	0	0	y ₀
	0	0	1	y ₁
	0	1	0	y ₂
	0	1	1	y ₃
A=1	1	0	0	y ₄
	1	0	1	y ₅
	1	1	0	y ₆
	1	1	1	y ₇

*. The above fig. shows the three TLPs are connected in parallel, with each other. The TLP $\bar{A}$ is used to enable one of the two 2x4 decoders. Also $\bar{A}$ is applied to the enable TLP of decoder (1) whereas $\bar{A}$ is applied directly to decoder (2).

A 4 Line to 16 Line decoder using 3:8 decoder:-

* A block diagram for a 4:16 decoder using 3:8 decoder has been shown in fig.

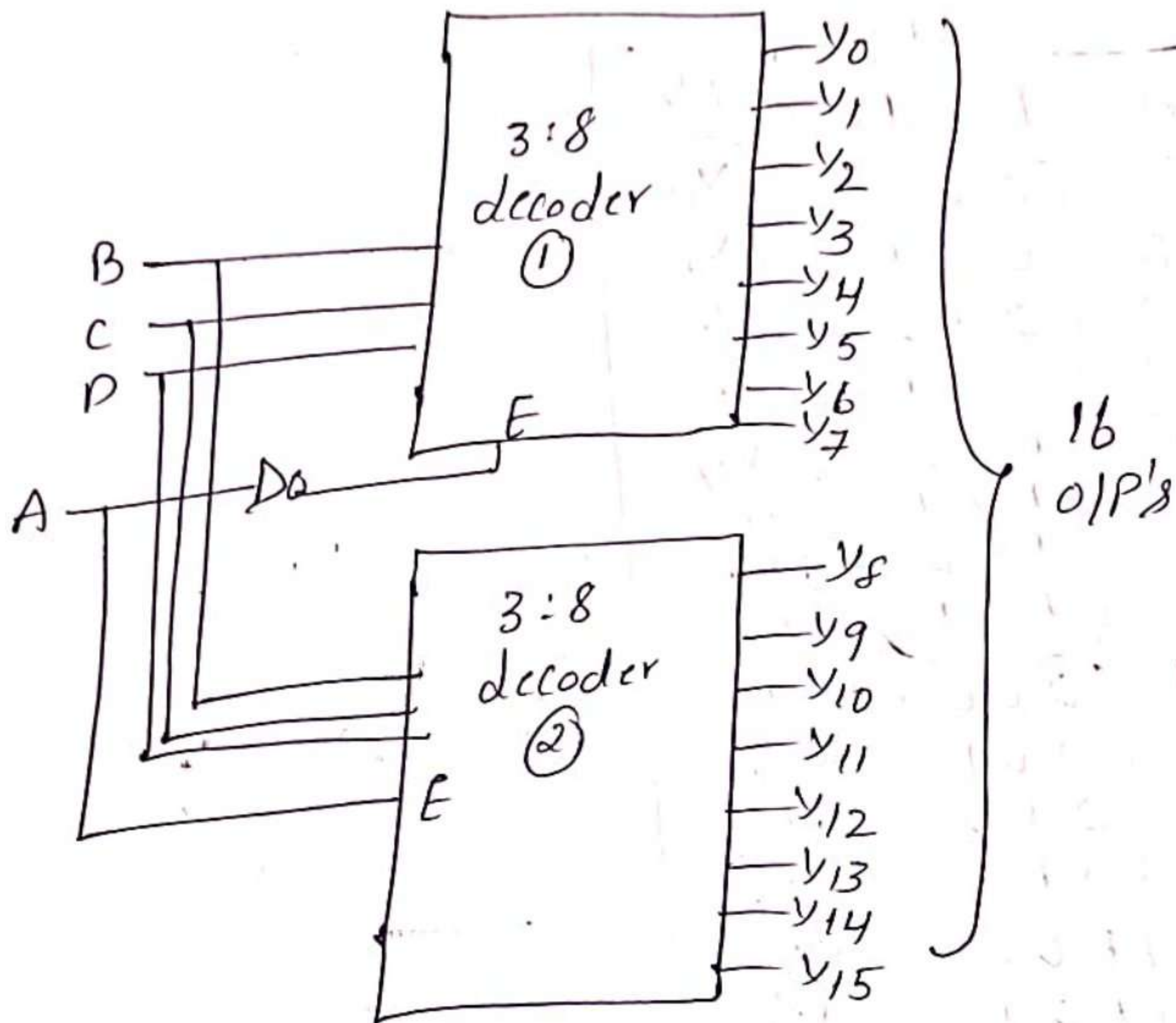


Fig:- A 4 to 16 decoder using 3 to 8 decoder.

- * The above fig shows that three TLP's are connected in parallel, with each other and then connected to the BCD TLP's.
- * The 4th TLP 'A' is used to enable one of the two 3X8 decoders. $\bar{A}$ is applied to the enable TLP of decoder (1), whereas 'A' is applied directly to decoder (2).

Truth table :-

		I/P's				O/P
		A	B	C	D	<u>y</u>
A=0		0	0	0	0	y ₀
		0	0	0	1	y ₁
		0	0	1	0	y ₂
		0	0	1	1	y ₃
		0	1	0	0	y ₄
		0	1	0	1	y ₅
		0	1	1	0	y ₆
		0	1	1	1	y ₇
A=1		1	0	0	0	y ₈
		1	0	0	1	y ₉
		1	0	1	0	y ₁₀
		1	0	1	1	y ₁₁
		1	1	0	0	y ₁₂
		1	1	0	1	y ₁₃
		1	1	1	0	y ₁₄
		1	1	1	1	y ₁₅

Classification of Encoders:-

15

1. Priority Encoder
2. 4×2 Encoder
3. 8×3 Encoder [Octal to Binary]
4. 10×4 Encoder [Decimal to Binary]
5. 16×4 Encoder [hexa decimal to Binary]

1. 4×2 Encoder:-

* A 4 to 2 line Encoder is also termed as 4×2 (or) $4:2$ Encoder.

* It has 4 I/P's or 2 O/P's

* ~~The inputs are y_0, y_1, y_2, y_3 or O/P's are A and B .~~

* The inputs are D_0, D_1, D_2 and D_3 , O/P's are y_0 and y_1 .

* The logic symbol and truth table of 4×2 Encoder are shown in below

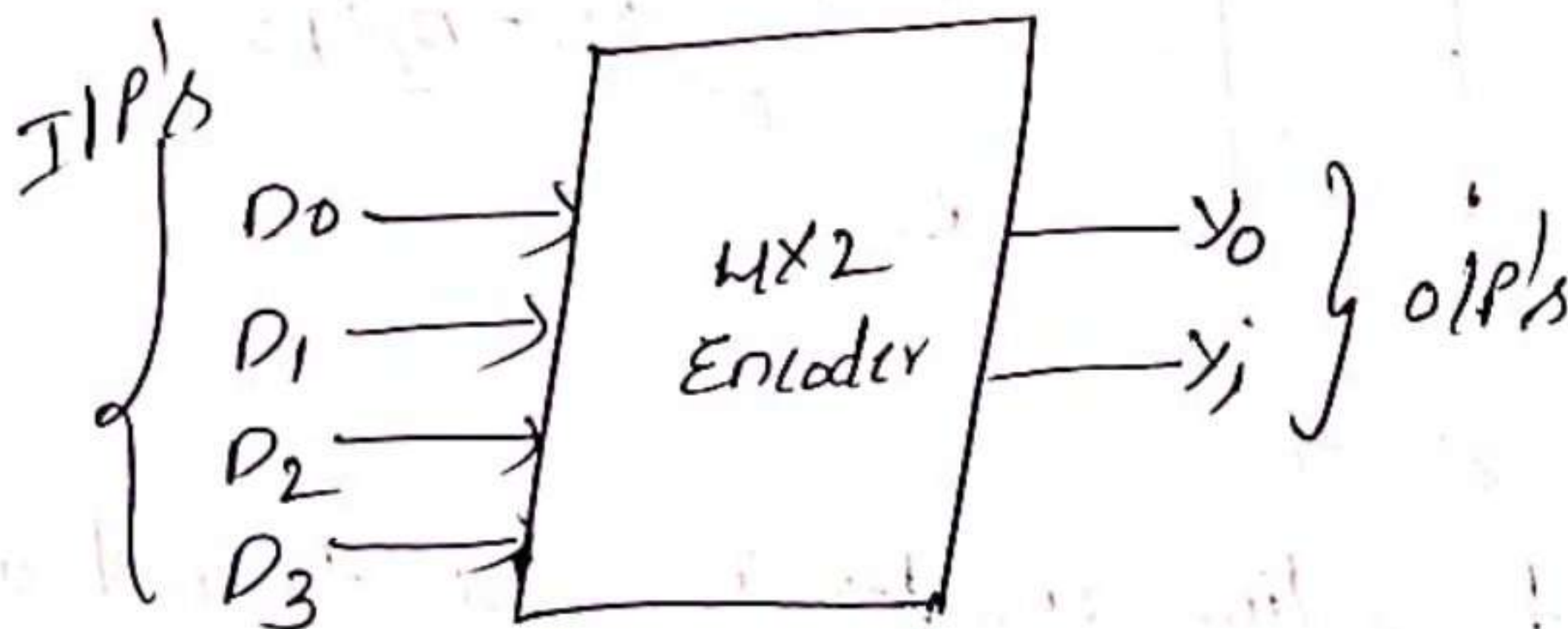


Fig:- logic symbol of 4×2 encoder.

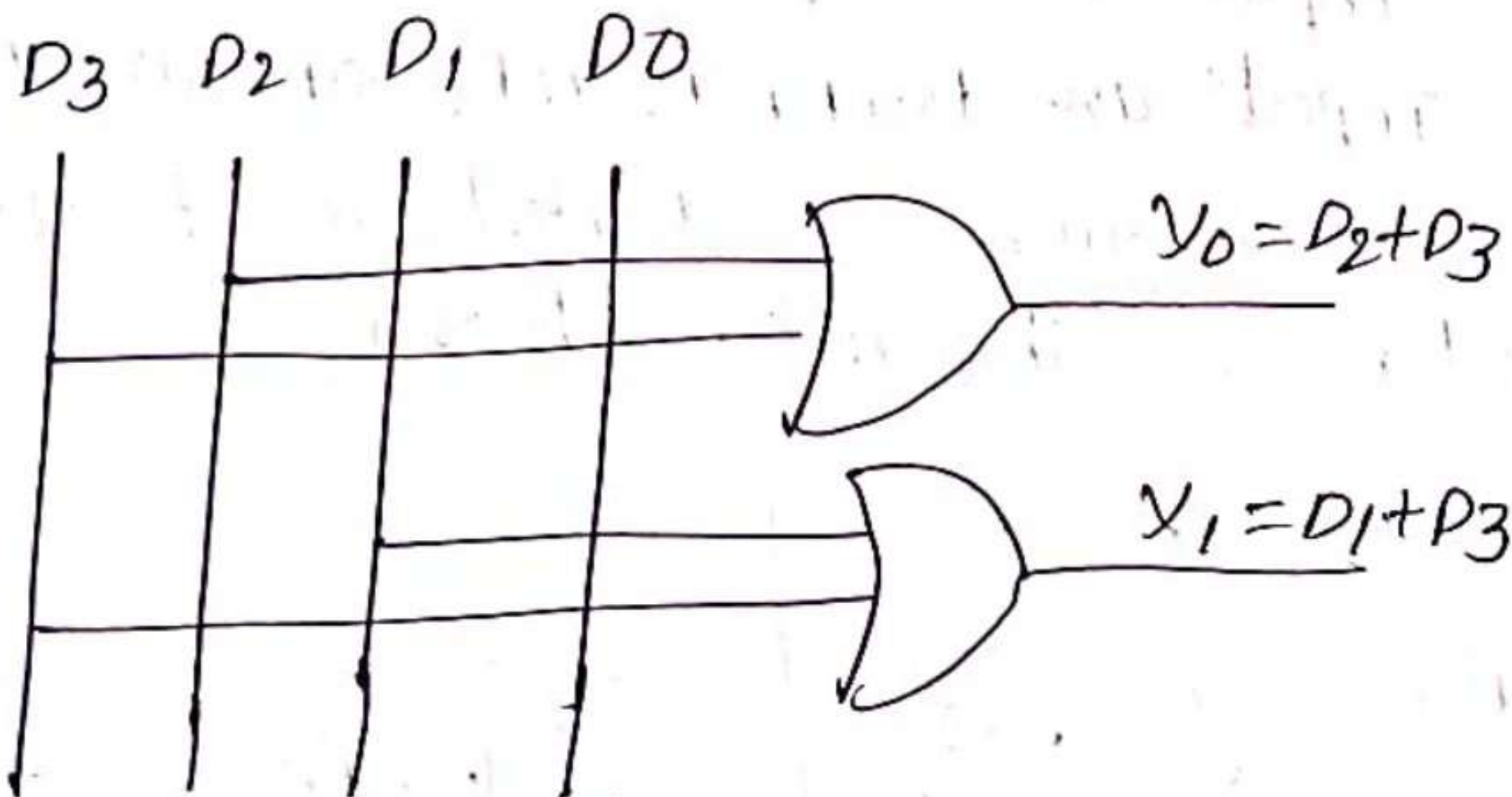
Truth table

Inputs				O/P's	
D ₀	D ₁	D ₂	D ₃	Y ₀	Y ₁
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

$\therefore Y_0 = D_2 + D_3$

$Y_1 = D_1 + D_3$

Logic diagram



2. 8X3 line Encoder:-

- * An 8 to 3 line Encoder is also termed as 8X3 Encoder (or) 8:3 Encoder.
- * An 8X3 encoder has eight I/P's & 3 O/P's
- * It is also called as Octal-to Binary Encoder.

* It converts an octal number to binary number. (16)

* The I/P's are $D_0, D_1, D_2, D_3, D_4, D_5, D_6$ & D_7 & the o/p's are Y_0, Y_1, Y_2 .

* The logic symbol & the Truth table as shown in fig.

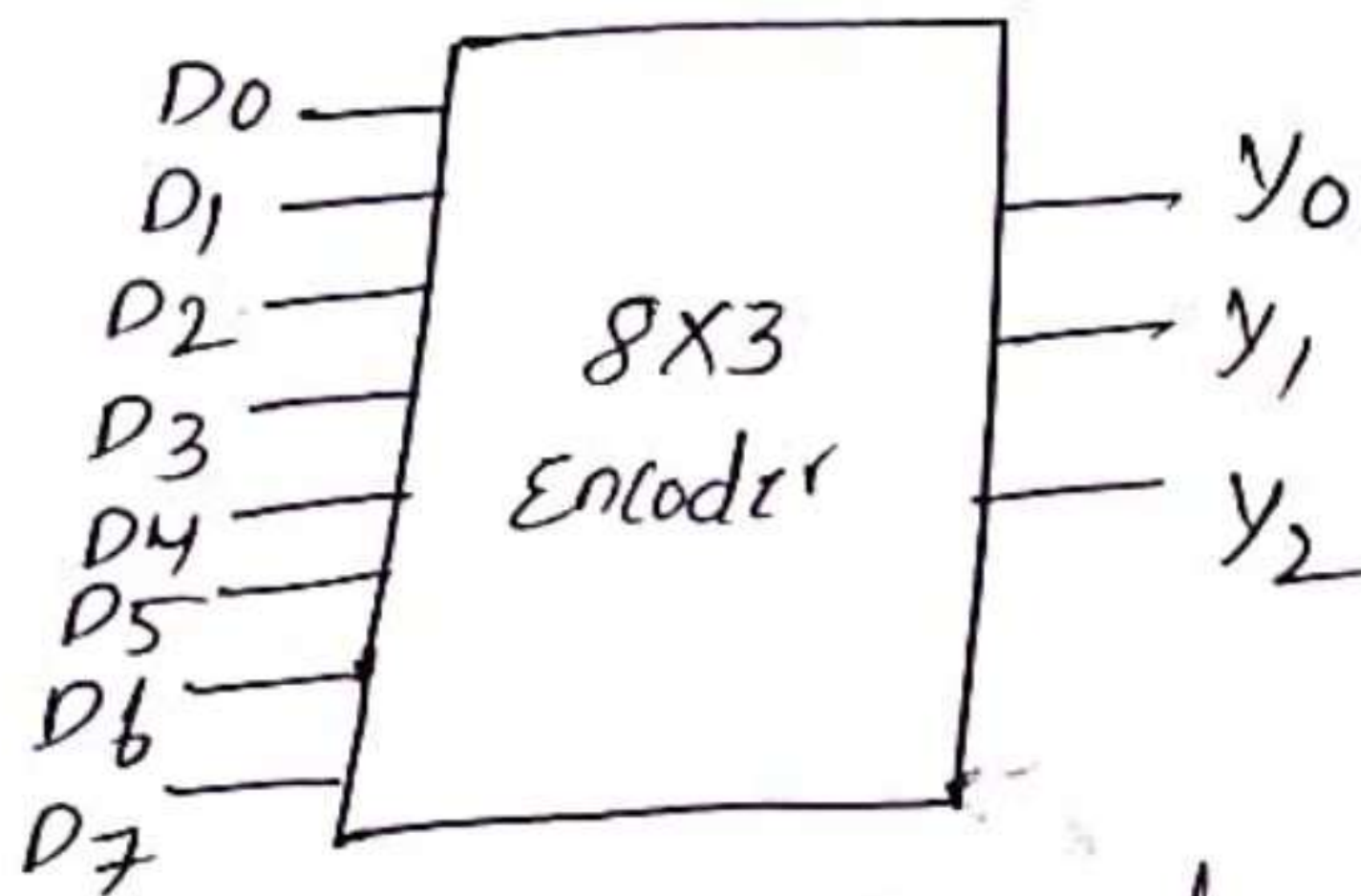


Fig:- logic symbol of 8X3 Encoder

Truth table:-

I/P's								o/p's		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

The o/p's are

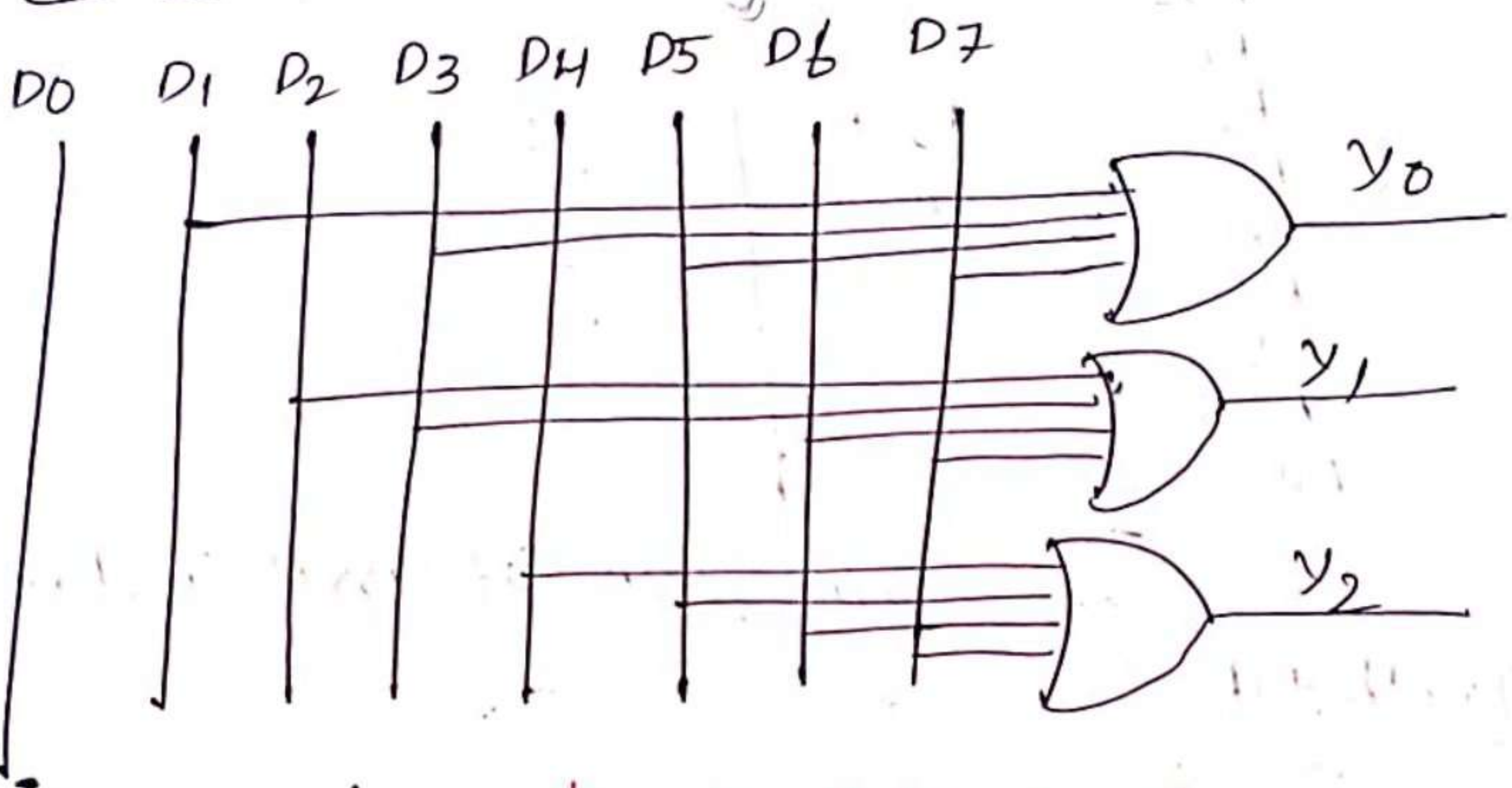
(17)

$$Y_0 = D_1 + D_3 + D_5 + D_7$$

$$Y_1 = D_2 + D_3 + D_6 + D_7$$

$$Y_2 = D_4 + D_5 + D_6 + D_7$$

Logic diagram:-



* Explain 10X4 Encoder or 16X4 Encoder

3. 10X4 Encoder:-

* A 10 to 4 line Encoder is also termed as 10X4 (or) 10:4 Encoder.

* An 10X4 Encoder has ten I/P's or 4 O/P's

* It is also called as Decimal to Binary (or) BCD Encoder.

* It converts Decimal number into Binary number.

* The I/P's are $D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9$. The o/p's are Y_0, Y_1, Y_2, Y_3 .

* The logic symbol of 10×4 is truth-table as shown in fig.

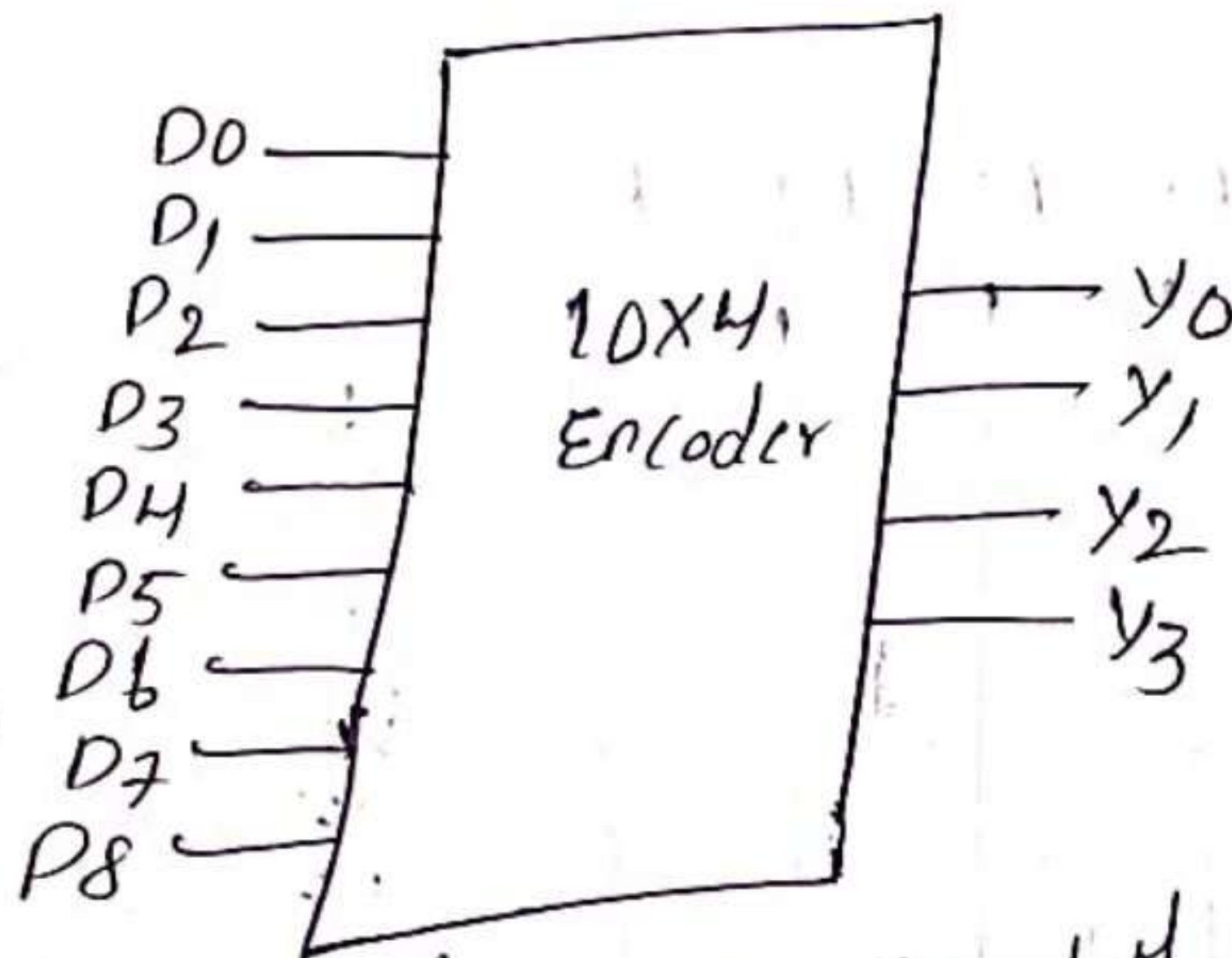


Fig:- logic symbol of 10×4 Encoder.

Truth table:-

D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

∴ The o/p's are

18

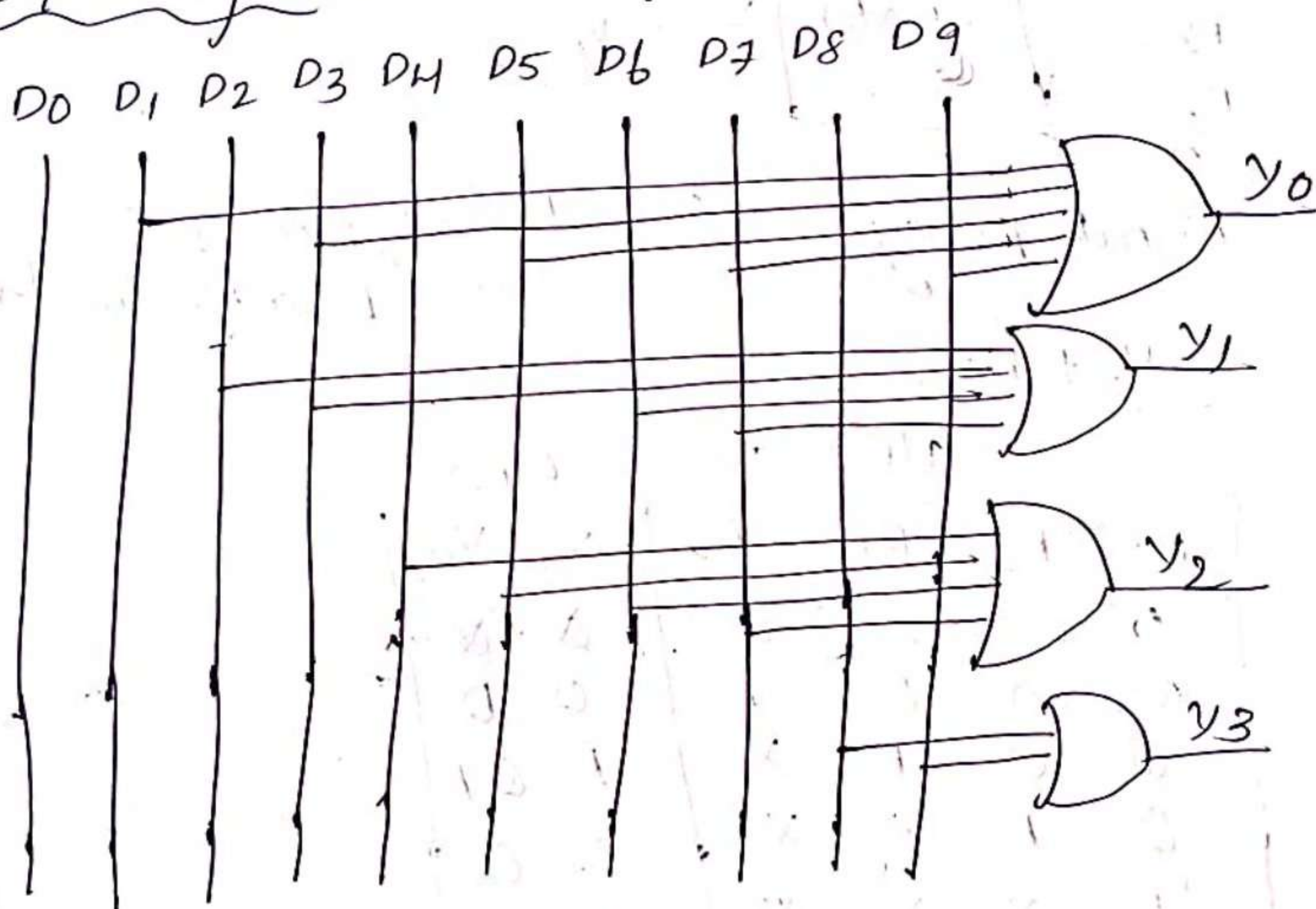
$$Y_0 = D_1 + D_3 + D_5 + D_7 + D_9$$

$$Y_1 = D_2 + D_3 + D_6 + D_7$$

$$Y_2 = D_4 + D_5 + D_6 + D_7$$

$$Y_3 = D_8 + D_9$$

Logic diagram:-



Priority Encoder:-

- * This is special type of Encoder. In this Encoder, priorities are given to the TLP lines.
- * If two (or) more TLP lines are '1' at the same time, then the TLP line with highest

K-Maps of O/P's

K-map for Y_1

D_3D_2	D_1D_0			
	00	01	11	10
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

→ D_2
→ D_3

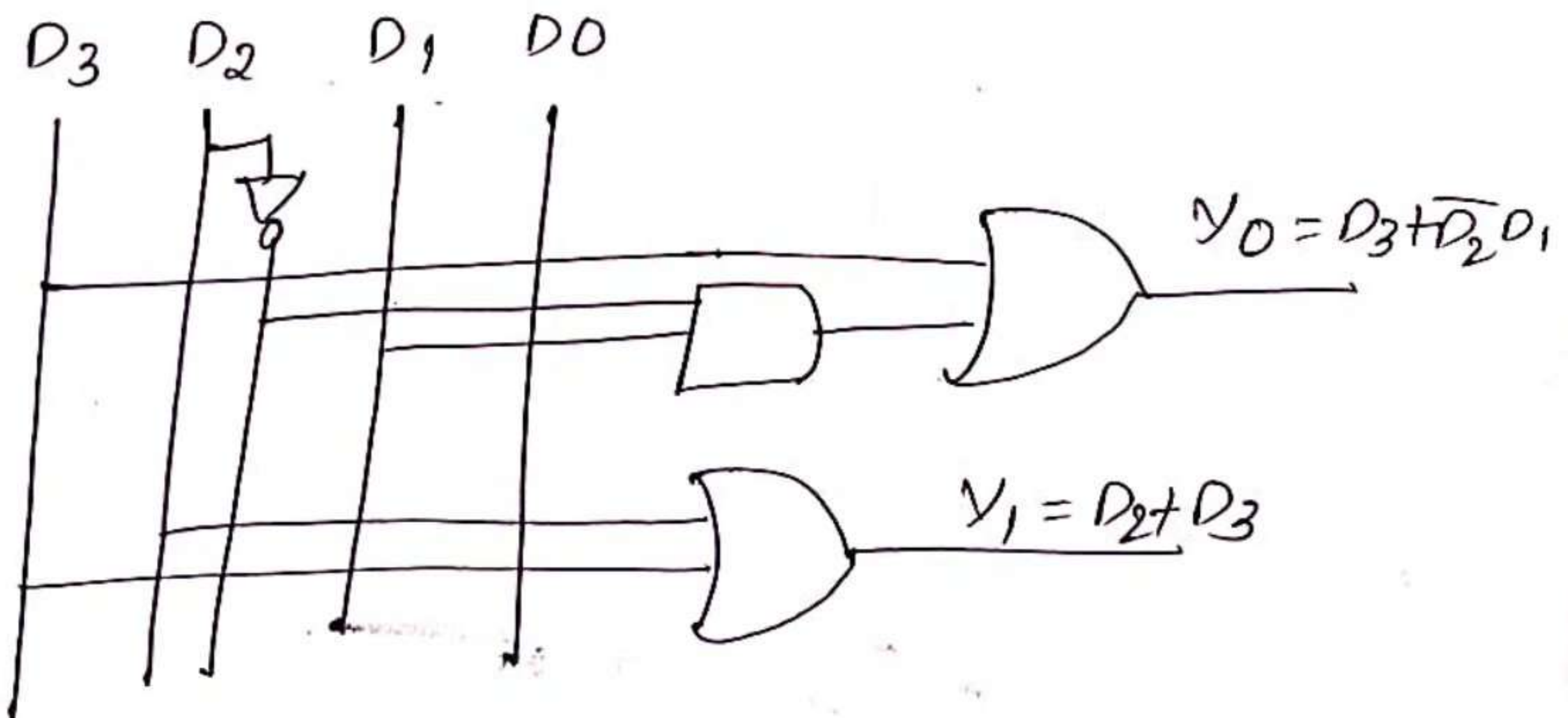
$$\therefore Y_1 = D_2 + D_3$$

K-map for Y_0

D_3D_2	D_1D_0			
	00	01	11	10
00	X	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$$Y_0 = D_3 + \overline{D_2}D_1$$

Logic diagram:-



*Binary multiplier:- (defn)

(20)

A Binary multiplier is a digital circuit used in digital electronics to multiply two binary numbers and provide the result as o/p. The two numbers A_1A_0 and B_1B_0 are multiplied together to produce a 4-bit o/p $P_3P_2P_1P_0$.

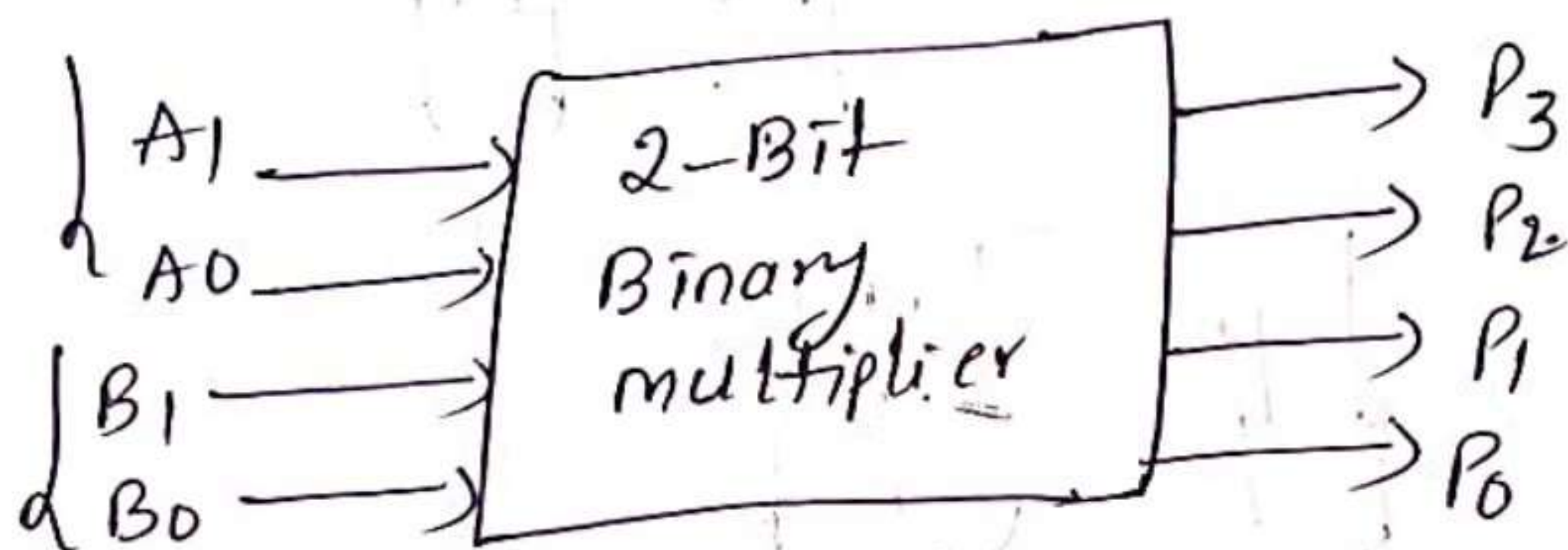


Fig:- 2-bit binary multiplier

2-bit binary multiplier unit implementation:-

Let us implement a two bit binary multiplier. Let the two binary numbers A_1A_0 and B_1B_0 . The multiplication table shown as below

$$\begin{array}{r}
 \begin{array}{cc} A_1 & A_0 \end{array} \\
 \times \begin{array}{cc} B_1 & B_0 \end{array} \\
 \hline
 \begin{array}{cc} B_0A_1 & B_0A_0 \end{array} \\
 \begin{array}{cc} B_1A_1 & B_1A_0 \end{array} \\
 \hline
 \begin{array}{ccc} & & \\ \text{6} & \text{6} & \text{6} \end{array}
 \end{array}$$

∴ Therefore.

$$C_0 = A_0 * B_0$$

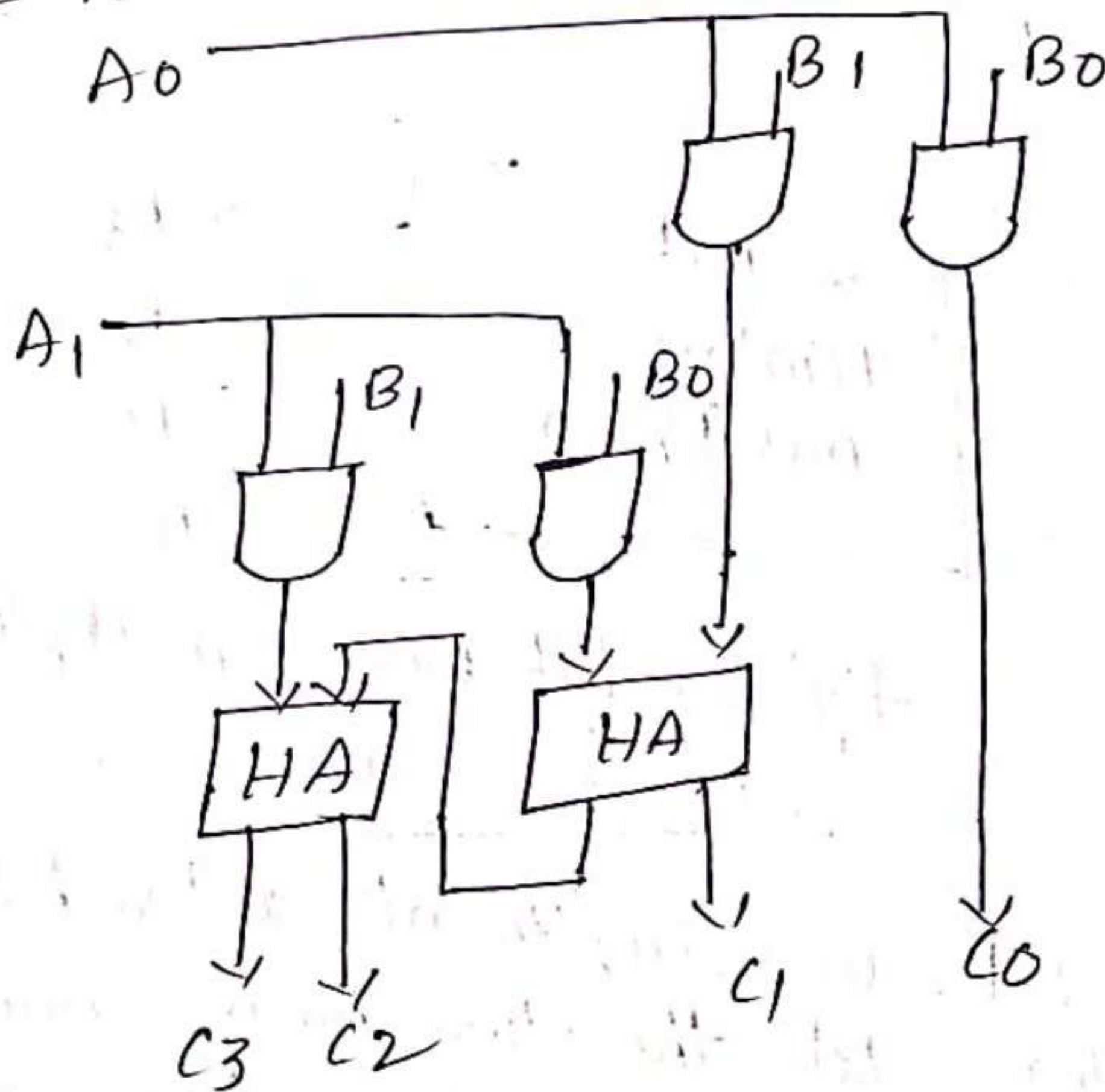
$$C_1 = A_0 * B_1 \oplus A_1 * B_0$$

$$C_2 = A_1 * B_1 \oplus (A_0 * B_1) * (A_1 * B_0)$$

$$C_3 = A_1 * B_1 \text{ and } (A_0 * B_1) * (A_1 * B_0)$$

∴ (carry generated here goes to next stage)

Logic diagram:-



A 4-bit by 3-bit multiplier:-

* Let us consider a multiplier circuit that multiplies a binary number of 4-bits by a number of 3-bits. Let the multiplicand be represented by $B_3 B_2 B_1 B_0$ and the multiplier by $A_2 A_1 A_0$.

Logic diagram:-

(21)

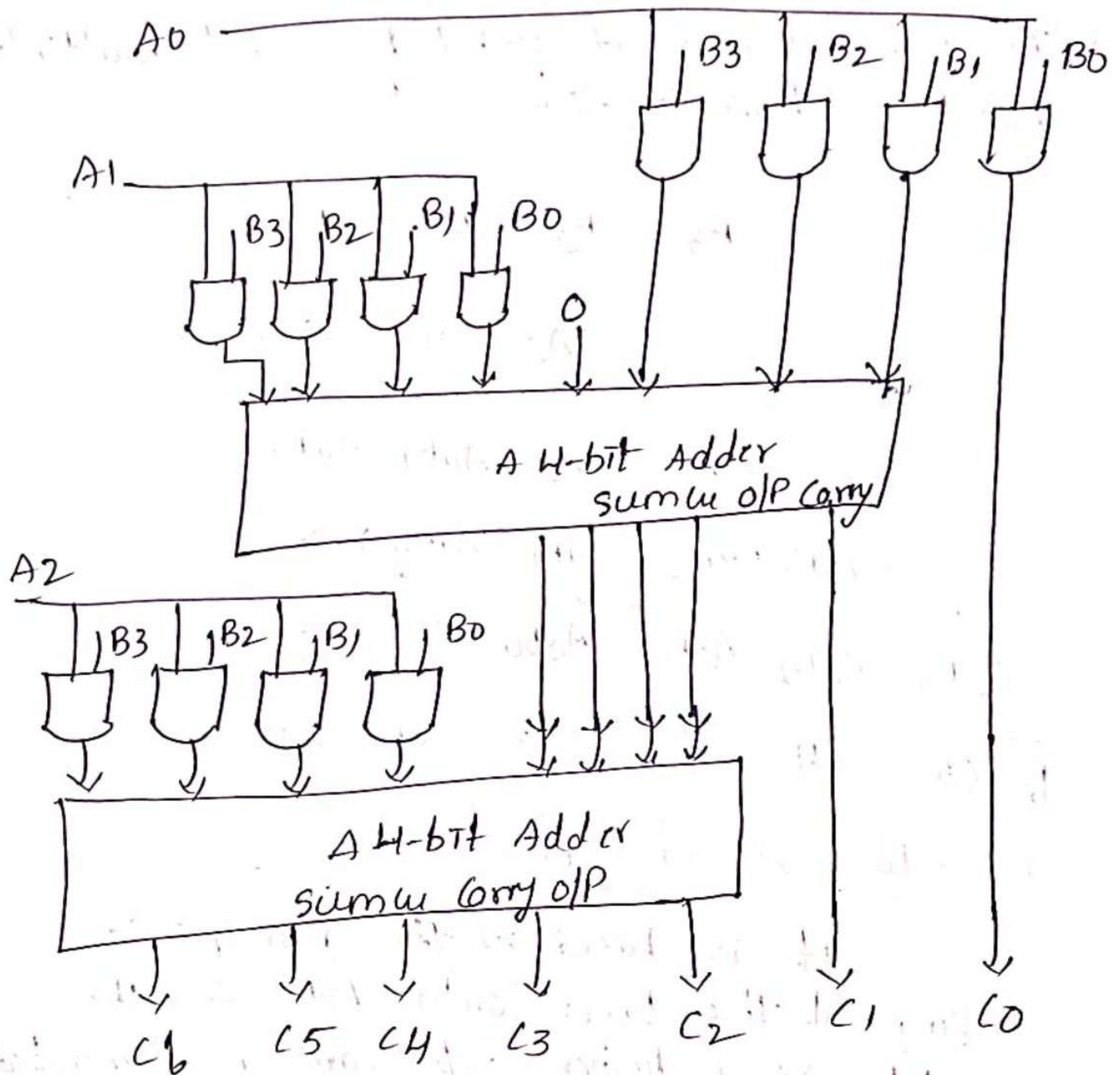


fig:- A 4-bit by 3-bit multiplier

We Required 12 AND gates and two 4-bit adders to produce a product of seven bit. (21)

* The logic diagram of 4-bit by 3-bit multiplier has been shown in fig.

$$\begin{array}{r}
 \begin{array}{cccc}
 B_3 & B_2 & B_1 & B_0 \\
 & A_2 & A_1 & A_0 \\
 \hline
 A_0 B_3 & A_0 B_2 & A_0 B_1 & A_0 B_0 \\
 A_1 B_3 & A_1 B_2 & A_1 B_1 & A_1 B_0 & X \\
 A_2 B_3 & A_2 B_2 & A_2 B_1 & A_2 B_0 & X & X \\
 \hline
 C_6 & C_5 & C_4 & C_3 & C_2 & C_1 & C_0
 \end{array}
 \end{array}$$

Carry-Look Ahead Adder:-

It is based on the principle of looking at the lower order bits of the inputs. If a higher order carry is generated.

* Two functions

* Carry propagation (Pi)

* Carry Generation (Gi)

(1)

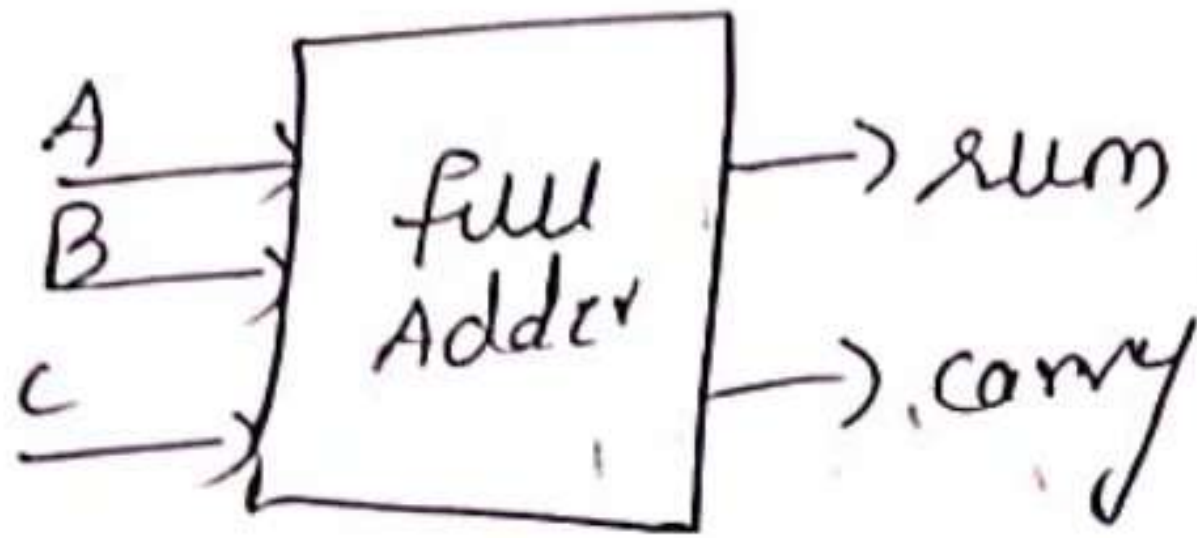


fig:- Logic symbol

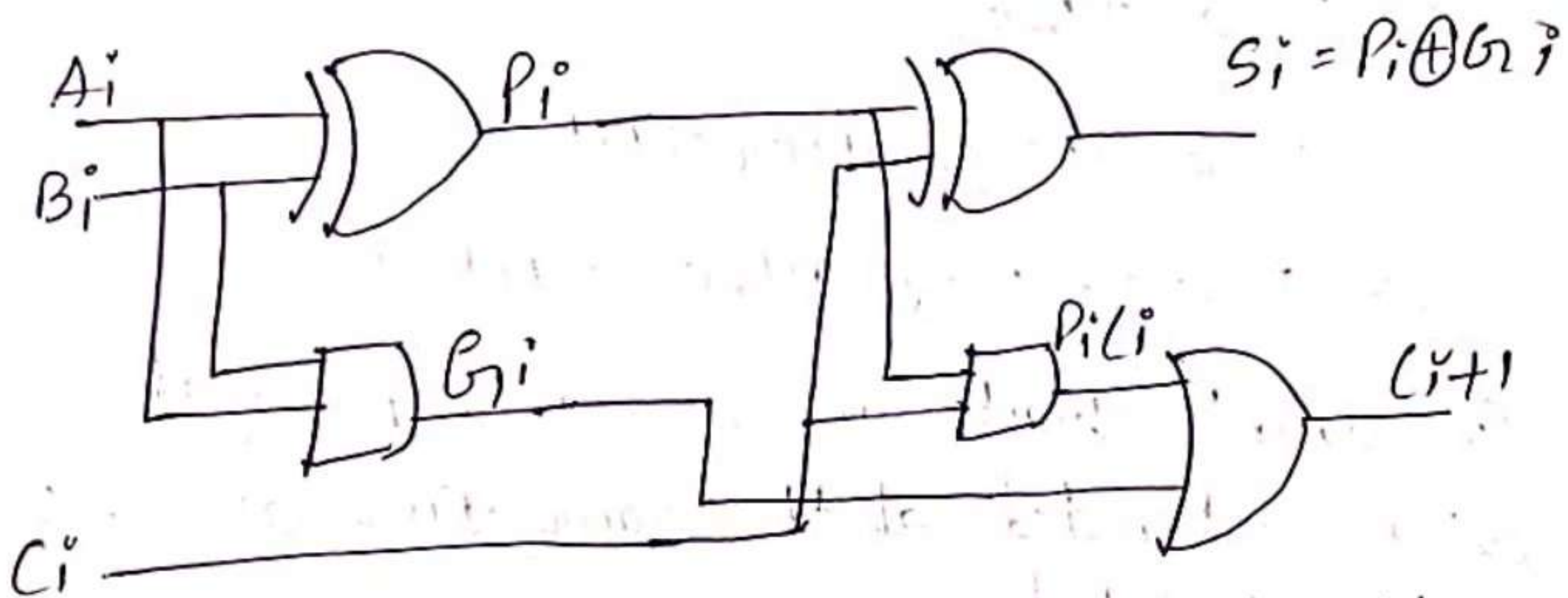


fig:- Realization of full adder

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

$$\text{sum}:- S_i = P_i \oplus C_i$$

$$\text{Carry}:- C_{i+1} = G_i + P_i C_i$$

* $G_i \rightarrow$ Carry generate $\rightarrow$ It produces carry
 $P_i \rightarrow$ Carry propagate $\rightarrow$ propagation of the carry from C_i to C_{i+1}

∴ The carry off of each stage :-

$C_0 \rightarrow$ the TLP carry

$$C_1 \rightarrow G_0 + P_0 C_0 \quad [C_{i+1} = G_i + P_i C_i]$$

$$C_2 \rightarrow G_1 + P_1 C_1 \Rightarrow G_1 + P_1 [G_0 + P_0 C_0] \\ = G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2$$

$$= G_2 + P_2 [G_1 + P_1 G_0 + P_1 P_0 C_0]$$

$$C_3 = G_2 + P_2 G_1 + P_1 P_2 G_0 + P_1 P_2 P_0 C_0$$

* C_3 doesn't wait for C_2 & C_1 to propagate.

* C_3 is estimated at the same time as C_2 & C_1 are estimated.

* The sum off of each stage.

$$S_0 = P_0 \oplus C_0 \Rightarrow A_0 \oplus B_0 \oplus C_0$$

$$S_1 = P_1 \oplus C_1 \Rightarrow A_1 \oplus B_1 \oplus C_1$$

$$S_2 = P_2 \oplus C_2 \Rightarrow A_2 \oplus B_2 \oplus C_2$$

$$S_3 = P_3 \oplus C_3 \Rightarrow A_3 \oplus B_3 \oplus C_3$$

Logic diagram :-

23

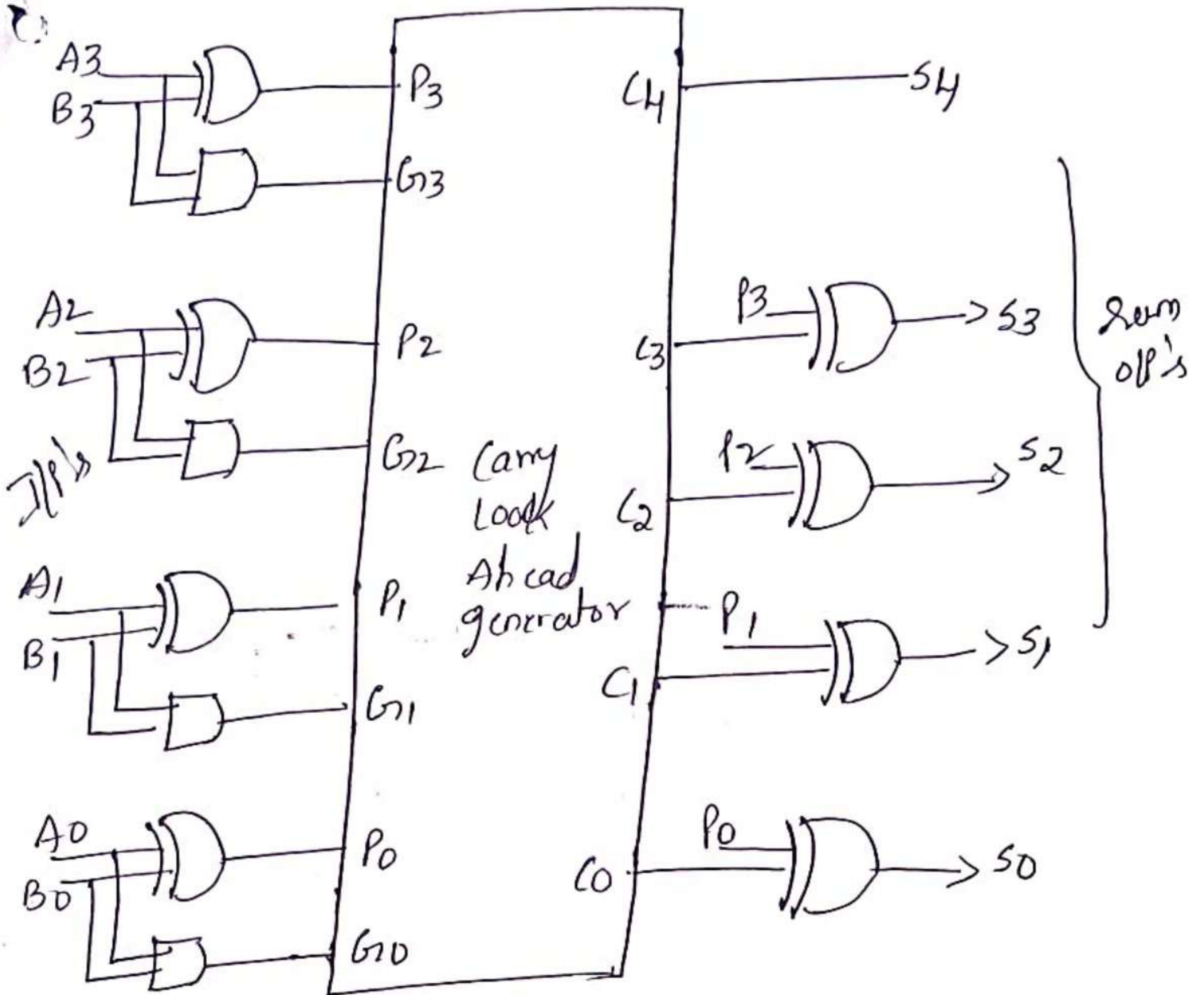


Fig.:- 4-bit Parallel Adder

UNIT-III: HARDWARE DESCRIPTION LANGUAGE

Introduction to Verilog - structural specification of logic circuits, behavioral specification of logic circuits, hierarchical Verilog Code, Verilog for combinational circuits - conditional operator, if-else statement, case statement, for loop using storage elements with CAD tools-using Verilog constructs for storage elements, flip-flop with clear capability, using Verilog constructs for registers and counters.

INTRODUCTION TO VERILOG:

In the 1980s rapid advances in integrated circuit technology lead to efforts to develop standard design practices for digital circuits. Verilog was produced as a part of that effort. The original version of Verilog was developed by Gateway Design Automation, which was later acquired by Cadence Design System.

In 1990 Verilog was put into the public domain, and it has since become the most popular language for describing digital circuits. In 1995 Verilog was adopted as an official IEEE Standard, called 1364-1995. An enhanced version of Verilog, called Verilog 2001, was adopted as IEEE Standard 1364-2001 in 2001.

Verilog was originally intended for simulation and verification of digital circuits. Subsequently, with the addition of synthesis capability, Verilog has also become popular for use in design entry in CAD systems. The CAD tools are used to synthesize the Verilog code into a hardware implementation of the described circuit. Verilog is a complex, sophisticated language. Learning all of its features is a daunting task. However, for use in synthesis only a subset of these features is important. Verilog has a number of constructs similar to the C programming language. The one who knows C will find Verilog easy to learn.

STRUCTURAL SPECIFICATION OF LOGIC CIRCUITS:

Verilog includes a set of gate level primitives that correspond to commonly used logic gates. A gate is represented by indicating its functional name, output, and inputs.

For example, a two-input AND gate, with inputs x1 and x2 and output y, is denoted as

and (y, x1, x2);

A four-input OR gate is specified as

or (y, x1, x2, x3, x4);

The keywords **nand** and **nor** are used to define the NAND and NOR gates in the same way.

The NOT gate given by

not (y, x);

implements $y = \sim x$. The gate level primitives can be used to specify larger circuits. A logic circuit is specified in the form of a **module** that contains the statements that define the circuit. The **module** may have inputs and outputs, which are referred to as its ports. The name port is a commonly used term that refers to an input or output connection to an electronic circuit. Consider the circuit

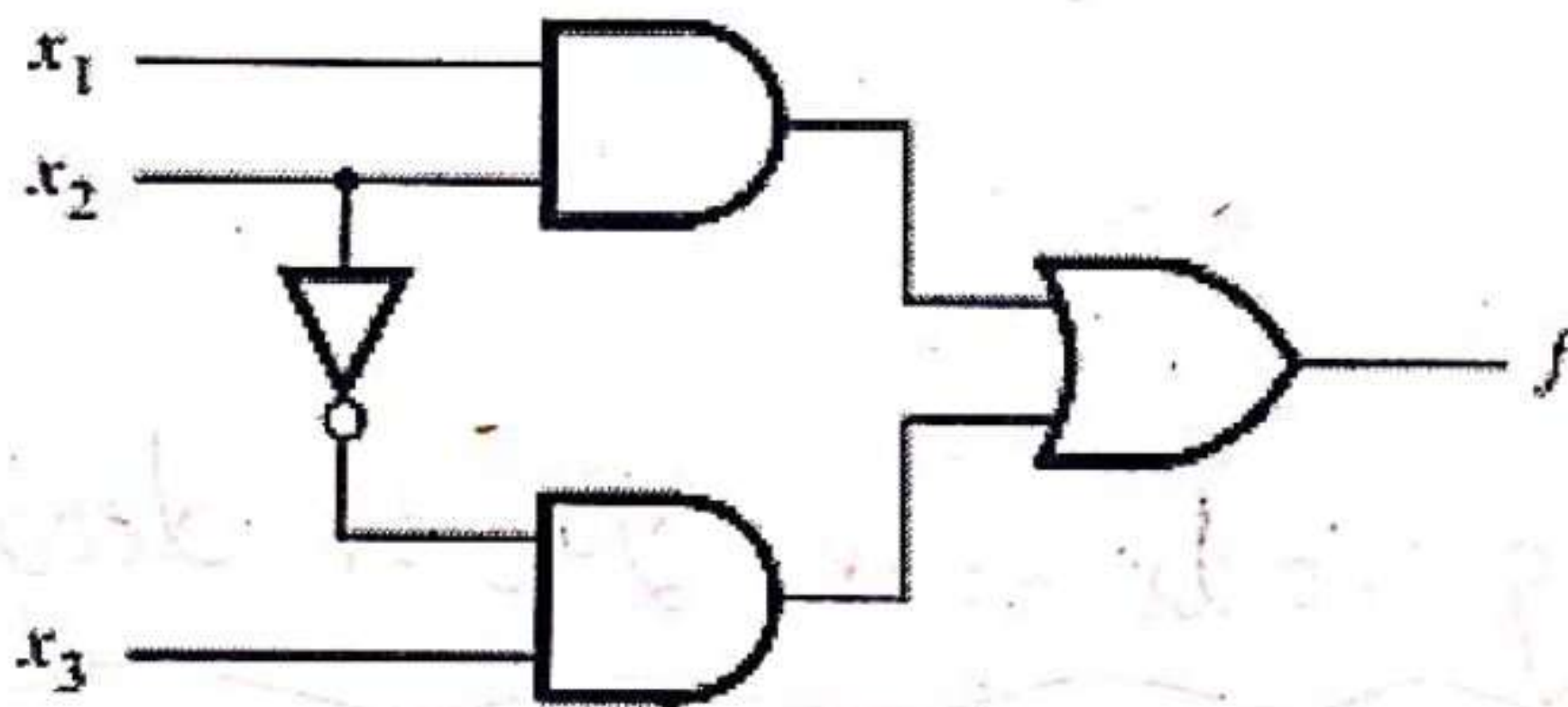


Fig 5.1: Simple logic circuit

The first statement gives the module a name, Simple logic circuit, and indicates that there are four port signals. The next two statements declare that x_1 , x_2 , and x_3 are to be treated as input signals, while f is the output. The actual structure of the circuit is specified in the four statements that follow. The NOT gate gives $k = \sim x_2$.

The AND gates produce $g = x_1 x_2$ and $h = \sim x_2 x_3$.

The outputs of AND gates are combined in the OR gate to form $f = g + h = x_1 x_2 + \sim x_2 x_3$

The module ends with the endmodule statement.

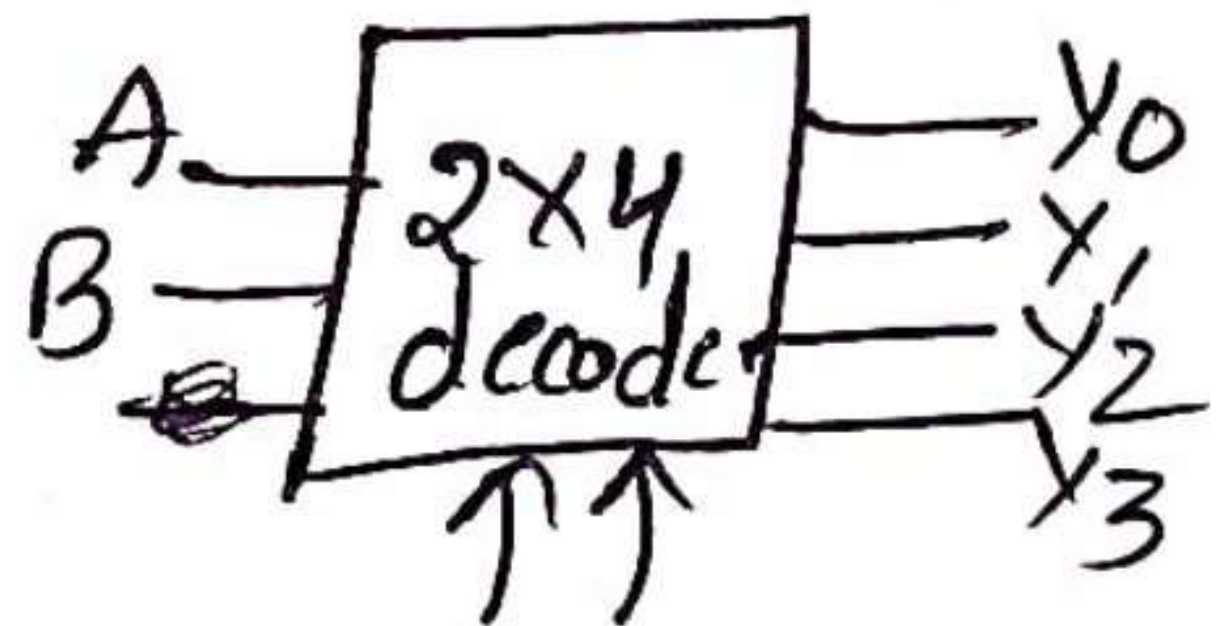
Verilog Syntax:

The names of modules and signals in Verilog code follow two simple rules: The name must start with a letter and it can contain any letter or number plus the “_” underscore and \$ characters. Verilog is case sensitive. Thus, the name k is not the same as K and Example1 is not the same as example1. The Verilog syntax does not enforce a particular style of code. For example, multiple statements can appear on a single line. White space characters, such as SPACE and TAB, and blank lines are ignored. As a matter of good style, code should be

Verilog code for Fig 5.1:

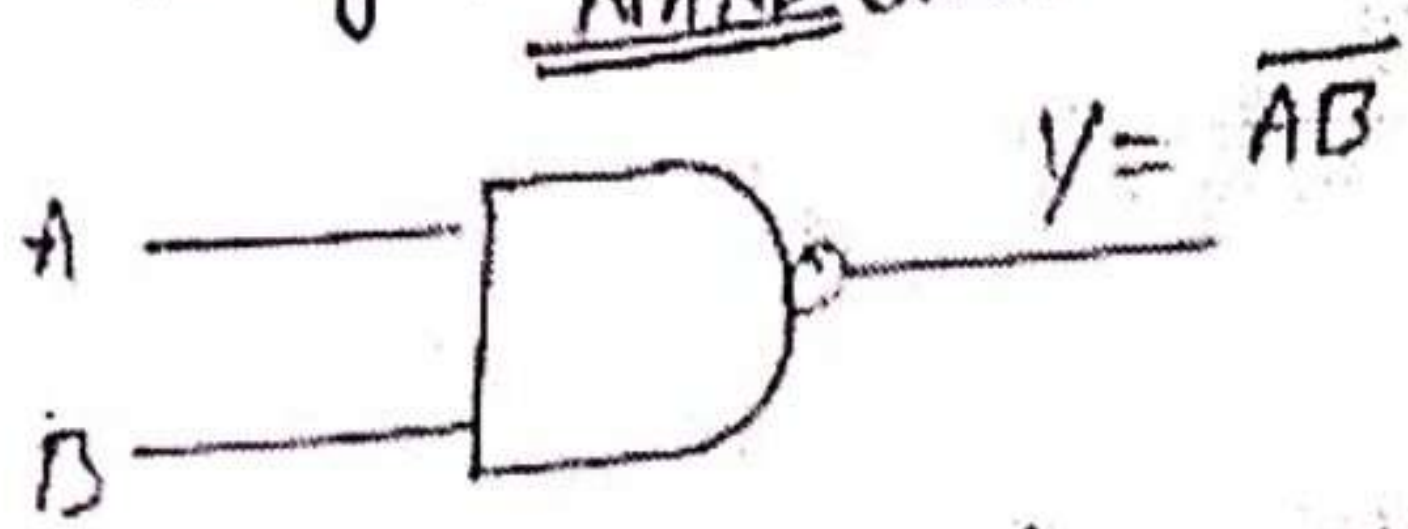
```
module Simple logic circuit (x1, x2, x3, f);  
    input x1, x2, x3;  
    output f; and (g, x1, x2);  
    not (k, x2);  
    and (h, k, x3);  
    or (f, g, h);  
endmodule
```

Verilog code for 2 to 4 decoder:-

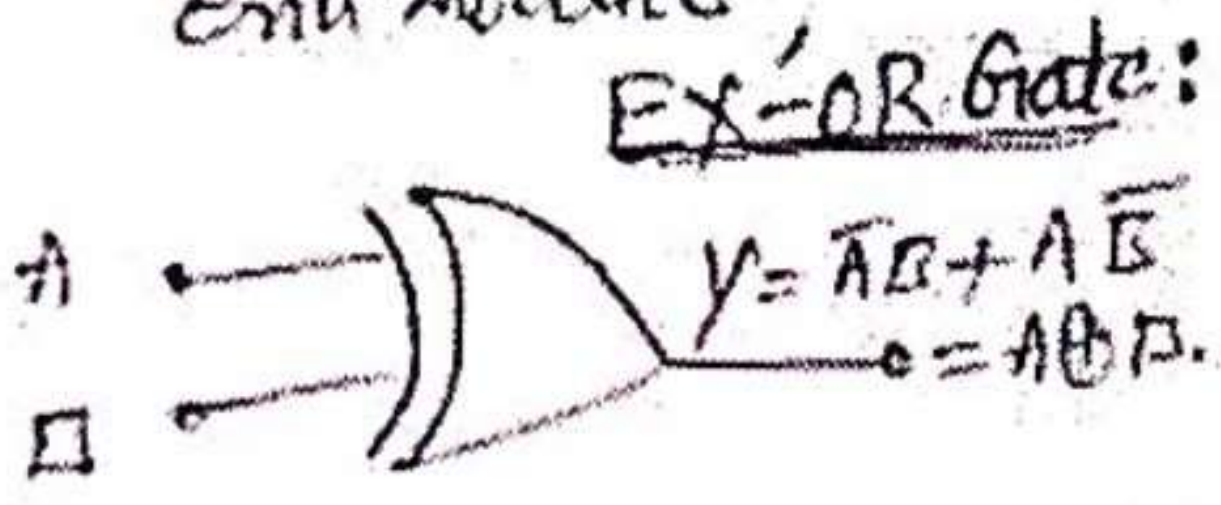


endmodule

→ Describe universal gates & complexable logic gates by using structural modeling.

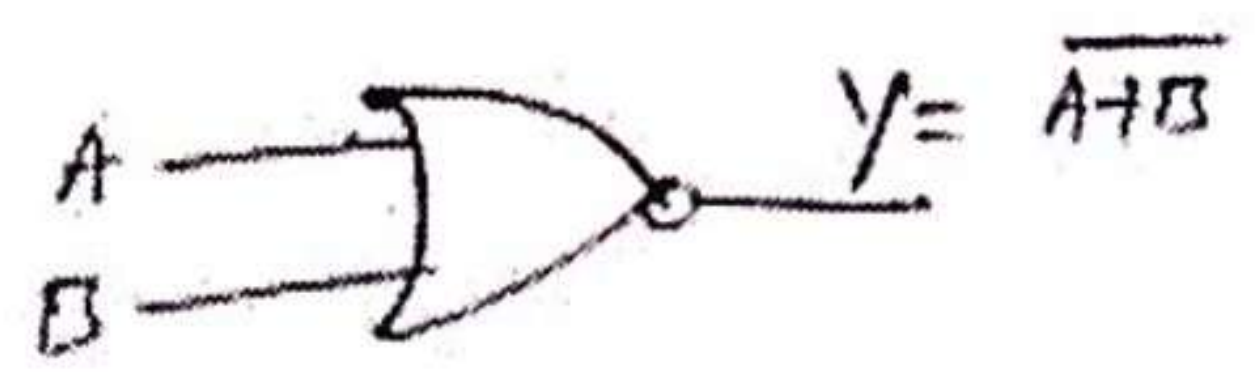


```
module NAND-ex(A, B, Y);
    input A, B;
    output Y;
    nand NI(A, B, Y);
endmodule;
```



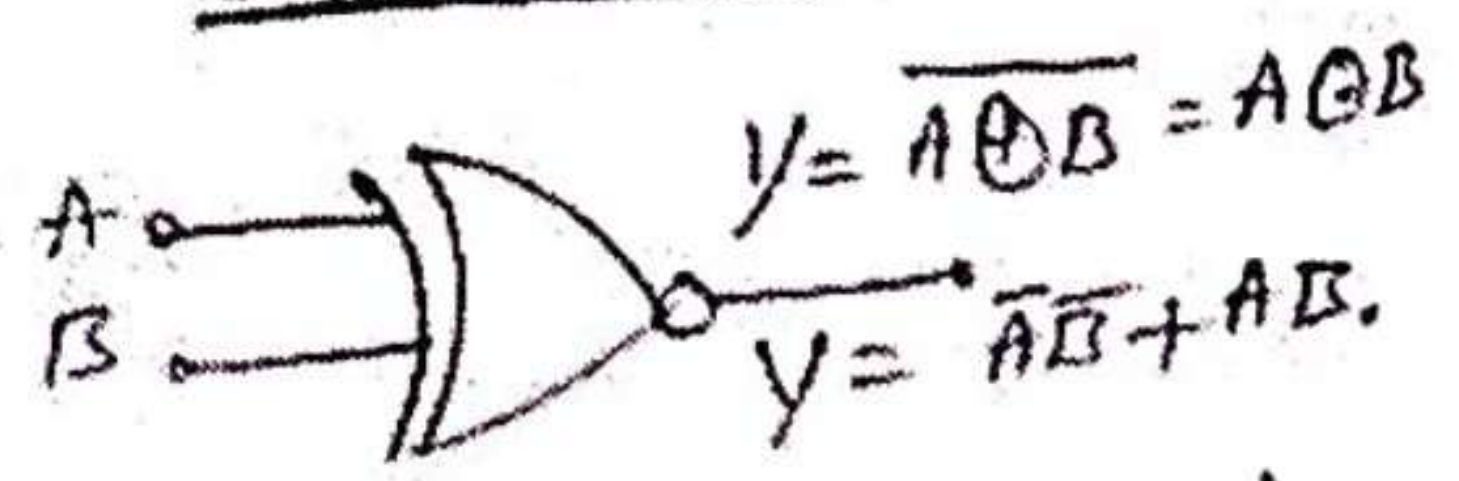
```
module x-or-ex(A, B, Y);
    input A, B;
    output Y;
    x-or(Y, A, B);
endmodule;
```

NOR Gate



```
module nor-ex(A, B, Y);
    input A, B;
    output Y;
    nor NI(Y, A, B);
endmodule.
```

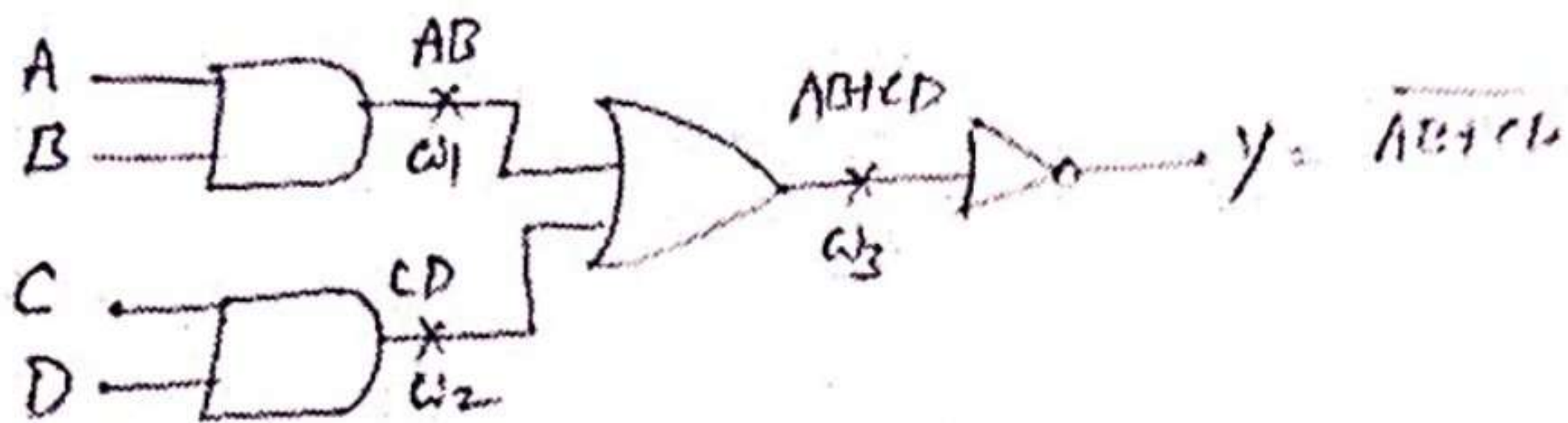
EX-NOR Gate:



```
module x-nor-ex(A, B, Y);
    input A, B;
    output Y;
    x-nor(Y, A, B);
endmodule
```


Structural modeling

AND-OR-INVERTER

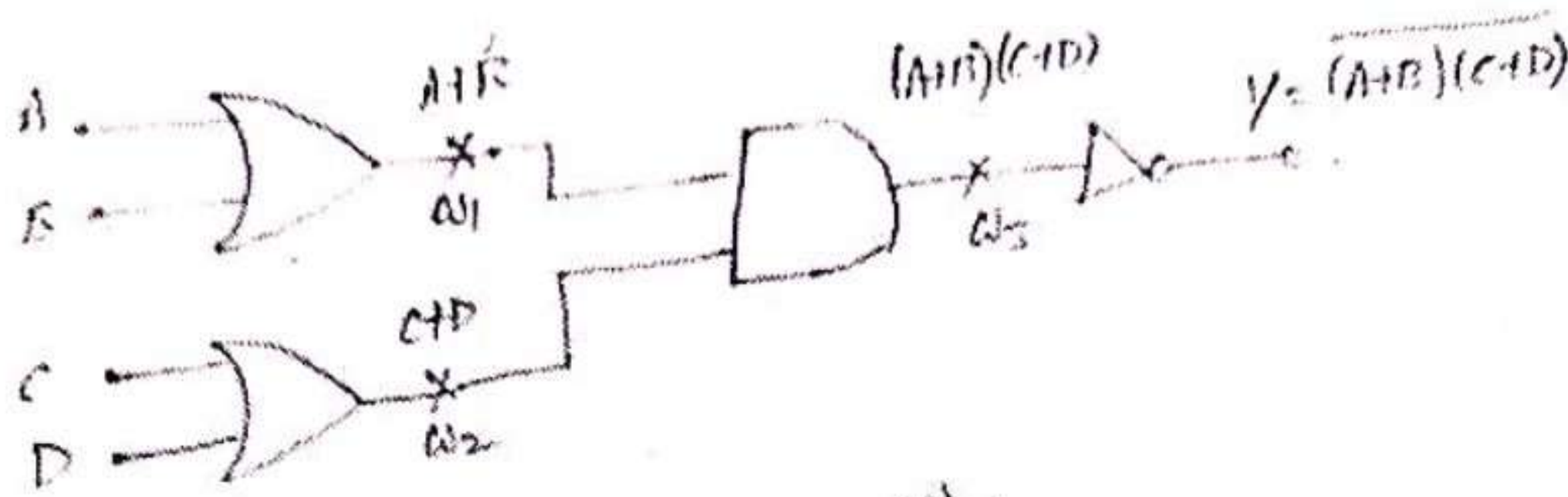


Fig(1): AND-OR-INVERTER.

```
module and-or-inv(A, B, C, D, Y);
    input A, B, C, D;
    output Y;

    wire w1, w2, w3;
    and A1(w1, A, B);
    and A2(w2, C, D);
    or O1(w3, w1, w2);
    not N1(Y, w3);
endmodule
```


OR-AND INVERTER:-

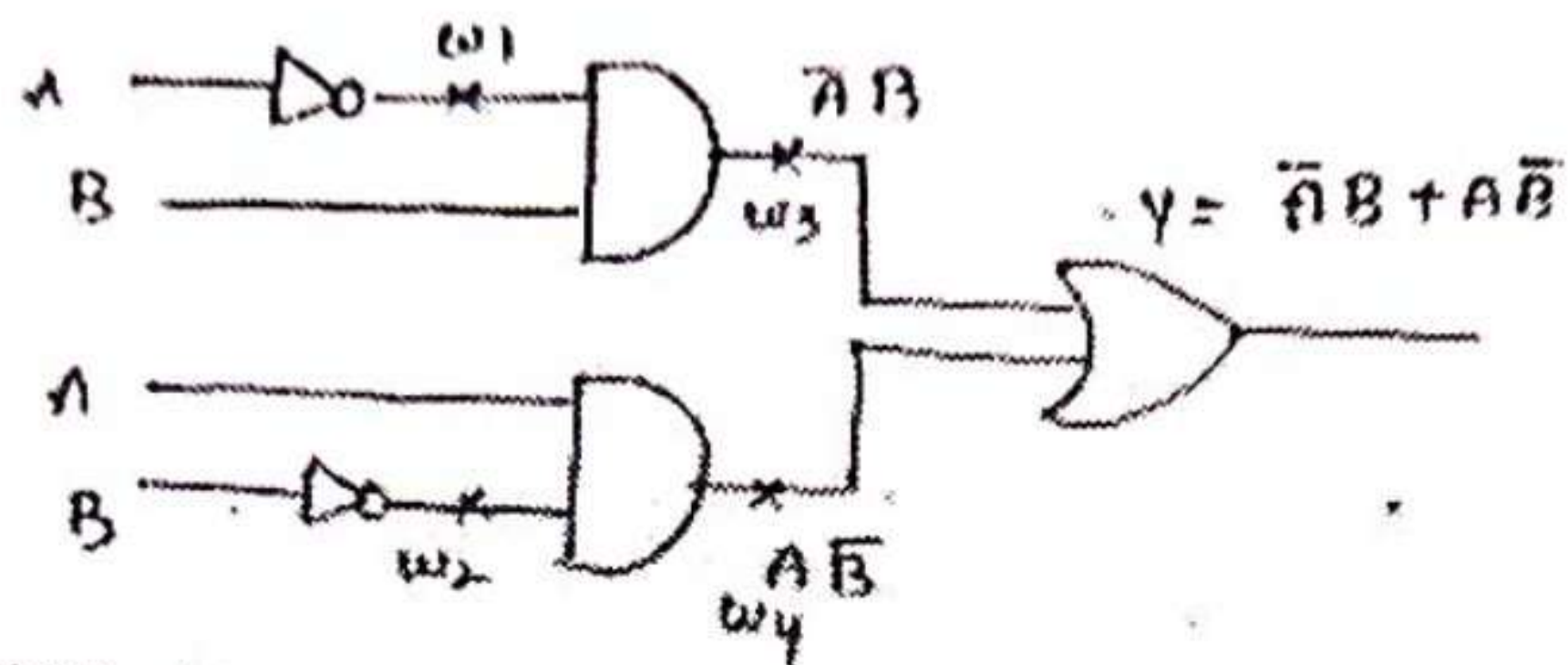


```
module or-and-inv(A,B,C,D,Y);  
    input A,B,C,D;  
    output Y;  
    wire w1,w2,w3;  
    A1 O1(w1,A,B);  
    A2 O2(w2,C,D);  
    And A1(w3,w1,w2);  
    not N1(Y,w3);  
endmodule
```

Example Verilog Structural modeling program for AND-OR-INVERTER

```
module and_or_inv(A,B,C,D,Y);  
    input A,B,C,D;  
    output Y;  
    wire w1,w2,w3;  
    and A1(w1,A,B); // AND gate instance A1 for A and B  
    and A2(w2,C,D); // AND gate instance A2 for C and D  
    or O1(w3,w1,w2); // OR gate instance O1 for the outputs of A1 and A2  
    not N1(Y,w3); // NOT gate instance N1 for the output of O1  
endmodule
```


$Y = \bar{A}B + A\bar{B}$. Describe given boolean function by using verilog HDL in structural Modelling.



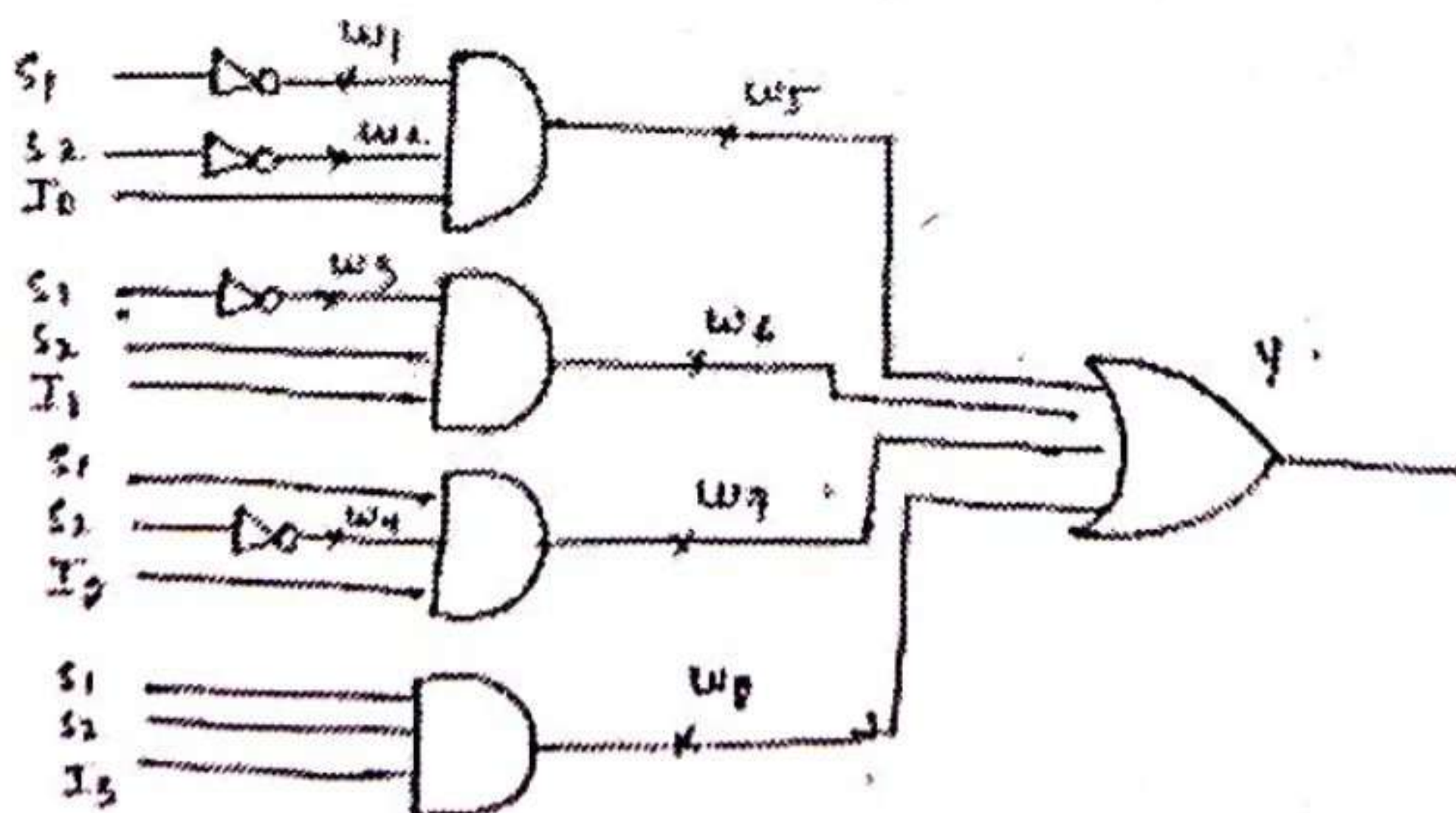
```

module AND OR (A,B,Y);
  input A,B;
  output Y;
  wire w1,w2,w3,w4;
  NOT N1(w1,A);
  NOT N2(w2,B);
  AND A1(w3,B,w1);
  AND A2(w4,A,w2);
  OR O1(Y,w3,w4);
end module

```

Describe 4-to-1 multiplexer by using Verilog HDL in Structural Modelling.

$$Y = \bar{S}_1 \bar{S}_2 I_0 + \bar{S}_1 S_2 I_1 + S_1 \bar{S}_2 I_2 + S_1 S_2 I_3$$




```

module AND_OP (s1, s2, J1, J2, J3, Y);
    input (s1, s2, J1, J2, J3);
    output Y;
    wire w1, w2, w3, w4, w5, w6, w7, w8;

    NOT N1 (w1, s1);
    NOT N2 (w2, s2);
    NOT N3 (w3, s1);
    NOT N4 (w4, s2);

    AND A1 (w1, w2, J1, w5);
    AND A2 (w1, w3, s1, J1);
    AND A3 (w7, s1, w4, J2);
    AND A4 (w8, s1, s2, J3);

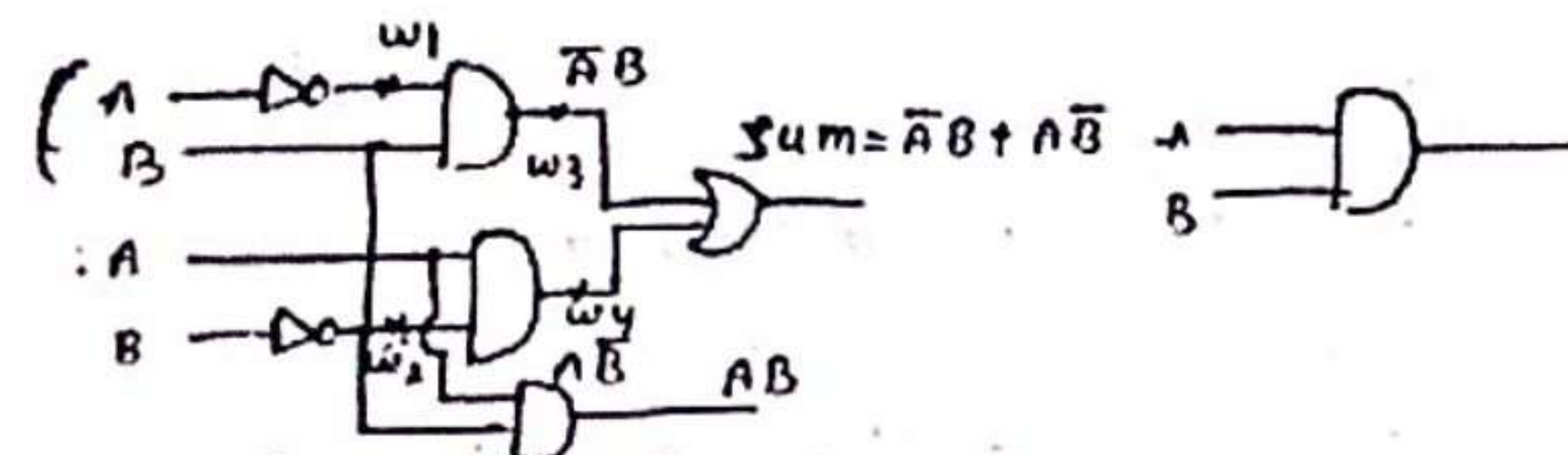
    OR o1 (w5, w6, w7, w8, Y);
end module

```

Display half adder for structural modeling.

$$\text{Sum} = \bar{A}B + A\bar{B}$$

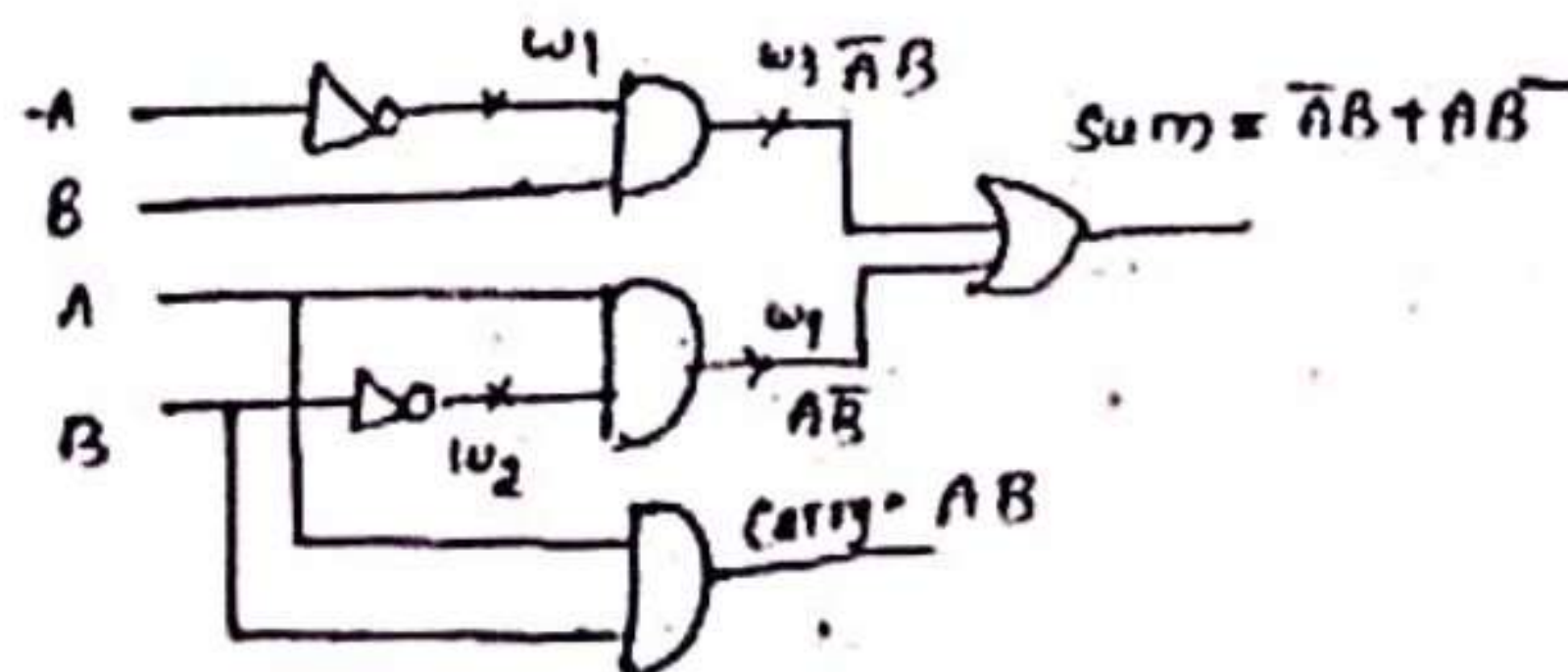
$$\text{Carry} = AB$$



```

module sum (A, B, sum);
    input A, B;

```



```

module HalfAdder (A, B, sum, carry);
    input A, B;
    output sum, carry;
    wire w1, w2, w3, w4;

    NOT N1 (w1, A);
    NOT N2 (w2, B);
    AND A1 (w3, w1, B);

```

```

    AND A2 (w4, A, w2);
    OR o1 (w3, w4, sum);
    AND A3 (A, B, carry);
end module

```


BEHAVIORAL SPECIFICATION OF LOGIC CIRCUITS

Using gate level primitives can be tedious when large circuits have to be designed. An alternative is to use more abstract expressions and programming constructs to describe the behavior of a digital circuit. One possibility is to define the circuit using logic expressions. Figure 5.1 can be defined with the expression $f = x_1x_2 + \sim x_2x_3$. The AND and OR operations are indicated by the "&" and "|" signs, respectively. The assign keyword provides a continuous assignment for the signal f. The word continuous stems from the use of Verilog for simulation; whenever any signal on the right-hand side changes its state, the value of f will be re-evaluated. Using logic expressions makes it easier to write Verilog code. The circuit can be described in words by saying that $f = x_1$ if $x_2 = 1$ and $f = x_3$ if $x_2 = 0$.

Note: In this behavioral modeling, the Verilog code is developed based on behavior of the digital logic circuit. The Verilog code is written by considering the truth table of the circuit and minimized Boolean expression of the output function.

Verilog code:

```
module Simple logic circuit (x1, x2, x3, f);  
    input x1, x2, x3;  
    output f;  
    assign f = (x1 & x2) | (~x2 & x3);  
endmodule
```

Describe all logic gates by using Verilog behavioral modeling.

AND Gate:

```
module AND_ex (A, B, Y);  
    input A, B;  
    output Y;  
    assign Y = A & B; // Perform AND operation on A and B  
endmodule
```


OR Gate:

```
module OR_ex (A, B, Y);  
input A, B;  
output Y;  
assign Y = A | B; // Perform OR operation on A and B  
endmodule
```

NOT Gate:

```
module NOT_ex (A, Y);  
input A;  
output Y;  
assign Y = ~A ; // Perform NOT operation on A and B  
  
endmodule
```

NAND Gate:

```
module NAND_ex (A, B, Y);  
input A, B;  
output Y;  
assign Y = ~(A & B); // Perform NAND operation on A and B  
  
endmodule
```

NOR Gate:

```
module NOR_ex (A, B, Y);  
input A, B;  
output Y;  
assign Y = ~(A | B); // Perform NOR operation on A and B  
  
endmodule
```


EX-OR Gate:

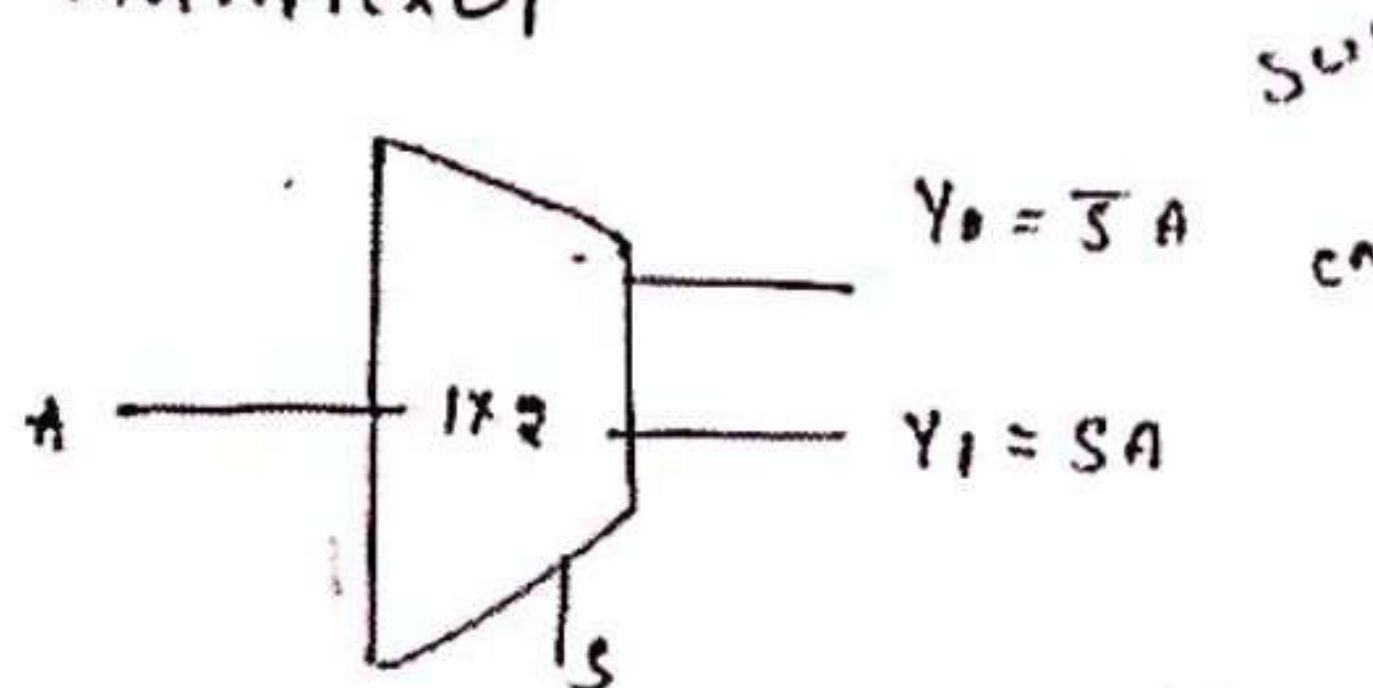
```
module X-OR_ex (A, B, Y);  
input A, B;  
output Y;  
assign Y = A ^ B; // Perform EX-OR operation on A and B  
  
endmodule
```

EX-NOR Gate:

```
module X-NOR_ex (A, B, Y);  
input A, B;  
output Y;  
assign Y = ~(A ^ B); // Perform EX-NOR operation on A and B  
  
endmodule
```

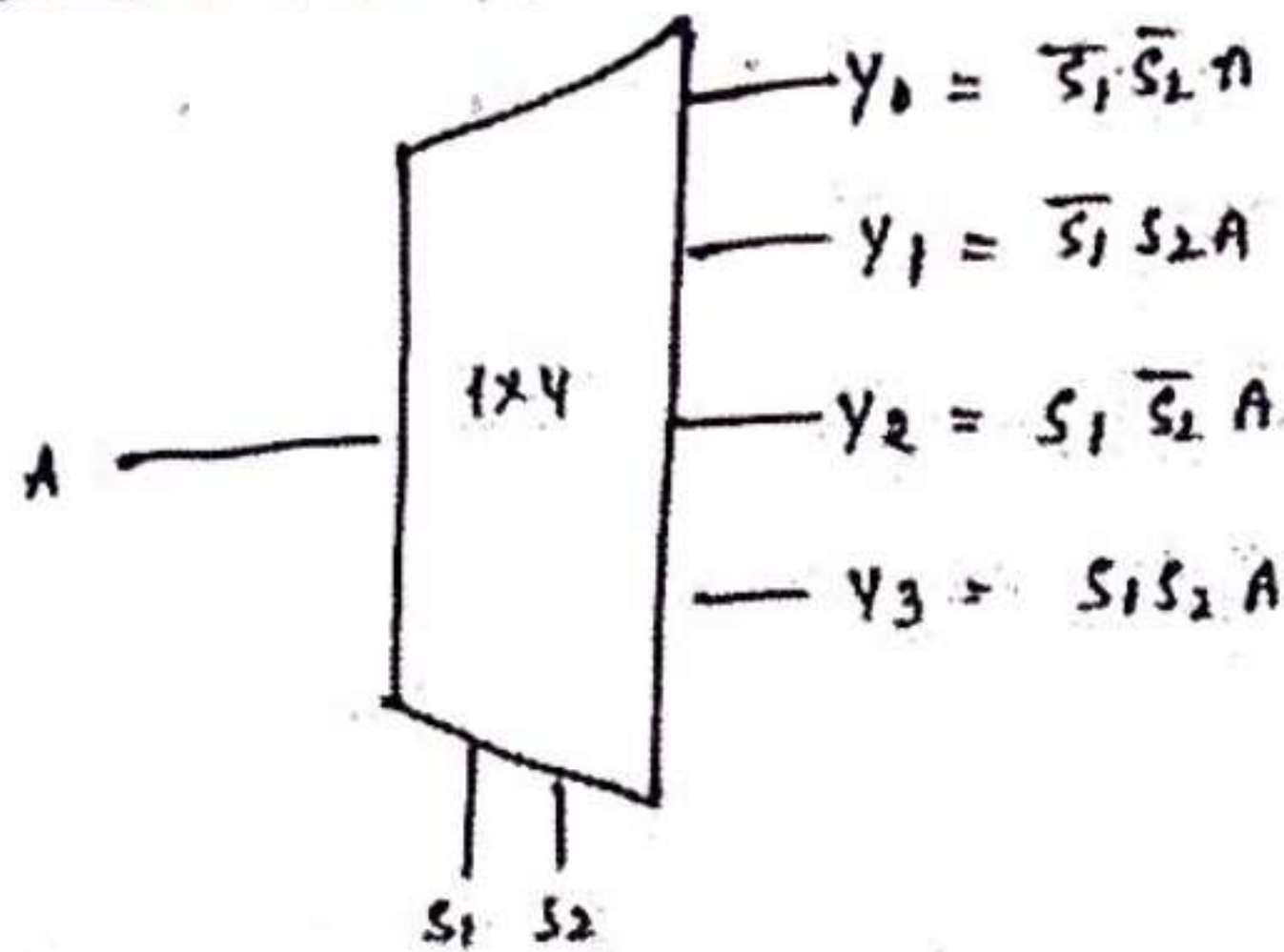
De multiplexer.

1x2 De multiplexer



```
module De muxer (A, S, Y0, Y1);  
input (A, S);  
output (Y0, Y1);  
assign Y0 = (~S & A);  
assign Y1 = (S & A);  
end module
```


1x4 Demultiplexer



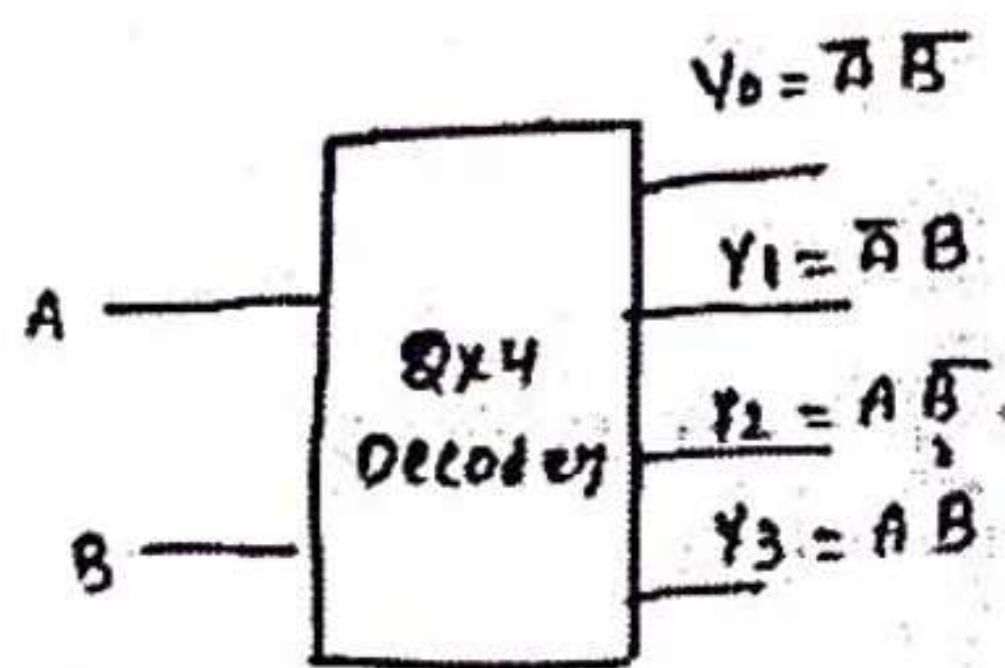
```

module Demuxer (A, S1, S2,
                Y0, Y1, Y2, Y3);
    input (A, S1, S2);
    output (Y0, Y1, Y2, Y3);
    assign Y0 = (~S1 & ~S2 & A);
    assign Y1 = (~S1 & S2 & A);
    assign Y2 = (S1 & ~S2 & A);
    assign Y3 = (S1 & S2 & A);
end module.

```

Decoder

2x4 Decoder



```

module Decoder (A, B, Y0, Y1, Y2, Y3);
    input (A, B);
    output (Y0, Y1, Y2, Y3);
    assign Y0 = (~A & ~B);
    assign Y1 = (~A & B);
    assign Y2 = (A & ~B);
    assign Y3 = (A & B);
end module.

```

VERILOG FOR COMBINATIONAL CIRCUITS

Having presented a number of useful building block circuits, we will now consider how such circuits can be described in Verilog. Rather than using gates or logic expressions, we will specify the circuits in terms of their behavior.

Example:

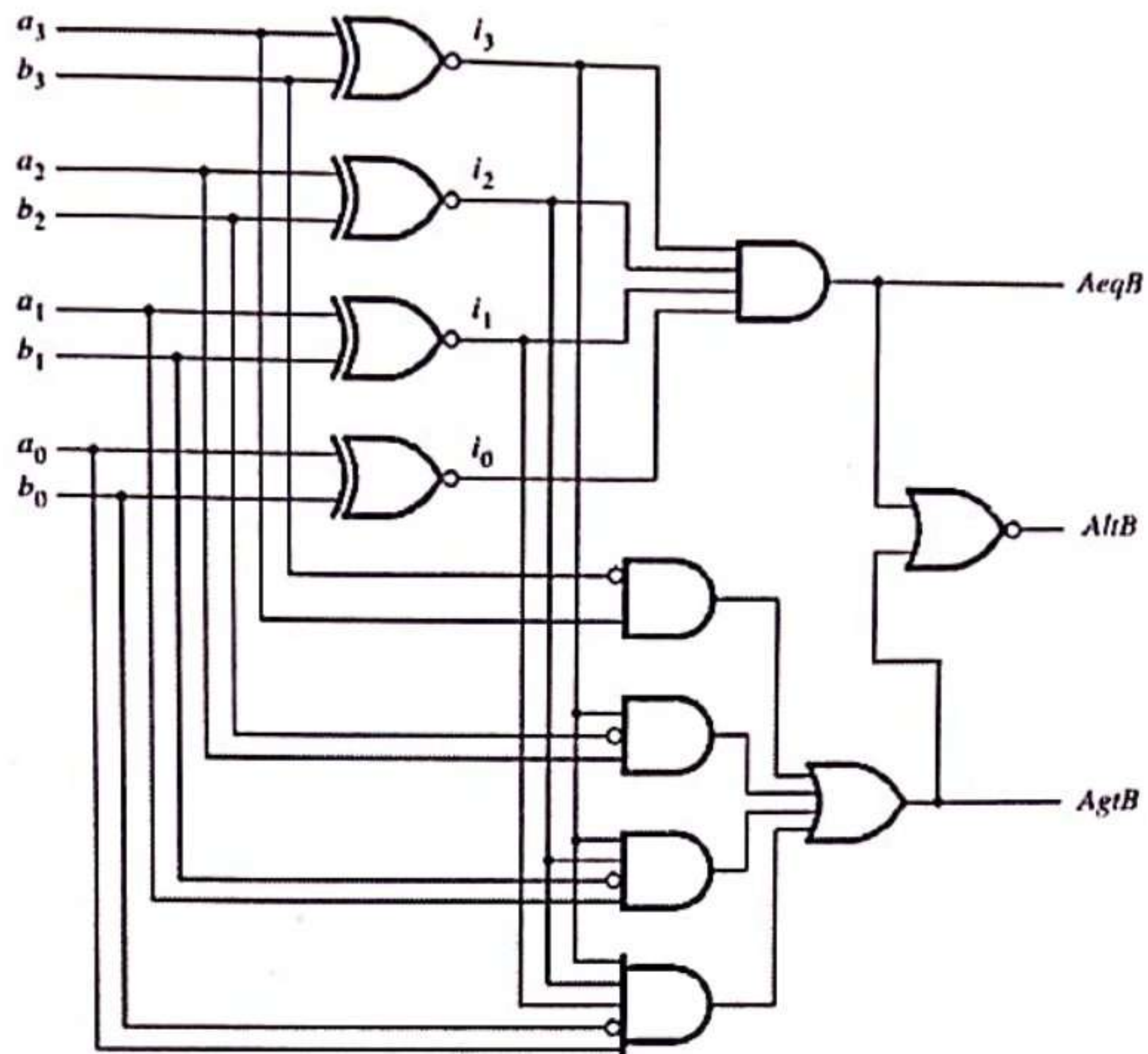


Figure 6.26 A four-bit comparator circuit.

```

module Comparator_4bit (
    input [3:0] A, // 4-bit input A
    input [3:0] B, // 4-bit input B
    output reg AeqB, // A = B
    output reg AltB, // A < B
    output reg AgtB, // A > B
);

    always @(*) begin
        // Default values for output
        AeqB = 0;
        AltB = 0;
        AgtB = 0;

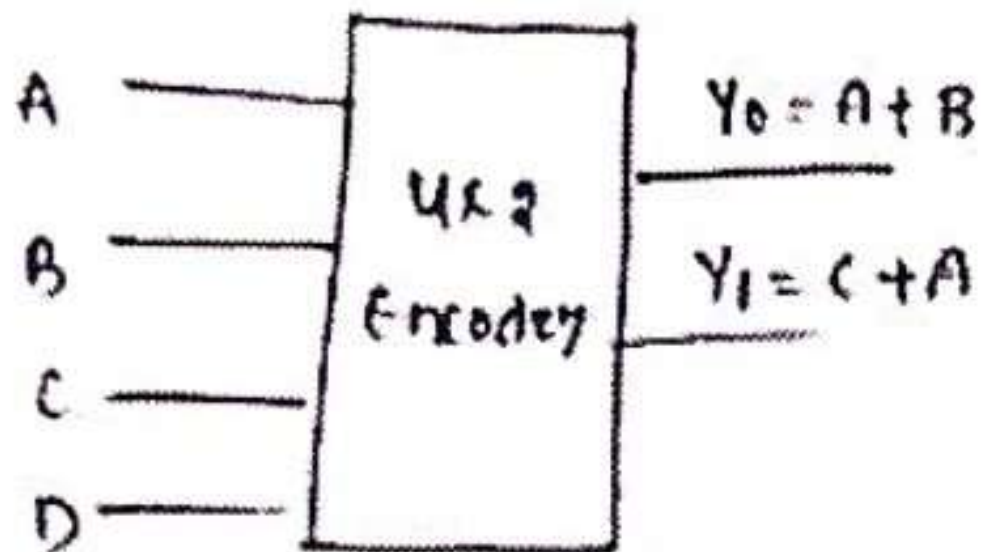
        // Compare A and B
        if (A == B)
            AeqB = 1;
        else if (A < B)
            AltB = 1;
        else
            AgtB = 1;
    end

endmodule

```


Encoder

4x2 Encoder



```
Module Encoder(A,B,C,D,Y0,Y1);
```

```
Input (A,B,C,D);
```

```
Output Y0,Y1;
```

```
assign Y0 = A + B;
```

```
assign Y1 = C + A;
```

```
end module.
```

Comparator

3-bit Magnitude Comparator.



```
module comparator(A,B,YL,YE,YH);
```

```
Input A,B;
```

```
Output YL,YE,YH;
```

```
assign YL = (~A & B);
```

```
YE = A ^ B;
```

```
YH = (A & ~B);
```

```
end module.
```

Parity generator.

```
module generator(A,B,C,even,odd);
```

```
Input (A,B,C);
```

```
Output even,odd;
```

```
assign even = A ^ B ^ C;
```

```
assign odd = ~even;
```

```
end Module.
```


Describe all logic gates in Verilog HDL using behavioral constructs

AND Gate:

```
module AND_ex (A, B, Y);
    input A, B;
    output reg Y;

    always @ (A or B) begin
        if (A == 1'b1 && B == 1'b1)
            Y = 1'b1;
        else
            Y = 1'b0;
        end
    endmodule
```

OR Gate:

```
module OR_ex (A, B, Y);
    input A, B;
    output reg Y;

    always @ (A or B) begin
        if (A == 1'b1 || B == 1'b1)
            Y = 1'b1;
        else
            Y = 1'b0;
        end
    endmodule
```

NOT Gate:

```
module NOT_ex (A, Y);
    input A;
    output reg Y;

    always @ (A) begin
        if (A == 1'b1)
            Y = 1'b0;
        else
            Y = 1'b1;
        end
    endmodule
```

NAND Gate

```
module NAND_ex (A, B, Y);
    input A, B;
    output reg Y;

    always @ (A or B) begin
        if (A == 1'b1 && B == 1'b1)
            Y = 1'b0;
        else
            Y = 1'b1;
        end
    endmodule
```


NOR Gate

```
module NOR_ex (A, B, Y);  
  input A, B;  
  output reg Y;  
  
  always @ (A or B) begin  
    if (A == 1'b0 && B == 1'b0)  
      Y = 1'b1;  
    else  
      Y = 1'b0;  
    end  
endmodule
```

X-OR Gate

```
module XOR_ex (A, B, Y);  
  input A, B;  
  output reg Y;  
  
  always @ (A or B) begin  
    if (A != B)  
      Y = 1'b1;  
    else  
      Y = 1'b0;  
    end  
endmodule
```

X-NOR Gate

```
module XNOR_ex (A, B, Y);  
  input A, B;  
  output reg Y;  
  
  always @ (A or B) begin  
    if (A == B)  
      Y = 1'b1;  
    else  
      Y = 1'b0;  
    end  
endmod
```



```

module decoder3to8_assign (en, a, b, c, y);
    input en, a, b, c;
    output [7:0] y;

    assign y[0] = en & ~a & ~b & ~c; // 000
    assign y[1] = en & ~a & ~b & c; // 001
    assign y[2] = en & ~a & b & ~c; // 010
    assign y[3] = en & ~a & b & c; // 011
    assign y[4] = en & a & ~b & ~c; // 100
    assign y[5] = en & a & ~b & c; // 101
    assign y[6] = en & a & b & ~c; // 110
    assign y[7] = en & a & b & c; // 111
endmodule

```

THE CONDITIONAL OPERATOR

In a logic circuit it is often necessary to choose between several possible signals or values based on the state of some condition. A typical example is a multiplexer circuit in which the output is equal to the data input signal chosen by the valuation of the select inputs. For simple implementation of such choices Verilog provides a conditional operator (?:) which assigns one of two values depending on a conditional expression. It involves three operands used in the syntax

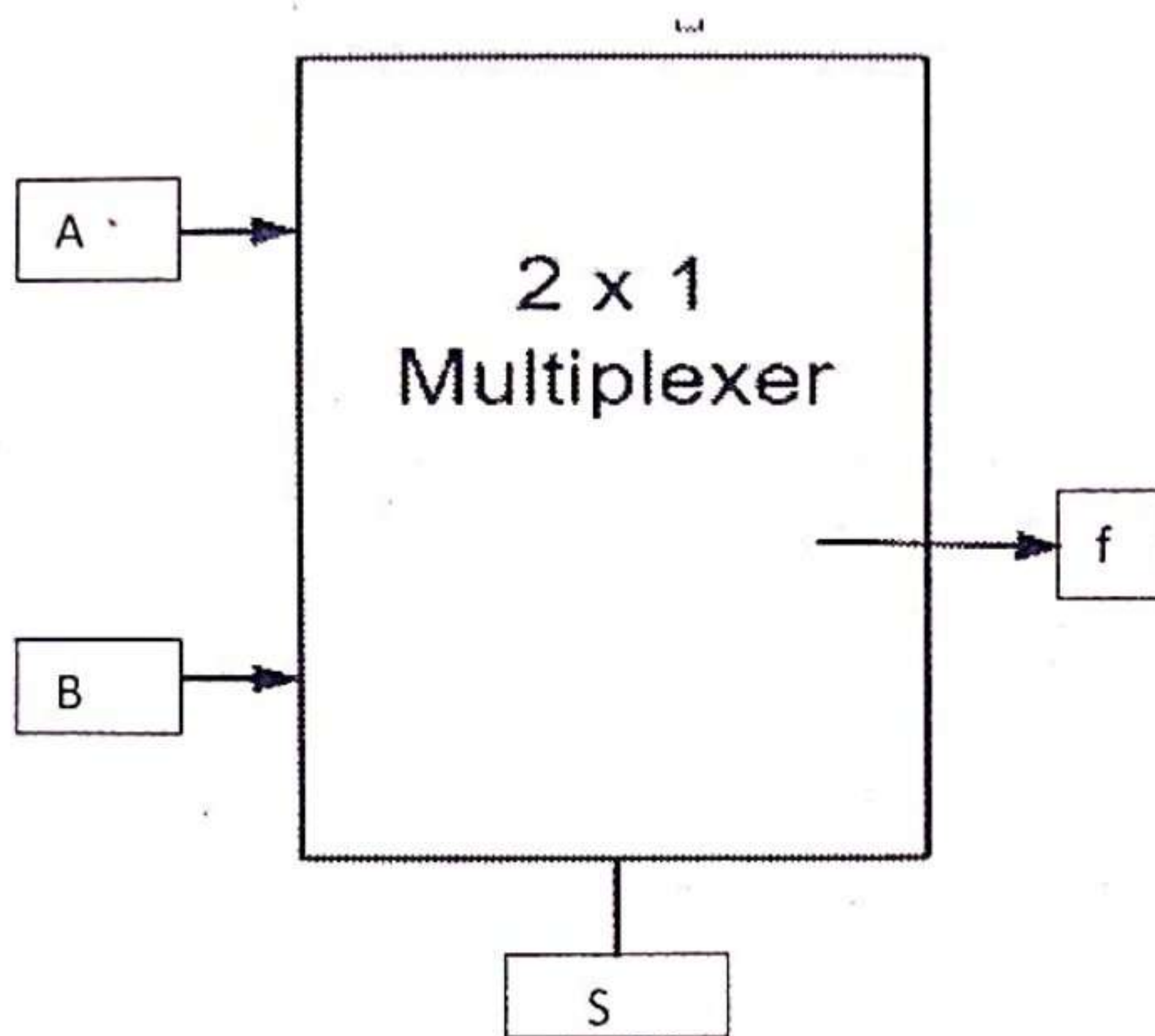
conditional_expression ? true_expression : false_expression

If the conditional expression evaluates to 1 (true), then the value of true_expression is chosen; otherwise, the value of false_expression is chosen. For example,

$$A = (B < C) ? (D + 5) : (D + 2);$$

the statement means that if B is less than C, the value of A will be D + 5 or else A will have the value D+2.

Verilog code for A 2-to-1 multiplexer specified using the conditional operator:



```
module mux2to1 (A, B, S, f);
```

```
    input A, B, S;
```

```
    output f;
```

```
    assign f = S ? B : A;
```

```
endmodule
```


4-Bit Comparator

A 4-bit comparator checks whether two 4-bit inputs (A and B) are equal. The output Y is 1 if A equals B, and 0 otherwise.

```
module Comparator_4bit (A, B, Y);  
    input [3:0] A, B;  
    output Y;  
  
    assign Y = (A == B) ? 1'b1 : 1'b0;  
  
endmodule
```

<result> = (<condition>) ? <value_if_true> : <value_if_false>;

Priority Encoder (4-to-2)

A 4-to-2 priority encoder outputs the binary code of the highest priority input that is high (1). If D[3] is high, it outputs 11, if D[2] is high and D[3] is low, it outputs 10, and so on.

```
module Priority_Encoder_4to2 (D, Y);  
    input [3:0] D;  
    output [1:0] Y;  
  
    assign Y = (D[3] == 1'b1) ? 2'b11 :  
               (D[2] == 1'b1) ? 2'b10 :  
               (D[1] == 1'b1) ? 2'b01 :  
               (D[0] == 1'b1) ? 2'b00 : 2'b00;  
  
endmodule
```

Explanation:

- The conditional operator is used here to check which input D[3:0] is high.
- The highest priority input is selected first. For example, if D[3] is high, the output Y is 11.
- If D[3] is low and D[2] is high, the output is 10, and so on.
- If none of the inputs are high, the output is 00.

THE IF-ELSE STATEMENT

It has the syntax

```
if (conditional_expression) statement;
```

```
else statement;
```

If the expression is evaluated to true then the first statement (or a block of statements delineated by **begin** and **end** keywords) is executed or else the second statement (or a block of statements) is executed.

Verilog code for A 2-to-1 multiplexer specified using The If-Else Statement:

```
module mux2to1 (w0, w1, s, f);  
    input w0, w1, s;  
    output f;  
    reg f;  
    always @(w0 or w1 or s)  
        if (s == 0) f = w0;  
        else f = w1;  
endmodule
```

HIERARCHICAL VERILOG CODE:

Hierarchical Verilog code for a 16-to-1 MUX using 4X1 multiplexer:

```
module mux16to1 (W, S16, f);  
    input [0:15] W;    // 16 input lines  
    input [3:0] S16;    // 4 select lines  
    output f;          // 1 output line  
    wire [0:3] M;      // Intermediate wires  
    // First level of 4-to-1 multiplexers  
    mux4to1 Mux1 (W[0:3], S16[1:0], M[0]);  
    mux4to1 Mux2 (W[4:7], S16[1:0], M[1]);  
    mux4to1 Mux3 (W[8:11], S16[1:0], M[2]);  
    mux4to1 Mux4 (W[12:15], S16[1:0], M[3]);  
    // Second level: a 4-to-1 multiplexer to select among the first four outputs  
    mux4to1 Mux5 (M[0:3], S16[3:2], f);  
endmodule
```


Hierarchical Verilog code for a 8-to-1 MUX using 4x1 multiplexer :

```
module mux16to1 (W, S, f);
    input [0:7] W;    // 8 input lines
    input [2:0] S;    // 3 select lines
    output f;        // 1 output line
    wire [0:1] M;    // Intermediate wires
    // First level of 4-to-1 multiplexers
    mux4to1 Mux1 (W[0:3], S[1:0], M[0]);
    mux4to1 Mux2 (W[4:7], S[1:0], M[1]);
    // Second level: a 2-to-1 multiplexer to select among the first two outputs
    Mux2to1 Mux3 (M[0:1], S[2], f);
endmodule
```

Hierarchical Verilog code for a 8-to-1 MUX using 2x1 multiplexer :

```
module mux16to1 (W, S, f);
    input [0:7] W;    // 8 input lines
    input [2:0] S;    // 3 select lines
    output f;        // 1 output line
    wire [0:5] M;    // Intermediate wires
    // First level of 2-to-1 multiplexers
    Mux2to1 Mux1 (W[0:1], S[0], M[0]);
    Mux2to1 Mux2 (W[2:3], S[0], M[1]);
    Mux2to1 Mux3 (W[4:5], S[0], M[2]);
    Mux2to1 Mux4 (W[6:7], S[0], M[3]);
    // Second level of 2-to-1 multiplexers
    Mux2to1 Mux5 (M[0:1], S[1], M[4]);
    Mux2to1 Mux6 (M[2:3], S[1], M[5]);
    // Third level: a 2-to-1 multiplexer to select among the two outputs
    Mux2to1 Mux7 (M[4:5], S[2], f);
endmodule
```


CASE STATEMENT

The if-else statement provides the means for choosing an alternative based on the value of an expression. When there are many possible alternatives, the code based on this statement may become awkward to read. Instead, it is often possible to use the Verilog case statement which is defined as

```
case (expression)
    alternative1: statement;
    alternative2: statement;
    .
    .
    .
    alternativej: statement;
    [default: statement;]
endcase
```

Verilog code:A 4-to-1 multiplexer defined using the case statement.

```
module mux4to1 (W, S, f);
    input [0:3] W;
    input [1:0] S;
    output f;
    reg f;
    always @(W or S)
    case (S)
        0: f = W[0];
        1: f = W[1];
        2: f = W[2];
        3: f = W[3];
    endcase
endmodule
```


Verilog code: A 8-to-1 multiplexer defined using the case statement.

```
module mux8to1 (W, S, f);  
    input [7:0] W;  
    input [2:0] S;  
    output reg f;  
    always @(*)  
        case (S)  
            3'd0: f = W[0];  
            3'd1: f = W[1];  
            3'd2: f = W[2];  
            3'd3: f = W[3];  
            3'd4: f = W[4];  
            3'd5: f = W[5];  
            3'd6: f = W[6];  
            3'd7: f = W[7];  
            default: f = 1'b0; // Optional but recommended  
        endcase  
endmodule
```

Verilog code: A 4-to-1 multiplexer defined using the case statement. (Method-2)

```
module mux4to1 (A, B, C, D, S1, S2, Y);  
    input A, B, C, D;  
    input S1, S2;  
    output reg Y; // Combined output and reg declaration  
  
    always @(*) begin  
        case ({S1, S2}) // Concatenated select lines  
            2'b00: Y = A;  
            2'b01: Y = B;  
            2'b10: Y = C;  
            2'b11: Y = D;  
            default: Y = 1'b0; // Ensures Y is defined for all cases  
        endcase  
    end  
endmodule
```


THE FOR LOOP

If the structure of a desired circuit exhibits a certain regularity, it may be possible to define the circuit using a for loop. The for loop has the syntax

for (initial_index; terminal_index; increment) statement;

Verilog code for a priority encoder:

```
module priority (W, Y, z);
```

```
    input [3:0] W;
```

```
    output [1:0] Y;
```

```
    output z;
```

```
    reg [1:0] Y;
```

```
    reg z;
```

```
    always @(W)
```

```
        begin
```

```
            z = 1;
```

```
            case (W)
```

```
                4'b1xxx: Y = 3;
```

```
                4'b01xx: Y = 2;
```

```
                4'b001x: Y = 1;
```

```
                4'b0001: Y = 0;
```

```
            default: begin z = 0;
```

```
                Y=2'bx;
```

```
            end
```

```
        endcase
```

```
    end
```

```
endmodule
```



```

module ripple_carry_adder (
    input [3:0] A, // 4-bit input A
    input [3:0] B, // 4-bit input B
    input Cin,    // Input carry (Cin)
    output [3:0] S, // 4-bit sum output
    output Cout   // Carry out
);

    // Internal wire to store carry between bits
    wire [4:0] C;

    // Assign the input carry Cin to the first carry wire
    assign C[0] = Cin;

    // Use a reg to store the result during the always block
    reg [3:0] sum_reg;
    reg carry_out_reg;

    // Combinational always block with a for loop
    always @(*) begin
        // Use a for loop to generate sum and carry for each bit
        integer i;
        for (i = 0; i < 4; i = i + 1) begin
            {C[i+1], sum_reg[i]} = A[i] + B[i] + C[i]; // Add A, B, and the carry from
the previous bit
        end
        carry_out_reg = C[4]; // Last carry is the final carry-out
    end

    // Assign the results to the output
    assign S = sum_reg;
    assign Cout = carry_out_reg;

endmodule

```

Explanation of the Code

1. Inputs and Outputs:

○ Inputs:

- A: A 4-bit input.
- B: A 4-bit input.
- Cin: Carry input.

○ Outputs:

- S: A 4-bit sum output.
 - Cout: Final carry output.
2. Carry Wires (C):
 - A 5-bit wire array is used to store carry values between each stage of the addition (C[0] to C[4]), where C[0] is the input carry, and C[4] is the final carry-out.
 3. always @(*) Block:
 - The block uses a for loop to perform the addition bit by bit.
 - For each bit i, it adds the A[i], B[i], and the previous carry (C[i]), generating the sum and the carry for the next stage (C[i+1]).
 4. Assignments:
 - The computed sum is stored in sum_reg, and the final carry (C[4]) is stored in carry_out_reg.
 - Both are assigned to the outputs S and Cout.

FOR LOOP USING STORAGE ELEMENTS WITH CAD TOOLS

USING VERILOG CONSTRUCTS FOR STORAGE ELEMENTS

A simple way of specifying a storage element is by using the if-else statement to describe the desired behavior responding to changes in the levels of data and clock inputs. Consider the always block

```
always @(Control or B)
  if (Control)
    A = B;
```

where A is a variable of reg type. This code specifies that the value of A should be made equal to the value of B when Control = 1. But the statement does not indicate an action that should occur when Control = 0. In the absence of an assigned value, the Verilog compiler assumes that the value of A caused by the if statement must be maintained until the next time this if statement is evaluated. This notion of implied memory is realized by instantiating a latch in the circuit.

CODE FOR A GATED D LATCH:

```
module D latch (D, Clk, Q);
  input D, Clk;
  output Q;
  reg Q;
  always @(D or Clk)
    if (Clk)
      Q = D;
endmodule
```


CODE FOR A D FLIP-FLOP:

```
module flipflop (D, Clock, Q);  
  input D, Clock;  
  output Q;  
  reg Q;  
  always @(posedge Clock)  
    Q = D;  
endmodule
```


FLIP-FLOPS WITH CLEAR CAPABILITY:

By using a particular sensitivity list and a specific style of if-else statement, it is possible to include clear (or preset) signals on flip-flops.

ASYNCHRONOUS CLEAR Figure 7.45 gives a module that defines a D flip-flop with an asynchronous active-low reset (clear) input. When **Resetn**, the reset input, is equal to 0, the flip-flop's Q output is set to 0. Note that the sensitivity list specifies the negative edge of **Resetn** as an event trigger along with the positive edge of the clock. We cannot omit the keyword **negedge** because the sensitivity list cannot have both edge-triggered and level-sensitive signals.

SYNCHRONOUS CLEAR

In this case the reset signal is acted upon only when a positive clock edge arrives. This code generates the circuit which has an AND gate connected to the flip-flop's D input.

VERILOG CODE: D flip-flop with asynchronous reset.

```
module flipflop (D, Clock, Resetn, Q);  
    input D, Clock, Resetn;  
    output Q;  
    reg Q;  
    always @(negedge Resetn or posedge Clock)  
        if (!Resetn) Q <= 0;  
        else Q <= D;  
endmodule
```

VERILOG CODE: D flip-flop with synchronous reset

```
module flipflop (D, Clock, Resetn, Q);  
    input D, Clock, Resetn;  
    output Q;  
    reg Q;  
    always @(posedge Clock)  
        if (!Resetn) Q <= 0;  
        else Q <= D;  
endmodule
```


USING VERILOG CONSTRUCTS FOR REGISTERS AND COUNTERS.

USING REGISTERS AND COUNTERS WITH CAD TOOLS AN N-BIT REGISTER

Since registers of different sizes are often needed in logic circuits, it is advantageous to define a register module for which the number of flip-flops can be easily changed. The code for an n-bit register is given below. The parameter n specifies the number of flip-flops in the register. By changing this parameter, the code can represent a register of any size.

Verilog Code :for an n-bit register with asynchronous clear

```
module regn (D, Clock, Resetn, Q);
    parameter n = 16;
    input [n-1:0] D;
    input Clock, Resetn;
    output [n-1:0] Q;
    reg [n-1:0] Q;

    always @(negedge Resetn or posedge Clock)
        if (!Resetn)
            Q <= 0;
        else
            Q <= D;

endmodule
```

Alternative code:

```
module shiftn (R, L, w, Clock, Q);
    parameter n = 16;
    input R;
    input L, w, Clock;
    output Q;
    reg Q;
    integer k;
    always @(posedge Clock)
        if (L) Q <= R;
```



```

    else
    begin
    for (k = 0; k < n-1; k = k+1) Q[k] <= Q[k+1];
    Q[n-1] <= w;
    end
endmodule

```

A FOUR-BIT SHIFT REGISTER

Code defines the four-bit shift register. The module Stage3 instantiates the leftmost flip-flop, which has the output Q3, and the module Stage0 instantiates the right-most flip-flop, Q0. When L = 1, the register is loaded in parallel from the R input; and when L = 0, shifting takes place in the left to right direction. Serial data is shifted into the most-significant bit, Q3, from the w input

```

module shift4 (R, L, w, Clock, Q);
input [3:0] R;
input L, w, Clock;
output [3:0] Q;
wire [3:0] Q;
muxdff Stage3 (w, R[3], L, Clock, Q[3]);
muxdff Stage2 (Q[3], R[2], L, Clock, Q[2]);
muxdff Stage1 (Q[2], R[1], L, Clock, Q[1]);
muxdff Stage0 (Q[1], R[0], L, Clock, Q[0]);
endmodule

```

alternative code:

```

module shift4 (R, L, w, Clock, Q);
input [3:0] R;
input L, w, Clock;
output [3:0] Q;
reg [3:0] Q;

```



```

always @(posedge Clock) if (L) Q <= R;
else
begin
Q[0] <= Q[1];
Q[1] <= Q[2];
Q[2] <= Q[3];
Q[3] <= w;
end
endmodule

```

4-BIT UP-COUNTER

Code represents a four-bit up-counter with a reset input, Resetn, and an enable input, E. The outputs of the flip-flops in the counter are represented by the vector named Q. The if statement specifies an asynchronous reset of the counter if Resetn = 0. The else if clause specifies that if E = 1 the count is incremented on the positive clock edge.

```

module upcount (Resetn, Clock, E, Q);
input Resetn, Clock, E;
output [3:0] Q;
reg [3:0] Q;
always @(negedge Resetn or posedge Clock)
if (!Resetn)
Q <= 0;
else if (E)
Q <= Q + 1;
endmodule

```

CODE FOR AN UP/DOWN COUNTER

Verilog code for an up/down counter It includes a control signal up_down that governs the direction of counting. It also includes an integer variable named direction, which is equal to 1 for up-count and equal to -1 for down-count.


```

module updowncount (R, Clock, L, E, up_down,
Q);
parameter n = 8;
input [n -1:0] R;
input Clock, L, E, up_down;
output [n -1:0] Q;
reg [n -1:0] Q;
integer direction;
always @(posedge Clock)
begin
  if (up_down)
    direction = 1;
  else
    direction = -1;
  if (L)
    Q <= R;
  else if (E)
    Q <= Q + direction;
end
endmodule

```

DOWN-COUNTER WITH PARALLEL LOAD

the code for a down-counter named downcount. A down-counter is normally used by loading it with some starting count and then decrementing its contents. The starting count is represented in the code by the vector R. On the positive clock edge, if L = 1 the counter is loaded with the input R, and if L = 0 the count is decremented. The counter also includes an enable input, E. Setting E = 0 prevents the contents of the flip-flops from changing when an active clock edge occurs.

```

module downcount (R, Clock, E, L, Q);
parameter n = 8;

```



```
input [n -1:0] R;  
input Clock, L, E;  
output [n -1:0]Q;  
reg [n -1:0]Q;  
always @(posedge Clock)  
if (L)  
    Q <= R;  
else if (E)  
    Q <= Q- 1;  
endmodule
```


Sequential Logic Circuits

def: Sequential logic ckt:-

Sequential logic circuit not only considers the current inputs but also retains a memory of previous states. It incorporates both combinational & memory elements

eg:- flip-flops, counters & registers

[or]
It is a logic circuit whose o/p depends on both present i/p's and previous o/p's

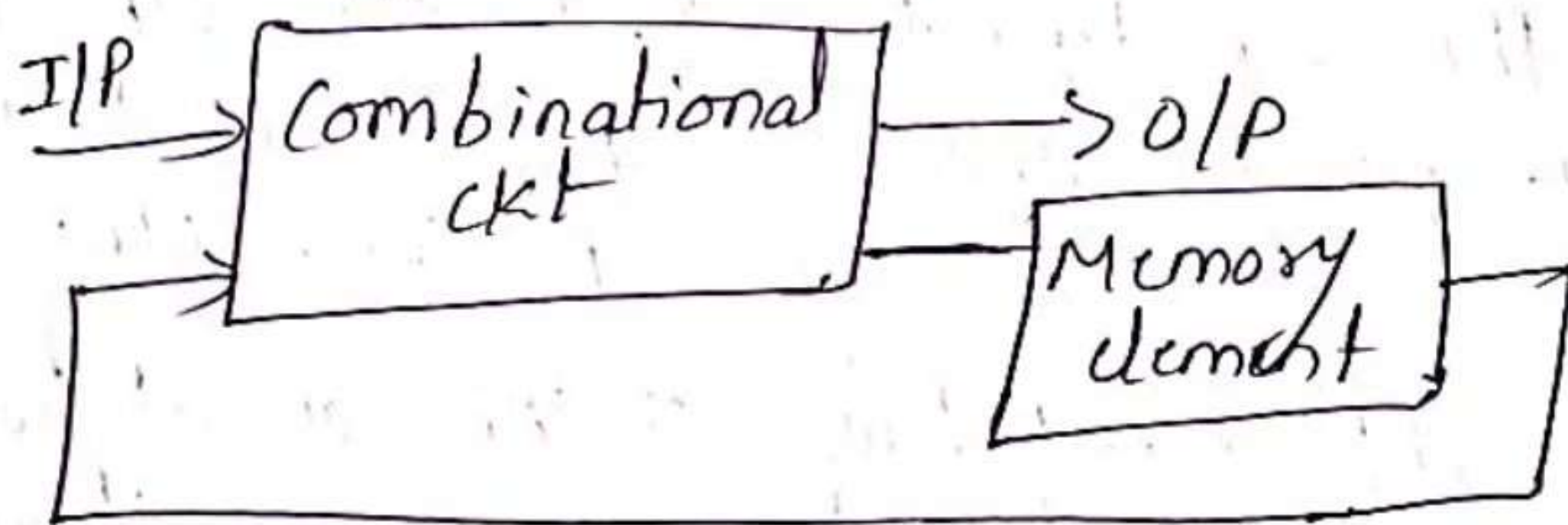


fig:- sequential logic circuit.

clock signal:-

The time at which any o/p can change states is determined by a signal called

"clock signal"

* clock pulse is normally a rectangular pulse train (or) square wave as shown in fig

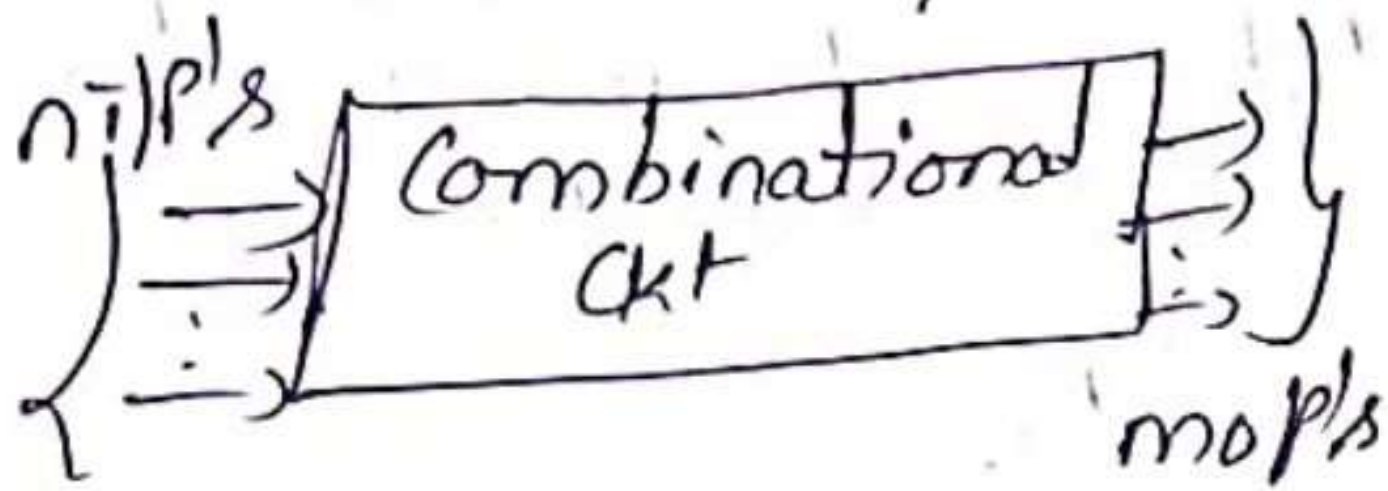
clk 

24/01/2024 Comparison b/w Combinational & Sequential

* The Comparison b/w Combinational & Sequential circuit is.

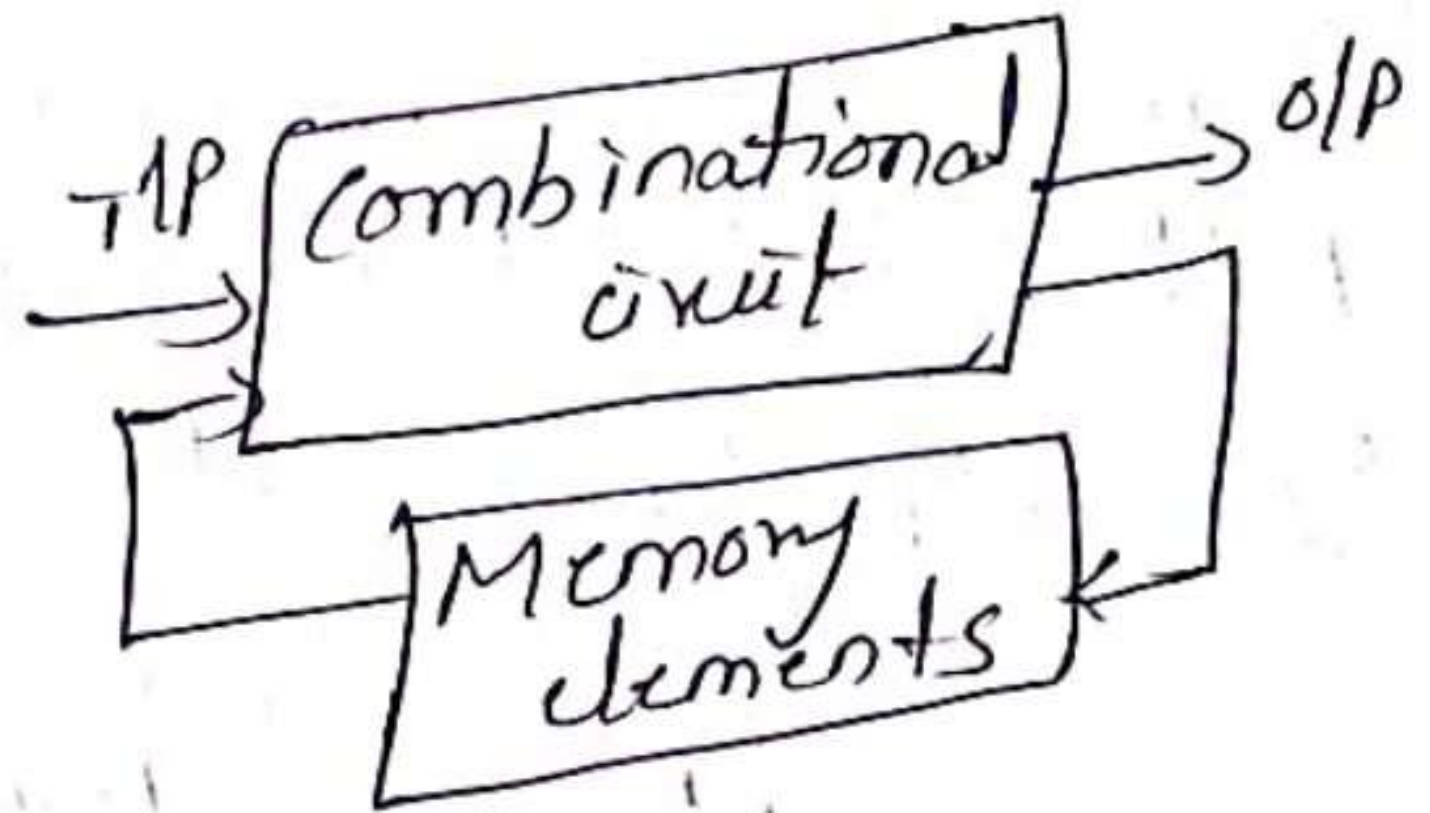
Combinational ckt

* Block diagram



Sequential ckt

* Block diagram.



* Examples:-
Adders

* In this o/p depends upon present i/p's only.

* This ckt doesn't have any memory element.

* Used for arithmetic as well as Boolean operations.

* There is no f/b b/w i/p & o/p.

* Ex: Counters.

* In this o/p depends present input as well as previous o/p.

* This circuit has memory element.

* Mainly used for storing data.

* There exists f/b b/w i/p & o/p.

Design procedure for sequential circuits:-

(2)

- * state diagram (or) timing diagram is given, which describes the behaviour of the ckt that is to be designed.
- * obtain the state table
- * The number of states can be reduced by state reduction method.
- * Do state assignment
- * determine the number of flip-flops
- * design the type of f/f to be used.
- * derive the ckt excitation table
- * Implement the circuit

* 2M Latch:-

Latch is a sequential circuit with two stable states as logic (1) or logic (0) which is which is sensitive to the T/P applied voltage or do not depend on the clock pulse.

- * flipflops do not use clock pulse.
- * They are 2 types 1. S-R Latch
2. D Latch.

1. Working of RS NAND Latch:-

- * Consider RS [R=Reset, S=Set] Latch formed with two cross coupled NAND gates as shown in fig.
- * R & S are the 2 T/P's, where as Q & $\bar{Q}$ are two O/P's
- * Here the O/P of one gate is connected to the T/P of the other gate, this produces the fb path.

Logic diagram:-

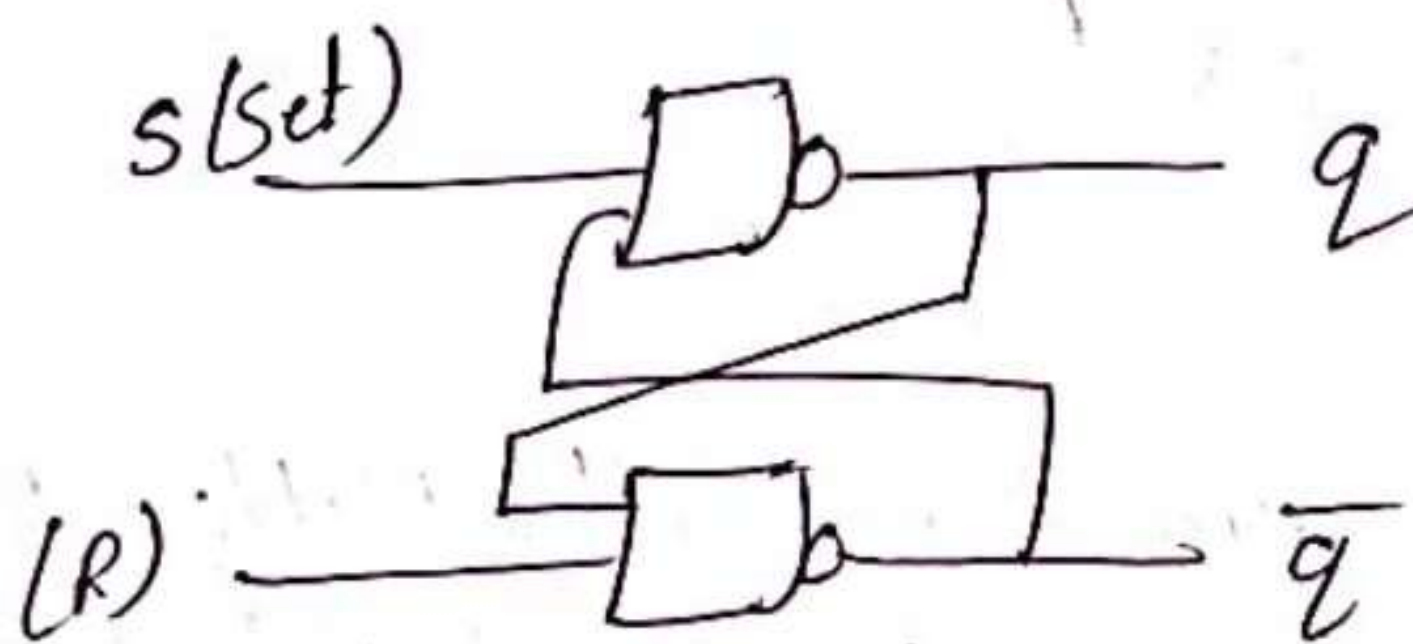


Fig:- SR Latch by using NAND

Truth table for:-

NAND T.T

A	B	$Y = A\bar{B}$
0	0	1
0	1	1
1	0	1
1	1	0

SR Latch T.T

S	R	Q	$\bar{Q}$	
0	0	Invalid		
0	1	0	1	Reset
1	0	1	0	Set
1	1	Q	$\bar{Q}$	No change

(3)

Case i):- When $S=0, R=0$

$Q=1, \bar{Q}=1 \rightarrow$ Invalid

Case ii) When $S=0, R=1$

$Q=0, \bar{Q}=1 \rightarrow$ Reset

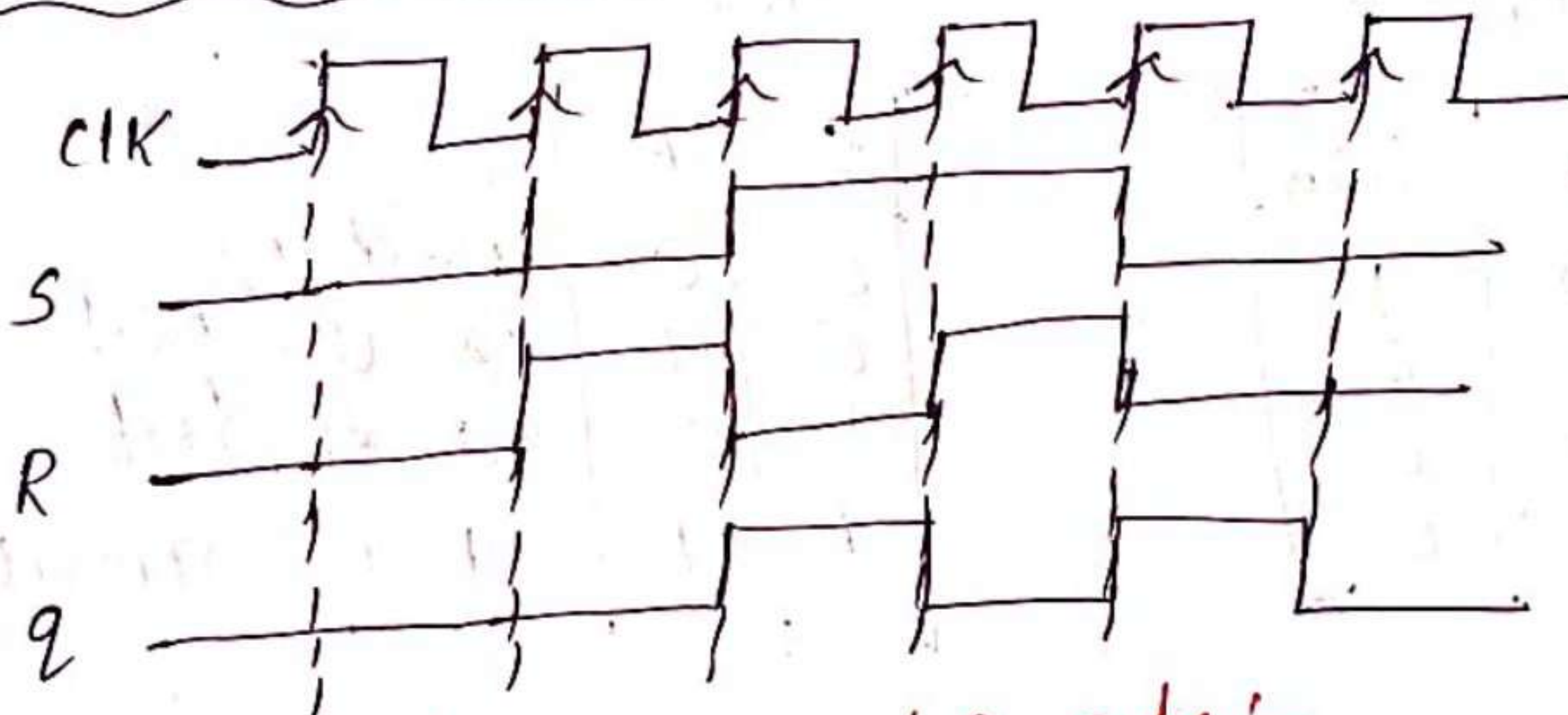
Case iii) When $S=1, R=0$

$Q=1, \bar{Q}=0 \rightarrow$ Set

Case iv) $S=1, R=1$, Assume $Q=0, \bar{Q}=1$

$Q=1, \bar{Q}=0$ } No change state
 $Q=0, \bar{Q}=1$ }

Timing diagram:-



SR Latch by using NOR gate:-

* Consider SR [set, Reset] Latch formed with two cross coupled NOR gates as shown in fig.

* R & S are the two i/p's where as Q & $\bar{Q}$ are the 2 o/p's

* The o/p of one gate is connected to the input of the other gate, this produces the fb path.

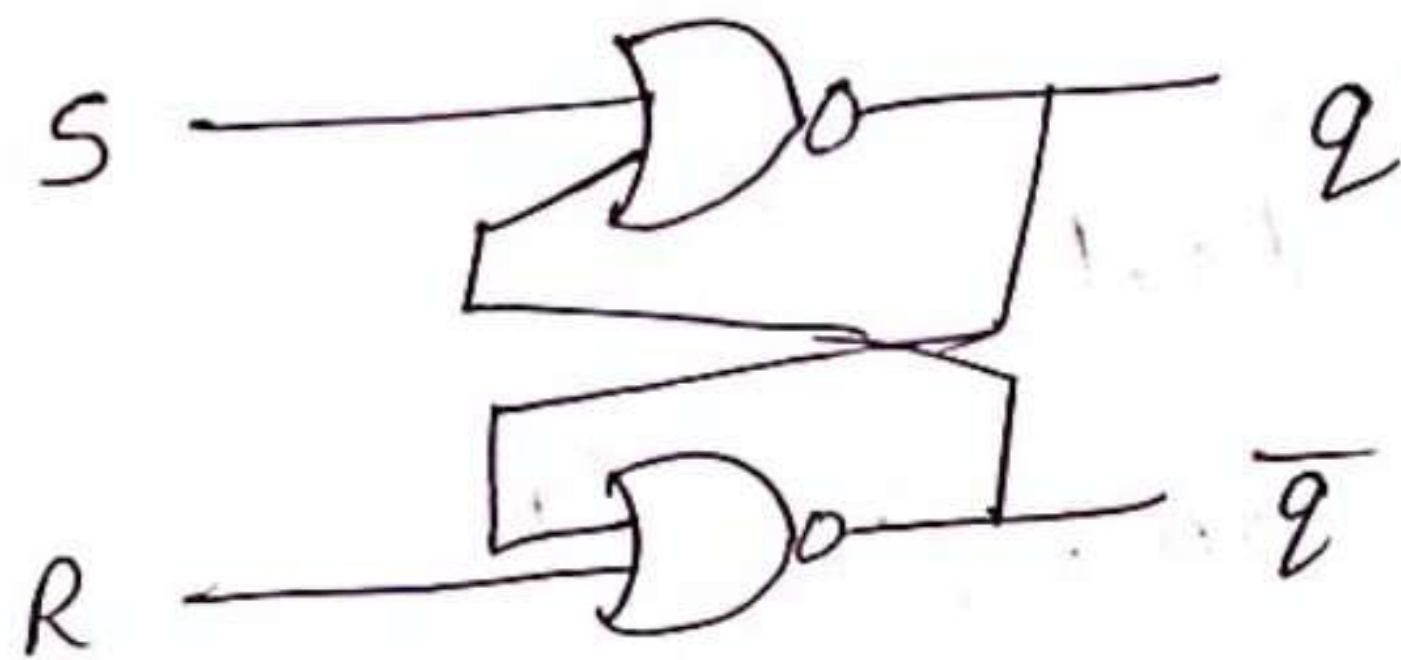


Fig:- SR latch by using NOR gate.

Truth tables:-

NOR T.T

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

SR Latch T.T

S	R	Q	$\overline{Q}$
0	0	No change	
0	1	0	1 $\rightarrow$ Set
1	0	1	0 $\rightarrow$ Set
1	1	1	1 $\rightarrow$ Invalid

Case i):- $S=0, R=0$, Assume $Q=0$

$Q=0, \overline{Q}=1$
 $Q=1, \overline{Q}=0$

} No change.

Case ii):- $S=0, R=1$

$Q=1, \overline{Q}=0 \rightarrow$ Set

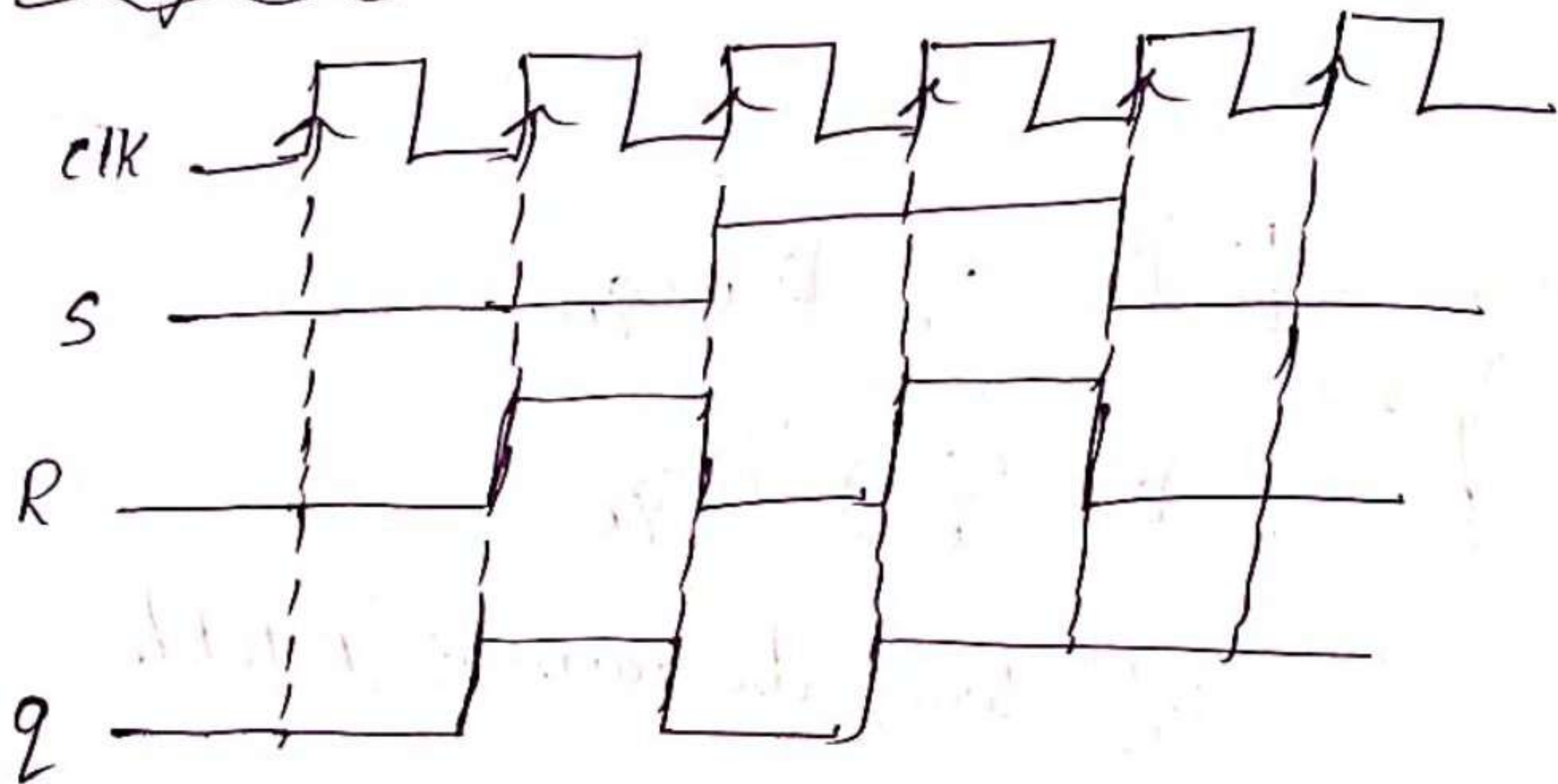
(Case iii) :- $S=1, R=0$

$Q=0, \bar{Q}=1 \rightarrow$ Reset

(Case iv) :- $S=1, R=1$

$Q=1, \bar{Q}=1 \rightarrow$ Invalid

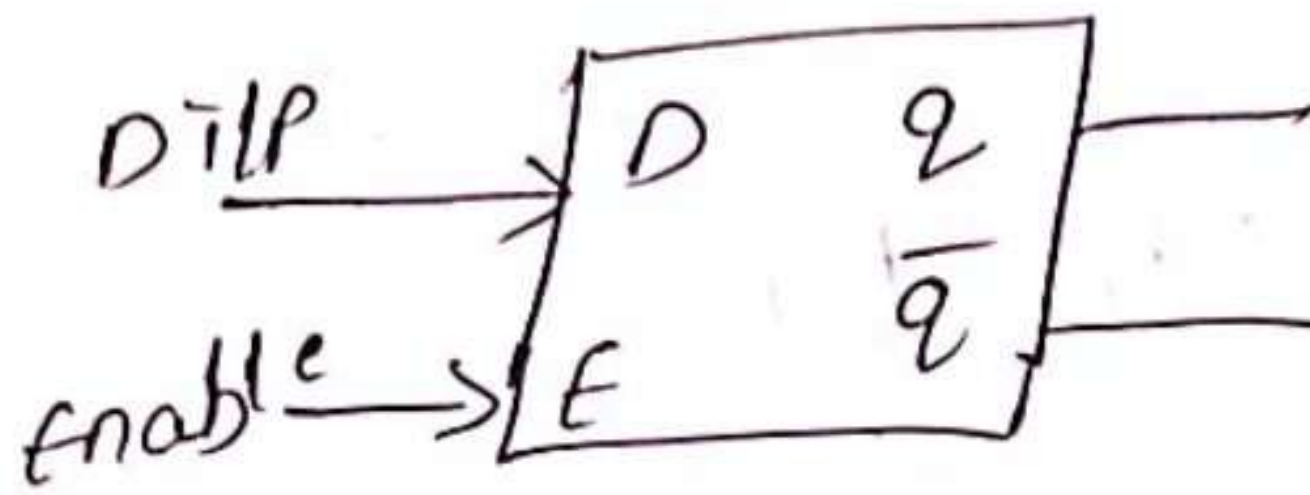
Timing diagram :-



D-latch :-

- * The ckt diagram of gated D-latch as shown in fig.
- * This is also called as gated Level Triggered D flip. Here D latch is the simplest gated SR latch, with a NAND Inverter connected b/w its S & R I/P's. This latch has only one T/P denoted by 'D'.
- * hence, $S=0, R=0$ (as $S=1, R=1$ these T/P conditions will never appear).

Logic symbol :-



Logic diagram :-

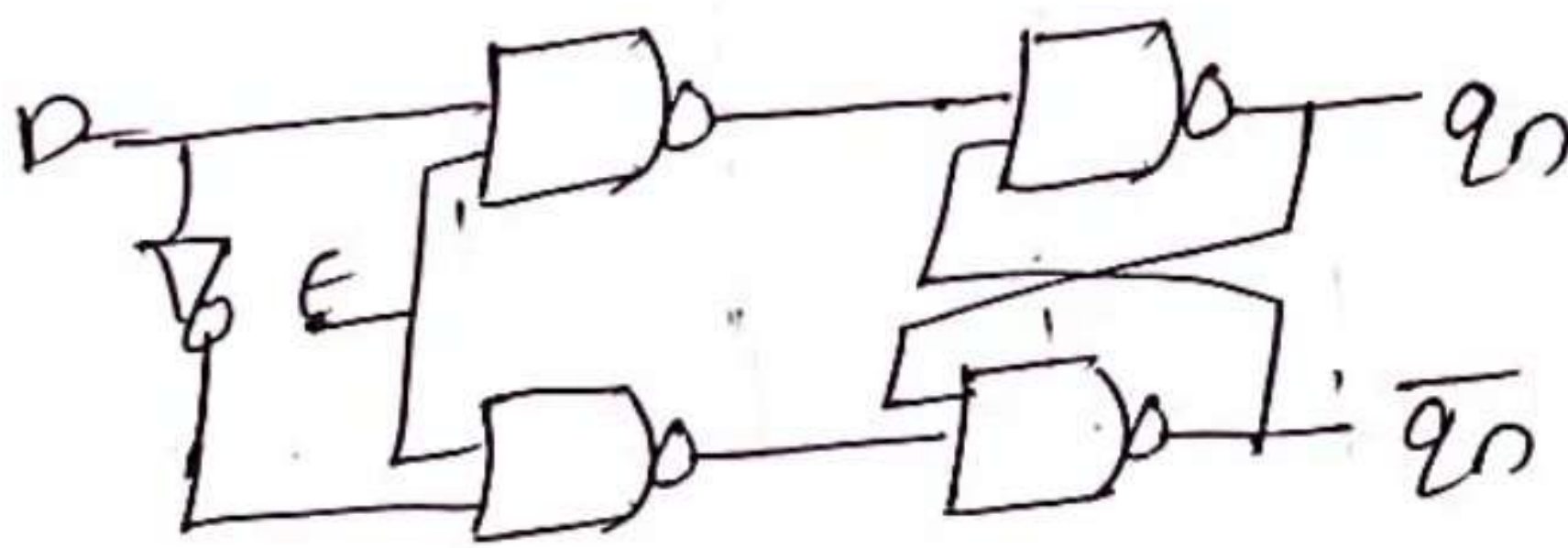


Fig:- Logic diagram of D latch

NAND T.T

A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

D-Latch T.T

E	D	Q	$\overline{Q}$
0	X	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

No change
 Reset
 Set

Case i):- When $D=0$,
 $Q=0, \overline{Q}=1$

Case ii):- When $D=1$,
 $Q=1, \overline{Q}=0$

Difference B/w Latches & Flip Flops:-

(5)

Latch

Flip Flop

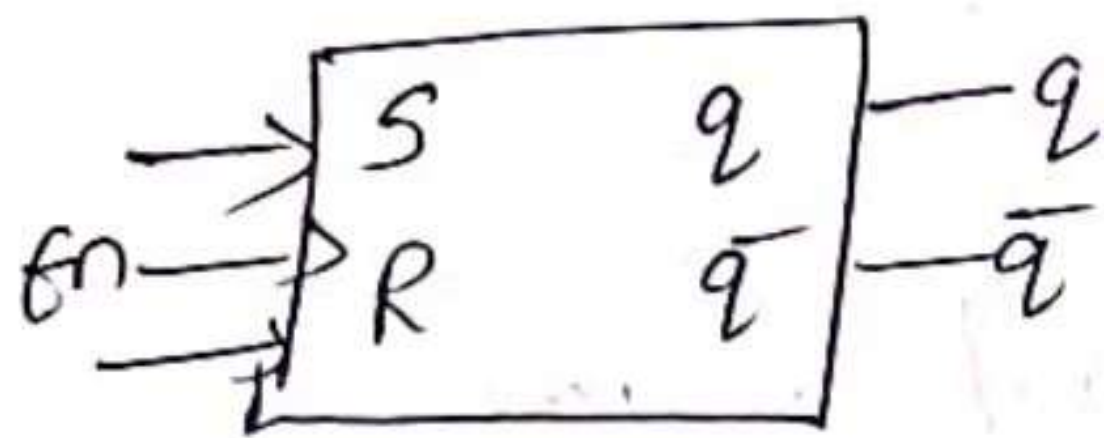
* Latch is the basic building blocks of any sequential circuit

* Flip-flop is also the building block of any sequential circuit.

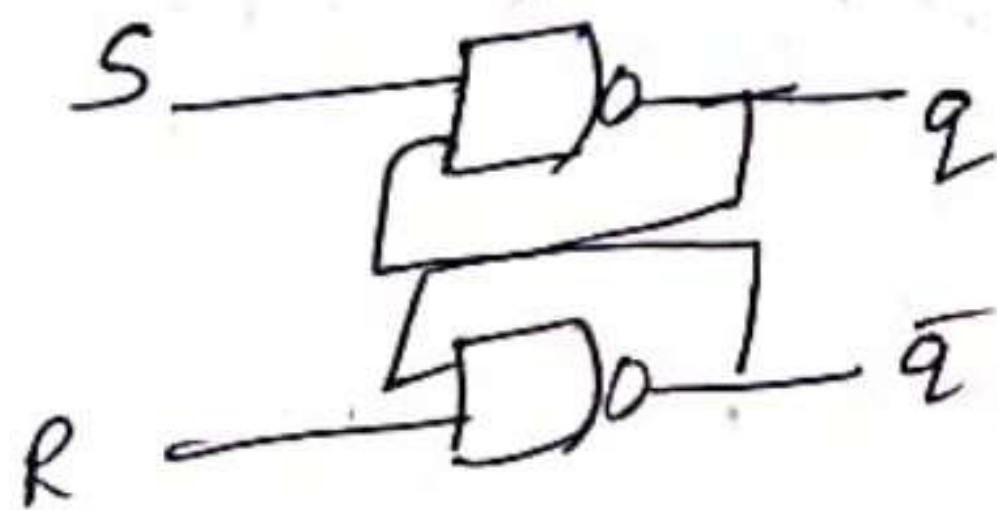
* Latch can be built by logic gates

* Flipflops can be built by latch with an additional control T/P.

* Logic symbol:-



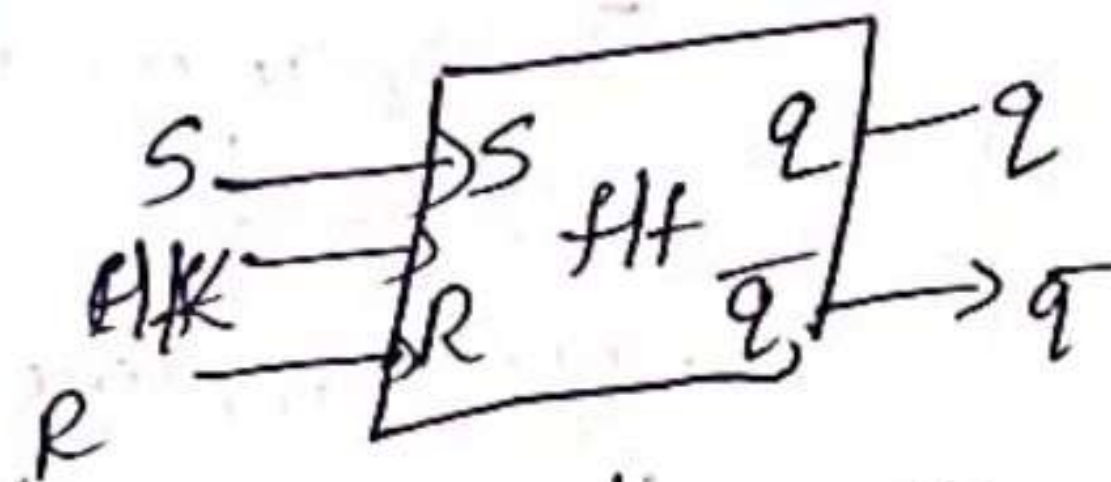
* Logic diagram



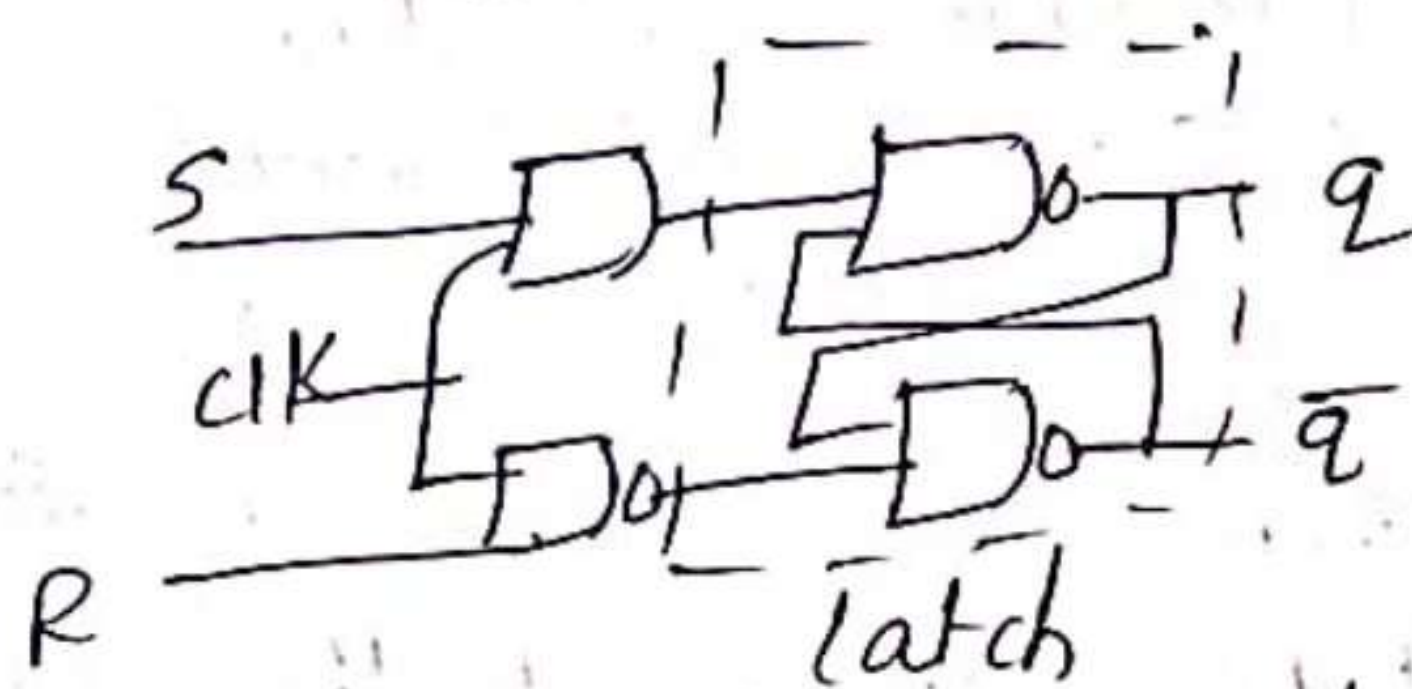
* Latch is a level triggered device



* Latch doesn't have a clock signal



* Logic diagram



* Flipflop is a edge triggered device



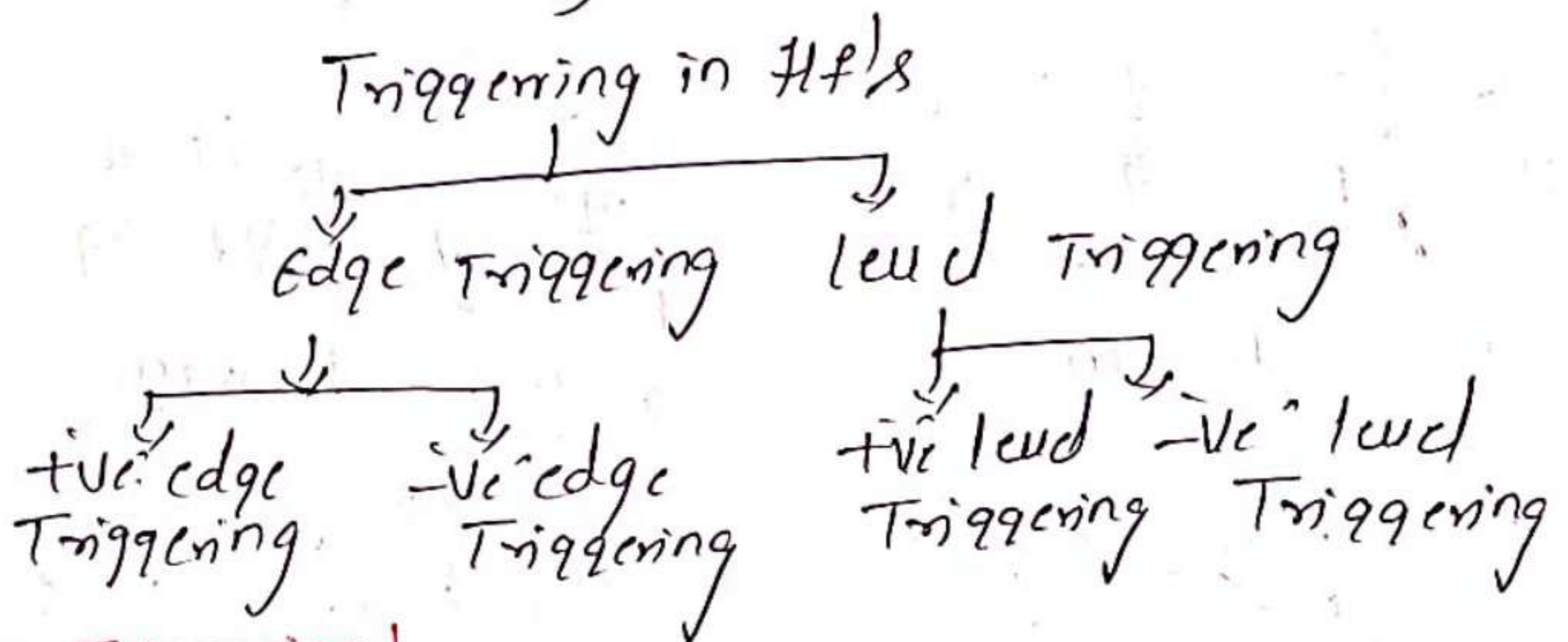
* Flipflop always have a clock signal.

Triggering in flipflops:-

- * When a clock pulse is applied along with the inp signal and the o/p changes at a particular instant of clock pulse is known as clocking (or) Triggering.

Types of Triggering:-

- * They are 2 types by which ff s can be Triggered
i) Edge Triggering
ii) level "



Edge Triggering:-

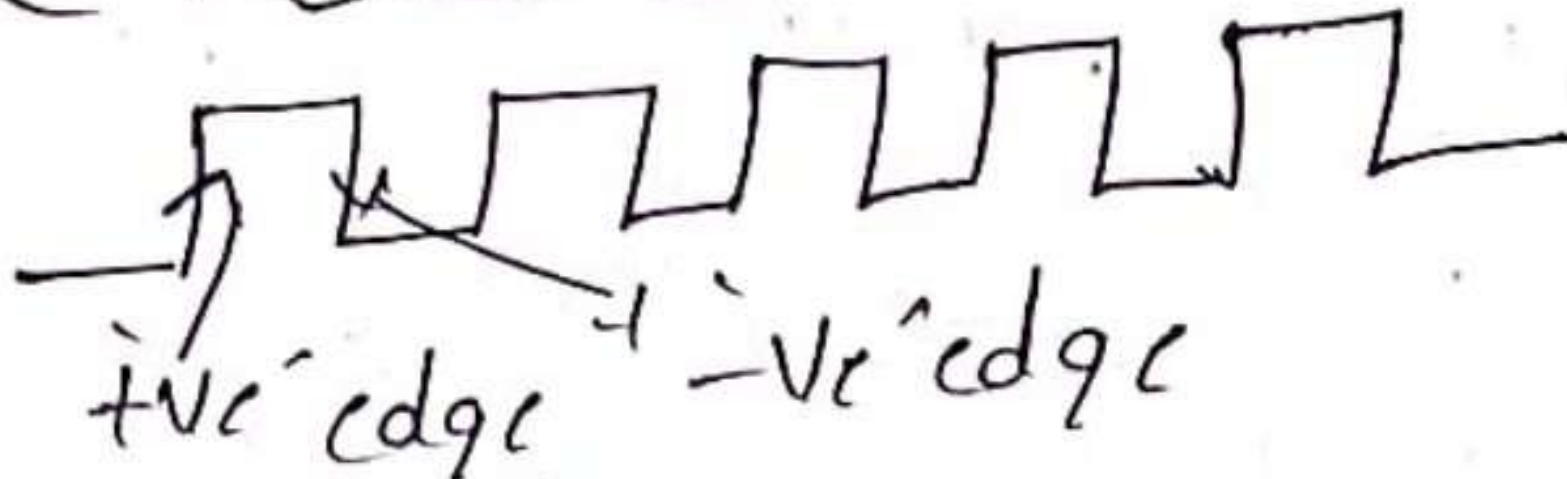
- * An edge Triggered flipflop changes states either at the $+ve$ edge (Rising edge) (or) at the $-ve$ edge (falling edge) of the clock pulse.
- * Edge Triggered is divided into 2 types
 $+ve$ edge:- When the Triggering occurs on the $+ve$ going edge of the clock. It is called

(6)

"+ve" edge Triggering"

b) -ve edge:- When the Triggering occurs on the -ve going edge of the clock, it is called

"-ve" edge Triggering"



Level Triggering:-

* Level Triggered ff changes state either at the +ve level (or) at the -ve level of the clock pulse

* Level Triggering in ff's can be divided into '2' types ✓

a) +ve Level:- When the Triggering occurs on the +ve level of the clock, it is called +ve level Triggering

b) -ve Level:- When the Triggering occurs on the -ve level of the clock, it is called -ve level Triggering

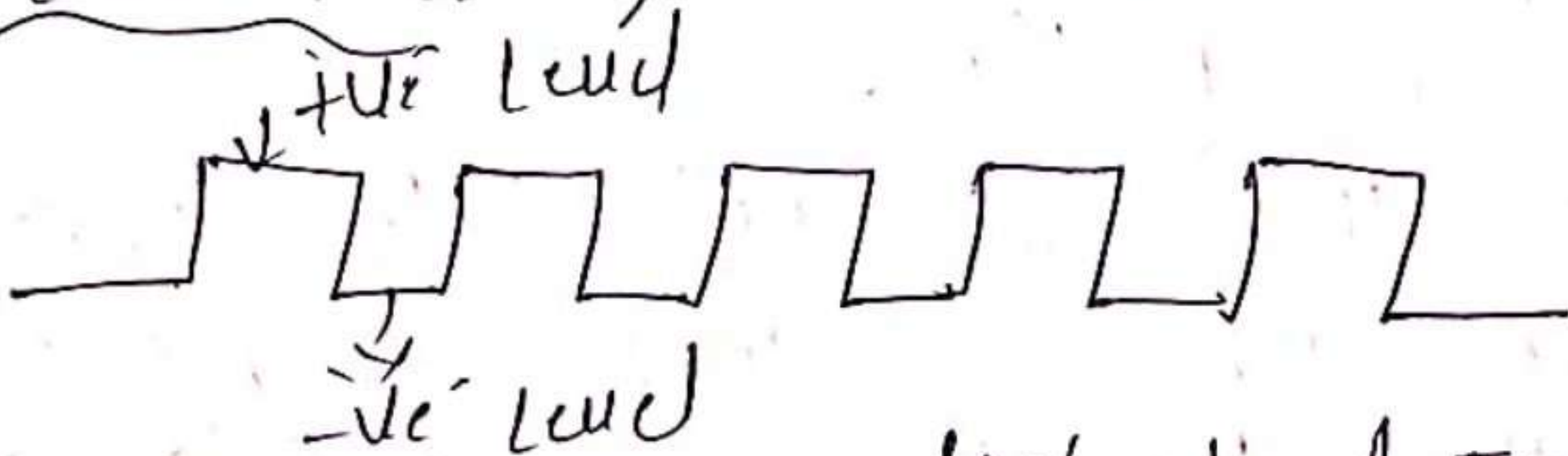


fig:- level Triggered clock pulse

* 2M Flipflops:-

def:- A flipflop is a fundamental building block of digital electronics, used to store binary information. It is capable of storing one bit information [1 (or) 0]. It is known as flipflop.

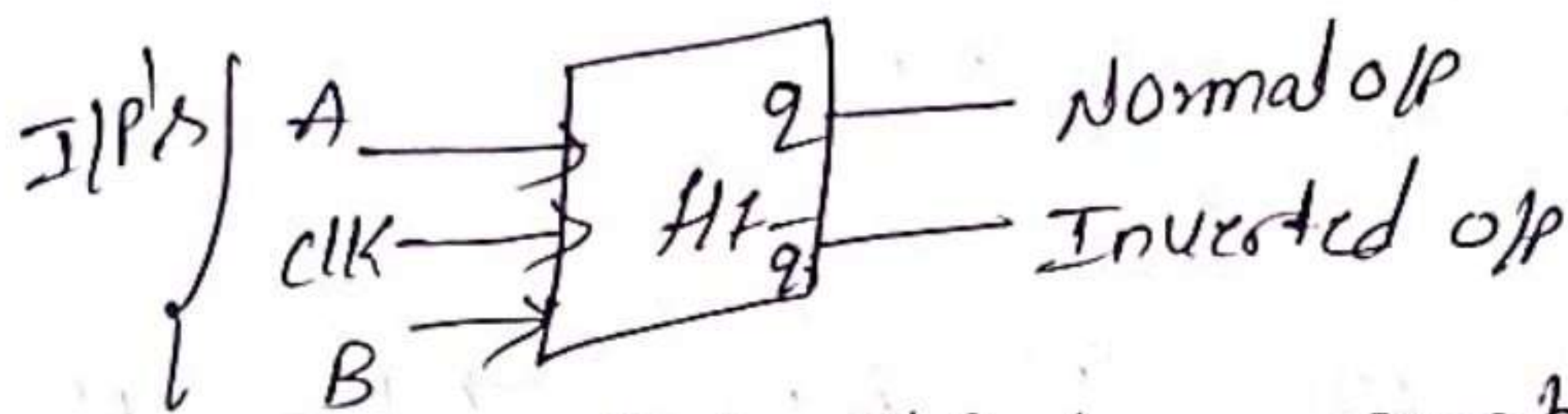


Fig:- flipflop logic symbol.

* There are different types of flipflops are:-

1. SR flipflop
2. JK flipflop
3. D flipflop
4. T flipflop
5. Master-slave JK flipflop.

1. SR flipflop ckt using NAND gates:-

* The type of flipflop which has two inputs namely S (Set) or R (Reset) is termed as an "SR flipflop".

* If the 'S' and 'R' inputs of the flipflop control its outputs when a clock pulse is present, then it is called a clocked "SR flipflop".

* two o/p's Q & $\bar{Q}$, as four NAND gates are used. (7)

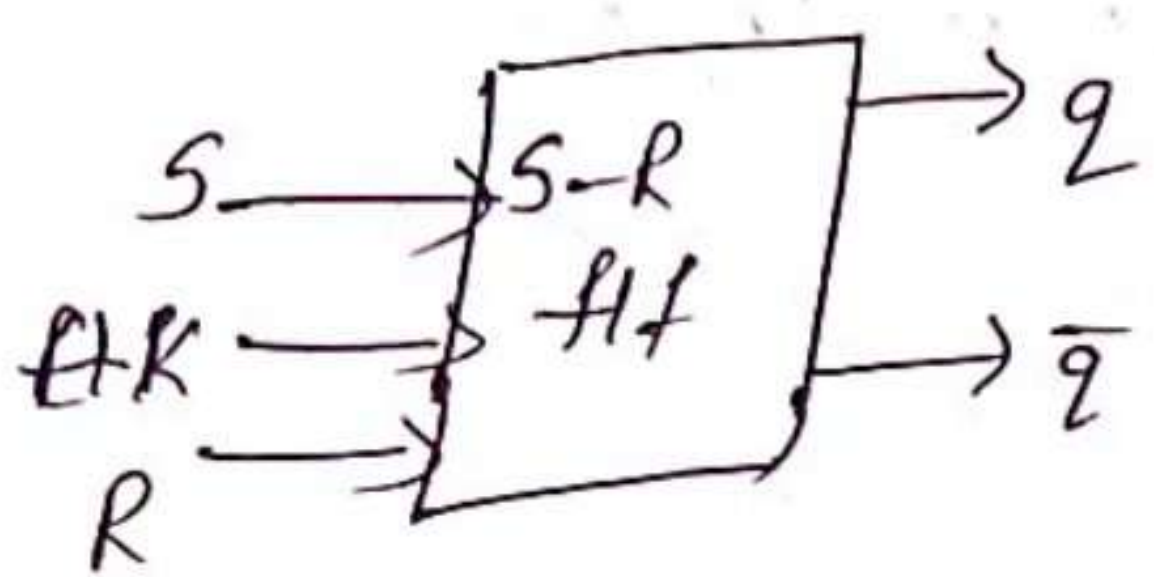


Fig:- Logic symbol

Logic diagram:-

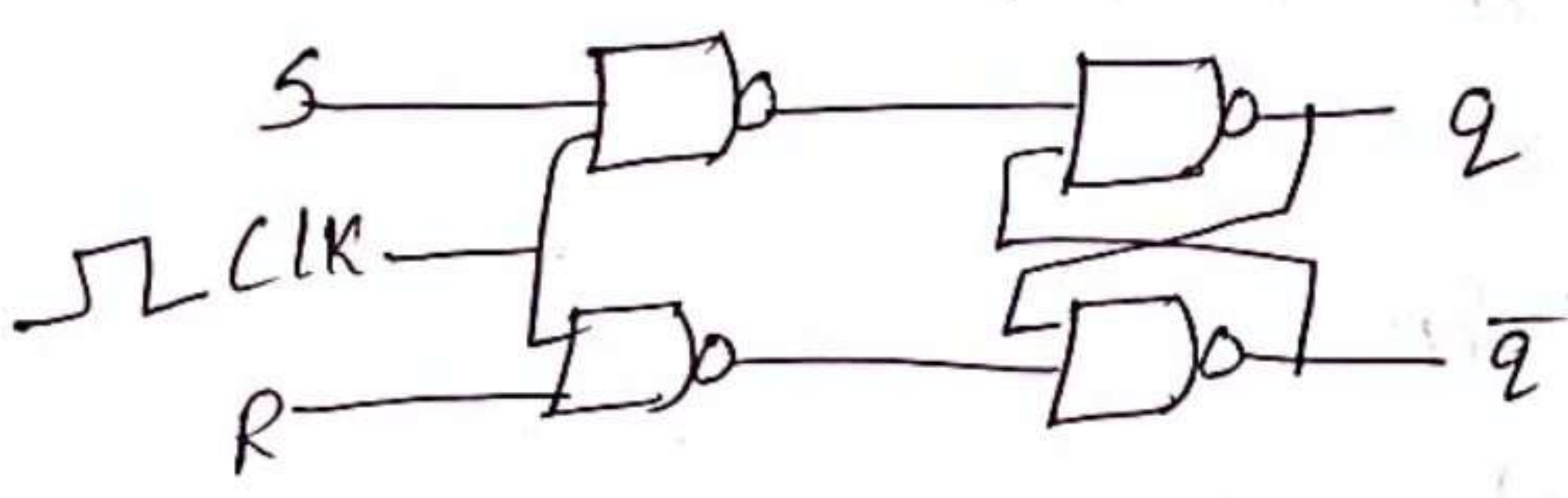


Fig:- SR FF by using NAND gates

NAND T.T

T/P		O/P
A	B	$Y = \overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

SR-FF T.T

T/P's		O/P's	
S	R	Q_n $\bar{Q}_n$	
0	0	Q_n Q_n	NO change
0	1	0 1	Reset
1	0	1 0	Set
1	1	1 1	Invalid

Case i): $S=0, R=0$

* Based on present o/p's q_n & $\bar{q}_n$, we get next stage o/p's, q_{n+1} & $\bar{q}_{n+1}$

$$q_{n+1} = 1 \cdot \bar{q}_n = \bar{q}_n = q_n$$

$$\bar{q}_{n+1} = 1 \cdot q_n = q_n$$

Case ii): $S=0, R=1$

$$q_{n+1} = 1 \cdot \bar{q}_n = \bar{q}_n = q_n = 0$$

$$\bar{q}_{n+1} = 0 \cdot q_n = 0 = 1$$

Case iii): $S=1, R=0$

$$q_{n+1} = 0 \cdot \bar{q}_n = 0 = 1$$

$$\bar{q}_{n+1} = 0$$

Case iv): $S=1, R=1$

$$q_{n+1} = 0 \cdot \bar{q}_n = 0 = 1$$

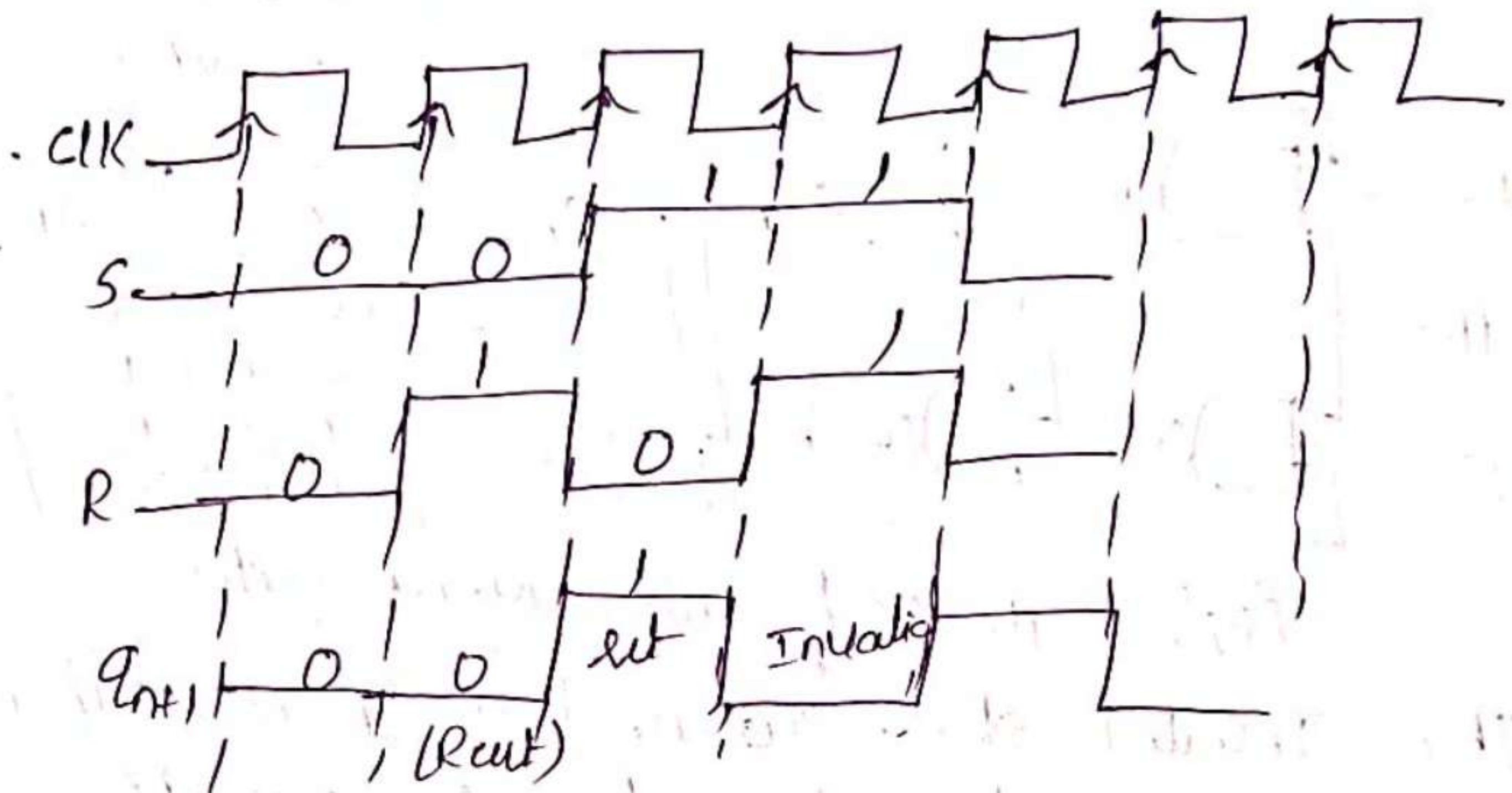
$$\bar{q}_{n+1} = 1 \cdot 0 = 0 = 1$$

SR FF T-T :-

S	R	q_{n+1}
0	0	$q_n \rightarrow$ No change
0	1	0 $\rightarrow$ Reset
1	0	1 $\rightarrow$ Set
1	1	X $\rightarrow$ Invalid

Timing diagram:-

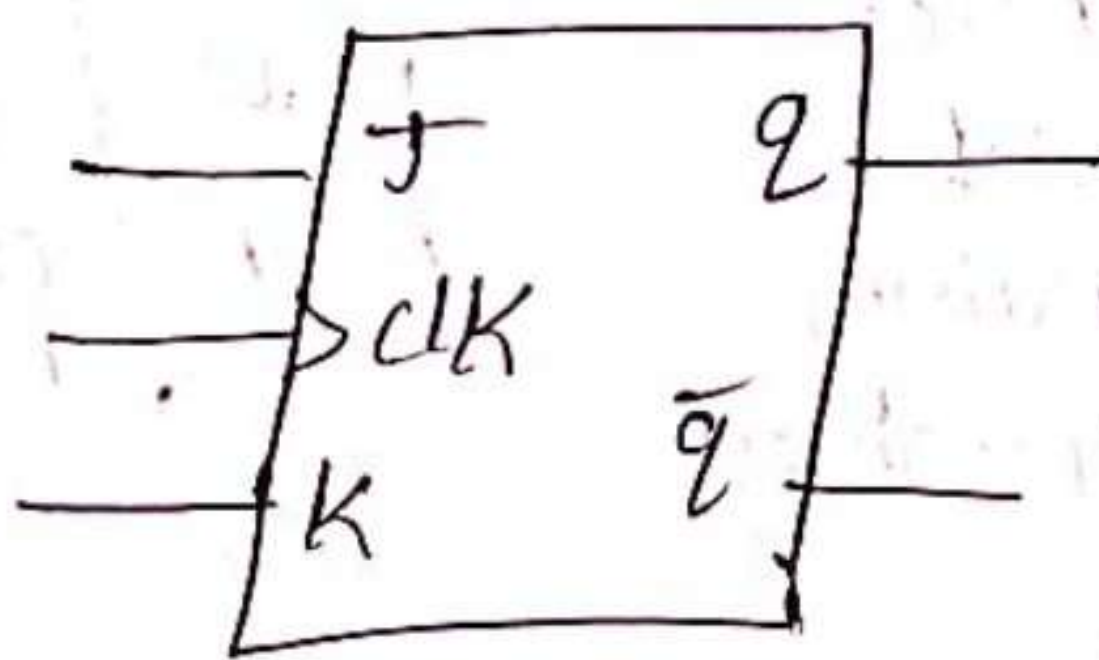
(8)



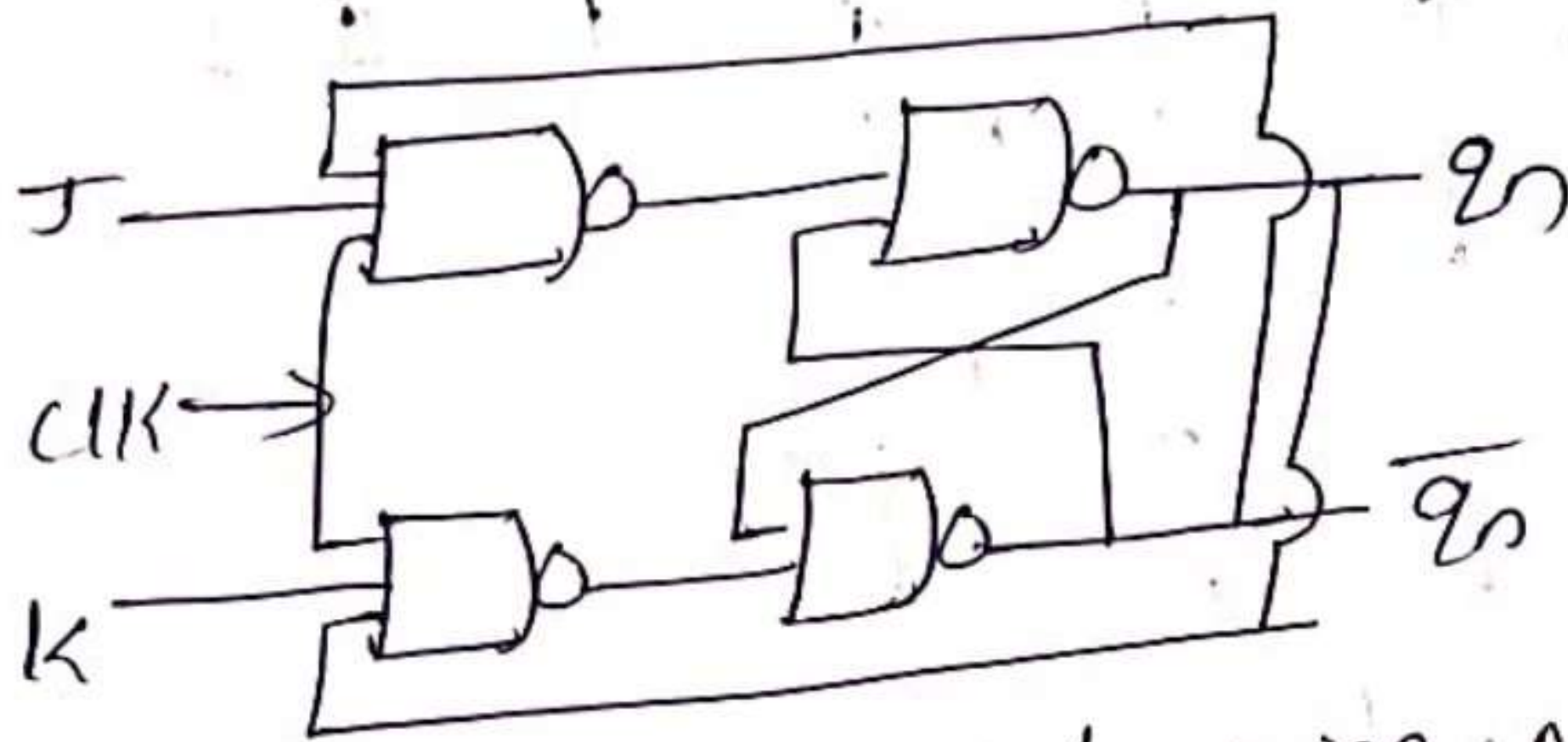
2. JK FlipFlop:-

JK FF is the Refinedment of SR FF to solve the problem of Indeterminate state. When both the TLP's are high

Logic Symbol:-



Logic diagram:-



NAND T-T

A	B	$Y = AB$
0	0	1
0	1	1
1	0	1
1	1	0

Fig:- JK ff by using NAND gates.

* The Invalid state occurred in SR ff, when $S=1, R=1$, can be eliminated by JK ff.

JK ff T-T:-

I/P's		O/P's		
J	K	Q_n	$\overline{Q_n}$	
0	0	Q_n	$\overline{Q_n}$	→ No change
0	1	0	1	→ Reset
1	0	1	0	→ Set
1	1	0	1	→ Raising operation

I/P		O/P
J	K	Q_{n+1}
0	0	No change
0	1	0 → Reset
1	0	1 → Set
1	1	0/1 → Raise operation

Case i):- $J=0, K=0$

$$Q_{n+1} = 1 \cdot \overline{Q_n} = \overline{Q_n} = Q_n$$

$$\overline{Q_{n+1}} = 1 \cdot Q_n = Q_n$$

Case ii):- $J=0, K=1$

$$Q_{n+1} = 1 \cdot \overline{Q_n} = \overline{Q_n} = Q_n \rightarrow 0 \text{ by Reset}$$

$$\overline{Q_{n+1}} = Q_n \cdot Q_n = 0 = 1$$

Case iii) $J=1, K=0$

(9)

$$\begin{aligned} q_{n+1} &= \overline{q_n} \cdot \overline{q_n} = \overline{0} = 1 \\ \overline{q_{n+1}} &= 1 \cdot 1 = 0 \end{aligned} \quad \left. \vphantom{\begin{aligned} q_{n+1} &= \overline{q_n} \cdot \overline{q_n} = \overline{0} = 1 \\ \overline{q_{n+1}} &= 1 \cdot 1 = 0 \end{aligned}} \right\} \text{set}$$

Case iv) :- $J=1, K=1$, Assume

a) $q_n=0, \overline{q_n}=1$

$$q_{n+1} = 0 \cdot \overline{q_n} = \overline{0} = 1$$

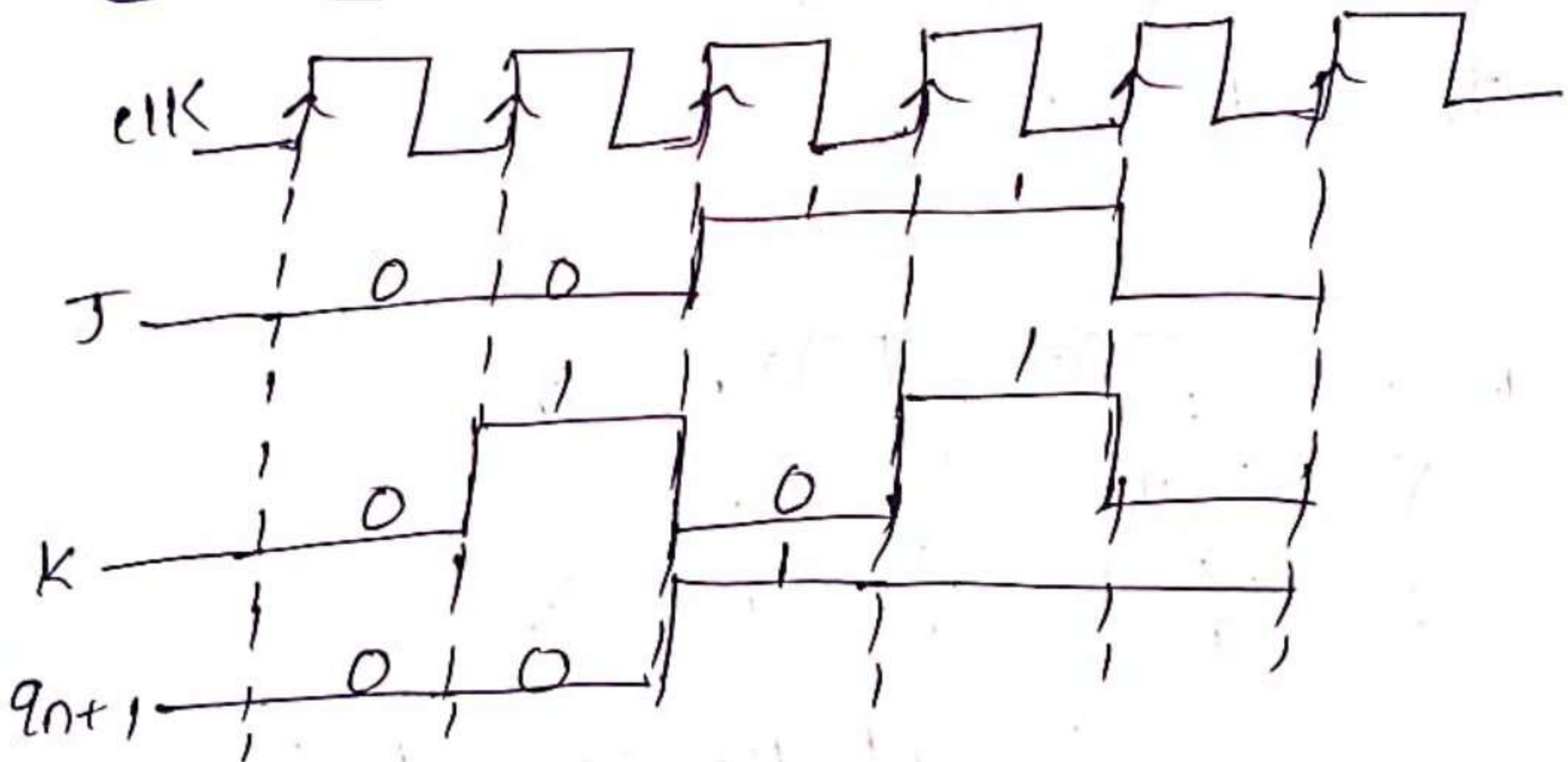
$$\overline{q_{n+1}} = 1 \cdot 1 = 0$$

c) b) $q_n=1, \overline{q_n}=0$

$$q_{n+1} = 1 \cdot \overline{q_n} = \overline{q_n} = q_n = 0$$

$$\overline{q_{n+1}} = 0 \cdot q_n = \overline{0} = 1$$

Wave forms:-



3-D FlipFlop:-

* D FlipFlop is obtained from SR flipFlop by connecting an Inverter b/w S & R inputs as shown in fig.

* hence D flip contains only one I/P (D), which stands for data.

* D flip is also called as delay (or) data flip. It is used to find delay b/w the set & Reset

Logic symbol:-

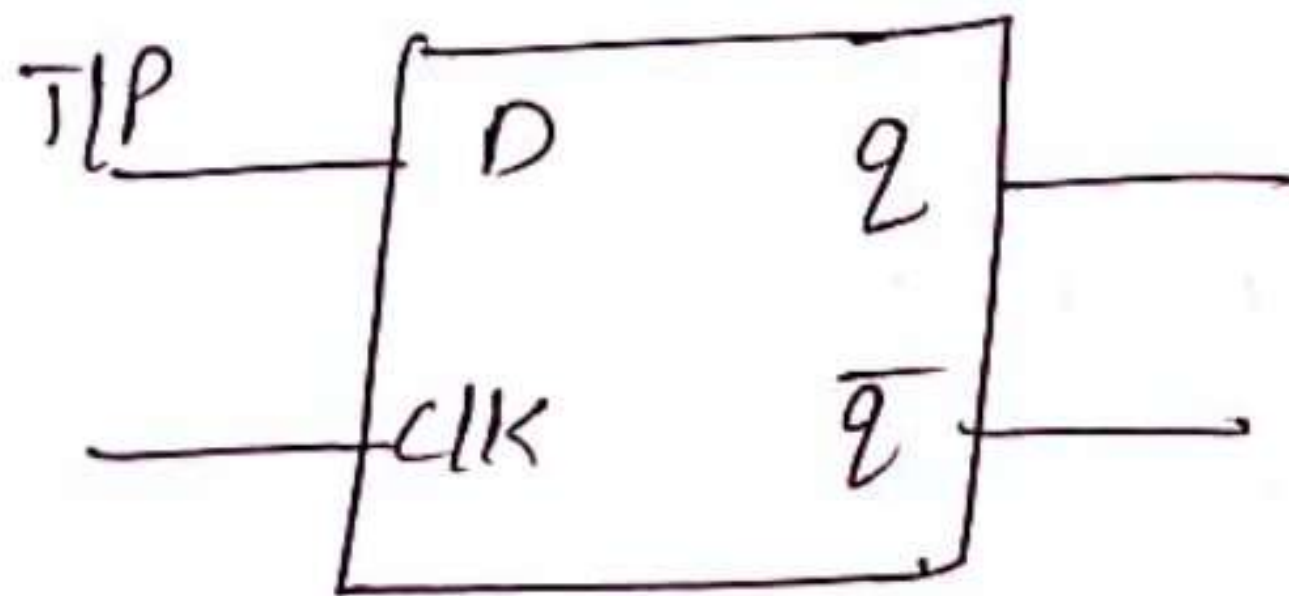


Fig:- Logic symbol

Logic diagram:-

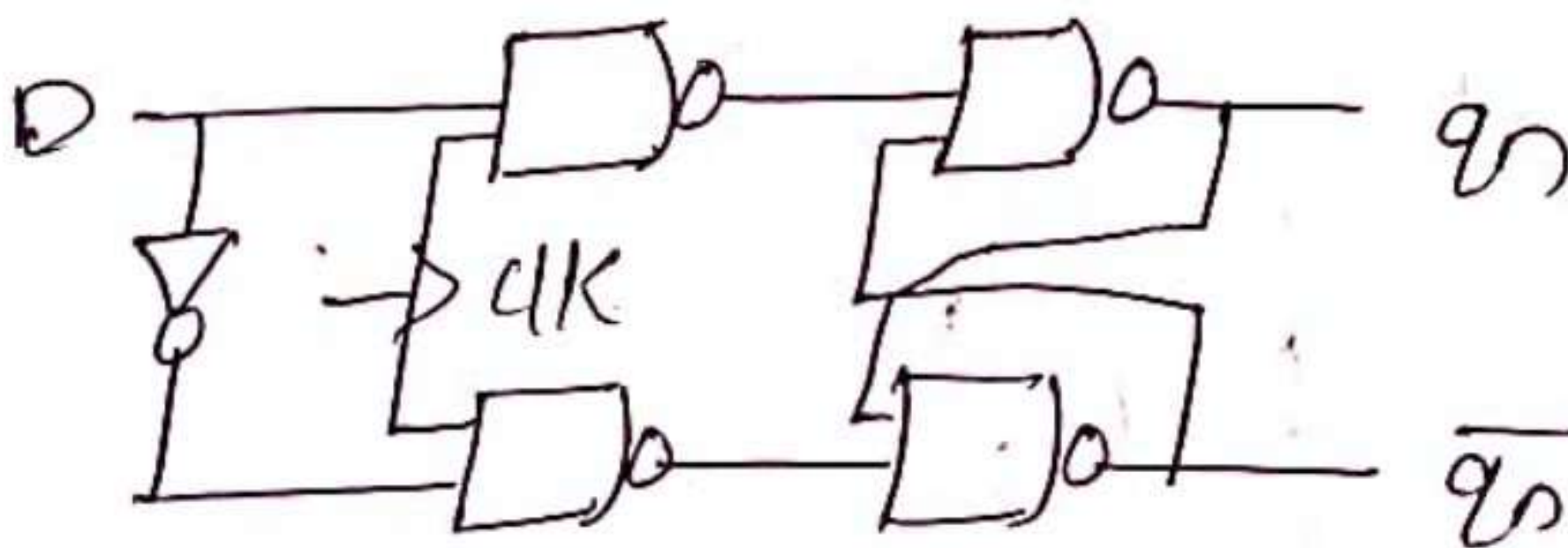


Fig:- ckt diagram of D flip

Case i):- $D=0$

$$Q_{n+1} = 1 \cdot \overline{Q_n} = \overline{Q_n} = Q_n \rightarrow 0$$

$$\overline{Q_{n+1}} = 0 \cdot Q_n = 0 = 1 \rightarrow 1$$

Result

(Case ii):- $D=1$

(10)

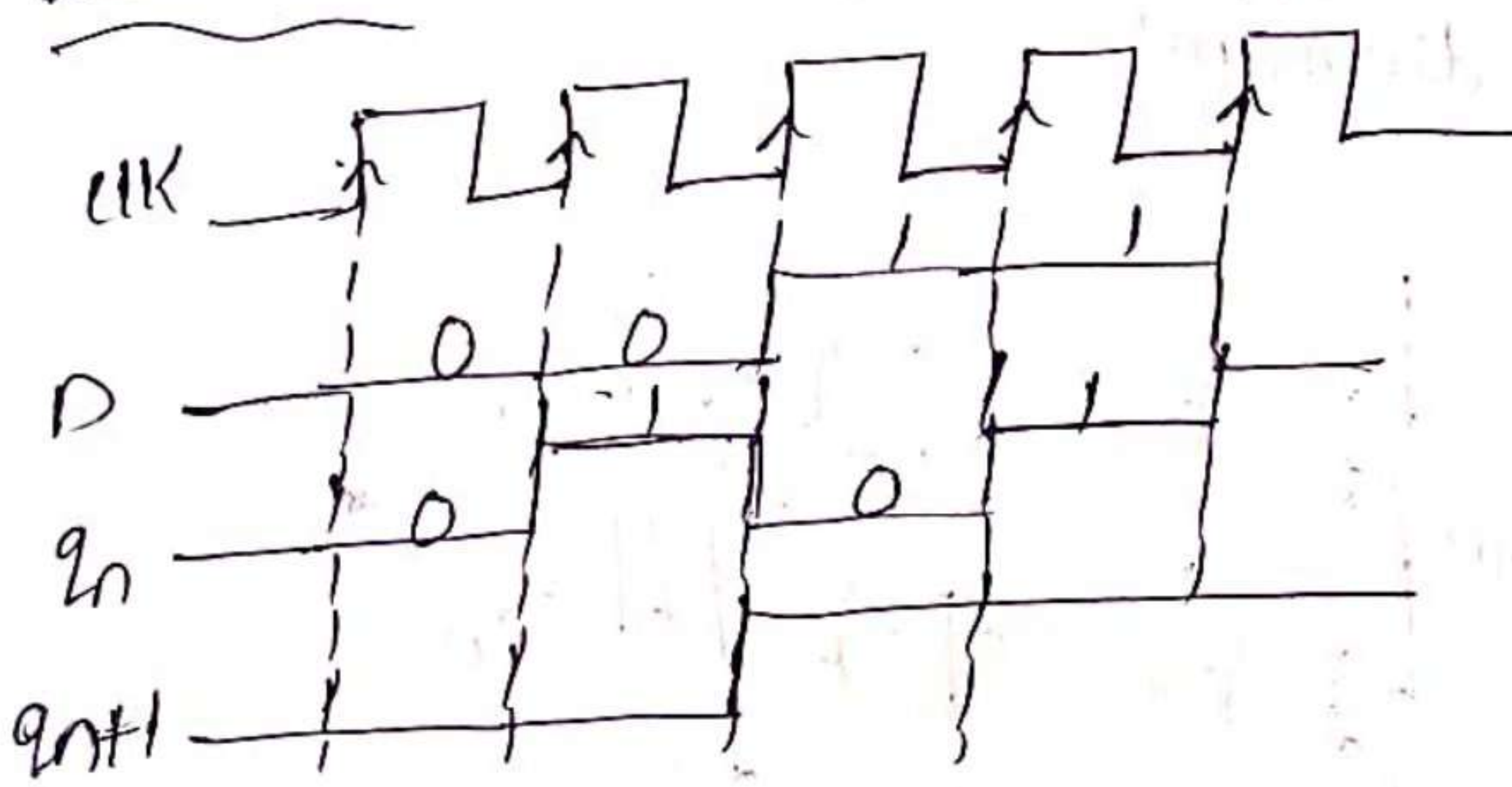
$$q_{n+1} = \overline{0} \cdot \overline{q_n} = \overline{0} = 1 \quad \text{set}$$

$$\overline{q_{n+1}} = 0$$

* T-T DFF:-
 I/P's $\rightarrow$ present stage o/p
 o/p's $\rightarrow$ Next stage o/p

D	q_n	q_{n+1}	
0	0	0	No change
0	1	0	reset
1	0	1	set
1	1	1	Invalid

Wave form:-



T-FlipFlop:-

* T FF is also known as Toggle-FF. It can be constructed from JK flipflop by connecting both T/P's 'J' & 'K' Together

* The single TLP version of JK FF is T FF. Generally, T FF IC are not available. It can be realized using JK, SR (or) D flip-flops.

logic symbol:-

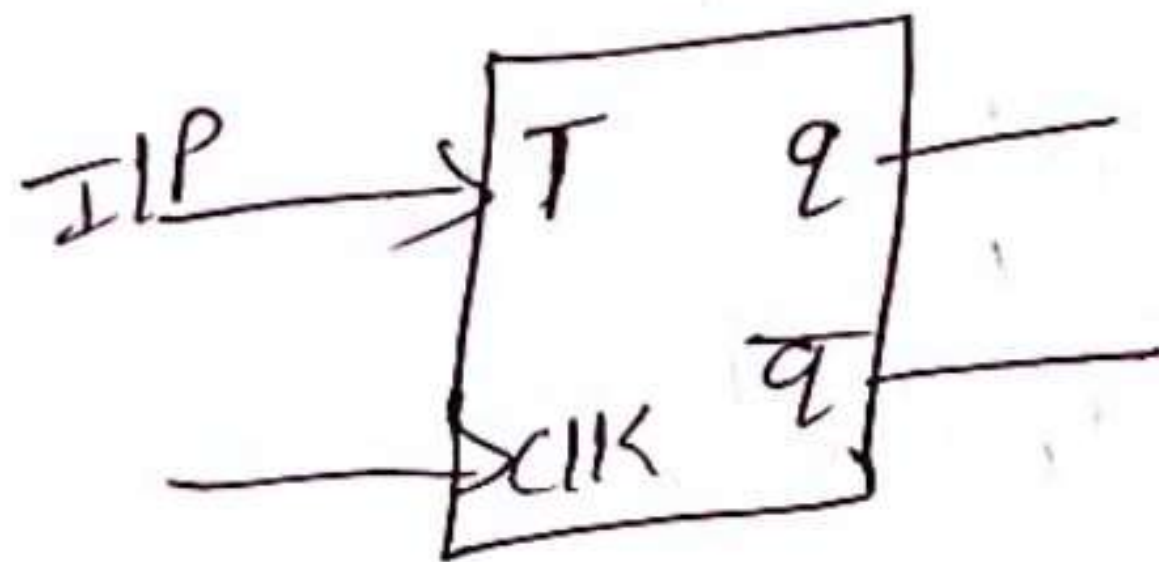


Fig:- logic symbol.

logic diagram:-

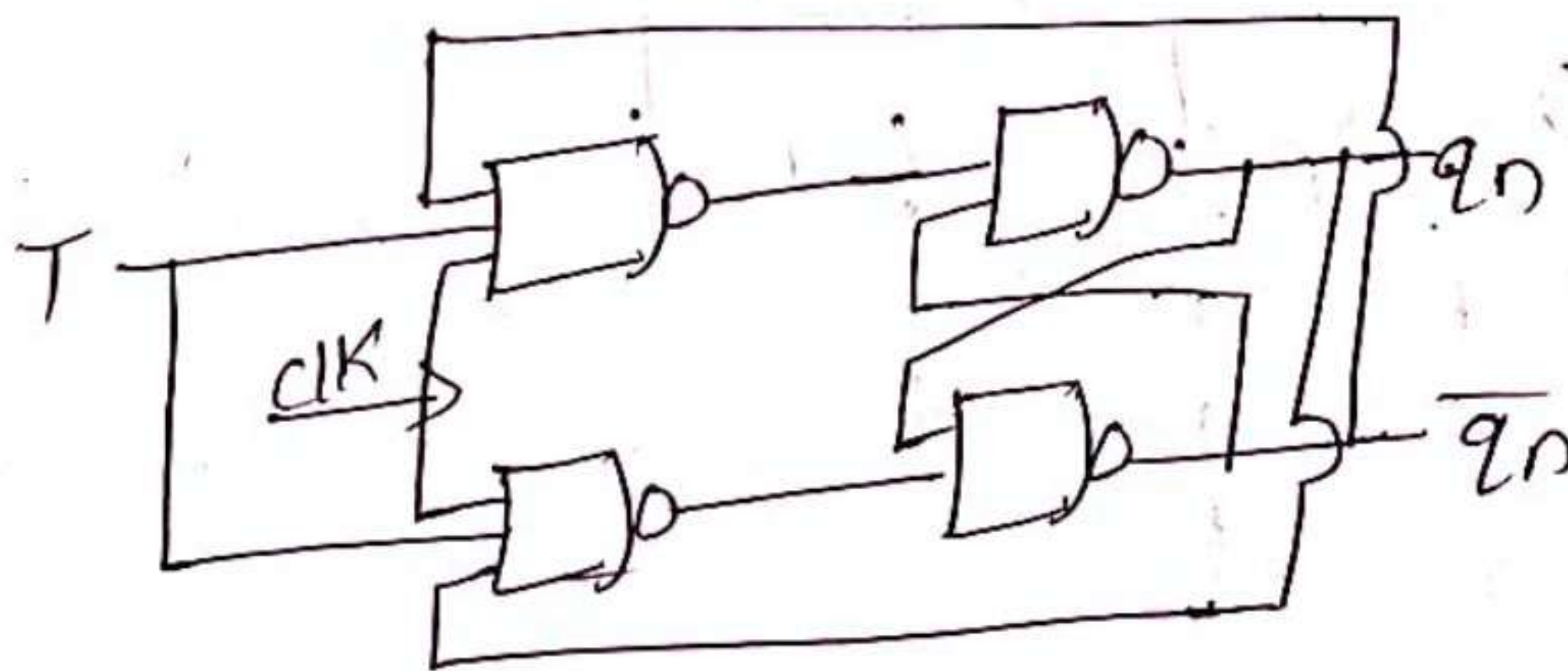


Fig:- T FF circuit diagram.

(aui):- $T=0$, Assume $q_n=0$
 $q_{n+1}=0$

(auii):- $T=0$, Assume $q_n=1$
 $q_{n+1}=1$

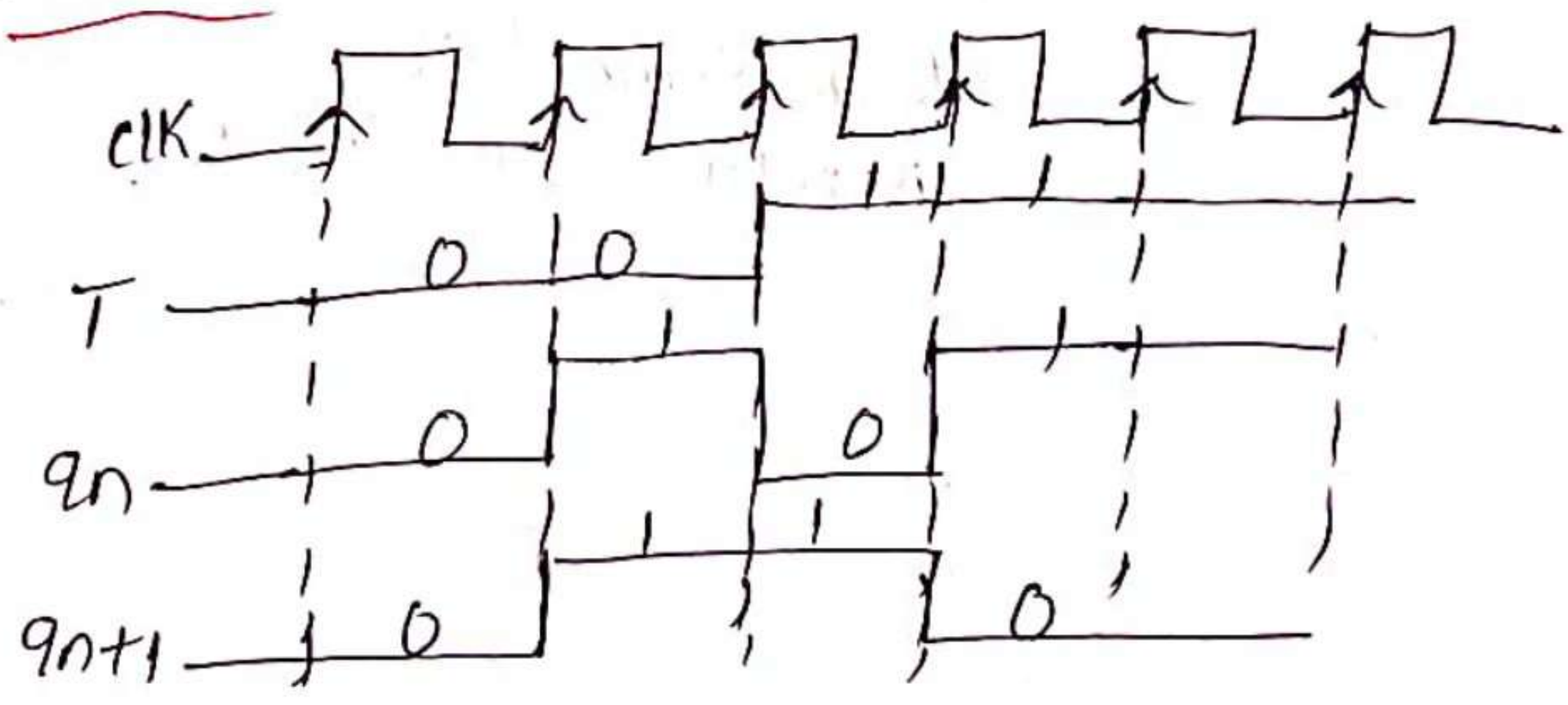
(auiii):- $T=1$, Assume $q_n=0$
 $q_{n+1}=1$

(auiV):- $T=1$, Assume $q_n=1$
 $q_{n+1}=0$

$T \neq T$

I/P's		O/P's
T	q_n	q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Waveform:-



Master-slave JK FF:-

- * The master-slave JK FF avoids the Race around problem.
- * It consists of a clocked JK flipflop as a Master and clocked SR flipflop as a slave.
- * The o/p of the Master FF is fed as an i/p to the slave FF. The o/p of the slave FF is connected as a third i/p of the Master JK FF.

Logic Symbol:-

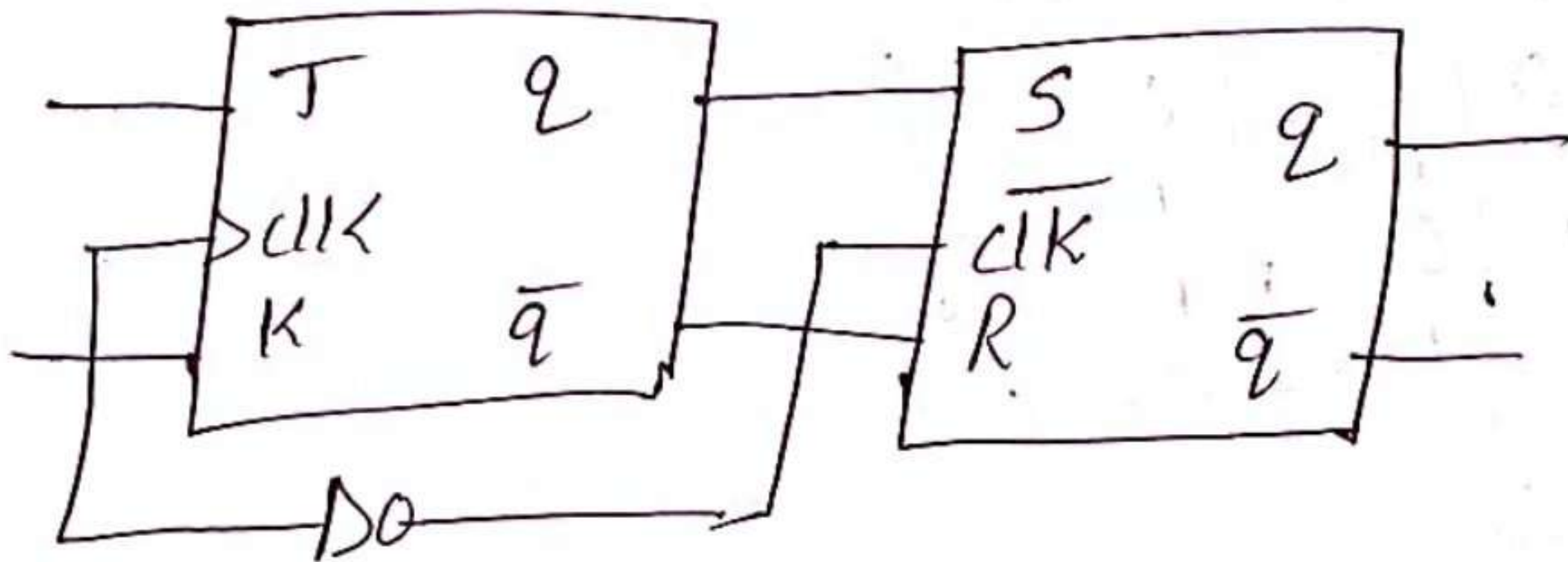


Fig:- Logic symbol of master-slave JK FF.

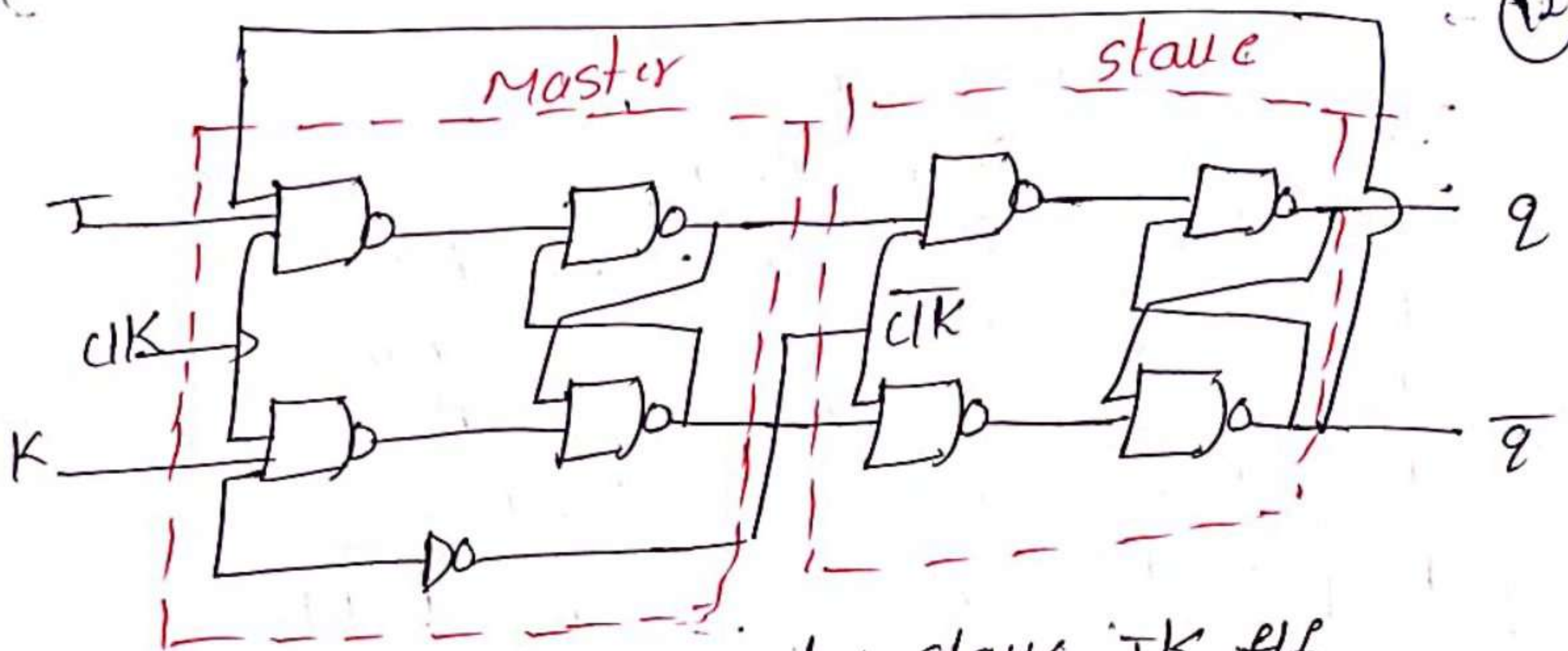


Fig. Master-slave JK flf.

Operation:-

- i) While the clock is high, the master is active and slave is inactive
- ii) While the clock is low, the master is inactive and slave is active.

JK flf T-T

clk	J	K	Q	Q̄
0	X	X	Q	Q̄
1	0	0	No change	
1	0	1	0	1 Reset
1	1	0	1	0 Set
1	1	1	0	1 Race
			1	0 Race

Toggle state

SR flf T-T

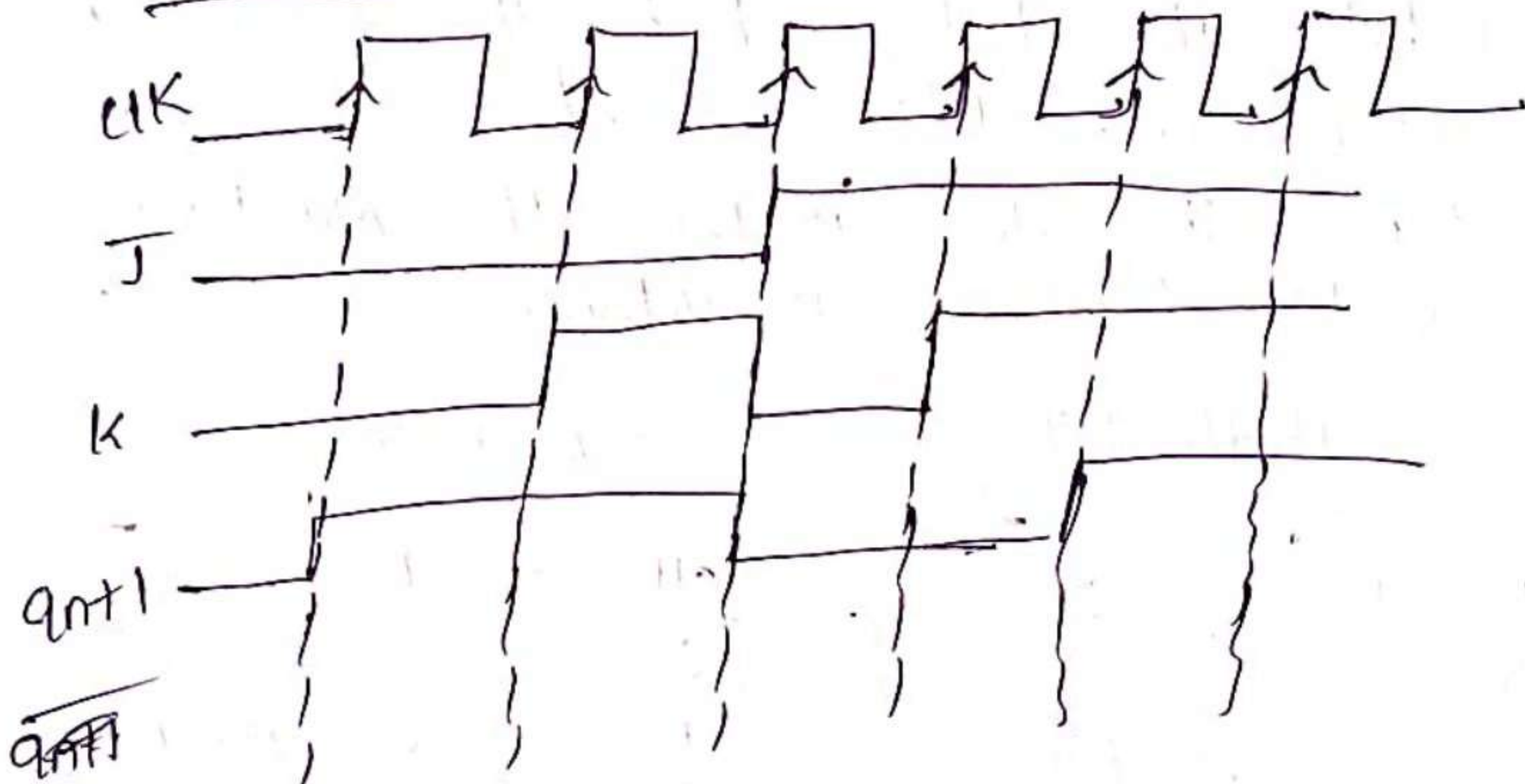
clk	S	R	Q	Q̄
0	X	X	Q	Q̄
1	0	0	No change	
1	0	1	0	1 → Reset
1	1	0	1	0 → Set
1	1	1	1	1 → Invalid

Master-slave JK FF T.T:-

I/P's			O/P's		
CLK	J	K	q_{n+1}	$\overline{q_{n+1}}$	
0	X	X	q_n	$\overline{q_n}$	→ No change
1	0	0	q_n	$\overline{q_n}$	→ No change
1	0	1	0	1	→ Reset
1	1	0	1	0	→ Set
1	1	1	$\overline{q_n}$	q_n	→ Toggle state

$\left\{ \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \right\}$

Waveform:-



Excitation tables of FF's:-

(13)

1) Excitation table of SR FF:-

Step 1:- Write the characteristic table of SR FF by using sr truth table

I/P's			O/P's
S	R	q_n	q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

→ No change
→ Reset
→ Set
→ Invalid

Step 2:- Write the excitation table of SR FF

I/P's		O/P's	
q_n	q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Step 3:- derive the characteristic equation of

S	q_n				
	00	01	11	10	
0	0	1	0	0	$\bar{R}q_n$
1	1	1	X	X	S

$$q_{n+1} = S + \bar{R}q_n$$

2. Excitation table of JK FF:-

Step 1:- Write the characteristic table of JK FF by using JK Truth table

I/P's			O/P's	
J	K	q_n	q_{n+1}	
0	0	0	0	→ No change
0	0	1	1	
0	1	0	0	→ Reset
0	1	1	0	
1	0	0	1	→ Set
1	0	1	1	
1	1	0	1	→ Toggle state
1	1	1	0	

Step 2:- Write the excitation table of JK FF

I/P's		O/P's	
q_n	q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Step 3:- derive the characteristic equation of q_{n+1}

J \ q_n	0	1	1	1
	0	1	1	1
0	0	0	1	1
1	0	1	1	1

$$q_{n+1} = J\bar{q}_n + Kq_n$$

3) D flip excitation table:-

(1H)

Step 1:- Write the characteristic table of D flip by using D flip T-T

Step 2:-

I/P's		O/P
D	q_n	q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

Step 2:- Write the excitation table of D flip

I/P's		O/P's
q_n	q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Step 3:- Write the characteristic equation of q_{n+1}

D	q_n
0	0
1	1

$$\therefore q_{n+1} = D$$

4) Excitation table of T FF:-

Step 1:- Write the characteristic table of T FF by using T FF T.T

I/P's		O/P's
T	q_n	q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Step 2:- Write the excitation table of T FF

I/P's		O/P's
q_n	q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Step 3:- Write the characteristic equation of q_{n+1}

T	q_n	0	1
0			1
1		1	

$$q_{n+1} = T \bar{q}_n + \bar{T} q_n$$

V.Vimp 5M
 Conversion of FF's:-

(15)

SR to other FF Conversions:-

1. SR to JK FF Conversion:-

Step 1:- Identify the available (SR) or Required (JK) FlipFlops

Step 2:- Write the excitation table of available (SR) FF

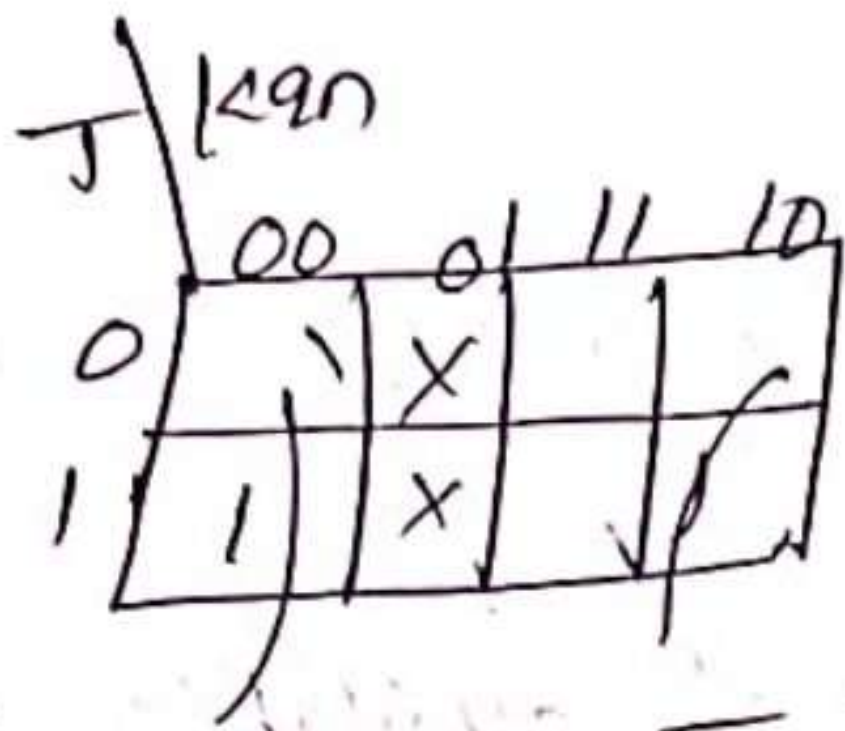
I/P's		O/P's	
q_n	q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Step 3:- Write the characteristic table of JK FlipFlop [Required FF]

I/P's			O/P's		
J	K	q_n	q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

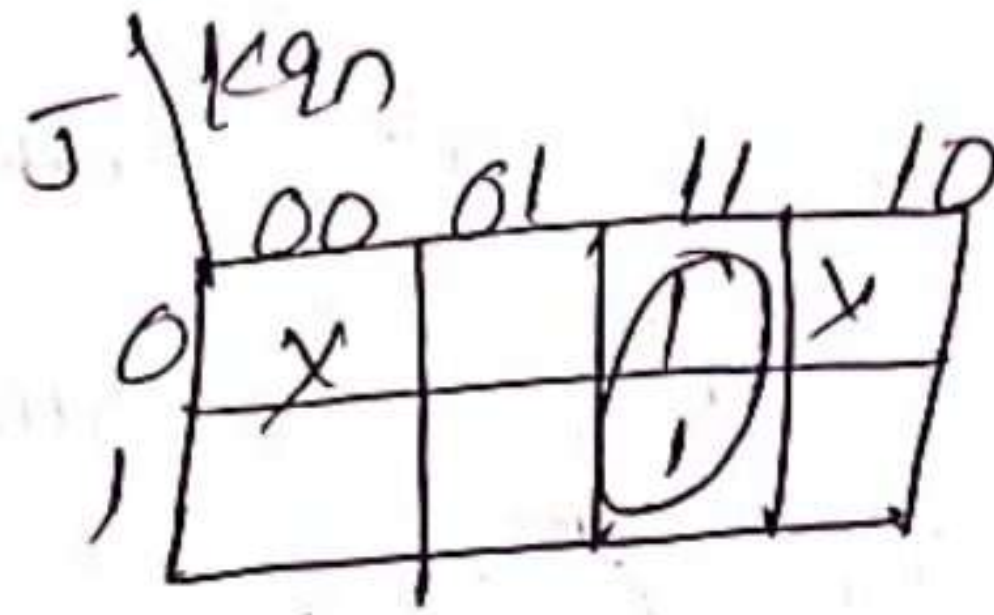
Step 4:- Find Boolean Expression S & R

K-Map for S



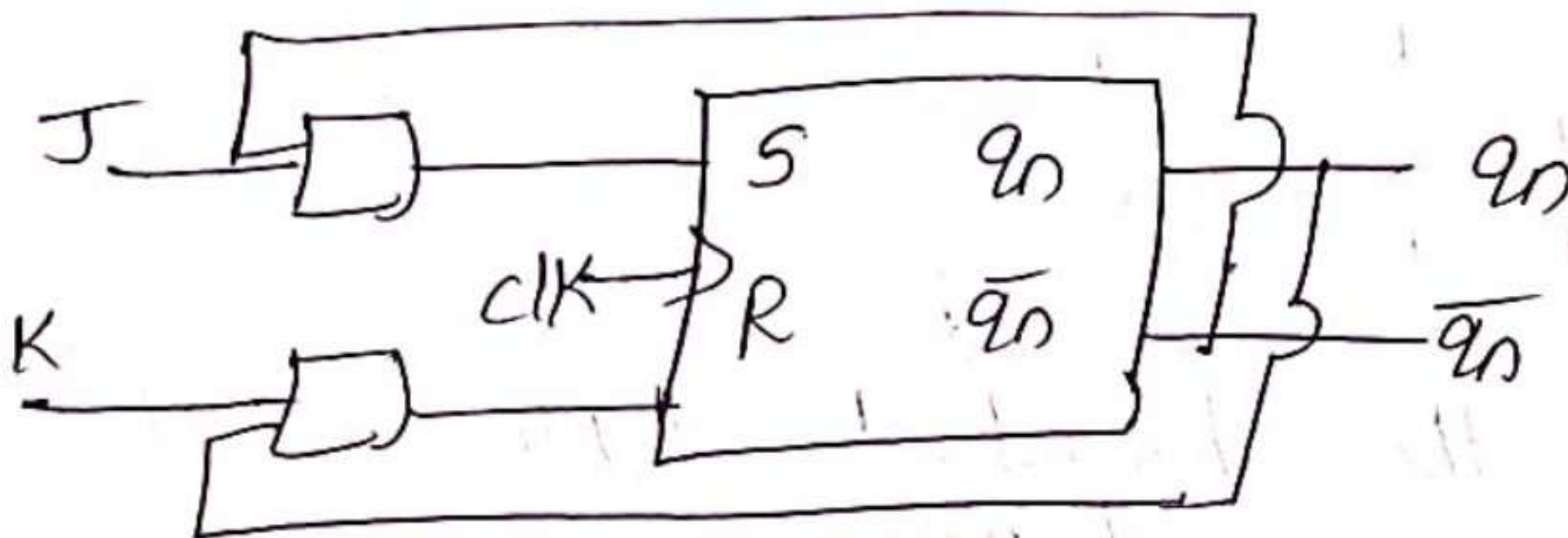
$$S = J \bar{q}_n$$

K-Map for R



$$R = Kq_n$$

Step 5:- Draw the ckt diagram



2) Convert SR FF to D FF:-

Step 1:- Identify the available (SR) & Required (D) FlipFlop

Step 2:- draw the Excitation table of SR FF

q_n	q_{n+1}	inp's	
		S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Step 3:- Write the characteristic equation of $\bar{D}$ FF (16)

I/P's		O/P's		
D	q_n	q_{n+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

Step 4:- Find the K-map for S and R

K-map for S

D \ q_n	0	1
0		
1	1	X

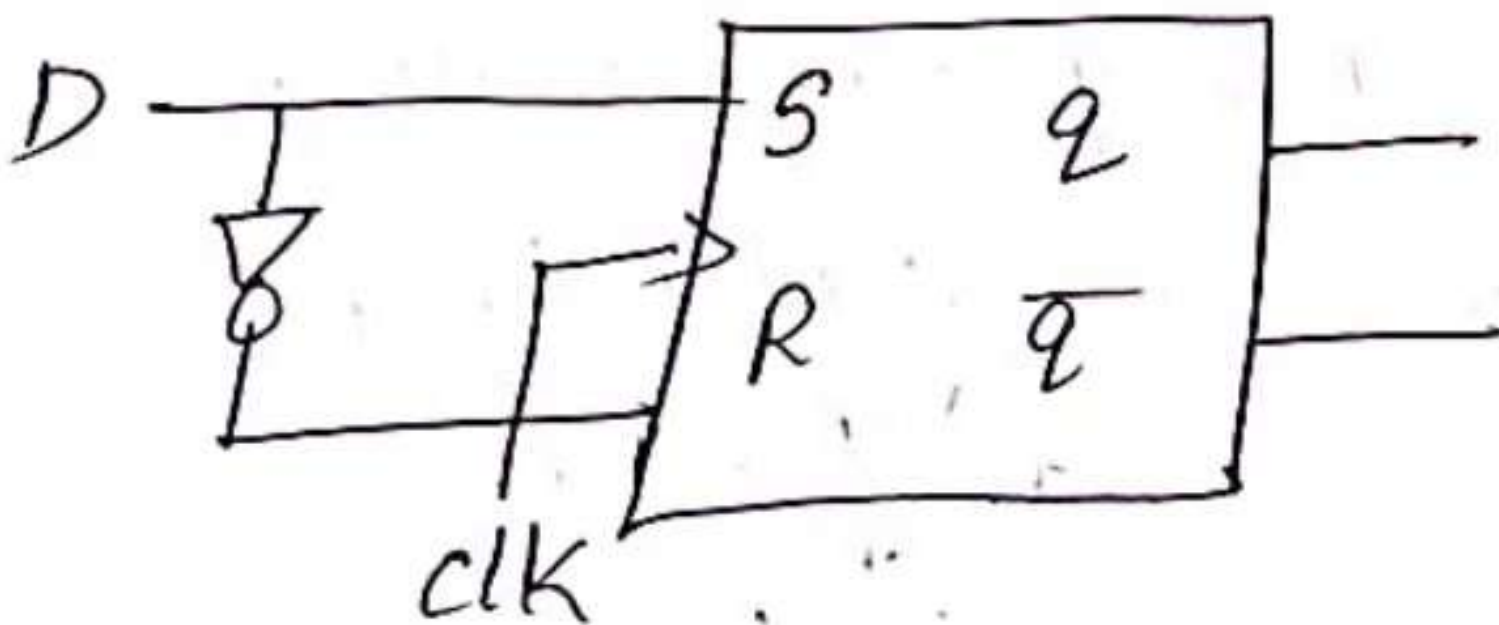
$$S = D$$

K-map for R

D \ q_n	0	1
0	X	1
1		

$$R = \bar{D}$$

Step 5:- Draw the ckt diagram of



3) Conversion of SR to T FF:-

Step 1:- Identify the available (SR) & Required (T) FF

Step 2:- Write the excitation table of SR FF

I/P's		O/P's	
q_n	q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Step 3:- Write the characteristic table of T FF

T	I/P's		O/P's	
	q_n	q_{n+1}	S	R
0	0	0	0	X
0	1	1	X	0
1	0	1	1	0
1	1	0	0	1

Step 4:- Find the K-map for 'S' & 'R'

K-map for 'S'

T \ q_n	0	1
0		X
1	1	

$$S = T \bar{q}_n$$

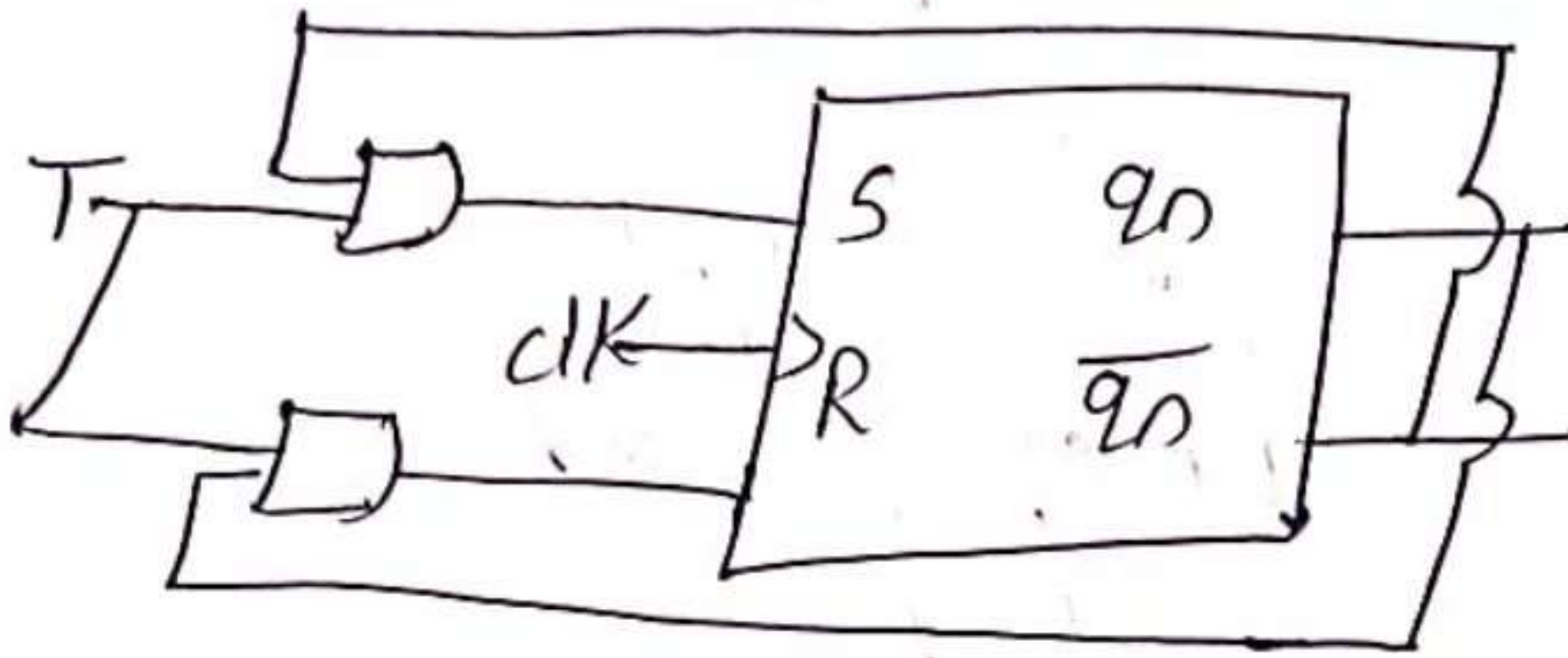
K-map for R

T \ q_n	0	1
0	X	
1		1

$$R = T q_n$$

step 5:- draw the ckt diagram

(17)



JK flipflop to other conversions:-

1) JK to SR flip conversion:-

step 1:- Identify the available (JK) & Required (SR) flipflop

step 2:- Write the excitation table for available (JK) flipflop

I/P's		O/P's	
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

step 3:- Write the characteristic table for SR flipflop

I/P's			O/P's		
S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	0	X	X	X
1	1	1	X	X	X

Step 4:- find Boolean Equations:

K-Map for J

S \ Rq _n	00	01	11	10
0		X	X	
1	1	X	X	X

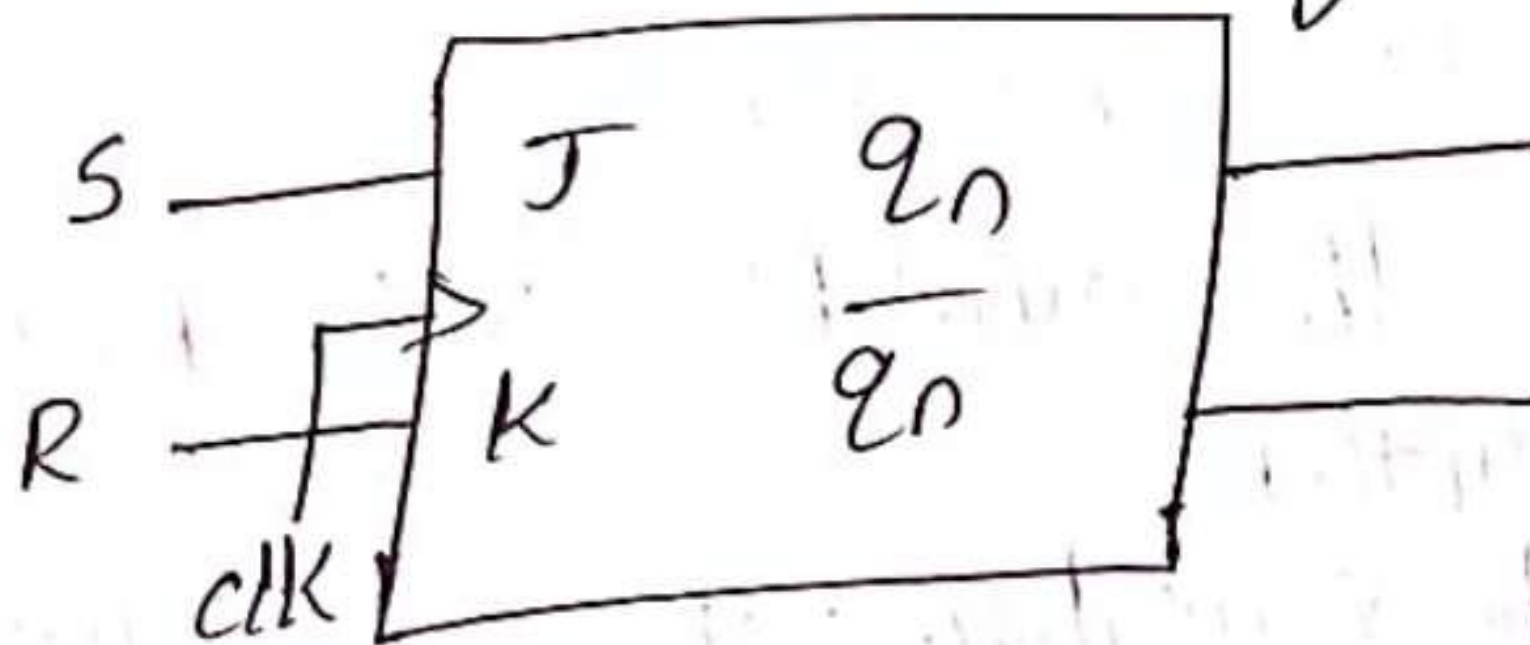
$$J = S$$

K-Map for K

S \ Rq _n	00	01	11	10
0			1	X
1	X		X	X

$$K = R$$

Step 5:- Draw the ckt diagram



2. JK to D Flipflop:-

Step 1:- Identify the available (JK) & Required (D) Flipflops

Step 2:- Write excitation table of JK flipflop

I/P's		O/P's	
Q _n	Q _{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Step 3:- Write the characteristic table of D-flip

I/P's		O/P's	
D	Q _n	Q _{n+1}	J K
0	0	0	0 X
0	1	0	X 1
1	0	1	1 X
1	1	1	X 0

Step 4:- find the Boolean expressions

(15)

K-Map for J

K-Map for K

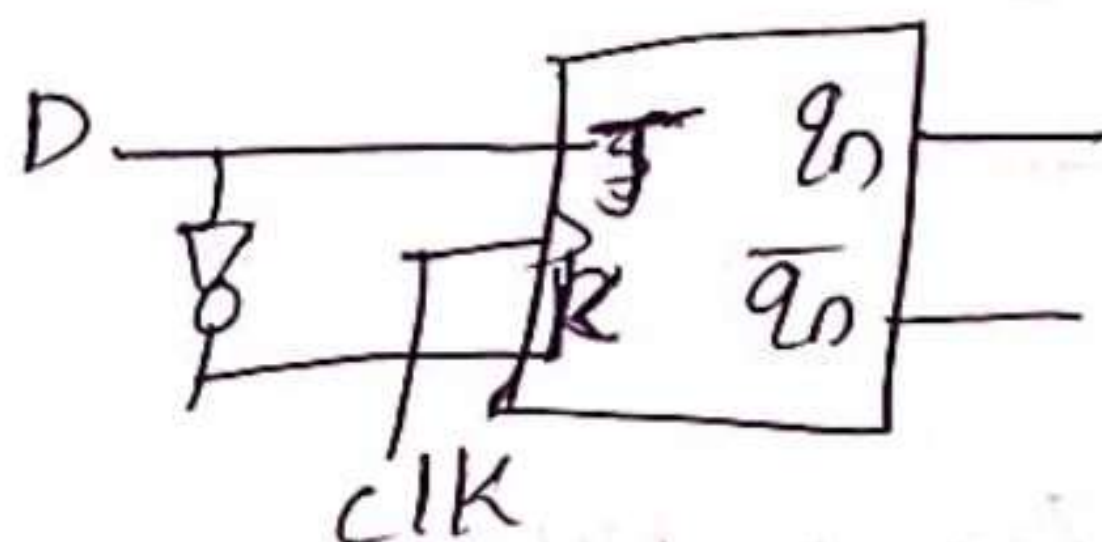
D \ q _n	0	1
0		X
1	1	X

$$J = D$$

p \ q _n	0	1
0	X	1
1	X	

$$K = \bar{D}$$

Step 5:- Draw the ckt diagram



3) JK to T flip conversions:-

Step 1:- Identify the available (JK) or Required (T) flipflops.

Step 2:- Write the excitation table of (JK) flip.

I/P's		O/P's	
q _n	q _{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Step 3:- Write the characteristic table for T flip

I/P's		O/P's	
T	q _n	q _{n+1}	J K
0	0	0	0 X
0	1	1	X 0
1	0	1	1 X
1	1	0	X 1

Step 4:- find Boolean Equations

K-Map for J

T \ q_n	0	1
0		X
1	1	X

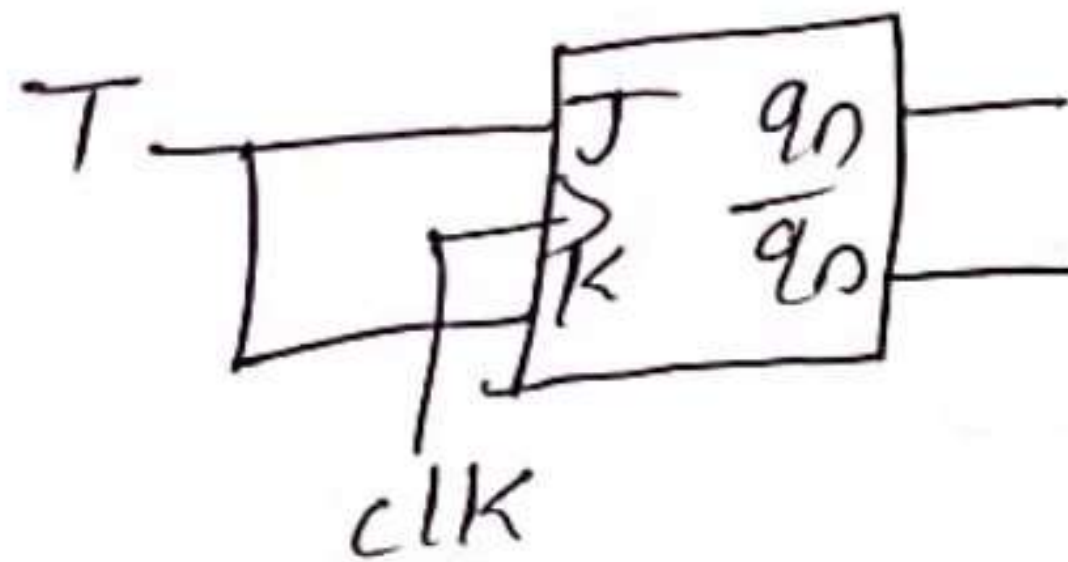
$$J = 1$$

K-Map for K

T \ q_n	0	1
0	X	
1	X	1

$$K = T$$

Step 5:- Draw the ckt diagram.



D FlipFlop To other conversions:-

D to SR flf Conversion:-

Step 1:- Identify the available (D) & Required (SR) flf's

Step 2:- Write the excitation table for D flf's

I/P's		O/P's
q_n	q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Step 3:- Write the characteristic table for SR FlipFlop.

(19)

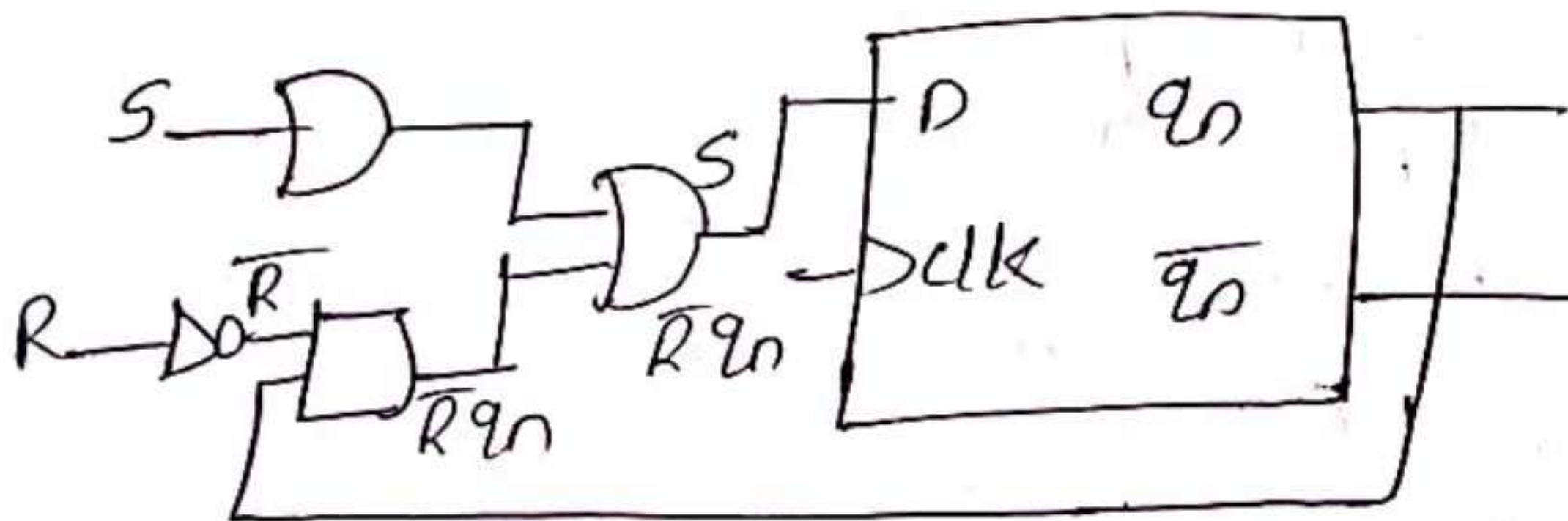
I/P's			O/P's	
S	R	q_n	q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	X	X
1	1	1	X	X

step 4:- Find the Boolean Expressions
K-Map for D

S	Rq_n			
	00	01	11	10
0		1		
1	1	1	X	X

$$D = S + \bar{R}q_n$$

step 5:- Draw the circuit diagram.



2. D to JK FF Conversion:-

Step 1:- Identify the available (D) we Required
JK FF

Step 2:- Write the Excitation table for ~~JK~~ FF

I/p's		O/p's
q_n	q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Step 3:- Write the characteristic table for JK
FF

I/p's			O/p's	
J	K	q_n	q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

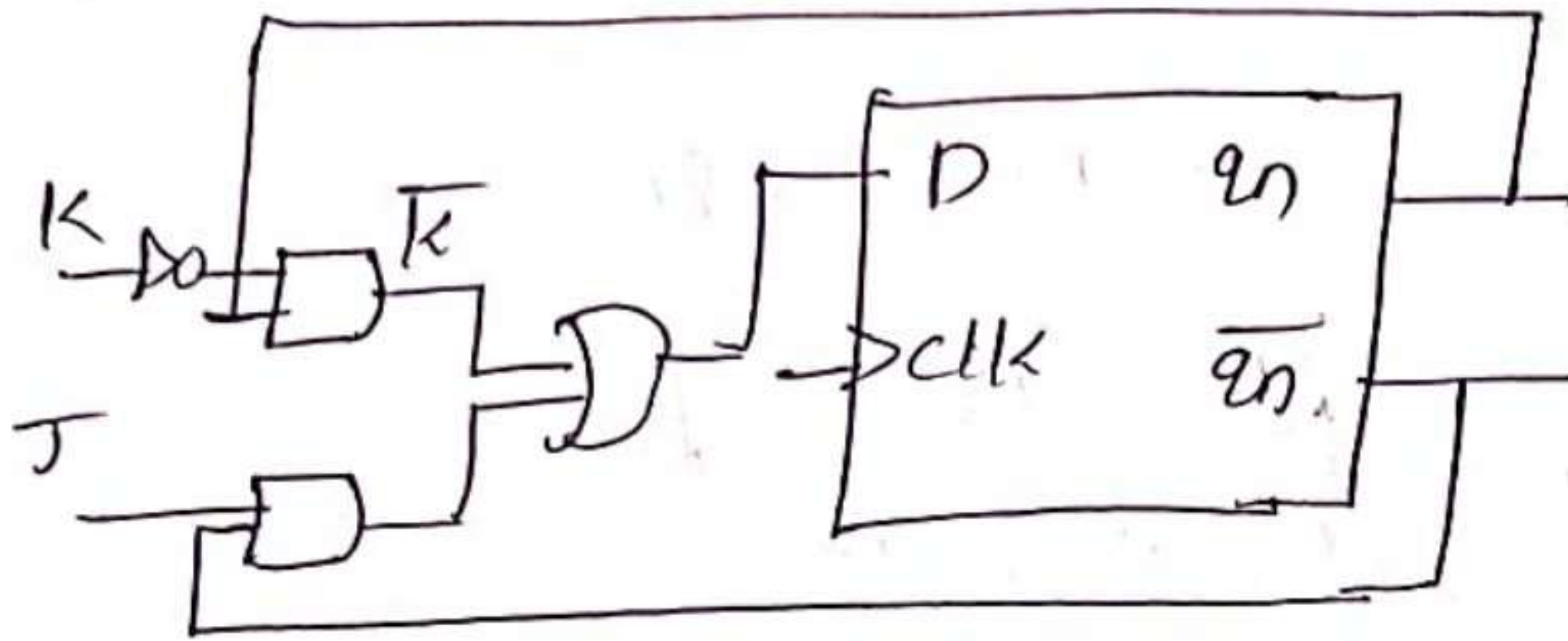
Step 4:- Find Boolean Expressions
K-Map for D

J	K q_n			
	00	01	11	10
0	0	1	0	0
1	1	1	1	0

$$D = \overline{K}q_n + J\overline{q}_n$$

Step 5:- Draw the ckt diagram.

(20)



3. D to T flip conversions:-

Step 1:- Identify the available (D) w. Required (T) flip flop.

Step 2:- Write ^{simulation} characteristic table for 'D' flip

I/P's		O/P's
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Step 3:- Write the characteristic table for 'T' flip

I/P's		O/P's
T	Q_n	Q_{n+1} D
0	0	0 0
0	1	1 1
1	0	1 1
1	1	0 0

Step 4:- find the Boolean expressions

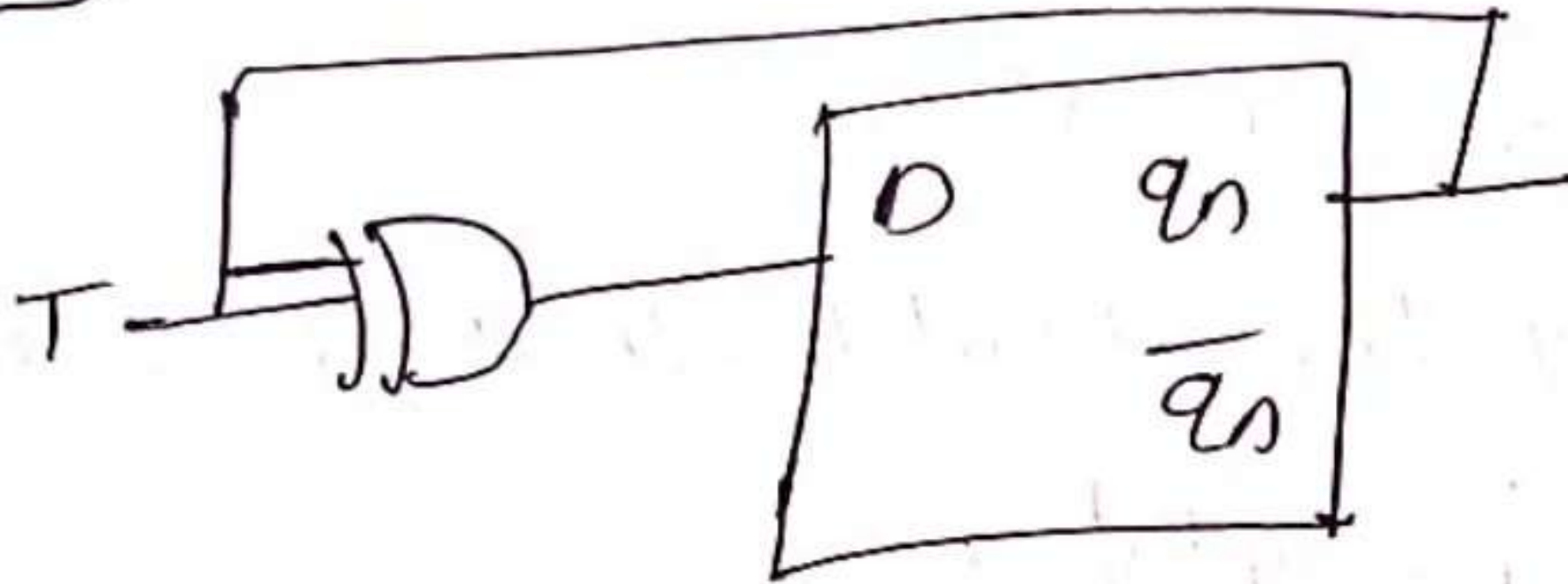
sk map for D

$T \backslash q_n$	0	1
0	0	1
1	1	0

$$D = T \bar{q}_n + \bar{T} q_n$$

$$D = T \oplus q_n$$

step 5:- Draw the ckt diagram



T ff to other conversions:-

1. T to SR ff conversion:-

step 1:- Identify the available (T) we Required (SR)

step 2:- Write the excitation table for T ff's

I/P's		O/P's
q_n	q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

step 3:- Write characteristic table for 'SR ff'

I/P's			O/P's	
S	R	q_n	q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1

(21)

1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	X	X
1	1	1	1	X	X

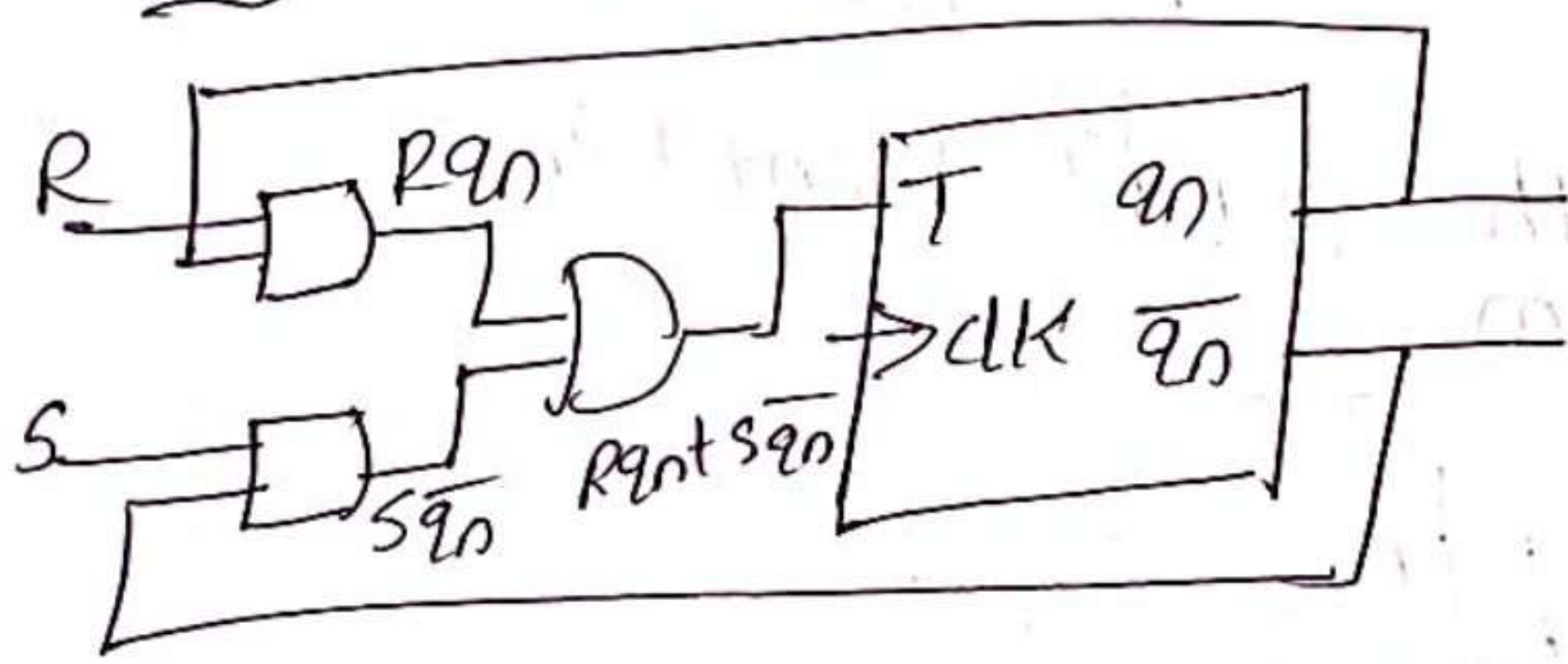
Step 4:- Find Boolean Expressions

K-Map for T

S \ Rq _n	00	01	11	10
0			1	
1	1		X	X

$$T = Rq_n + S\bar{q}_n$$

Step 5:- Draw the ckt diagram



2. T to JK HF Conversions:-

Step 1:- Identify the available (T) w Required CLK FlipFlops.

Step 2:- Write the excitation table for T FlipFlop

q _n	q _{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Step 3:- Write the characteristic table for JK (HA)

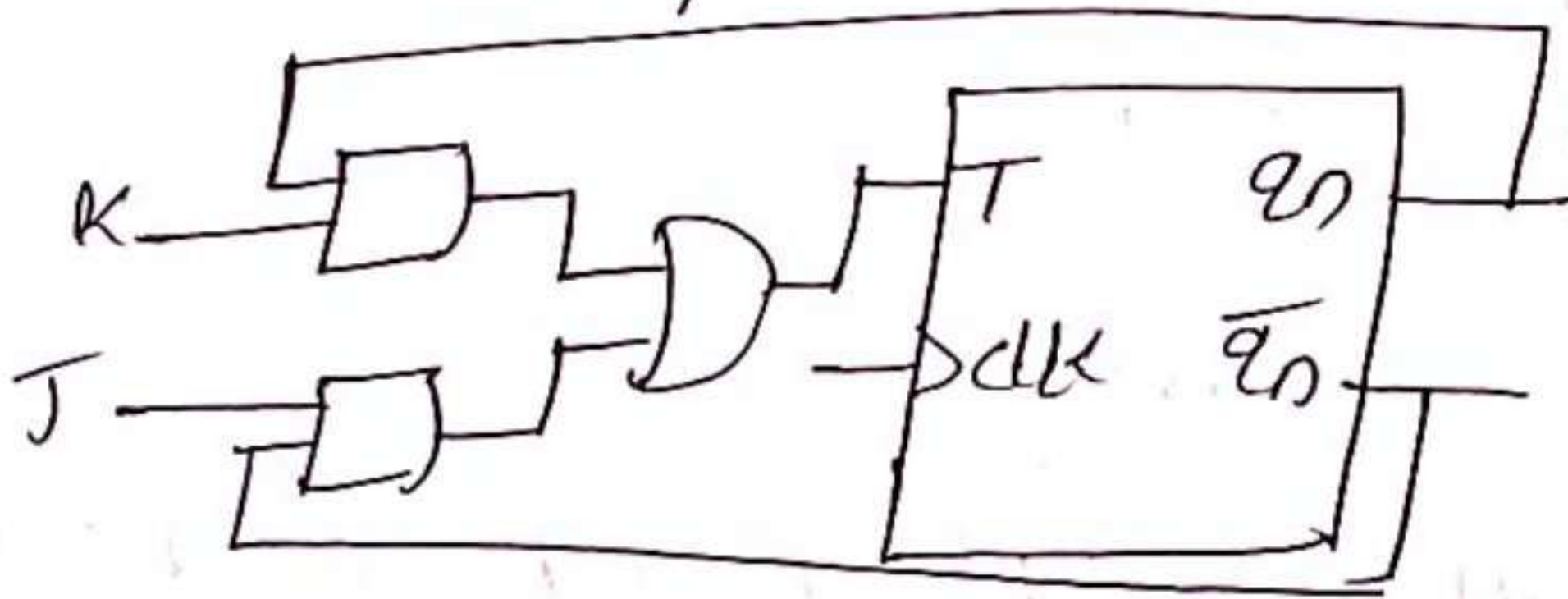
I/Ps			O/Ps	
J	K	q_n	q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1

Step 4:- Find Boolean Expression for T
K-Map for T:-

J	K q_n			
	00	01	11	10
0			1	1
1	1		1	1

Step 5:- Draw the circuit diagram

$$T = Kq_n + J\bar{q}_n$$



T to D Flip-Flop Conversion:-

Step 1:- Identify the available (T) or Required (D) flip flops

Step 2:- Write the excitation table for T ff

I/P's		O/P's
q_n	q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Step 3:- Write the characteristic table for D (ff)

D	I/P's		O/P's	
	q_n	q_{n+1}	T	
0	0	0	0	
0	1	0	1	
1	0	1	1	
1	1	1	0	

Step 4:- find Boolean expression

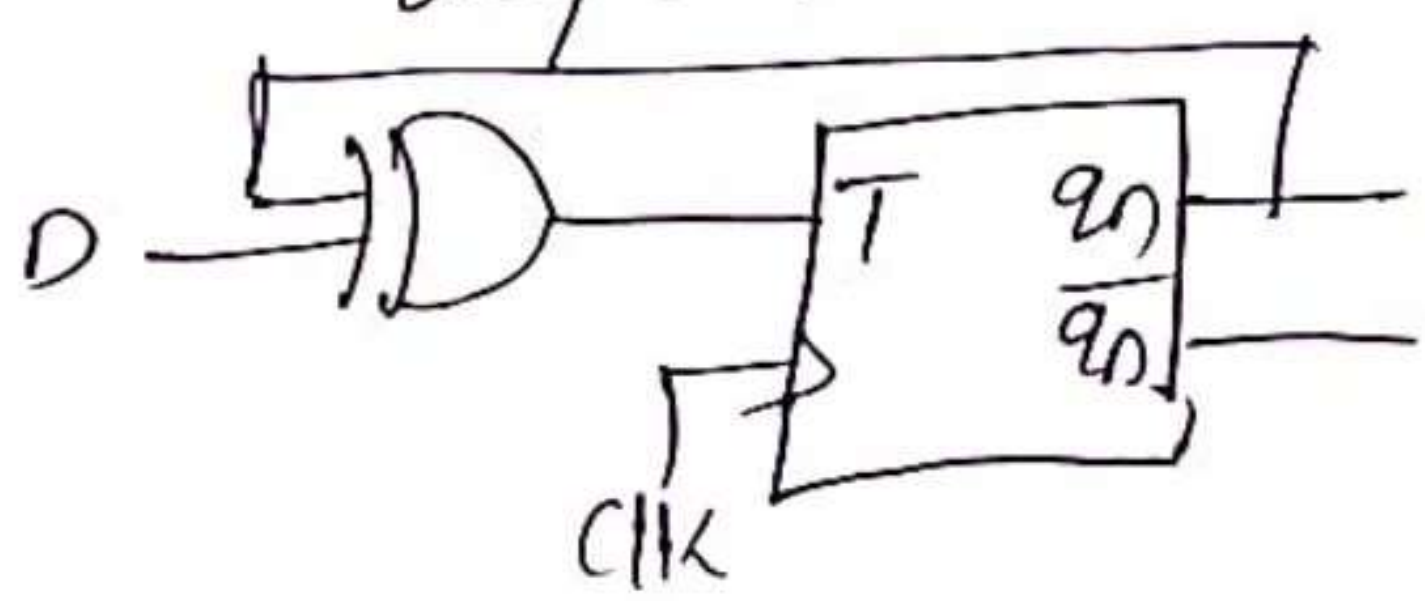
K-map for T:-

D \ q_n	0	1
0	0	1
1	1	0

$$T = \overline{D}q_n + D\overline{q_n}$$

$$= D \oplus q_n$$

Step 5:- draw the ckt diagram.



* ^{2M} Registers:-

Def:- flipflop is '1'-bit Memory cell. To Increase the storage capacity. We have to use group of flipflop. This group of flip-flops is known as "register".

* The n-bit Register consists of 'n' number of flipflops and is capable of storing 'n' bit word.

* ^{2M} Shift Register:-

def:- It can shift the data from one Register to another Register.

* Shift Registers are classified into 4 type.

1. Serial IN - serial out [SISO]
2. Serial IN - Parallel out [SIPO]
3. Parallel IN - serial out [PISO]
4. Parallel IN - Parallel out [PIPO]

1. Serial IN - serial out [SISO]:-

* SISO stands for Serial IN serial out shift Register.

* Here it is a four-bit shift Register. Here (23)
 We have clock pulse. This clock pulse will be applied on all the flipflops.

* Serial IN means, the I/P will be applied in serial manner & serial out means, the O/P will get in serial manner. bit by bit

* This is the shift Register. Q_3, Q_2, Q_1, Q_0 are O/P's of the flt's, D_3, D_2, D_1, D_0 are I/P's to the flt's

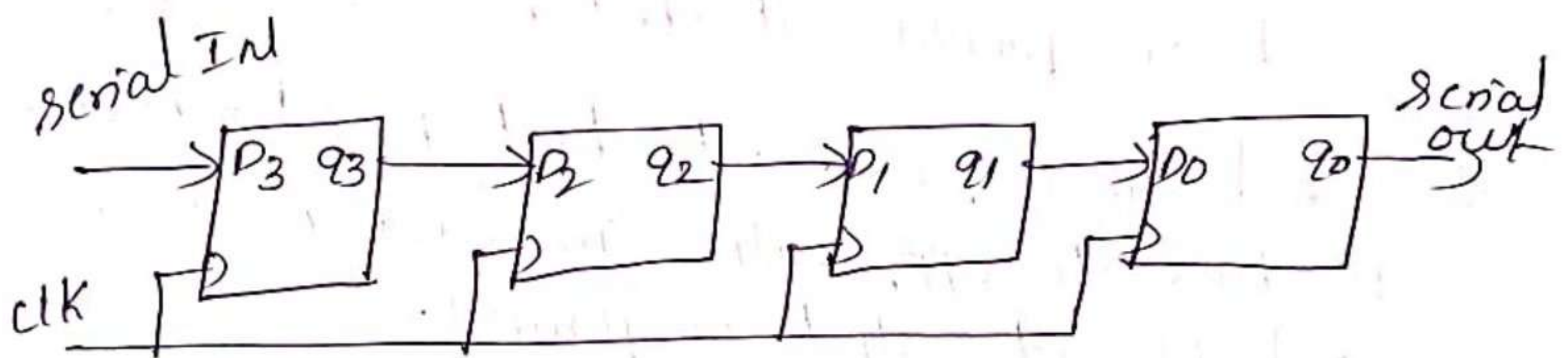


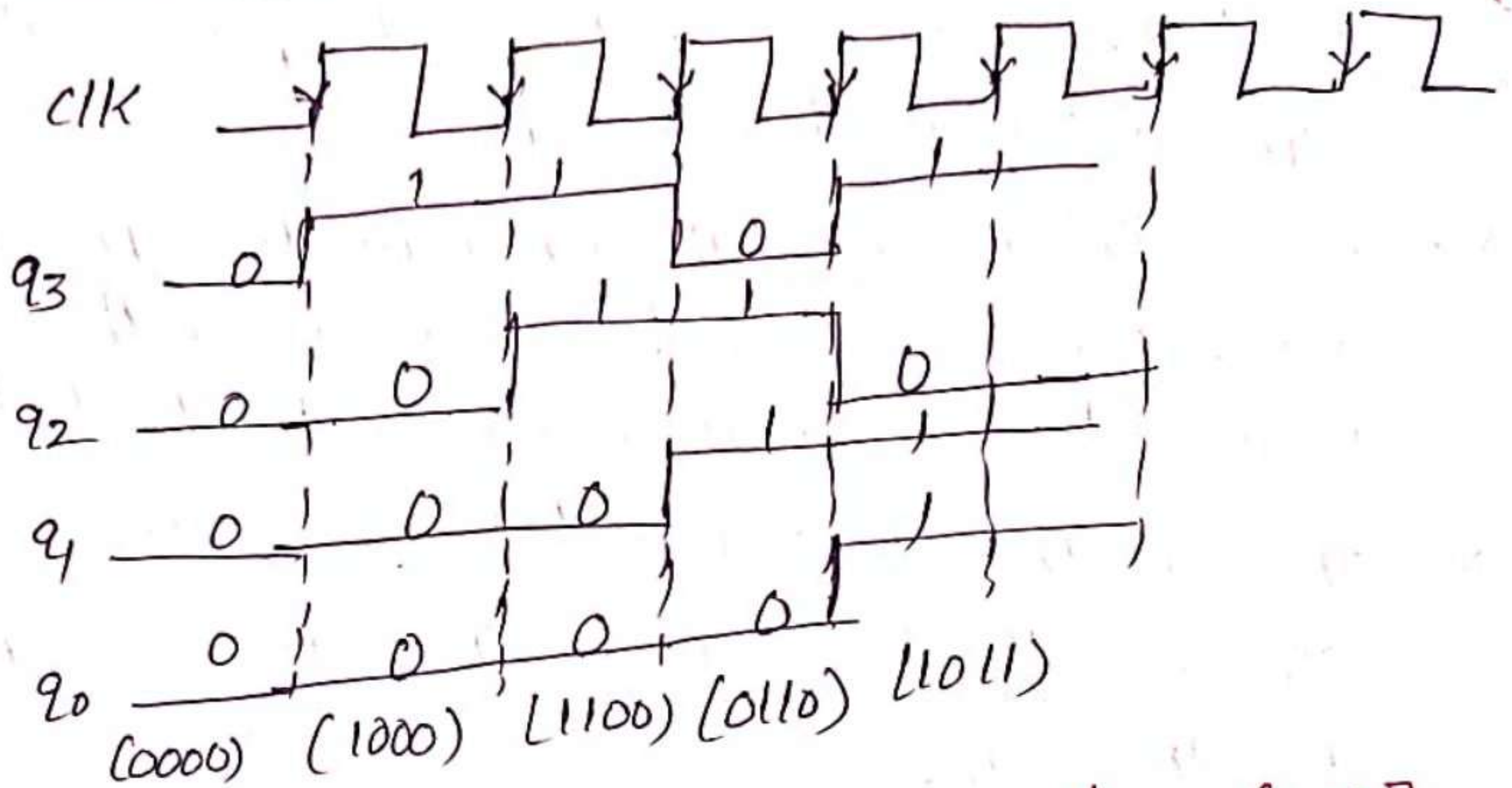
Fig:- Serial IN Serial out S.R

* Let us assume our Register contain 1011
 4-bit Register, here 1011

* Now we have MSB & LSB
 to perform shift right operation

clk	SIP	Q_3	Q_2	Q_1	Q_0	
0	X	0	0	0	0	→ Neglect
1	1	1	0	0	0	→ "
2	1	1	1	0	0	→ "
3	0	0	1	1	0	→ "
4	1	1	0	1	1	

Waveform :-



2. Serial In parallel out shift Register:- [SIPO]

* Serial In parallel out shift Register means, we have to apply I/P's in serial manner, and we get output in parallel.

* Let us take 4-bit shift register. so we need '4' flip's. We are using D flip's. Means it is used to storing the data.

* The I/P's are D_3, D_2, D_1, D_0 as the o/p's are q_3, q_2, q_1, q_0

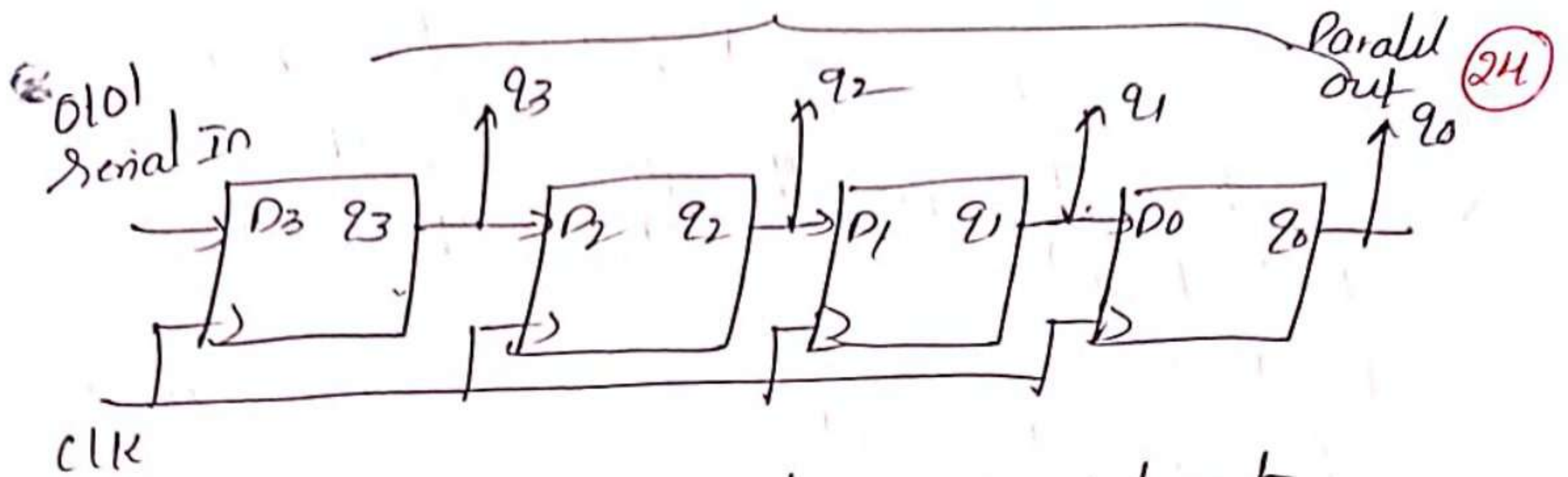
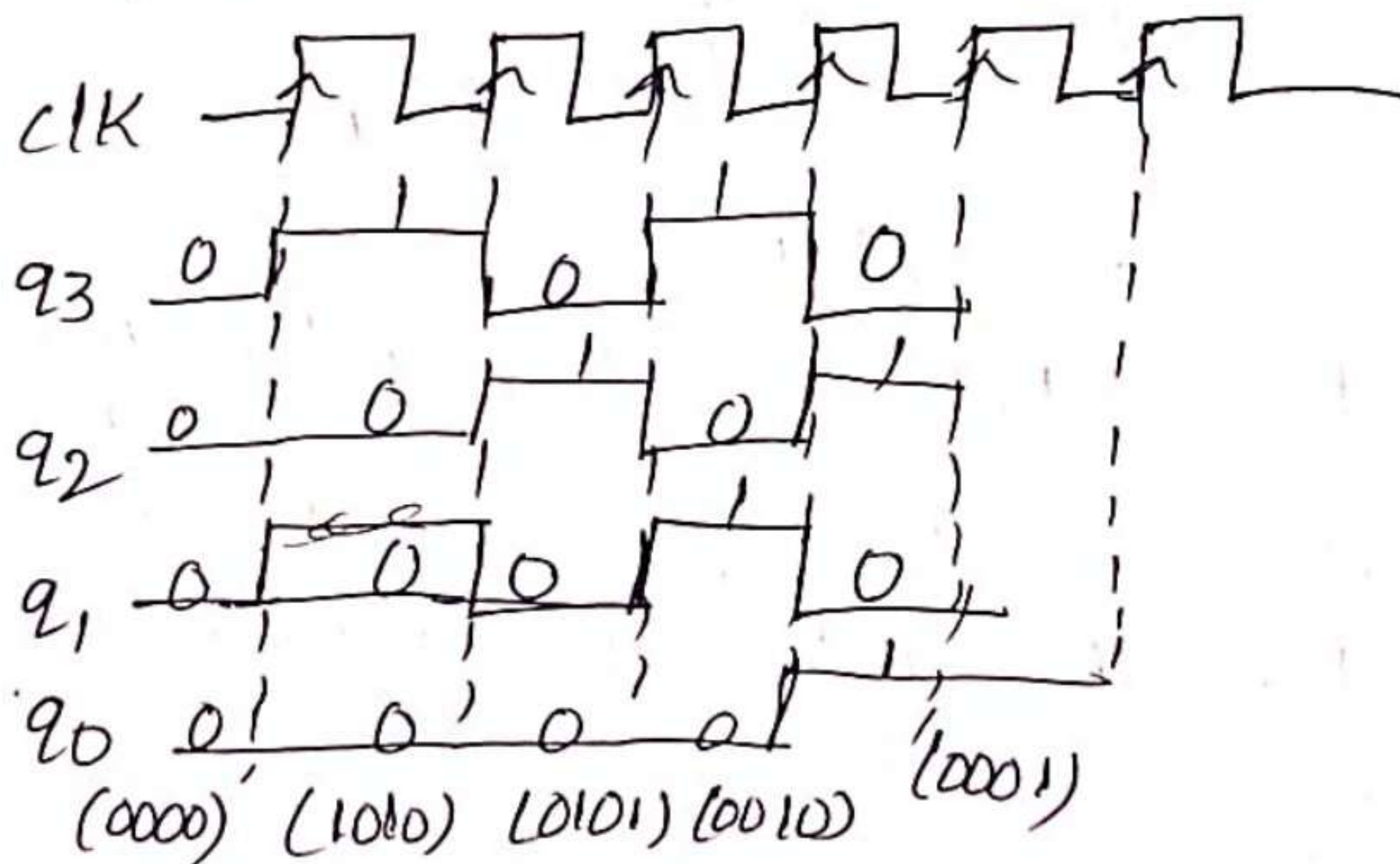


Fig:- Serial In - Parallel out

* Let us assume 4-bit Register 0101

CLK	SIP	q ₃	q ₂	q ₁	q ₀	
0	—	0	0	0	0	→ Neglect
1	1	1	0	0	0	→ "
2	0	0	1	0	0	→ "
3	1	1	0	1	0	→ "
4	0	0	1	0	1	

* 'n' clock pulses are required in order to store (or) load in the data Register



3. Parallel IN - Parallel out shift Register:- [PIPO]

- * We have to apply the I/P in parallel. The Shift Register will produce the O/P in Parallel manner.
- * The Register contains 4-bits. so we need four flipflops.
- * Here we are using D flipflop to store the data.
- * Here the I/P's are D_3, D_2, D_1, D_0
- * " " O/P's are Q_3, Q_2, Q_1, Q_0

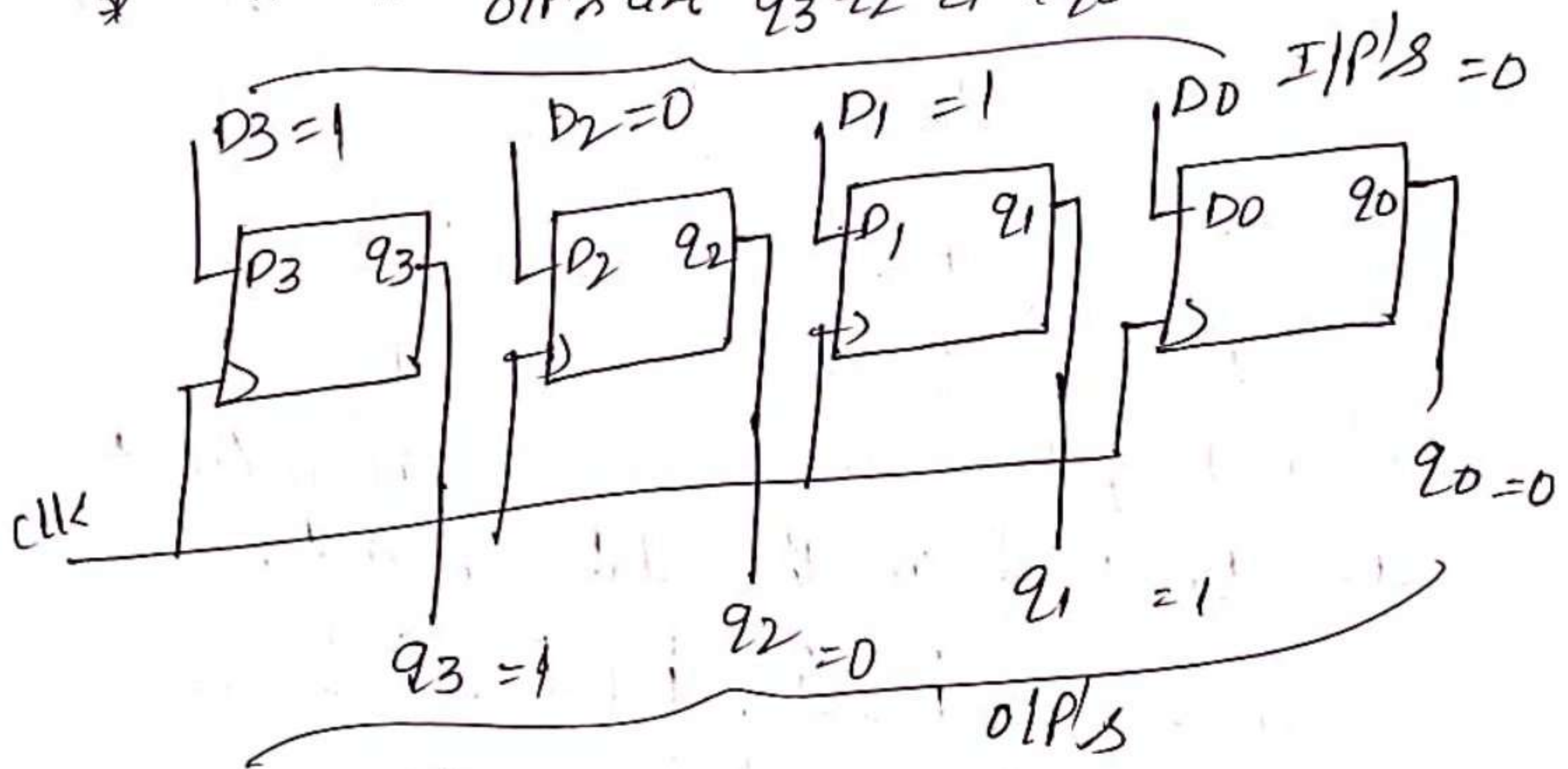
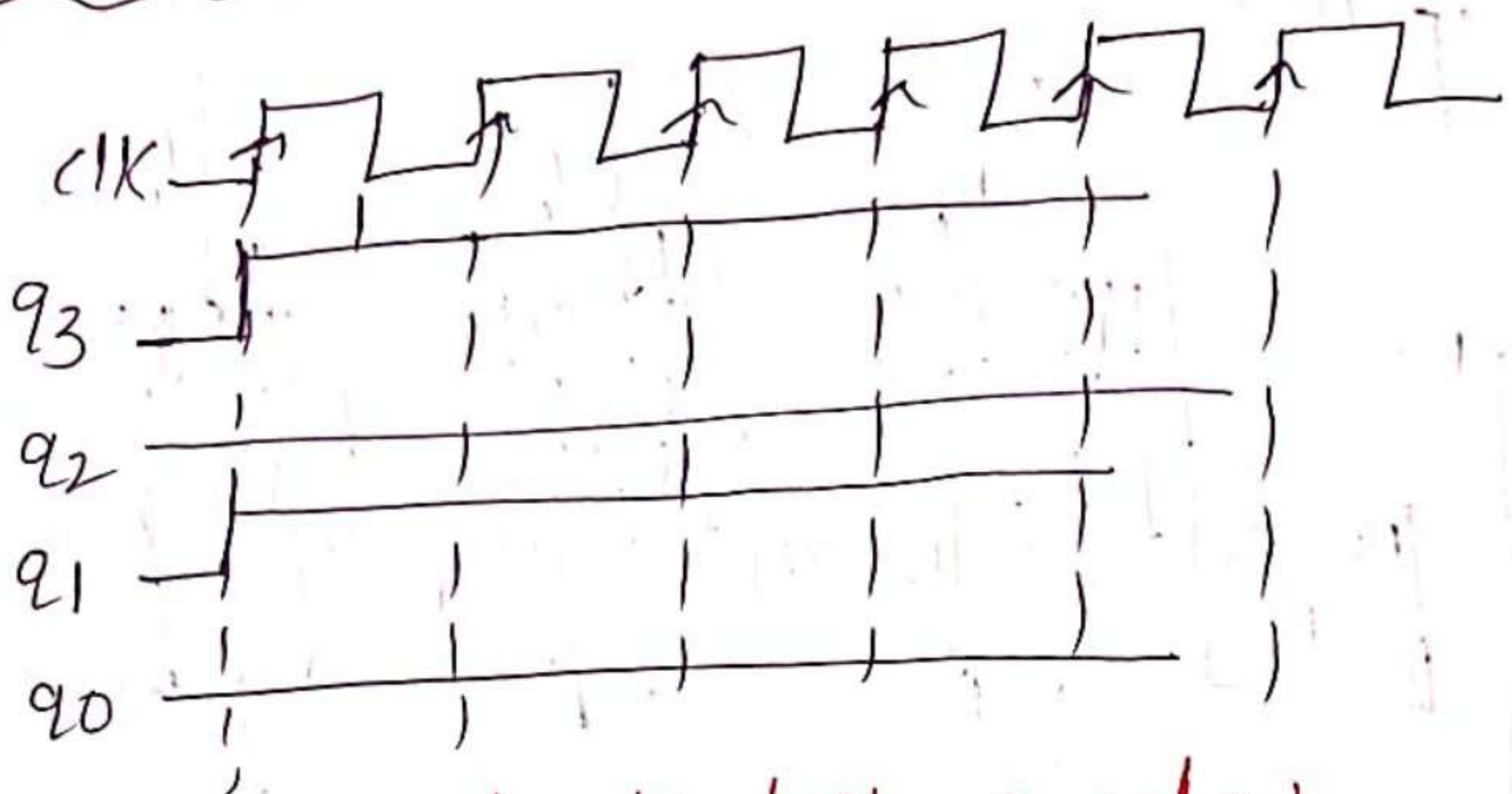


Fig:- PIPO

- * Let us take 4 bit Register 1010
MSB LSB
- * only one clock pulse is Required to store the data in Register.

Waveform

(25)



4. Parallel In - Serial out shift Register:-

- * We have to apply the T/P in parallel as we get o/p in serial manner.
- * Here the T/P's of the flip's are $D_3 D_2 D_1 D_0$
- * " " o/p's of the flip's are $Q_3 Q_2 Q_1 Q_0$
- * We are using D flip means Data Register.
- * Let us take the T/P 4-bits 1010
MSB LSB
- * The T/P's are $B_3 B_2 B_1 B_0$
- * In order to perform the operation, we are using control operation i.e. shift/load

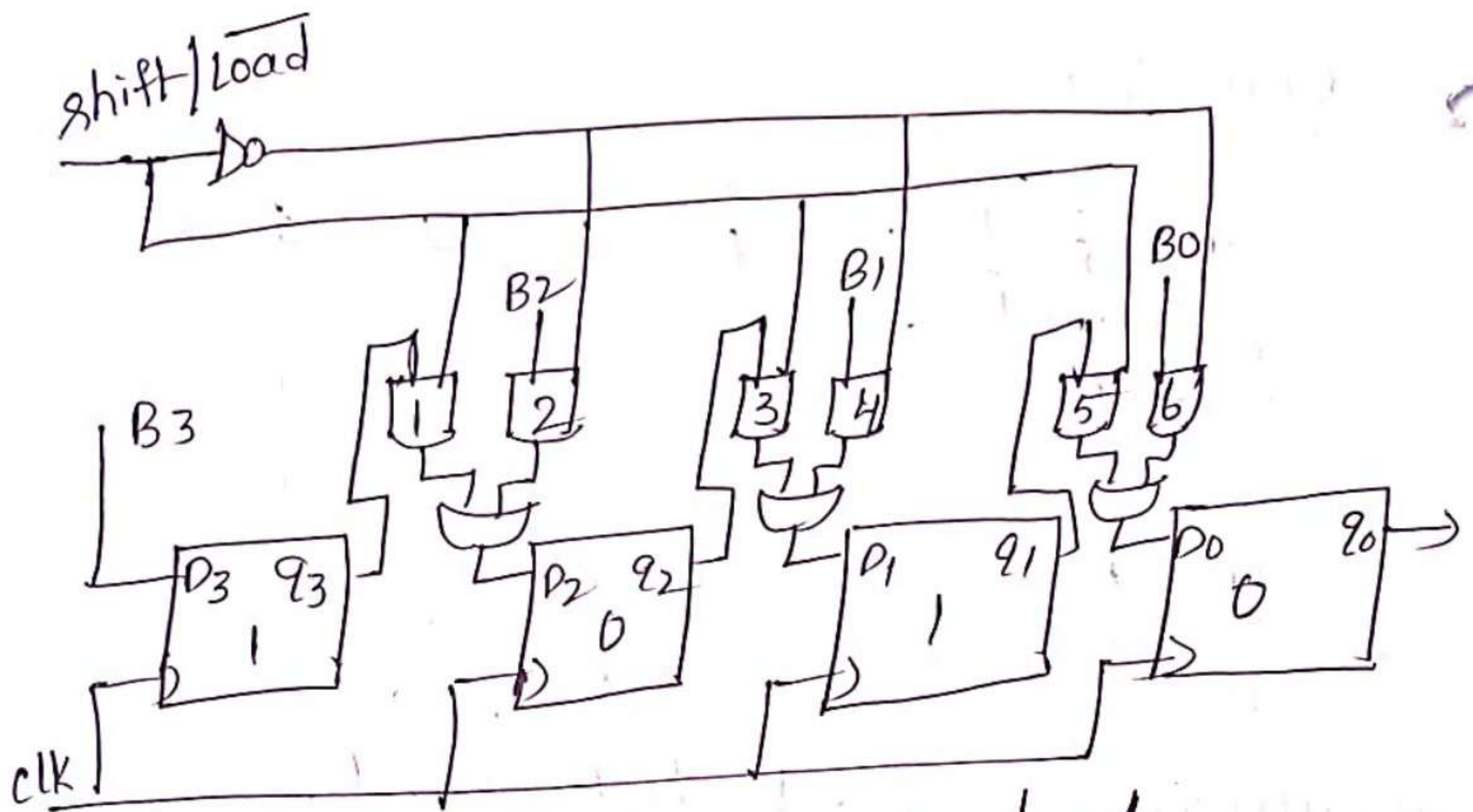


Fig. 1 - Parallel Input - Serial Output.

Load operation:- [When $T/P = 0$]

- * When $T/P = 0$, the ckt performs the load operation.
- * The odd numbers AND gates are [1, 3, 5] T/P is zero. The even number AND gates [2, 4, 6] T/P is 1.
- * The OR operation is performed b/w two AND gates.
- * Only one clock pulse is required for load operation.

Shift operation! - [When $T/P=1$] (26)

- * When the T/P is '1', the ckt performs the shift operation.
- * The even Number AND gates [2, 4, 6] T/P is '0'
- * The odd number AND gates [1, 3, 5] the T/P is '1'.
- * Therefore 'n' clock pulses are Required to shift the data.

2M det
Universal shift register!

- * A Register Capable of shifting in one direction only is a "Unidirectional Shift Register"
- * A Register Capable of shifting in both directions is a "bidirectional Shift Register"
- det 1M
* The Register which can be used to shift the data in both the directions like left, right and can load parallel data as well is called "Universal Shift Register"

Operation!

- * S_0, S_1, S_2 are the select pins that are used to select the mode of operation. i.e. shift left, Right, Parallel mode.

- * A clk TLP is used to synchronize all operations.
- * A clear control is used to clear the Register to '0'.
- * The TLP's D_1, D_2, D_3 & D_4 are connected in parallel & the OLP's are Q_1, Q_2, Q_3 & Q_4 are connected in parallel.

Truth table

Mode control		operation
S_1	S_0	y
0	0	No change
0	1	Shift Right
1	0	Shift Left
1	1	Parallel load

ckt diagram:-

(27)

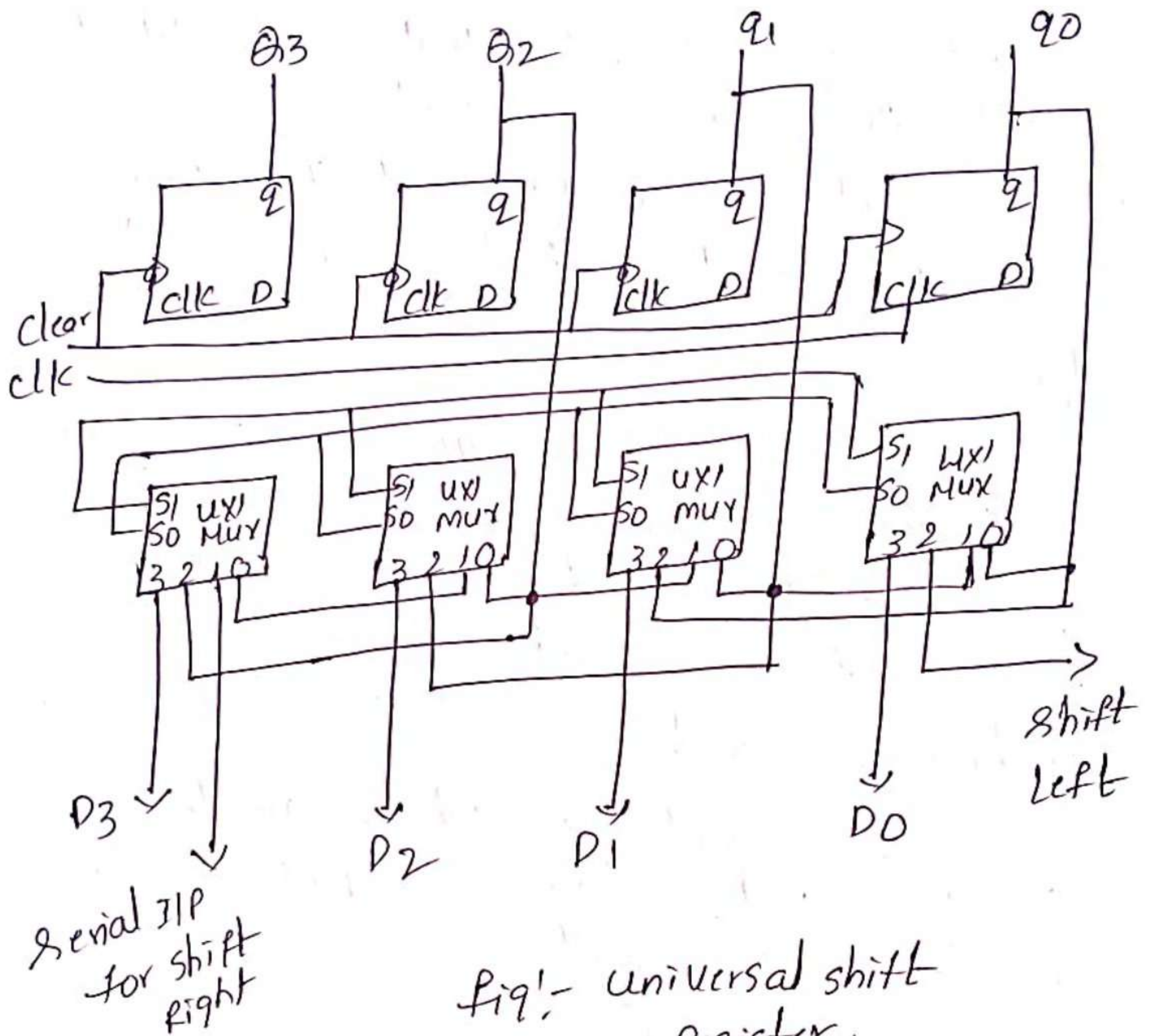


Fig:- Universal shift Register.

Counters:-

def:- A Counter is a Capable of Counting the number of clock pulses that arrives at its clock T/P is known as "Counter"

- * The counters are mostly constructed using JK flts in toggle mode (or) T flts.

Types of Counters:-

- * According to mode of applying clock pulses counters are classified as
 1. Synchronous Counters.
 2. Asynchronous Counters.

Synchronous Counter:-

The Counter in which all the flts are triggered by same clock pulse known as "Synchronous Counter"

Asynchronous Counter:-

Asynchronous Counter is a Cascaded arrangement of flts, where the O/P of one flt drives the CLK T/P of the following flts.

Diffⁿ BtW Synchronous & Asynchronous Counters:-

(28)

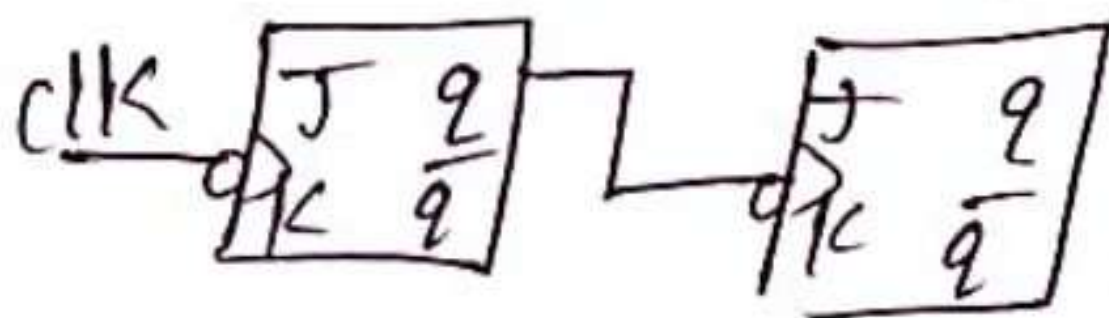
Asynchronous

* All the FF's are Triggered by different clock sources at a different times

* operation is slow

* propagation delay is more

* ckt diagram



Eg:- serial counter

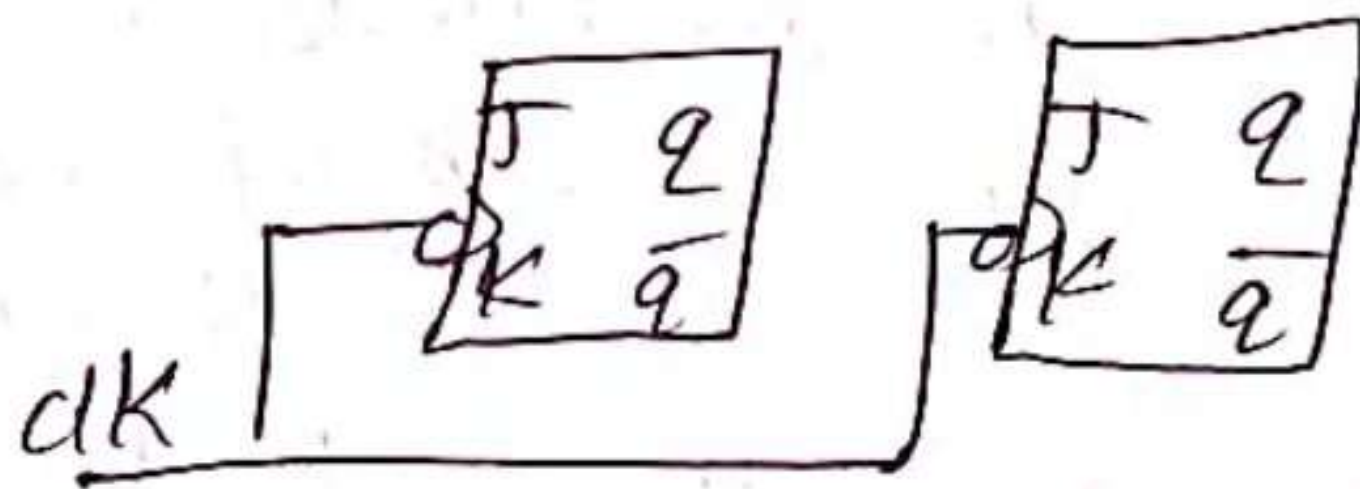
Synchronous

* All the flipflops are Triggered by the same clock source at the same time.

* operation is fast

* propagation delay is less

* ckt diagram



Eg:- parallel counters.

Synchronous Counter:-

* In synchronous counter we use

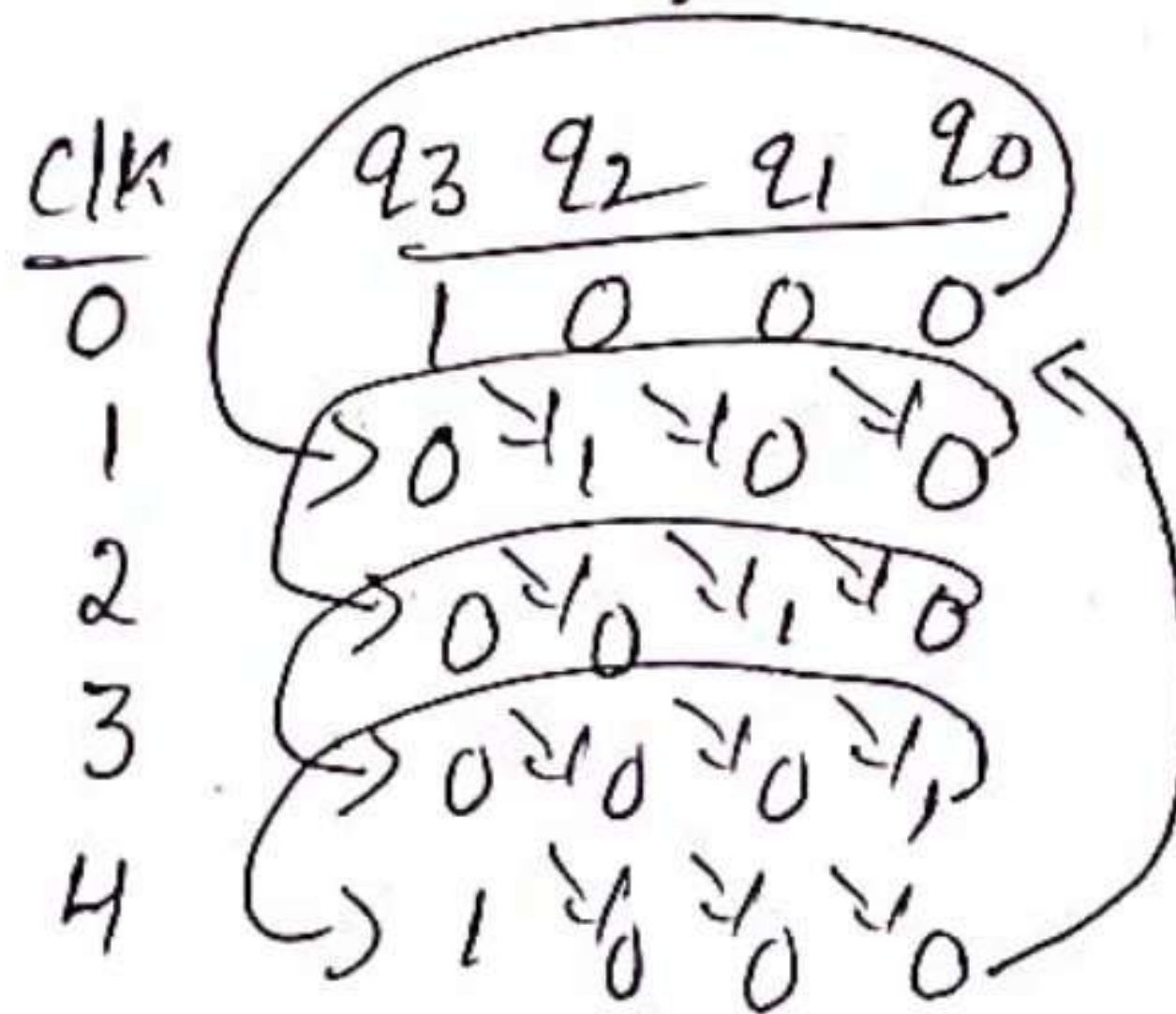
1. Ring Counter
2. Johnson Counter.

Ring Counter:-

- * A Ring Counter is a shift Register with only one flip being set at any particular time, all others are cleared.
- * The first flip is set at '1' and the single bit is shifted from one flip to the other on every clock pulse to produce the sequence of timing signal.
- * The bit arrived at the last flipflop is fed back to the 1st flip to form a ring, hence it is called as "Ring Counter".

operation:-

- * The logic diagram for 4-bit Ring Counter in which the output of each stage is connected to the input of the next stage.
- * The output of the last stage is fed back to the input of the first stage.



Logic diagram:-

(29)

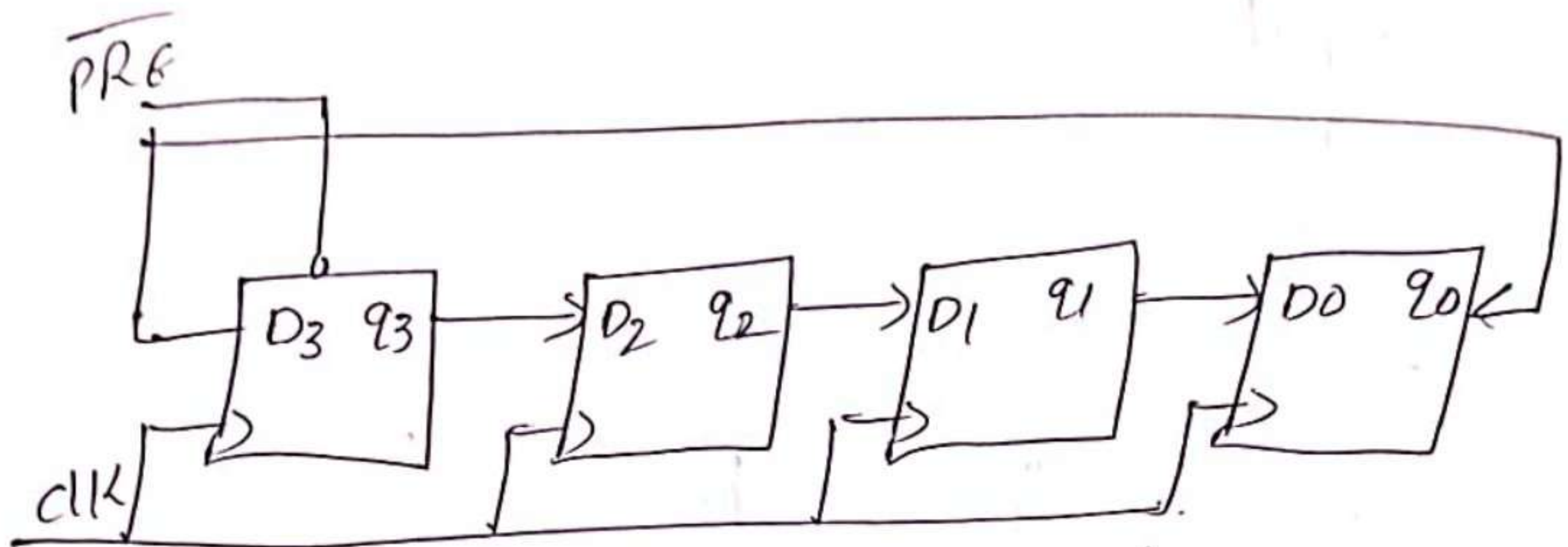
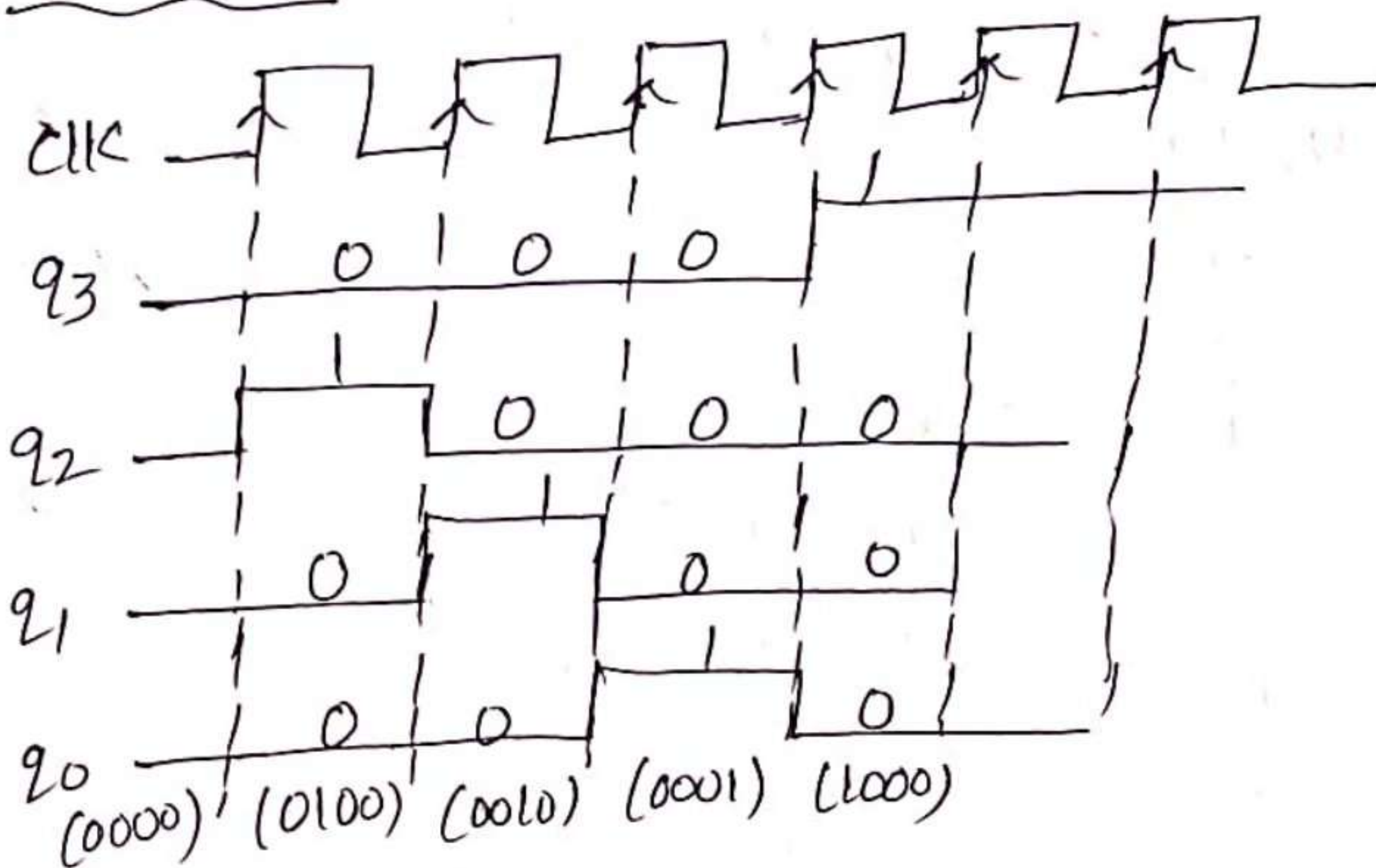


Fig:- 4-bit Ring Counter

Waveform



Johnson Counter:-

* It is similar to the Ring Counter

* The difference b/w Ring Counter & Johnson Counter is, In Ring Counter the last HF output is passed to the first HF as IP,

but Johnson Counter the last flip of Inverted output is passed to the front flip as Input

Logic diagram:-

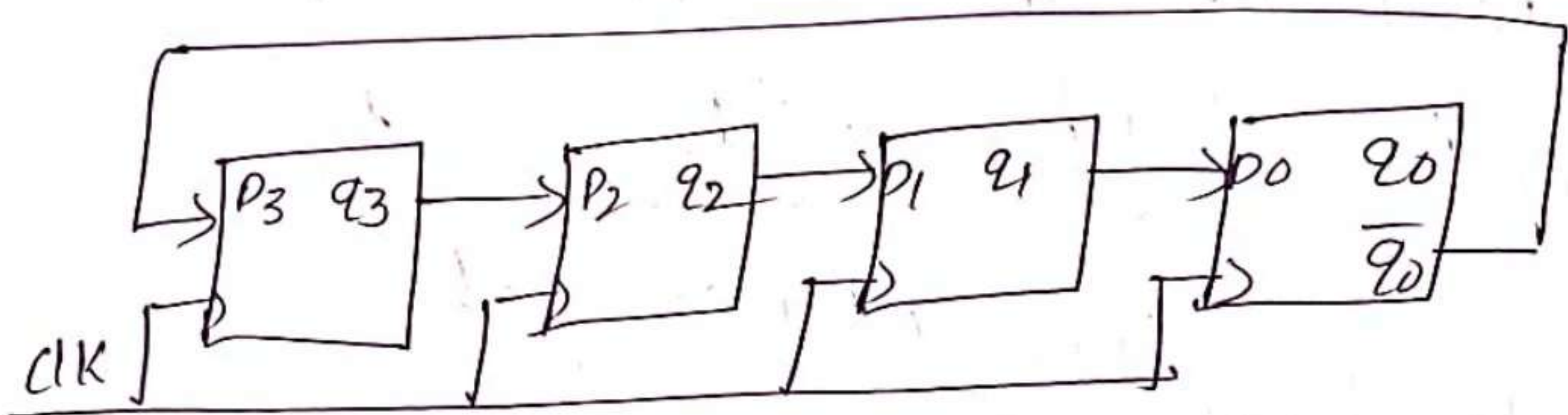
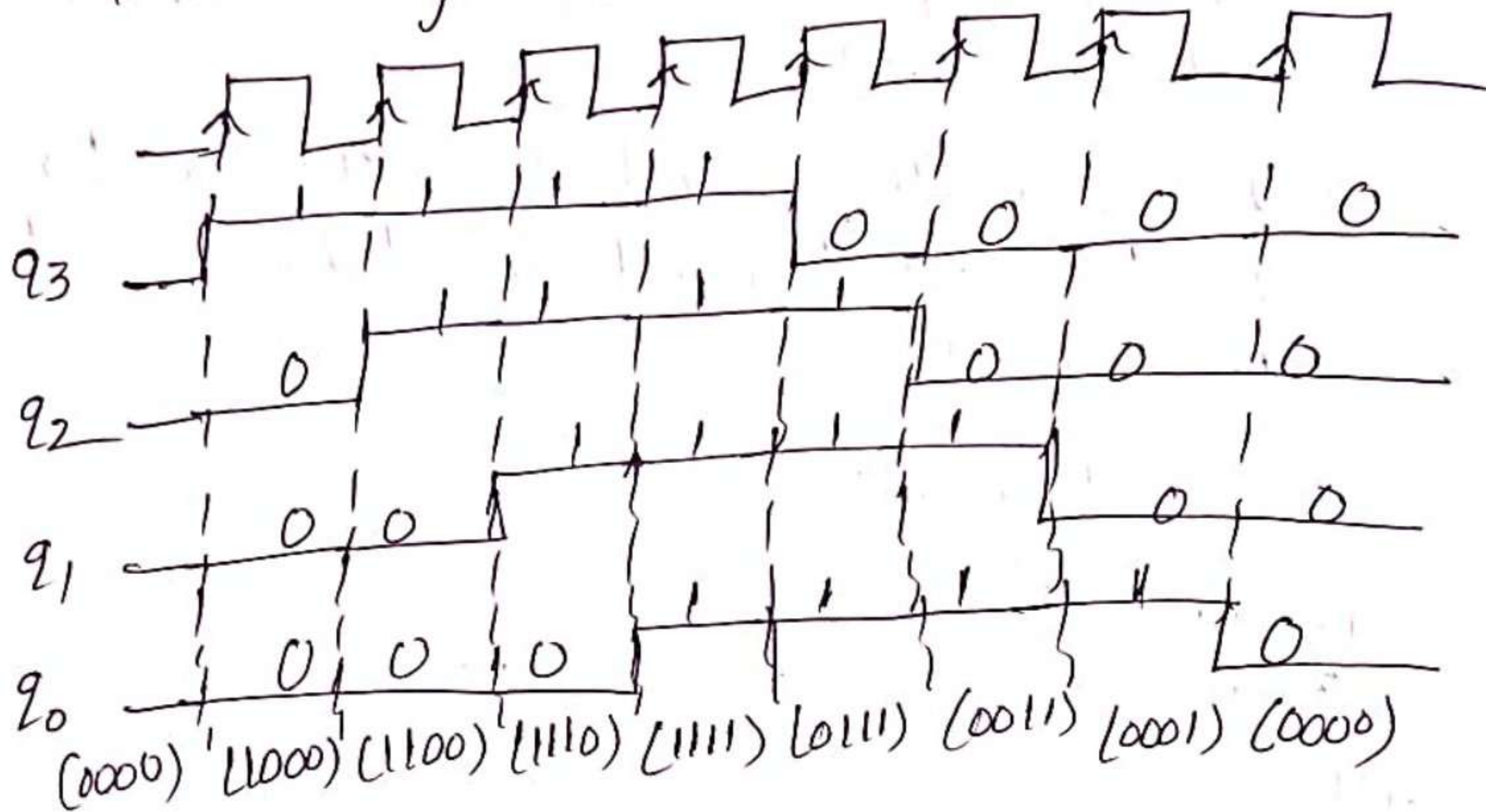


Fig.:- Johnson Counter

Truth table

clk	$\overline{Q_3}$	$\overline{Q_2}$	$\overline{Q_1}$	$\overline{Q_0}$
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

* It is also called as Johnson Counter (30)
Twisted Ring Counter



Asynchronous Counter types:-

Asynchronous Counter types:-

1. 3-bit Ripple up Counter
2. 3-bit Ripple down Counter
3. 4-bit Ripple up Counter
4. 4-bit " down Counter

3-bit Asynchronous up Counter:- [Ripple up Counter]

* 3-bit Ripple up Counter, we require to use 3 flipflops

* The Working operation of a 3-bit Ripple up Counter takes place in exactly similar manner as that of a 2-bit Counter

* The Asynchronous preset and clear terminals are also being used. Both of them are active low $\overline{IP}$.

* hence, for the normal o/p of the Counter preset & clear terminals must be connected to Logic 1

Logic diagram:-

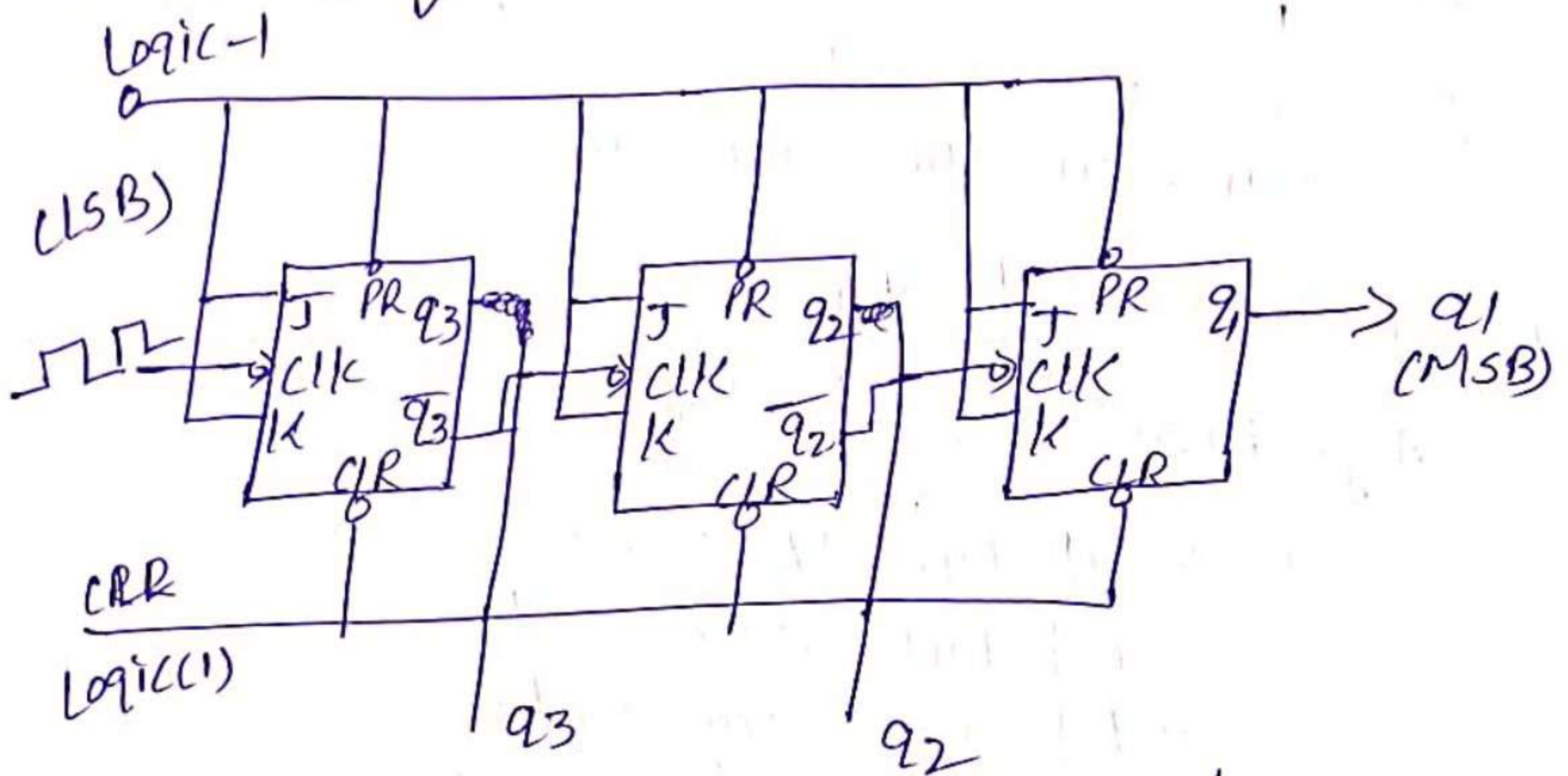
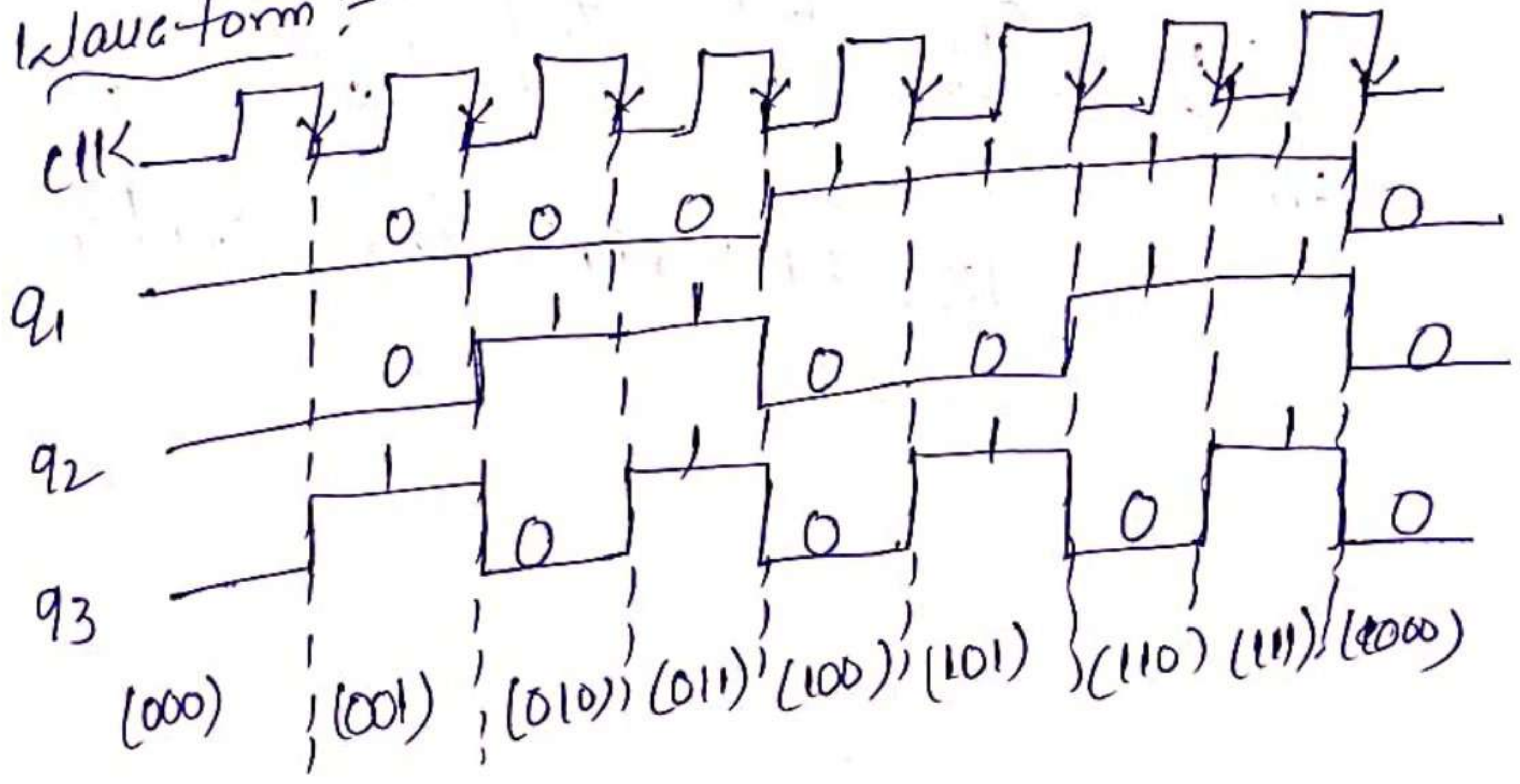


Fig. 3-bit ripple up counter.

Truth table :-

Clock	flip-flops			state	decimal equivalent
Initially	q ₁ (MSB)	q ₂	q ₃ (LSB)		
Initially	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	2	2
3	0	1	1	3	3
4	1	0	0	4	4
5	1	0	1	5	5
6	1	1	0	6	6
7	1	1	1	7	7
8	0	0	0	0	0

Waveform :-



3-bit Ripple down Counter:-

def:- The counters which can count in the 'downward' direction, i.e. from the 'maximum count to zero' are called "down counters".

* The count down sequence for a 3-bit synchronous down counter is as under below.

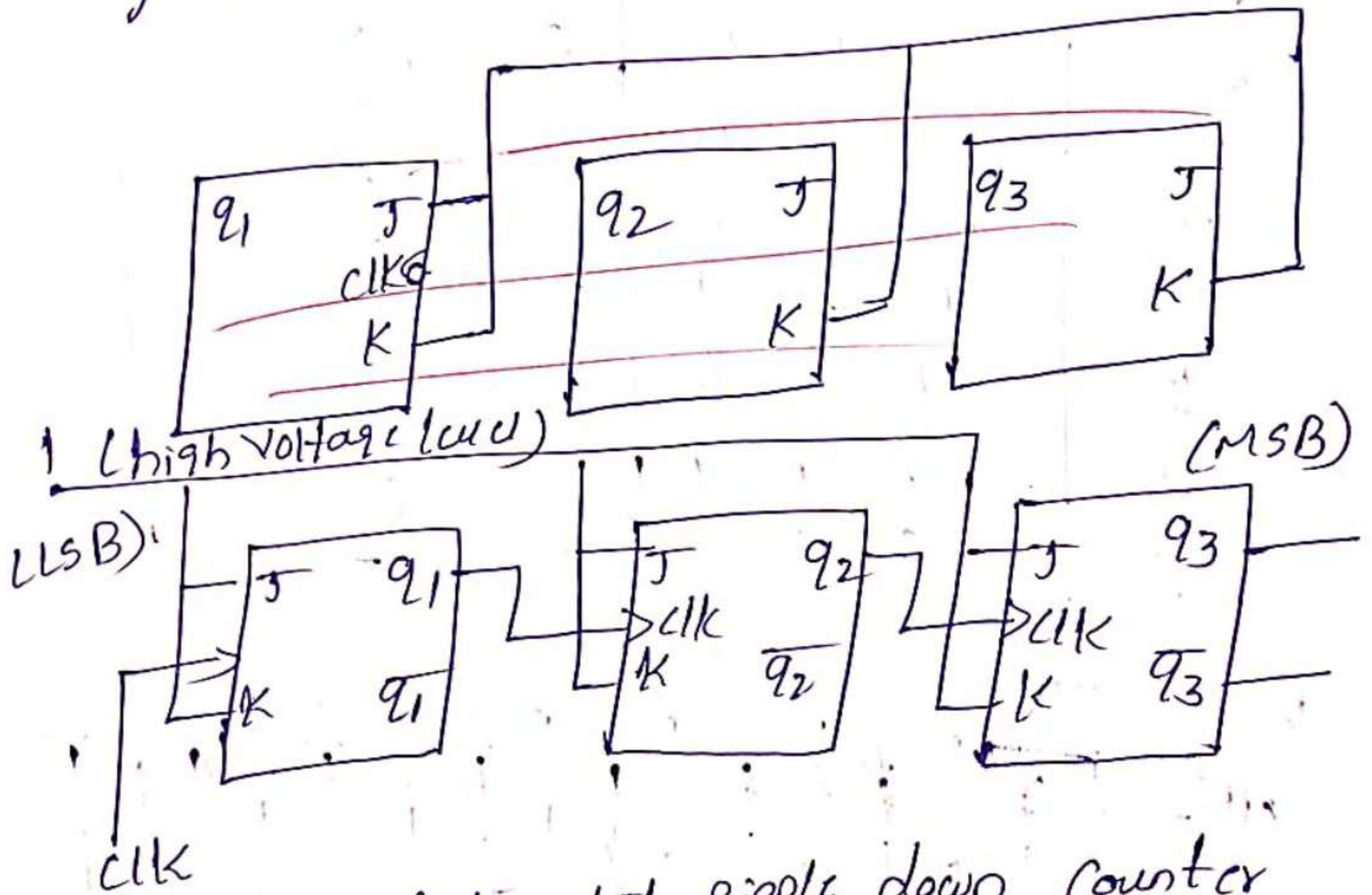


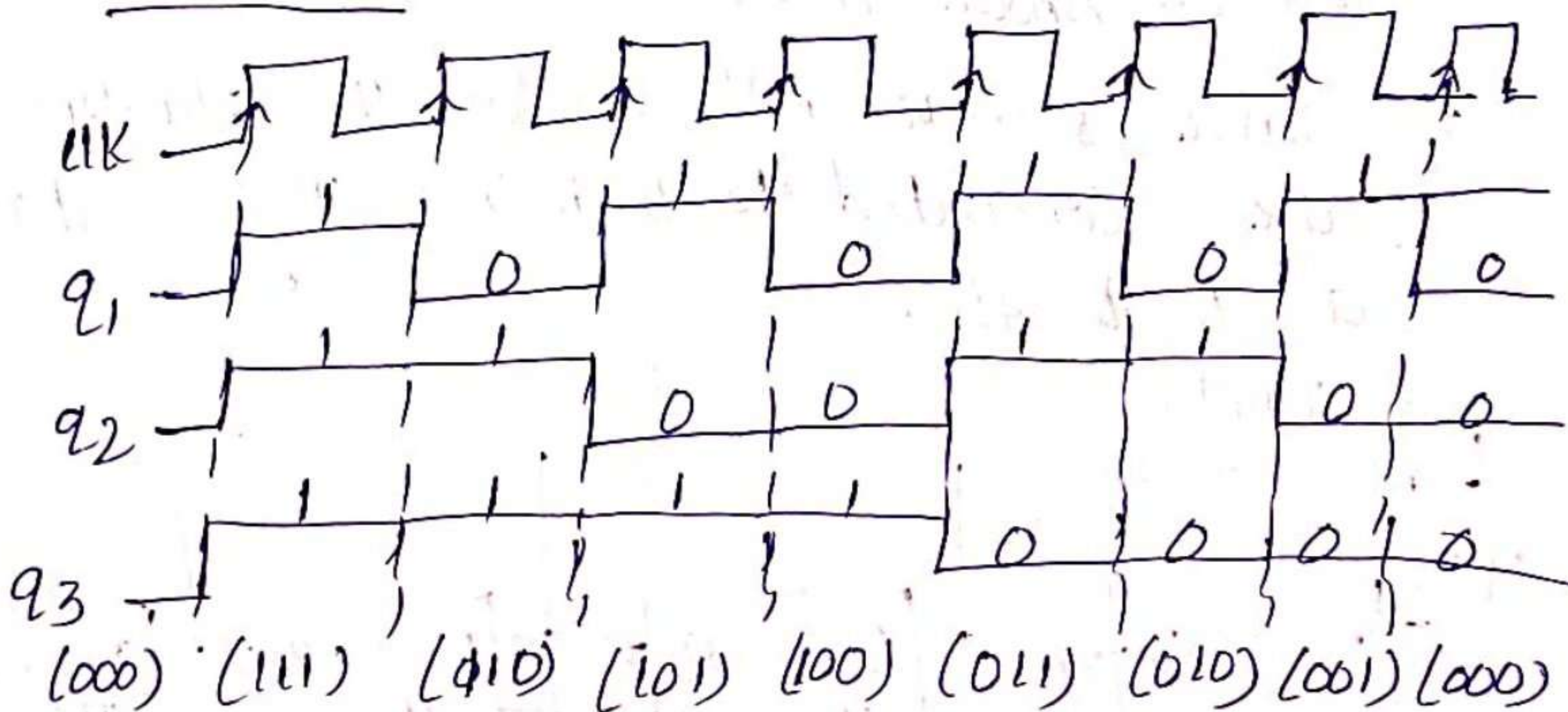
Fig 1:- 3 bit ripple down counter

Truth table:-

clk	q ₃	q ₂	q ₁	
0	0	0	0	Initial
1	1	1	1	7
2	1	1	0	6
3	1	0	1	5
4	1	0	0	4
5	0	1	1	3
6	0	1	0	2
7	0	0	1	1
8	0	0	0	0

down
Counter

Waveform



N-bit Ripple-up Counter:-

- * A N-bit Asynchronous Counter Consists of N flip-flops connected in Cascade. The number of states is given by.
- * The ckt gives a Maxⁿ count of 16. This Counter gives a natural binary count from 0 to 15 and Resets to Initial Condition on the 16th TLP pulse.
- * The clock TLP is applied to the first flip-flop and its Q output as clock input for 2nd flip-flop and so on shown in fig.
- * Since J and K TLPs of all the flip-flops are connected to logic 1, They act as a toggle flip-flop.

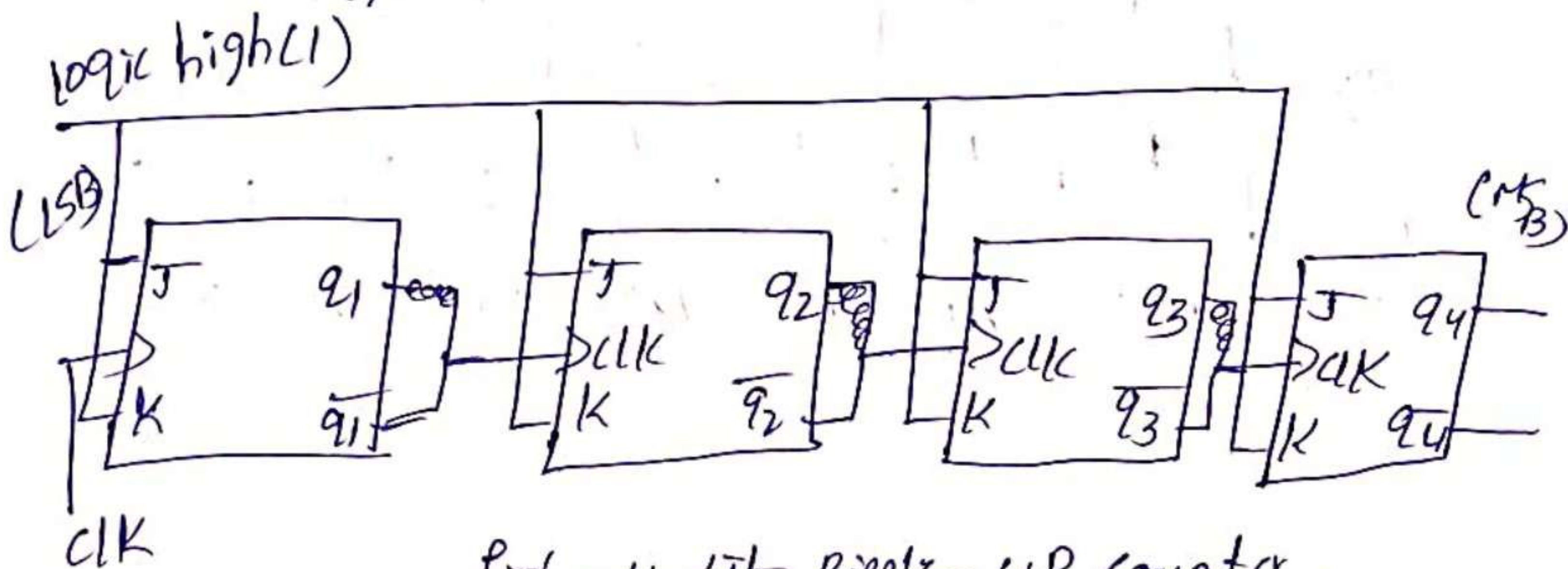
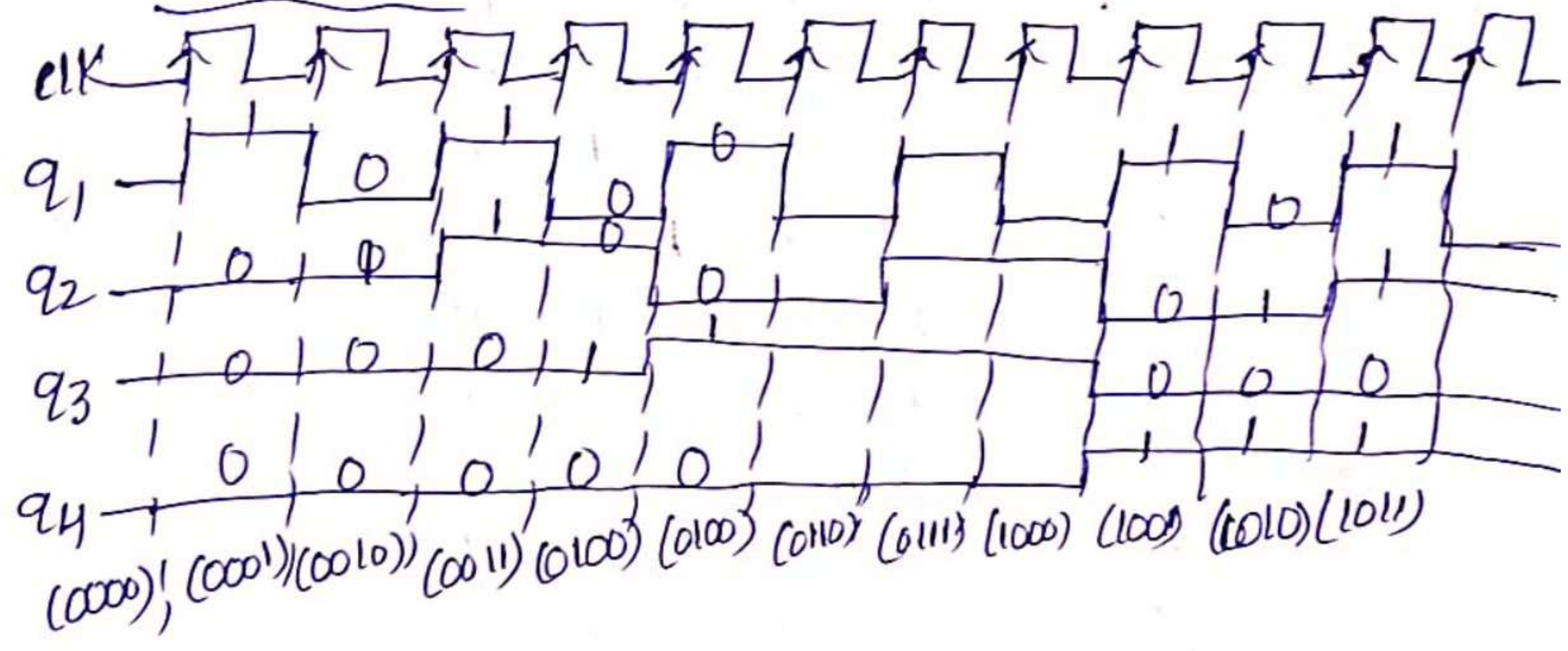


Fig:- N-bit Ripple up Counter.

	(MSB)				(LSB)
CLK	q ₄	q ₃	q ₂	q ₁	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	

Waveform:-



4-bit Ripple down Counter:-

* The ckt diagram of a 4-bit ripple down counter as shown in fig 4

* here we are using 4 f/f's are connected in cascaded form as shown in fig 4

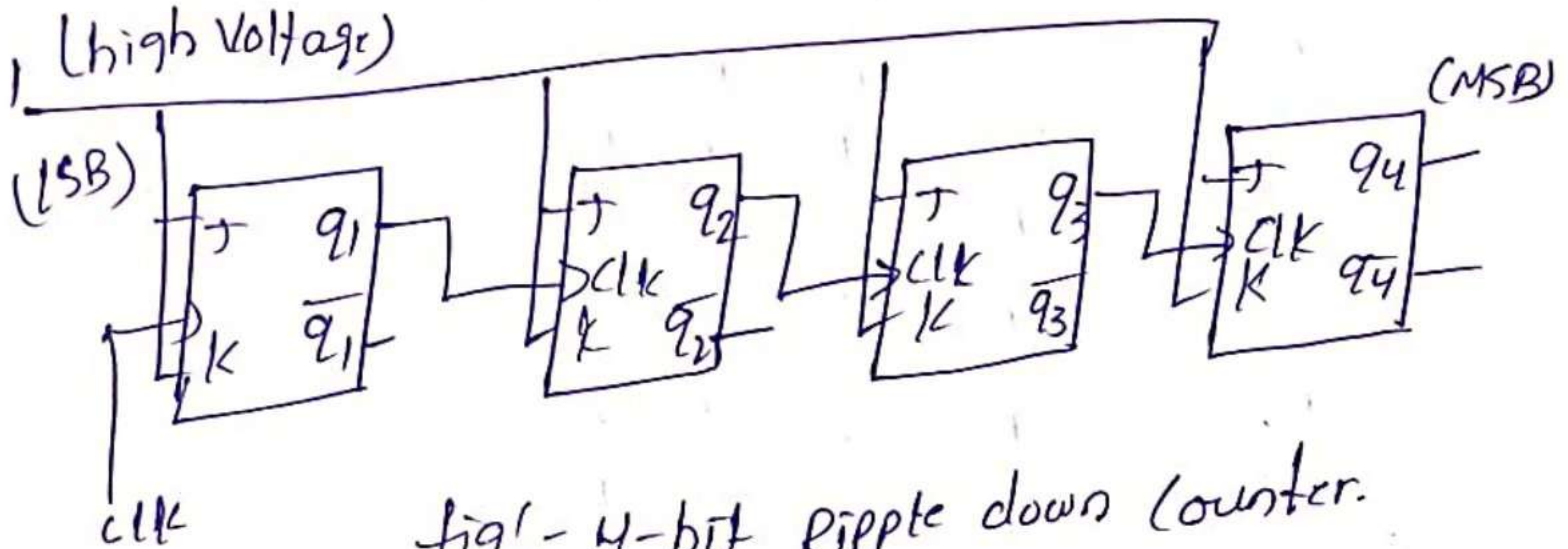


fig 4:- 4-bit ripple down counter.

Truth table:-

clk	Q ₄	Q ₃	Q ₂	Q ₁	
0	0	0	0	0	0
1	1	1	1	1	15
2	1	1	1	0	14
3	1	1	0	1	13
4	1	1	0	0	12
5	1	0	1	1	11
6	1	0	0	1	10
7	1	0	0	0	9
8	1	0	0	0	8
9	0	1	1	1	7
10	0	1	1	0	6
11	0	1	0	1	5
12	0	1	0	0	4

13.	0	0	1	1	3
14.	0	0	1	0	2
15.	0	0	0	1	1

34

Waveform :-

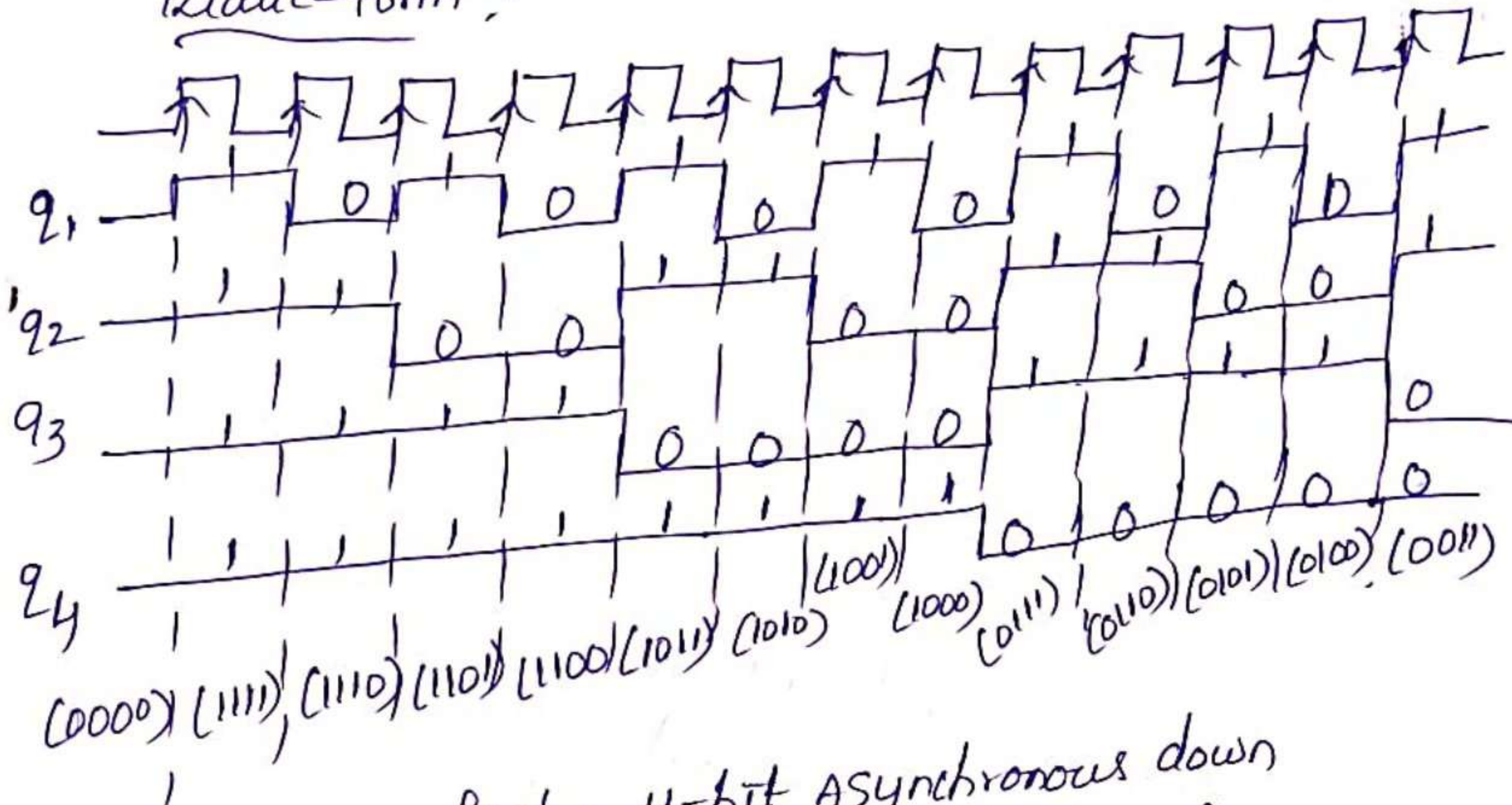


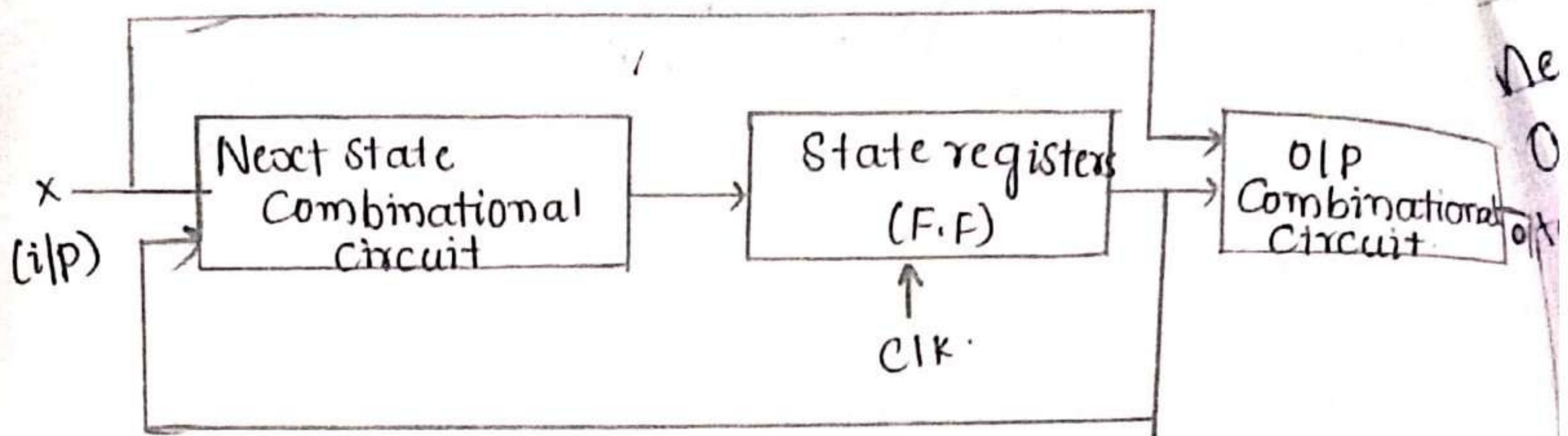
Fig:- 4-bit ASynchronous down counter waveform.

UNIT-4

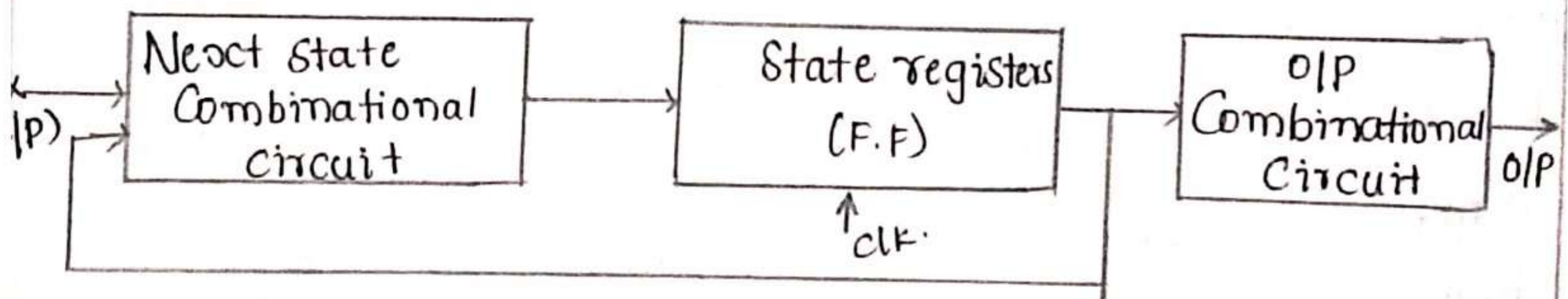
FSM and PLD's

Types of FSM:-

- Since we know (or) we have two types of circuits. They are Combination and Sequential circuits. In Combinational circuits feed back is not present and where as Sequential circuits are having feed back.
- Now, all the Sequential circuits are F.S.M's. FSM means "Finite State Machine".
A machine which consists of finite number of states are called FSM.
- FSM gives the abstract view of the Sequential circuits. Abstract means, they are present in reality, but we can represent in pictorially.
- FSM are of two types, They are:
 - * Mealy Machines
 - * Moore Machines.
- In mealy machines outputs are depends on the inputs and flipflop outputs.



→ Moore machines outputs don't depend on the inputs and it is just depends only on flipflop states.



Differences between Mealy and Moorely Machines:-

Mealy State Machine

1) Output depends on present State as well as present input.

2) If input changes, Output also changes.

3) Less number of States are required.

4) There is more hardware requirement for circuit implementation.

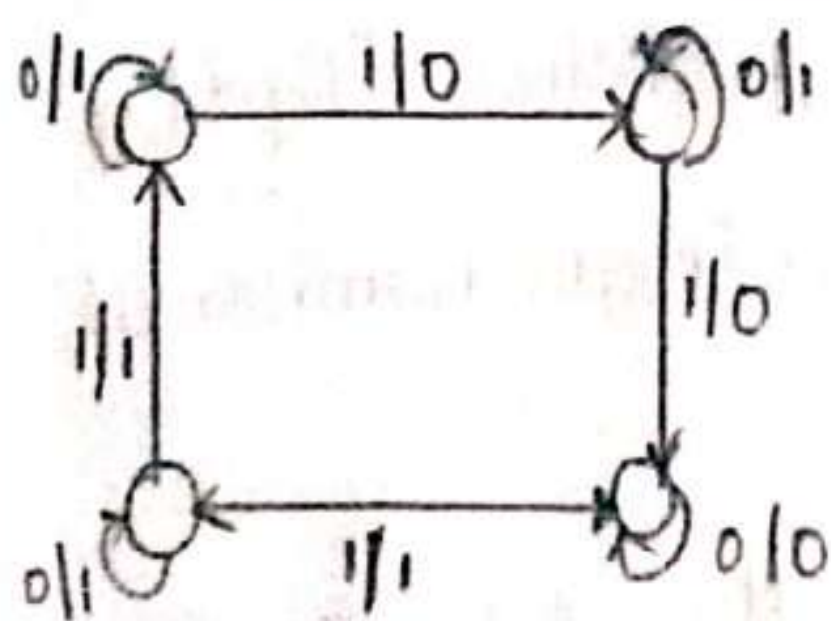
5) They react faster to inputs.

6) A Synchronous output generation.

7) Output is placed on transitions.

8) It is difficult to design.

9) State Diagram:-



$$10) Z(t) = \lambda \{ S(t), x(t) \}.$$

Moore State Machine.

(1) Output depends only upon present state.

(2) If input changes, output does not change.

(3) More number of States are required.

(4) There is less hardware requirement for circuit implementation.

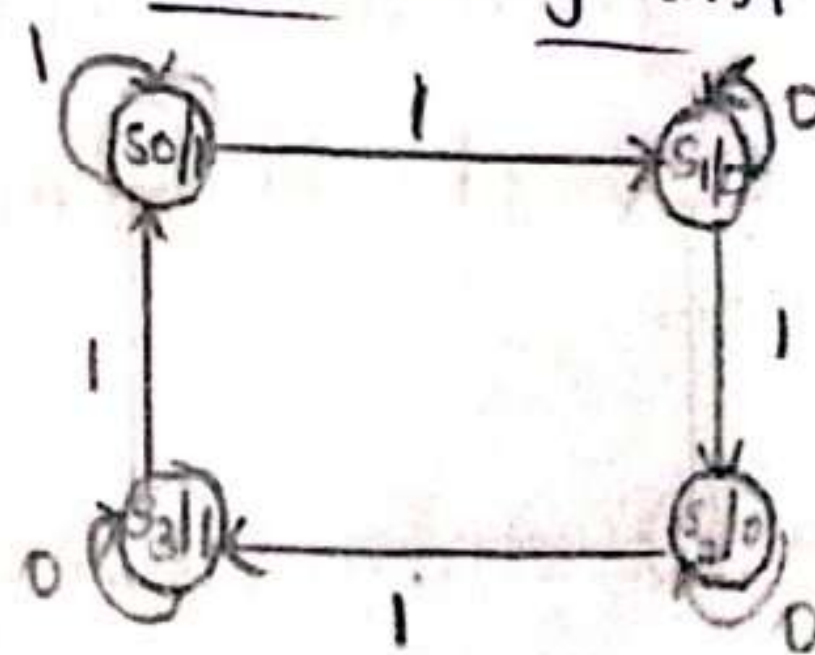
(5) They react slower to input (one clock cycle later).

(6) Synchronous Output & State generation.

(7) Out put is placed on States.

(8) Easy to design.

(9) State Diagram:-



$$10) Z(t) = \lambda \{ S(t) \}.$$

Capabilities and Limitations of finite State Machine

(1) Periodic Sequence of finite States:-

With n -State machine we can generate a sequence of n States (or) smaller than n -States.

Ex:- For $n=5$, the state machine can generate a maximum sequence of numbers are 5 only, i.e., 0,1,2,3,4, 0,1,2,3,4....

(2) No infinite Sequence:-

Infinite sequence cannot be produced by the finite state machine.

(3) Limited Memory:-

The finite state machine has a limited memory and due to limited memory it can not produce certain outputs. Because of limited memory it is not sufficient to store arbitrarily large partial products during multiplication.

State Assignment:-

State Assignment refers to the process of assigning binary values to the states of a sequential machine. The binary values should be given to the states in such a way that flipflop input functions may be implemented with a minimum number of logic gates.

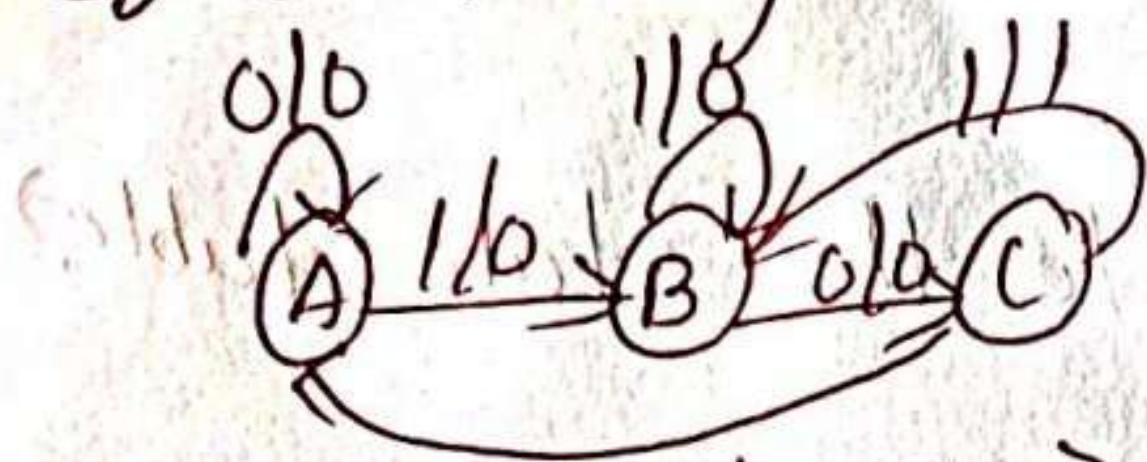
For a circuit with m states & n bits, then the codes must contain $m \leq 2^n$. for $n=3$ $8 \leq 2^3$ i.e., 000 to 111.

Machine realization of FSM using flip flops:- (Steps):-

- 1) To realize a FSM, one needs a state table with present state, next state, inputs & outputs.
- 2) After writing (or) obtaining the state table, the next step is to find the excitation table for the required flip flop.
- 3) Number of flip flops that are required to design a FSM (or) sequential circuit depends on the number of bits that are assigned to the states (or) the number of states that are present in the state diagram.
- 4) After obtaining the complete state (or) table, find the Boolean equations for the next state flip flop inputs by using K-map. If possible, also find output equations also.
- 5) Now draw the sequential circuit according to the Boolean equations required.

* 2M

Melay state diagram:-



0/0 → write the state table?

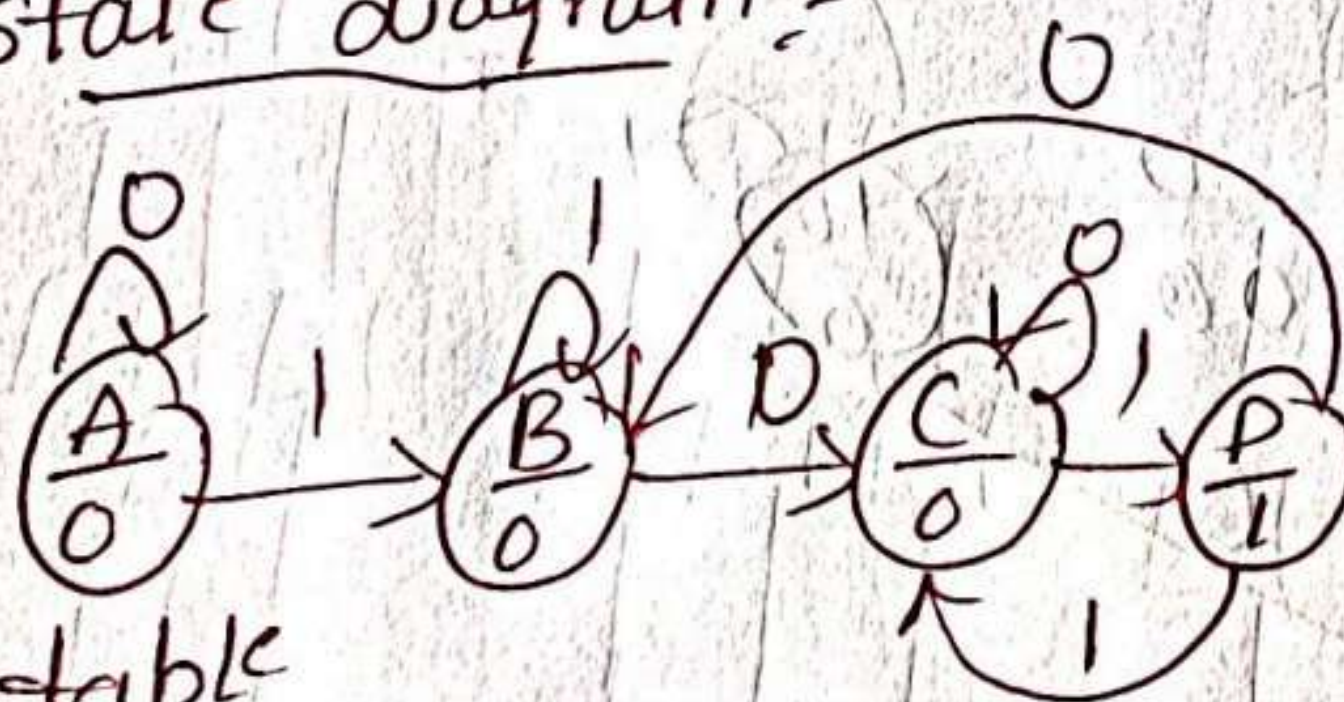
Ans:-

Present state	Next state o/p $x=0$ $x=1$		Present state	Next state
	$x=0$	$x=1$		
A	A, 0	B, 0	00	00
B	C, 0	B, 0	01	10
C	A, 0	B, 1	00	01
			11	11

* 2M

Moore

state diagram:-

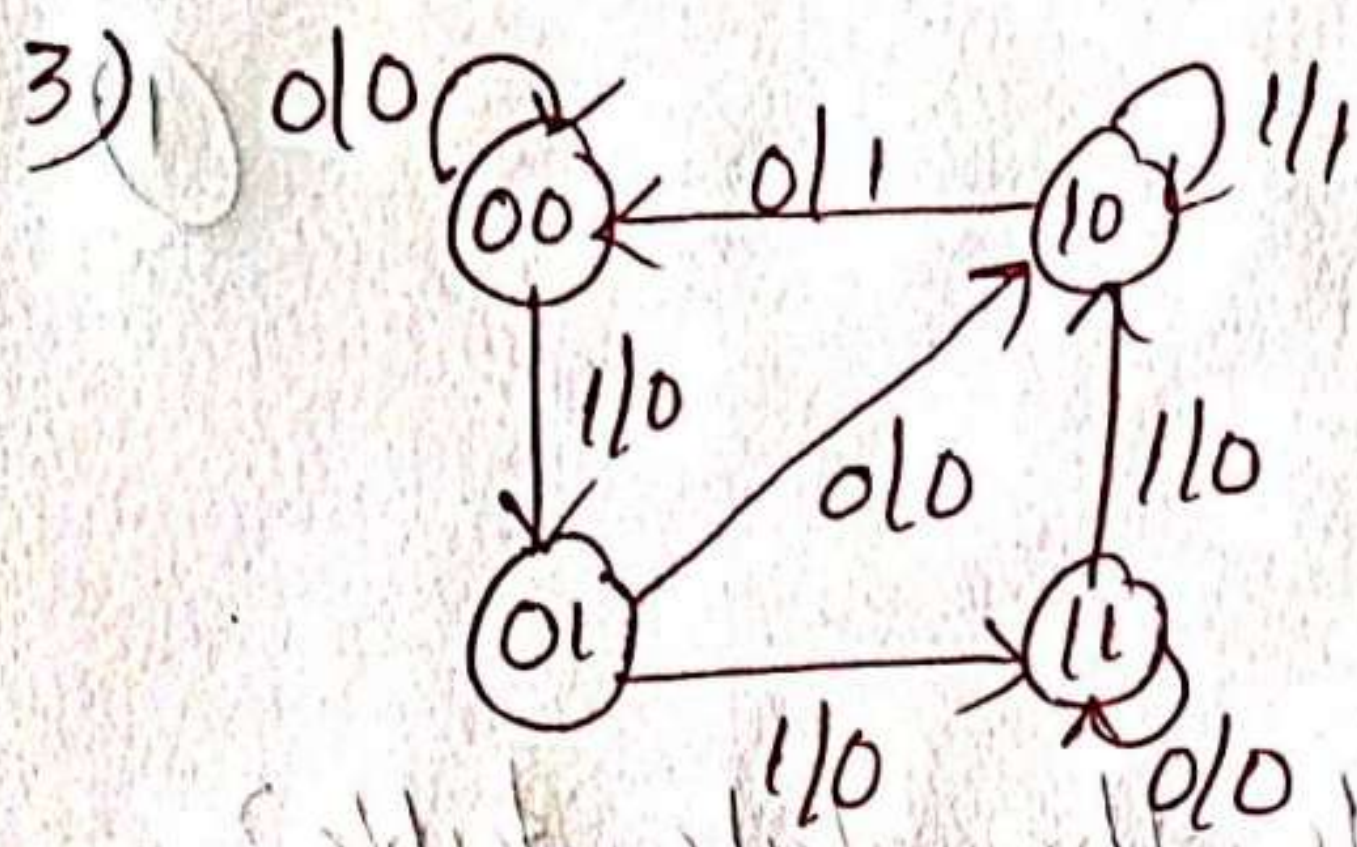


state table

Write the

Ans:-

Present state	Next state $x=0$ $x=1$		o/p	Present state	Next state
	$x=0$	$x=1$			
A	A	B	0	00	00
B	C	B	0	01	10
C	C	D	0	00	01
D	B	C	1	10	01

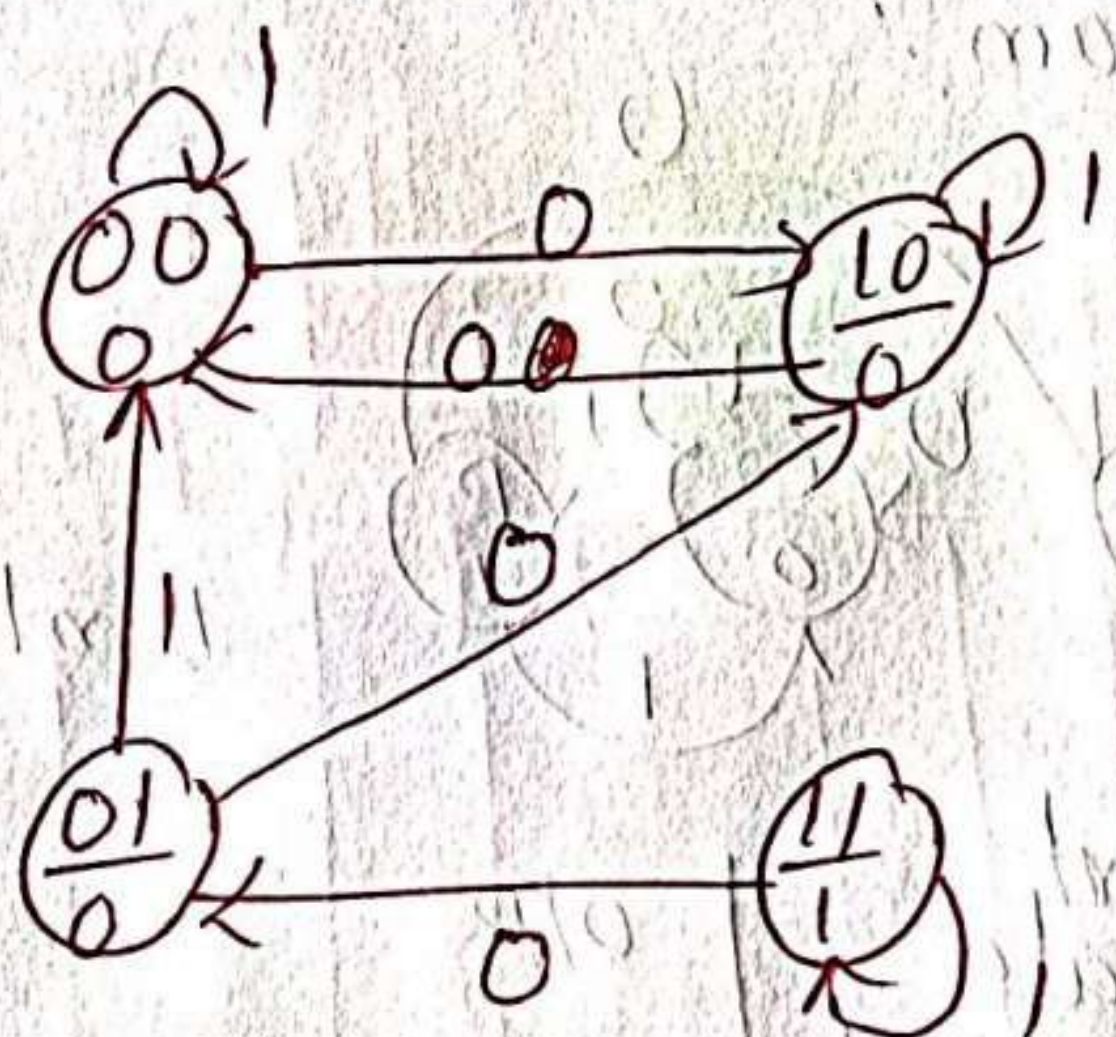


Write the state table

Sol:-

Present state	next state		O/P
	$x=0$	$y=0$	
00	00	01	0
01	10	11	0
10	00	10	1
11	11	10	0

4)



Write the state table

Sol:-

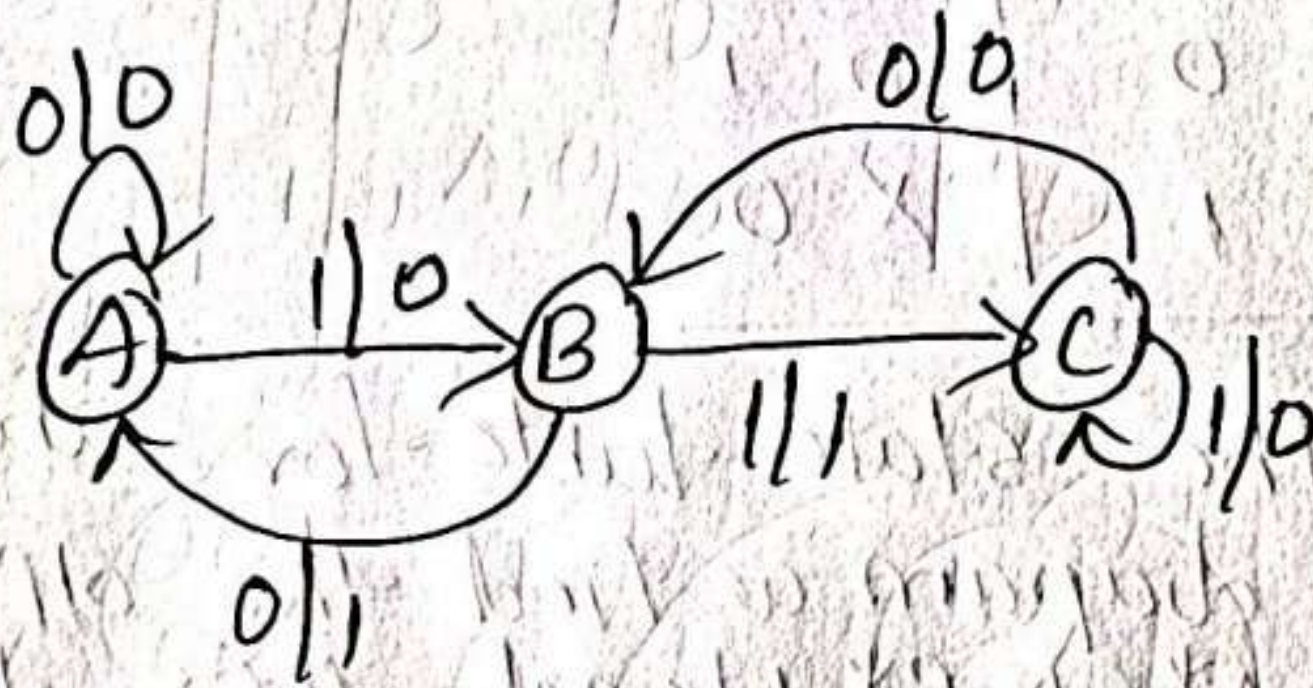
present state	Next state		O/P
	$x=0$	$x=1$	
00	10	00	0
01	10	00	0
10	00	10	0
11	01	11	1

5) ^{2M} By using state table write the state diagram in mealy model.

state table.

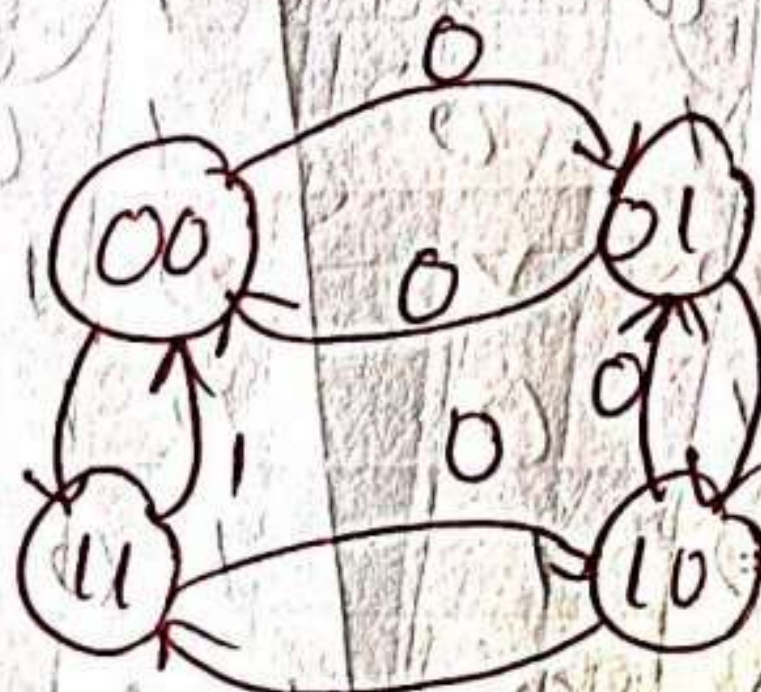
P.S	N.S		O/P	
	x=0	x=1	z=0	z=1
A	A	B	0	0
B	A	C	1	1
C	B	C	0	0

Ans:- state diagram in mealy model:-

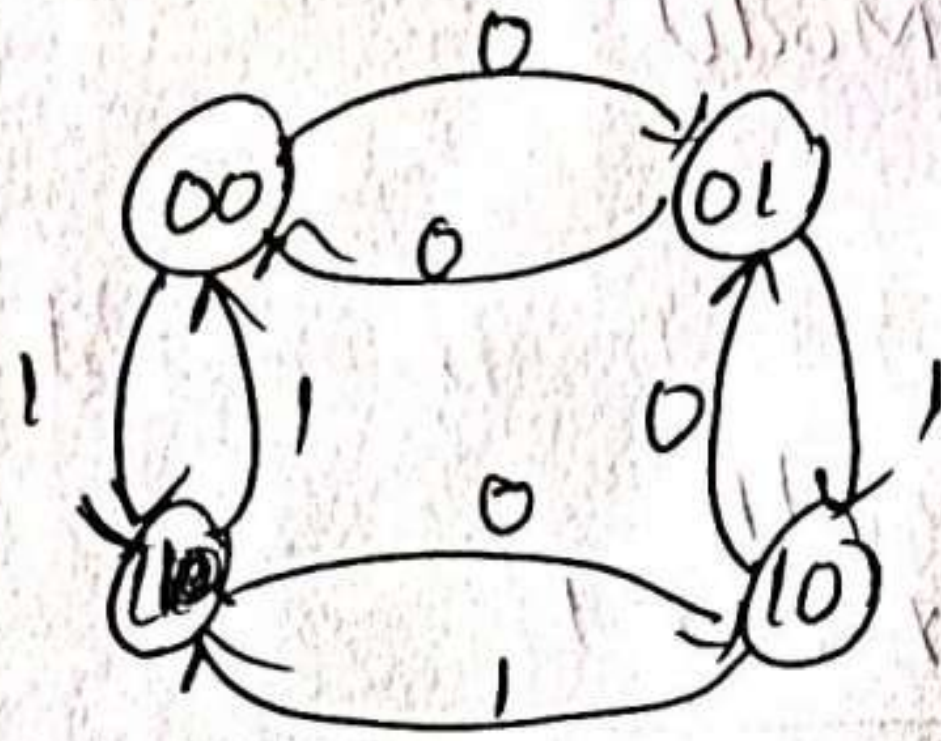


Realization of fsm by using f/f's:-

1- Realization of fsm by using SR f/f:-



sol:- Given data



step 1:- Write the excitation table of SR FF

I/P's		O/P's	
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

step 2:- Write the state table for above state diagram we find the o/p of flip's
Compare A w $A \oplus B, B \oplus A$ we get flip's o/p's

Present state		TIP	Next state		Flip's			
A	B	X	A*	B*	SA	RA	SB	RB
0	0	0	0	1	0	X	1	0
0	0	1	1	1	1	0	1	0
0	1	0	0	0	0	X	0	1
0	1	1	1	0	1	0	0	1
1	0	0	0	1	0	1	1	0
1	0	1	1	1	X	0	1	0
1	1	0	1	0	X	0	0	1
1	1	1	0	0	0	1	0	1

K-map for S_A

A \ B	0	1	2	3
0	0	1	1	2
1	4	5	7	6

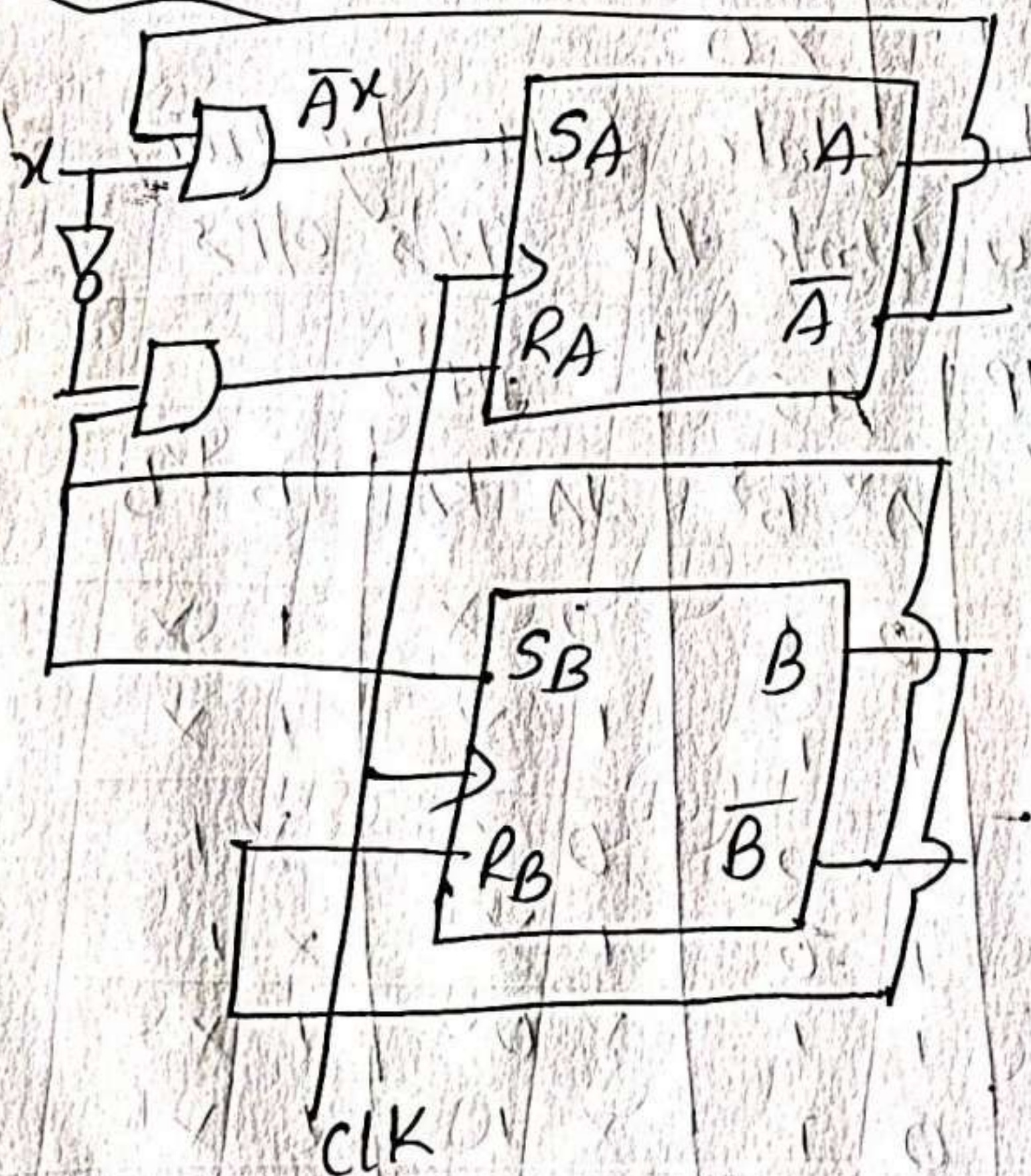
$$S_A = \bar{A}X$$

K-map for S_B

A \ B	0	1	2	3
0	1	1		
1	1	1		

$$S_B = \bar{B}$$

Logic diagram:-



K-map for R_A

(3)

A \ B	0	1	2	3
0	X	1	3	X
1	4	5	7	6

$$R_A = \bar{B}X$$

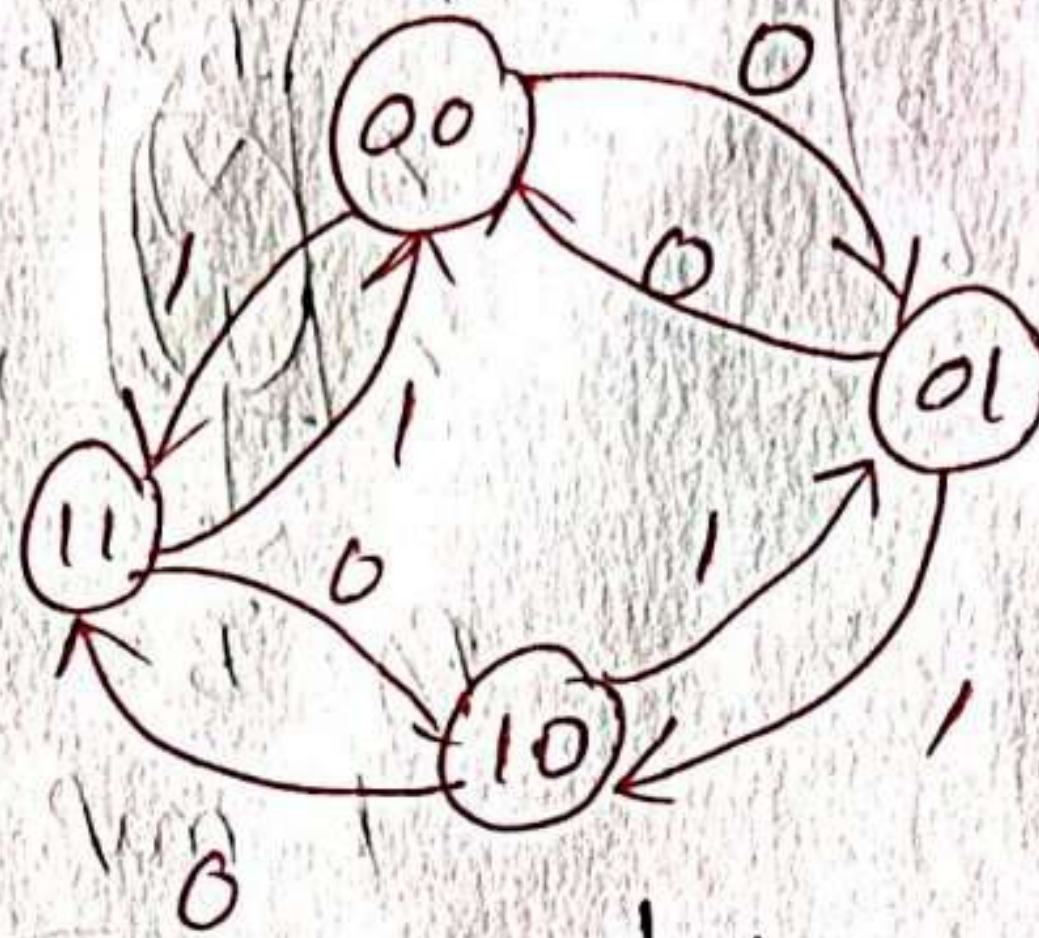
K-map for R_B

A \ B	0	1	2	3
0			1	1
1			1	1

$$R_B = B$$

2. FSM using JK flip-flops:-

Given data:-



Ques:- FSM using JK flip-flops:-

Step 1:- Write the Excitation table of JK flip

I/P		O/P	
q_n	q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Step 2:- Write the state table by using state diagram or write the flip-flop O/Ps

present state A B		I/P X	N/S $A^* B^*$		flip-flops			
					JA	KA	JB	KB
{	0 0	0	0	1	0	X	1	X
	0 0	1	1	1	1	X	1	X
{	0 1	0	0	0	0	X	X	1
	0 1	1	1	0	1	X	X	1
{	1 0	0	1	1	X	0	1	X
	1 0	1	0	1	X	1	1	X
{	1 1	0	1	0	X	0	X	1
	1 1	1	0	0	X	1	X	1

Step 3:- K-map for T_A

A \ B	0	1	1	0
0		1	1	
1	X	X	X	X

$$T_A = X$$

K-map for T_B

A \ B	0	1	1	0
0	1	1	X	X
1	1	1	X	X

$$T_B = 1$$

K-map for K_A

A \ B	0	1	1	0
0	X	X	X	X
1	1	1		

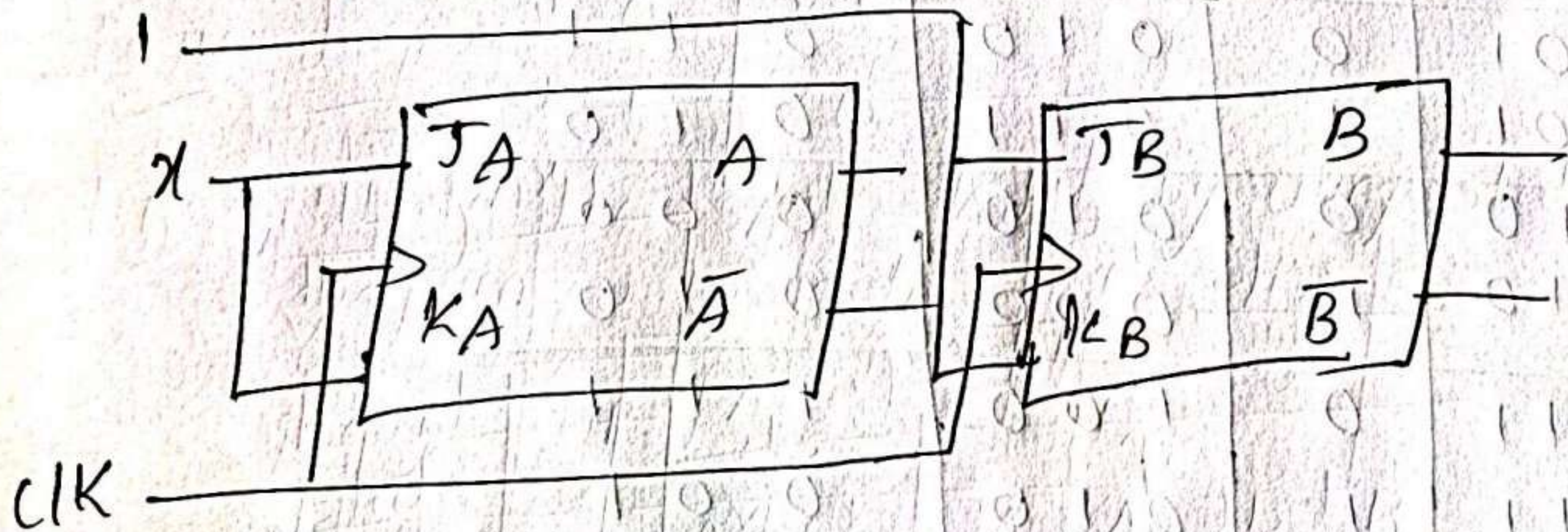
$$K_A = X$$

K-map for K_B

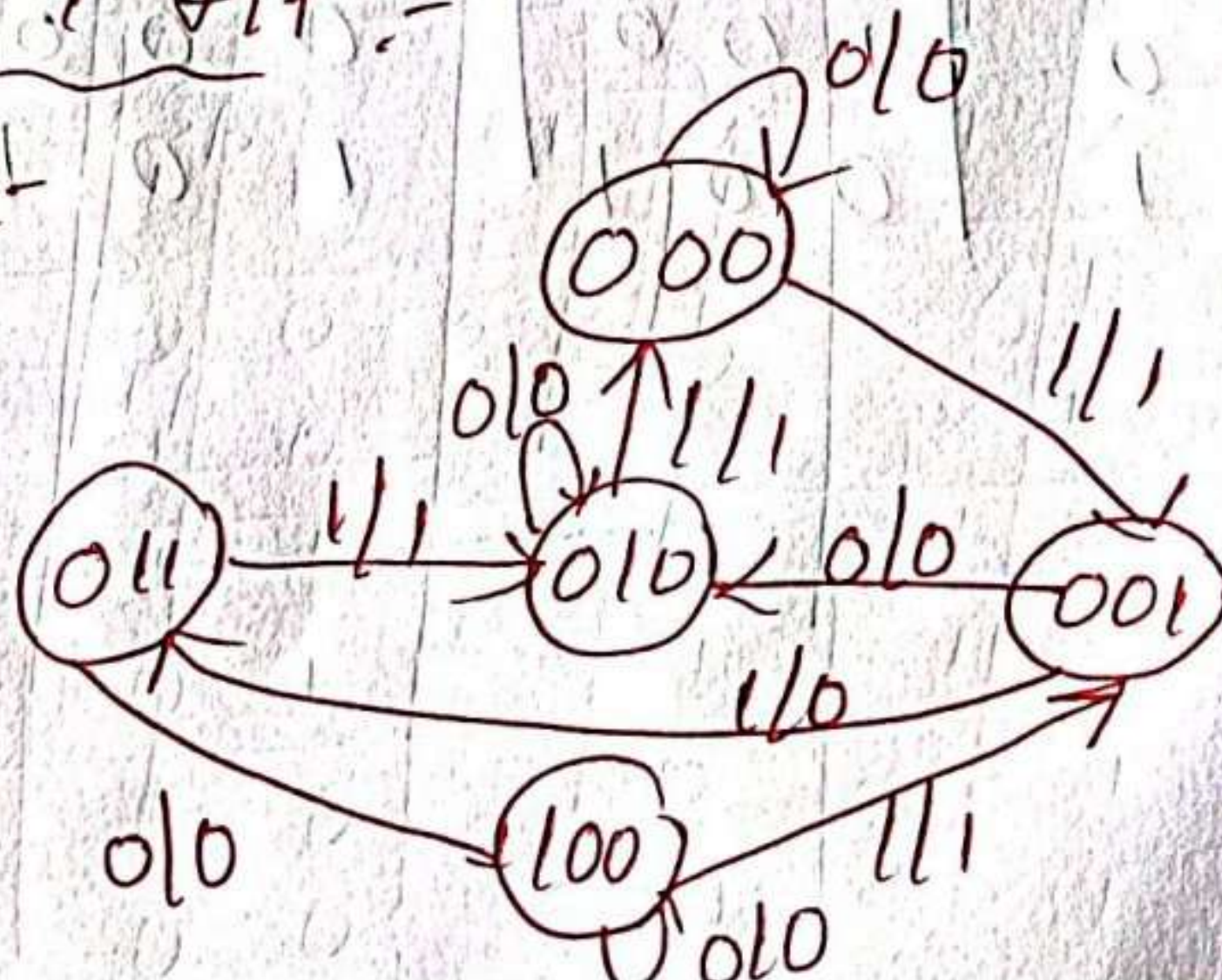
A \ B	0	1	1	0
0	X	X	1	1
1	X	X	1	1

$$K_B = 1$$

Step 4:- Draw the logic diagram.



3) FSM using T flip:-
Given data:-



Sol:- Given data

Step 1:- Write the excitation table of $\bar{T}$ flip

I/P's		O/P's
q_n	q_{n+1}	$\bar{T}$
0	0	0
0	1	1
1	0	1
1	1	0

Step 2:- Write the state table by using state diagram and write the flip's

P.S			I/P	N.S			Flip's		
A	B	C		A*	B*	C*	TA	TB	TC
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	1
0	0	1	1	0	1	1	0	1	0
0	1	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	1	0
0	1	1	0	1	0	0	1	1	1
0	1	1	1	0	1	0	0	0	1
1	0	0	0	1	0	0	0	0	0
1	0	0	1	0	0	1	1	0	1

Step 3:- K-map for T_A, T_B, T_C (5)

K-map for T_A :-

AB \ C	0	1	2	3
$\bar{A}\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10

$$T_A = \bar{A}B\bar{C} + A\bar{B}\bar{C}$$

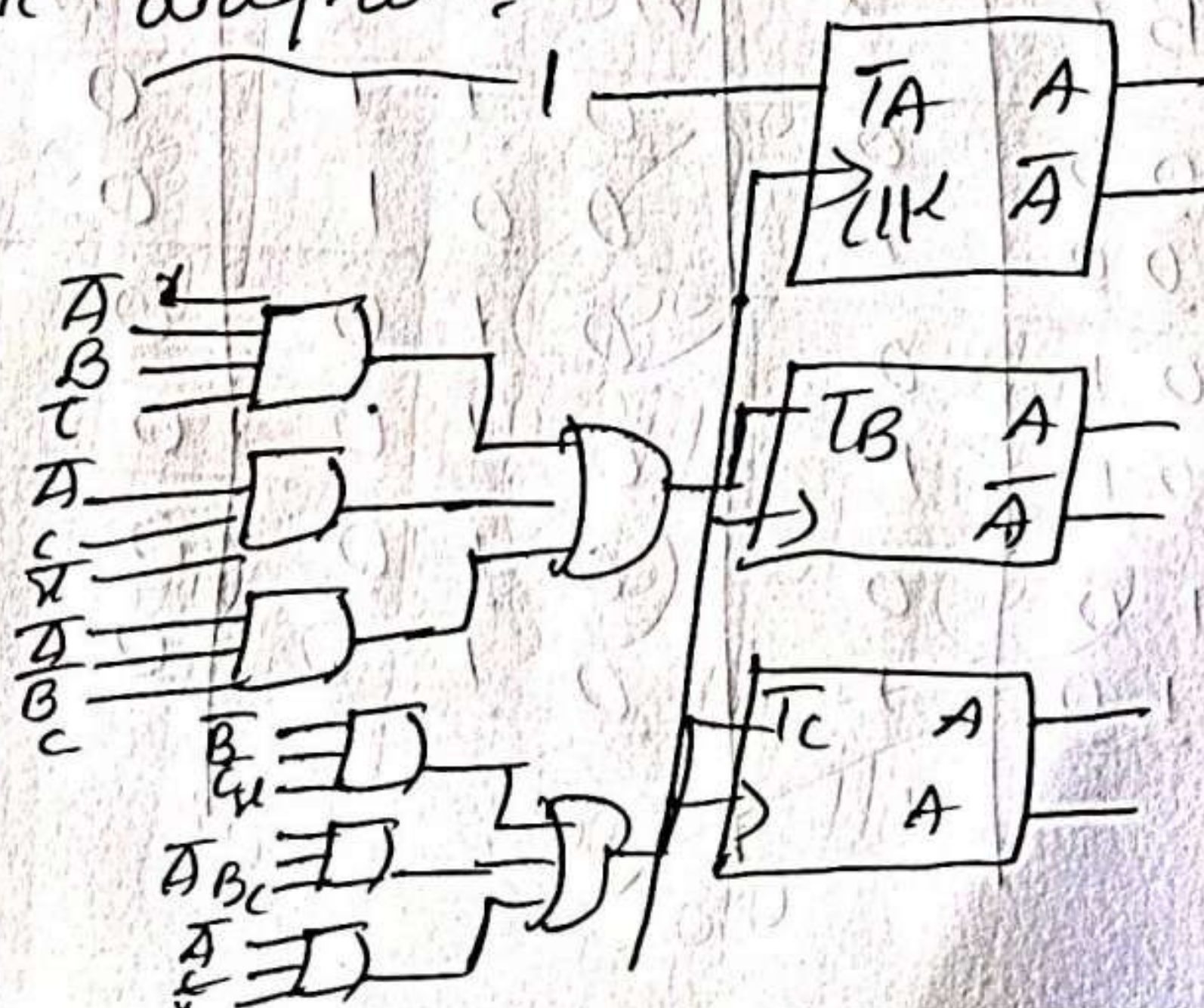
$T_A = 1$

AB \ C	0	1	2	3
$\bar{A}\bar{B}$		1		1
$\bar{A}B$			1	1
AB				
$A\bar{B}$		1		

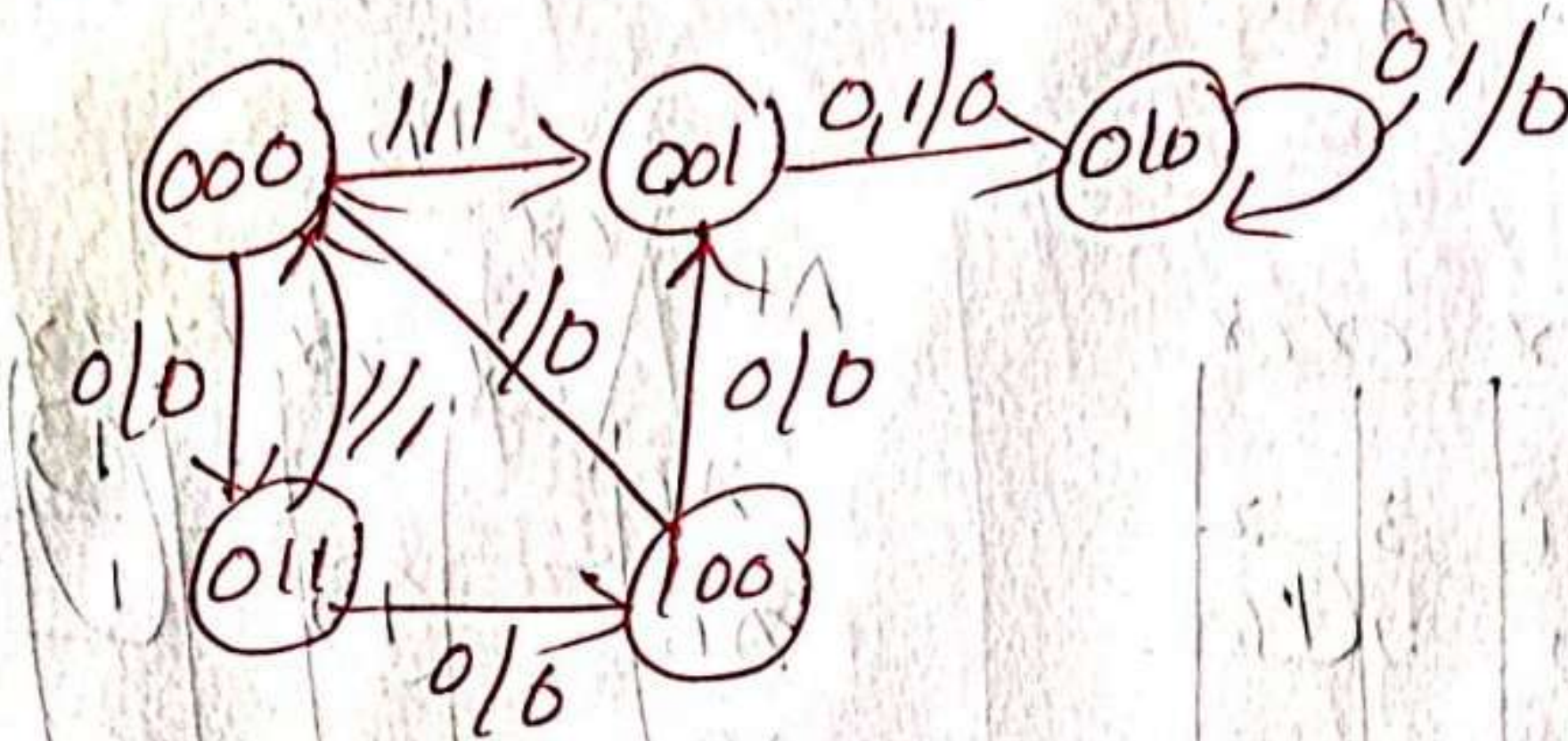
$$T_C = \bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}C\bar{B}$$

$$T_C = \bar{B}\bar{C} + \bar{A}C(B + \bar{B})$$

draw the logic diagram:-



4. FSM using D FF:-



Ans):- Given data

Step 1:- Write the excitation table of D FF

I/P's		O/P's
q_n	q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Step 2:- Write the state table by using state diagram or write the FF's O/P's

P.S			I/P	N.S			FF's		
A	B	C	X	A*	B*	C*	PA	DB	DC
0	0	0	0	0	1	1	0	1	1
0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	0
0	0	1	1	0	1	0	0	1	0
0	1	0	0	0	1	0	0	1	0
0	1	0	1	0	1	0	0	1	0

P.S	X T/P	N.S			DA	DB	DC
		A*	B*	C*			
0 1 1	0	1	0	0	1	0	0
0 1 1	1	0	0	0	0	0	0
1 0 0	0	0	0	1	0	0	1
1 0 0	1	0	0	0	0	0	0

6

Step 3:- find the K-map for DA, DB, DC

Step 4:- Draw the logic diagram

* Conversions from Mealy to Moore & Moore to Mealy

I) Convert Mealy to Moore :-

1) Given

P.S	N.S w/o P	
	x=a	x=b
q ₀	q ₃ , 0	q ₁ , 1
q ₁	q ₀ , 1	q ₃ , 0
q ₂	q ₂ , 1	q ₂ , 0
q ₃	q ₁ , 0	q ₀ , 1

Ans: Step 1:- Given data

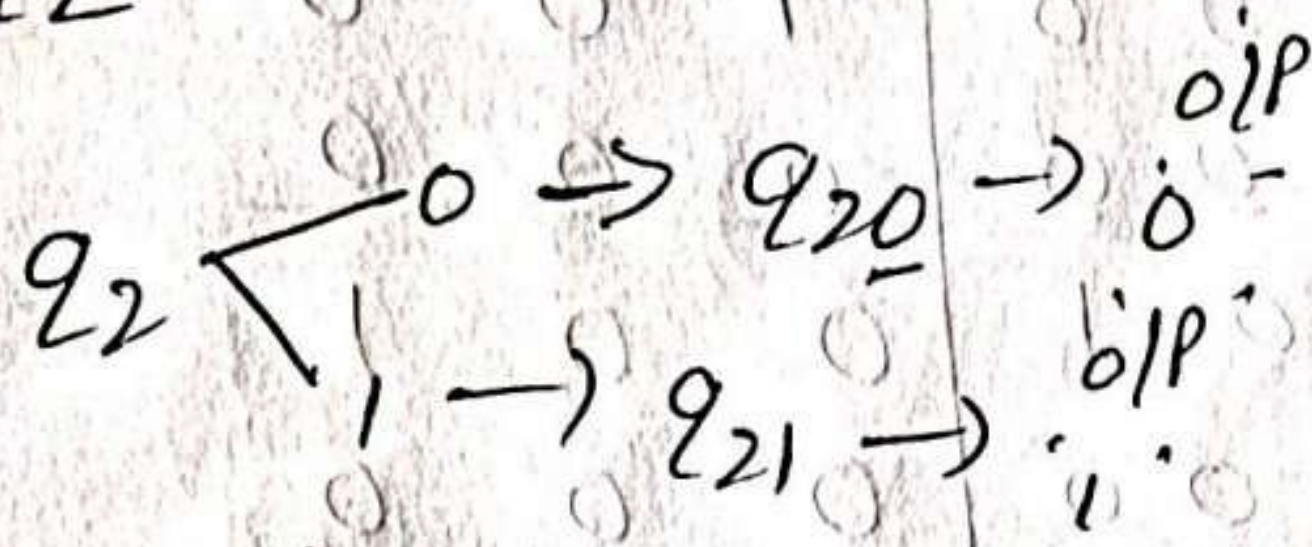
Write the equal o/p's of above state table

* q₀ = 1, only one o/p

* q₁ = have 2 o/p's, so split into

q₁ 0 → q₁₀ → o/p
 q₁ 1 → q₁₁ → o/p

* q_2 = have 2 o/p's, so split into

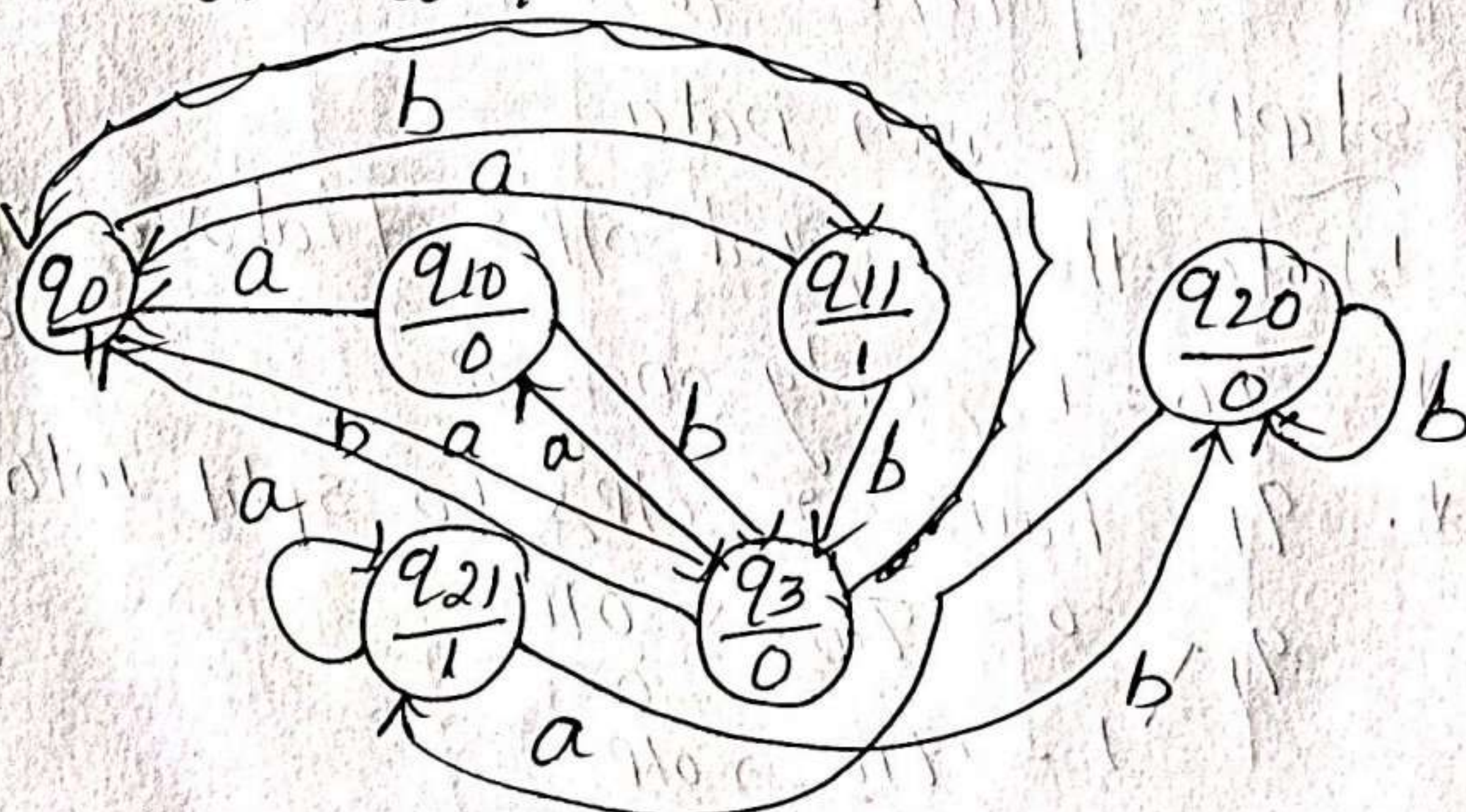


* q_3 = have 1 o/p

step 2:- Draw the new state table.

P.S	N.S		o/p
	$x=a$	$x=b$	
q_0	q_3	q_{11}	1
q_{10}	q_0	q_3	0
q_{11}	q_0	q_3	1
q_{20}	q_{21}	q_{20}	0
q_{21}	q_{21}	q_{20}	1
q_3	q_{10}	q_0	0

step 3:- Draw the state diagram by using new state table & convert into Moore state diagram



2) Convert Mealy to Moore:- (7)

<u>P.S</u>	<u>N.S</u> o/p	
	$x=0$	$x=1$
q_1	$q_3, 1$	$q_2, 0$
q_2	$q_1, 1$	$q_4, 1$
q_3	$q_2, 0$	$q_1, 1$
q_4	$q_4, 1$	$q_3, 0$

Sol:- Step 1:- Write the Equal o/p of P.S

* $q_1 = 1$, have only one o/p

* $q_2 = 0$, have only one o/p

* $q_3 =$ have 2 o/p's, so split

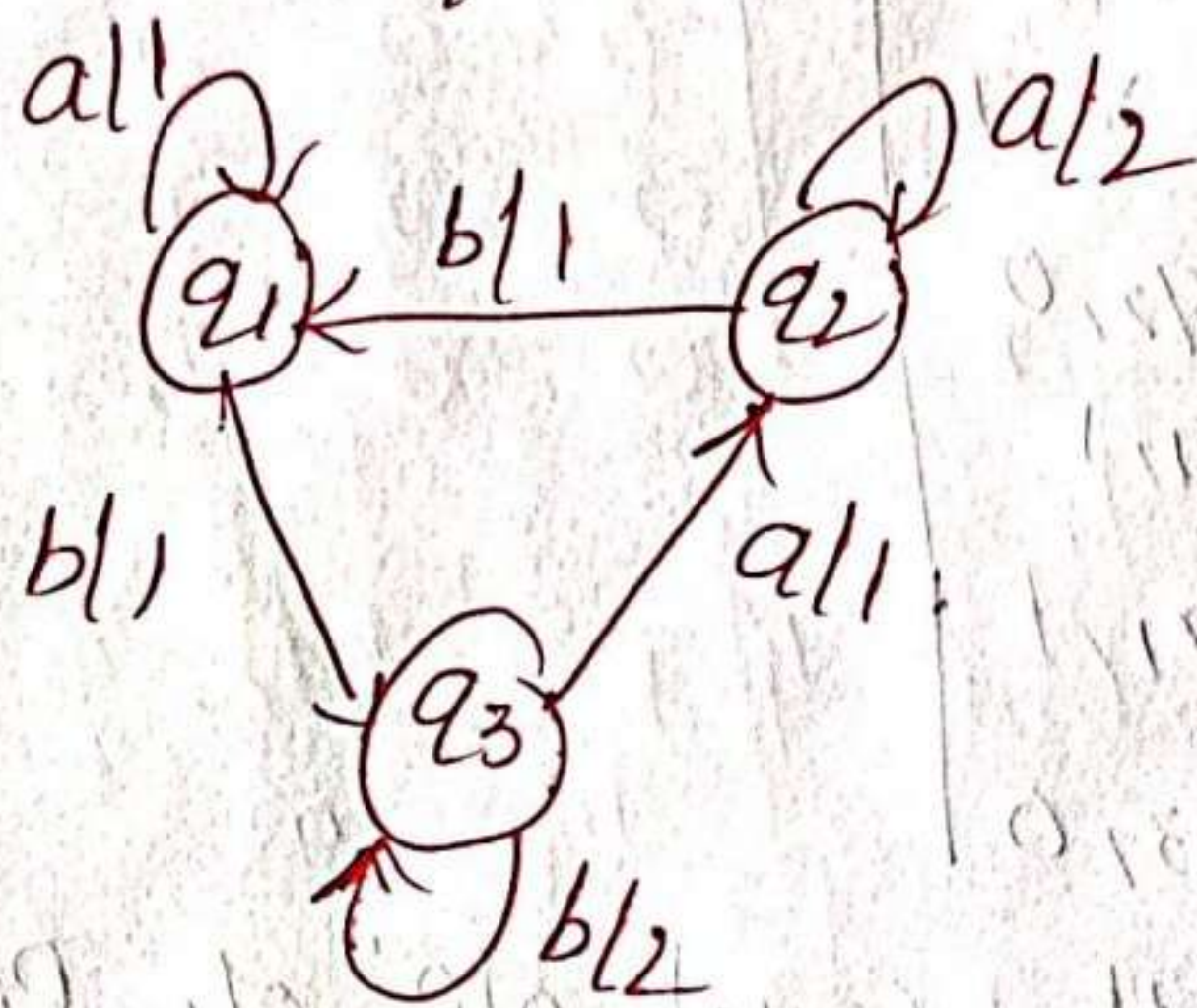
$q_3 \begin{cases} 0 \rightarrow q_{30} \rightarrow 0 \\ 1 \rightarrow q_{31} \rightarrow 1 \end{cases}$

* $q_4 = 1$ have one o/p

Step 2:- Write the New State table

<u>P.S</u>	<u>N.S</u>		<u>O/P</u>
	$x=0$	$x=1$	
q_1	q_{31}	q_2	1
q_2	q_1	q_4	0
q_{30}	q_{20}	q_1	0
q_{31}	q_2	q_1	1
q_4	q_4	q_{30}	1

3) Convert Mealy state diagram into Moore state diagram.



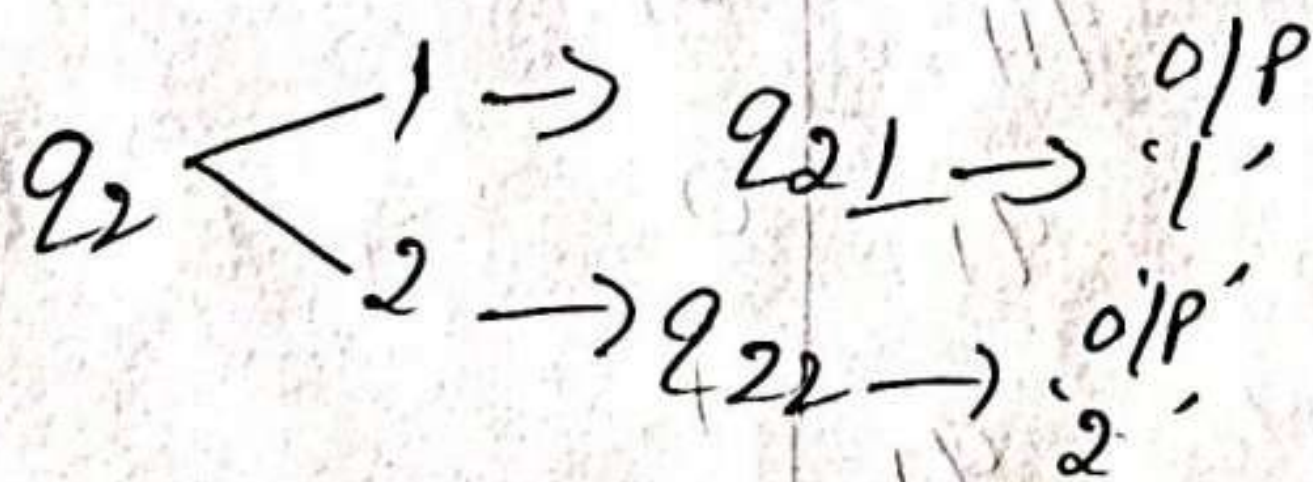
sol: step 1:- Write the state ~~diag~~ table for above state diagram

<u>P.S</u>	<u>N.S/O/P</u>	
	<u>x=a</u>	<u>x=b</u>
q1	<u>q1, 1</u>	q3, 1
q2	q2, 2	<u>q1, 1</u>
q3	q2, 1	q3, 2

step 2:- Write the equal o/p's of above state table

* q1 = 1 have only 1 o/p

* q2 = have 2 o/p's, so split into 2 parts



* $q_3 = \text{have } 2 \text{ o/p's}$

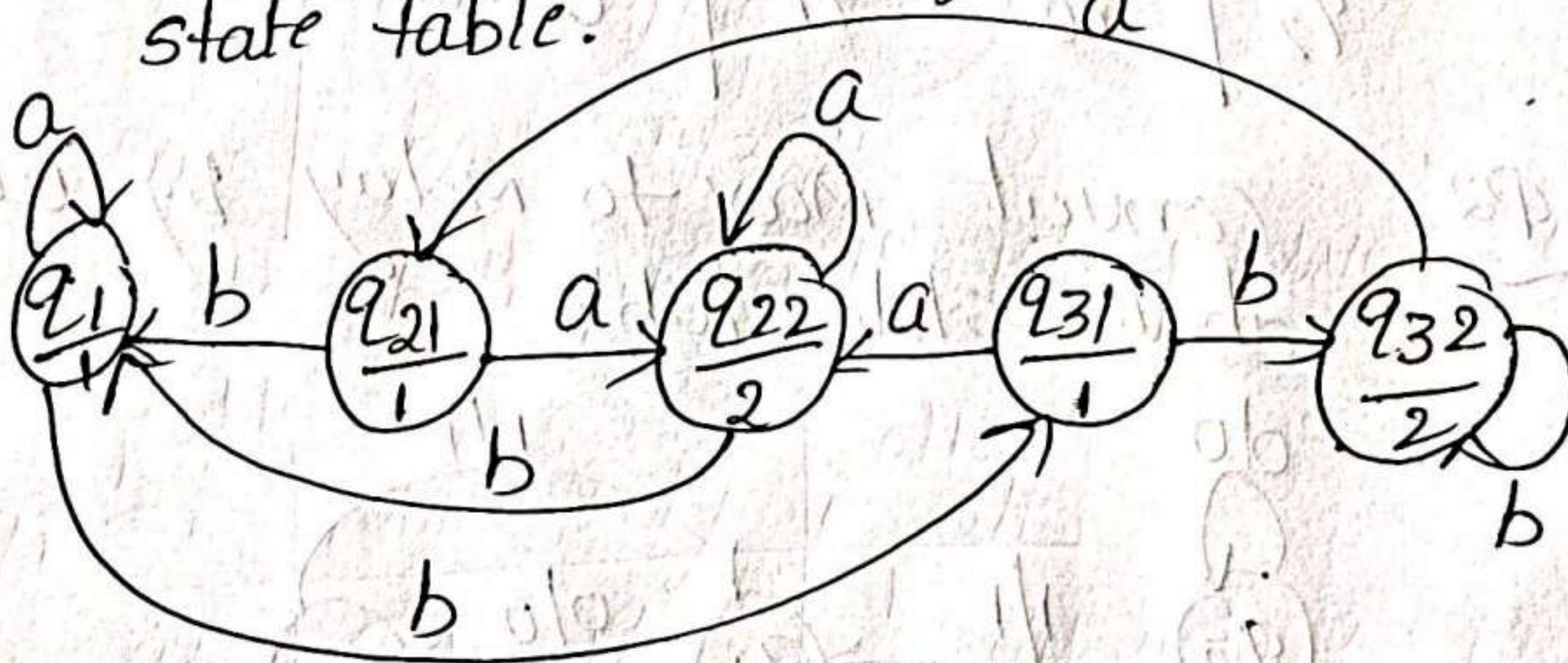
$q_3 < \begin{cases} 1 \rightarrow q_{31} = 1 \\ 2 \rightarrow q_{32} = 2 \end{cases}$

(8)

step 3:- Write the new state table

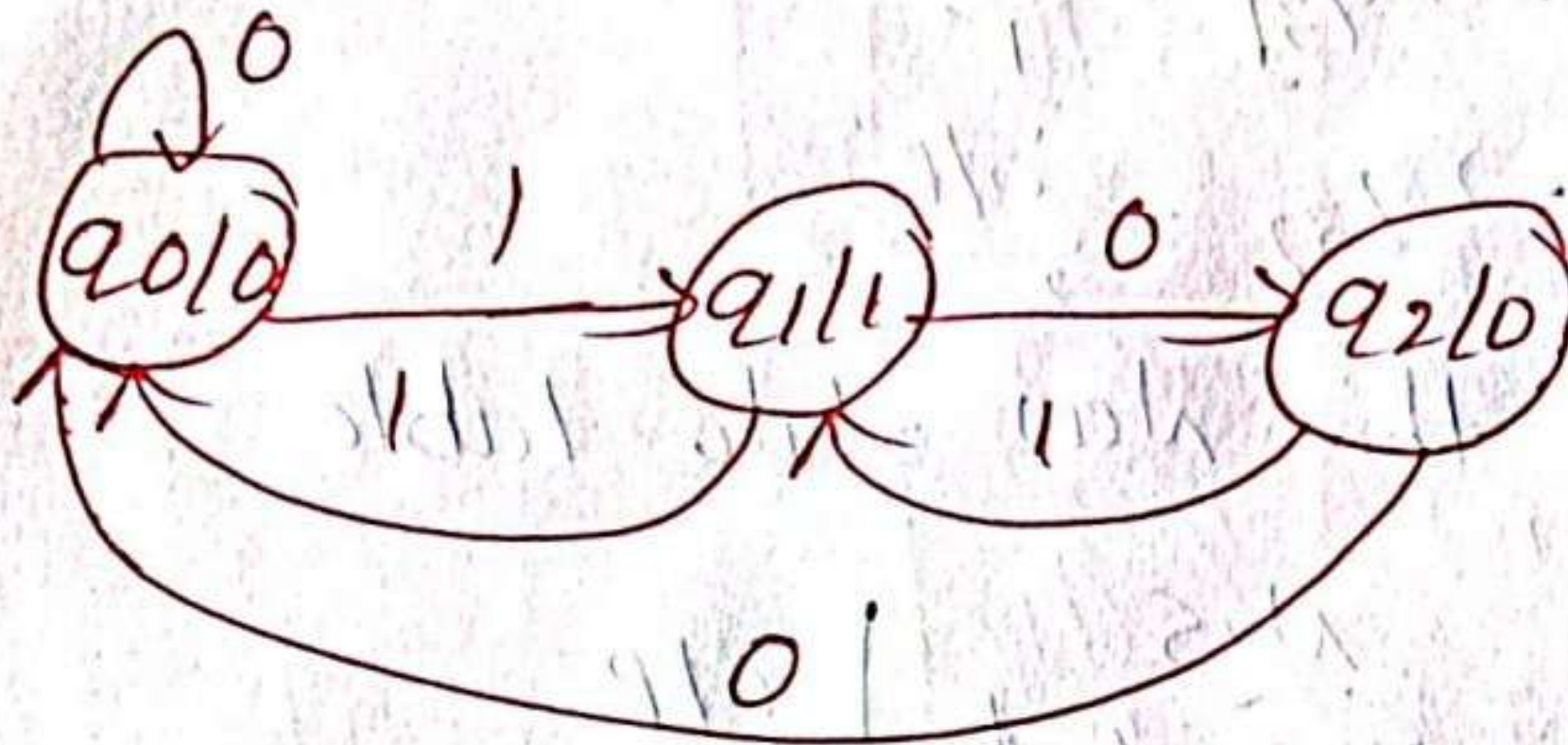
<u>P.S</u>	<u>N.S</u>		<u>O/P</u>
	$x=a$	$x=b$	
q_1	q_1	q_{31}	1
q_{21}	q_{22}	q_1	1
q_{22}	q_{22}	q_1	2
q_{31}	q_{21}	q_{32}	1
q_{32}	q_{21}	q_{32}	2

step 4:- Draw the state diagram for above state table.



Convert Moore to Mealy!:-

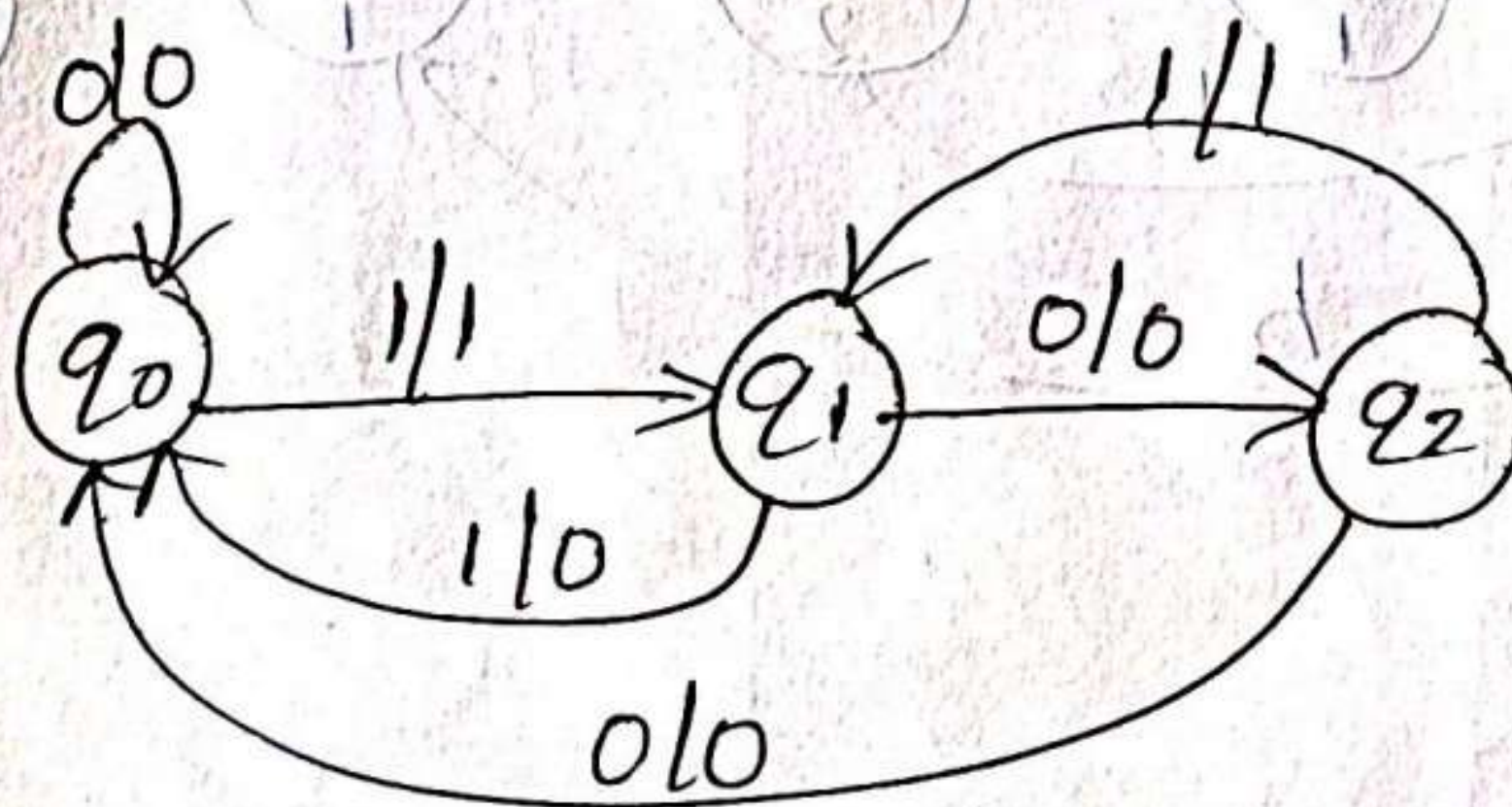
1)



Sol:- Step 1:- Write the state table by using above state diagram.

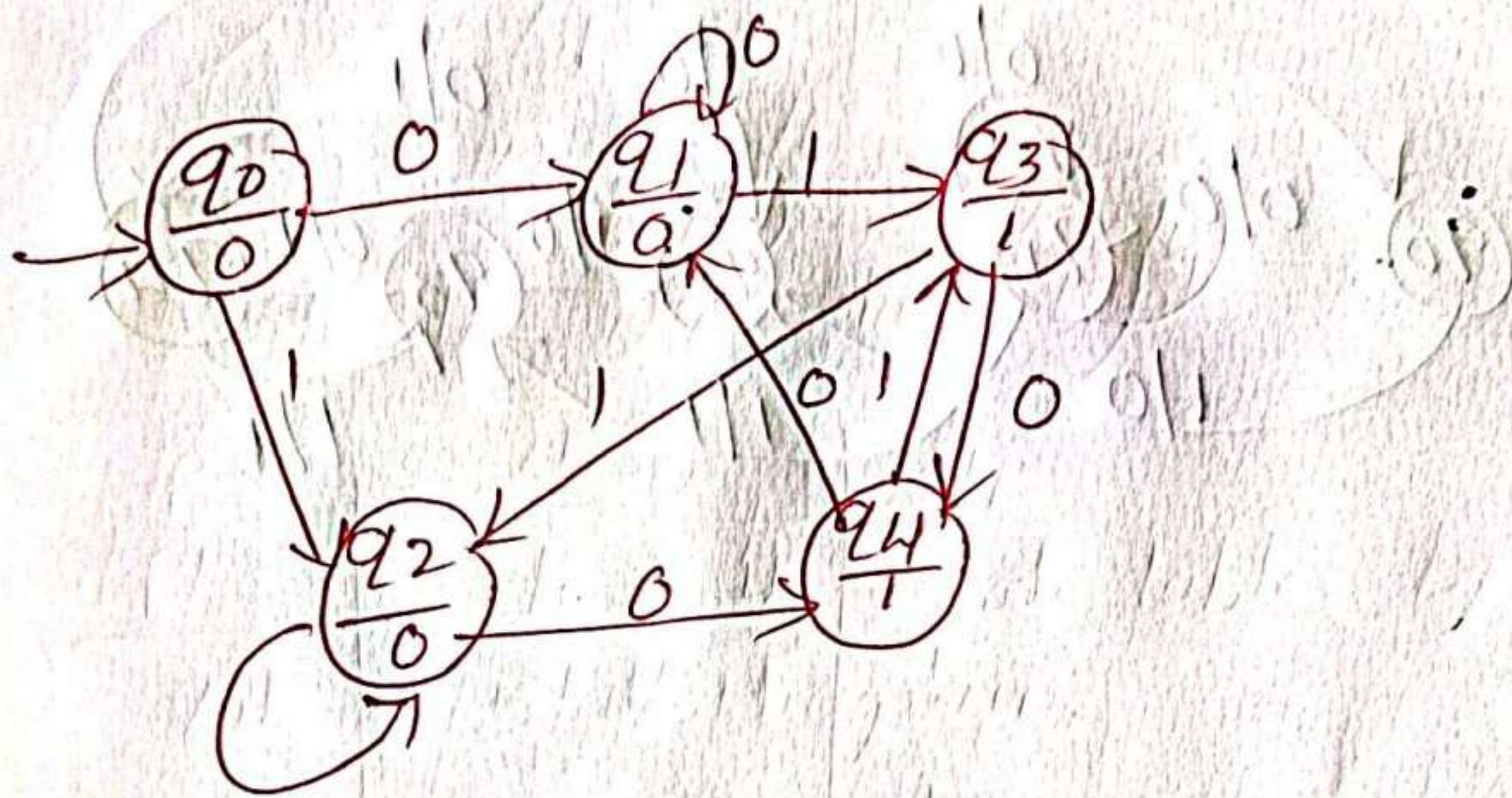
<u>P.S</u>	<u>N.S</u>		<u>O/P</u>
	$x=0$	$x=1$	
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_0	q_1	0

Step 2:- Convert Moore to Mealy by using above state table



2) Convert Moore to Mealy!

(9)

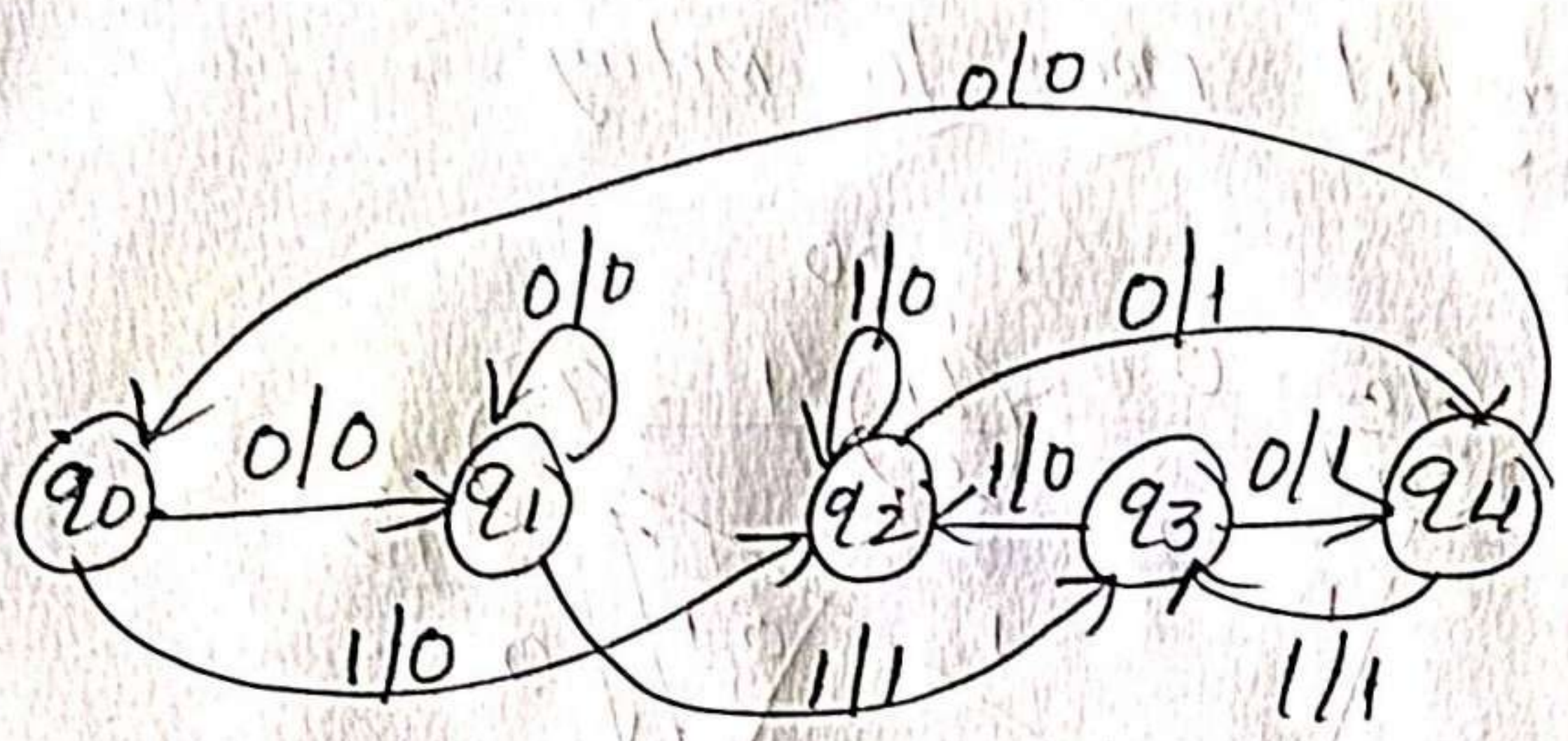


Ans:- Step 1:- Write the state table by using state diagram.

<u>P.S</u>	<u>N.S</u>		<u>O/P</u>
	$x=0$	$x=1$	
q0	q1	q2	0
q1	q1	q3	0
q2	q4	q2	0
q3	q4	q2	1
q4	q1	q3	1

Step 2:- draw the state diagram of above state table to convert Moore to Mealy

1)



	1/0	0/0	0/1	1/1
0	SP	IP	OP	
1	SP	IP	IP	
2	SP	IP	SP	
3	SP	IP	SP	
4	SP	IP	IP	

states of automata states all words of length 4 that are accepted by the automata

5. Finite State Machine - Part-II

①

PLD's (Programmable Logic Devices):-

PLD is an IC having design structure consisting of AND arrays at input followed by OR arrays at Output.

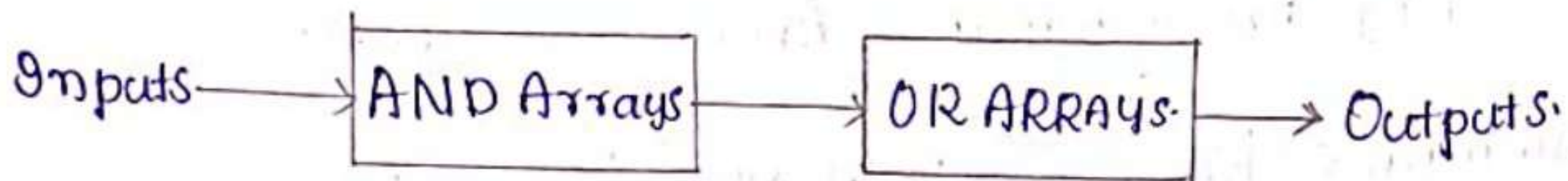


Fig. Programmable Logic Device (PLD).

Types of PLD's:-

There are three types of PLDs:

- (1) Programmable Logic Array (PLA)
- (2) Programmable Array Logic (PAL)
- (3) Programmable Read Only Memory (PROM).

(1) Programmable Logic Array (PLA):-

PLA is a programmable logic device with design structure consisting of programmable AND gates at input followed by programmable OR gates at Output.

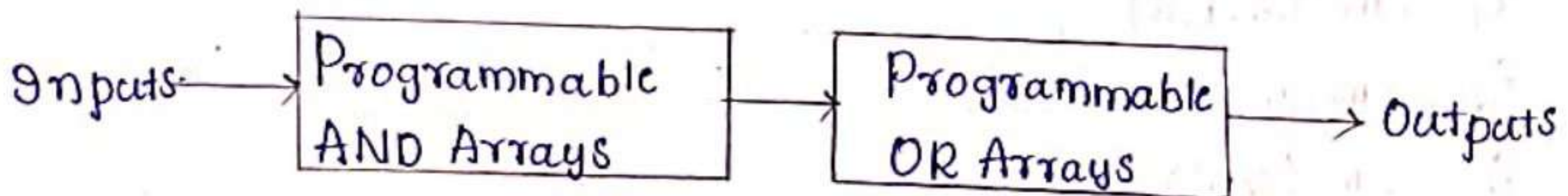


Fig. Programmable Logic Array (PLA).

(2) PROGRAMMABLE ARRAY LOGIC (PAL):

PAL is a programmable logic device with design structure consisting of programmable AND gates at input followed by fixed OR gates at output.

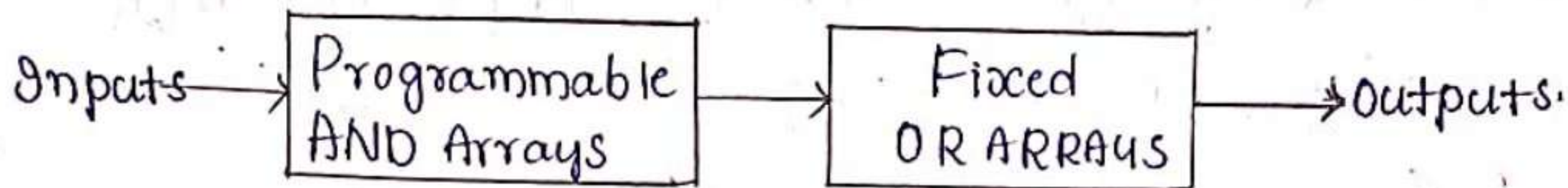


Fig. Programmable Array Logic (PAL).

(3) PROGRAMMABLE READ ONLY MEMORY (PROM):

PROM is a programmable logic device with design structure consisting of fixed AND gates at input followed by programmable OR gates at output where fixed AND gates are connected using decoders.

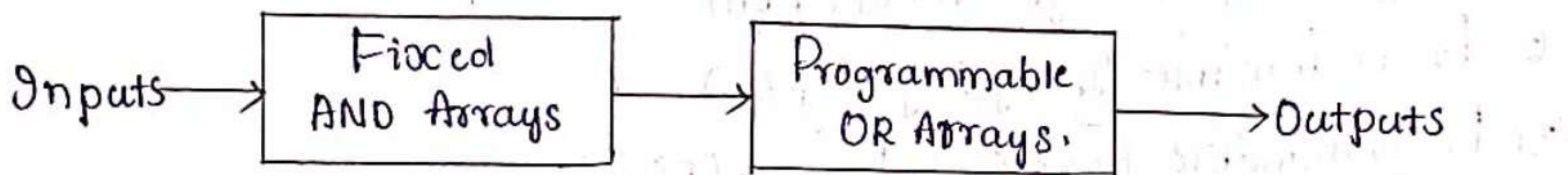


Fig. Programmable Read Only Memory (PROM).

Problems:-

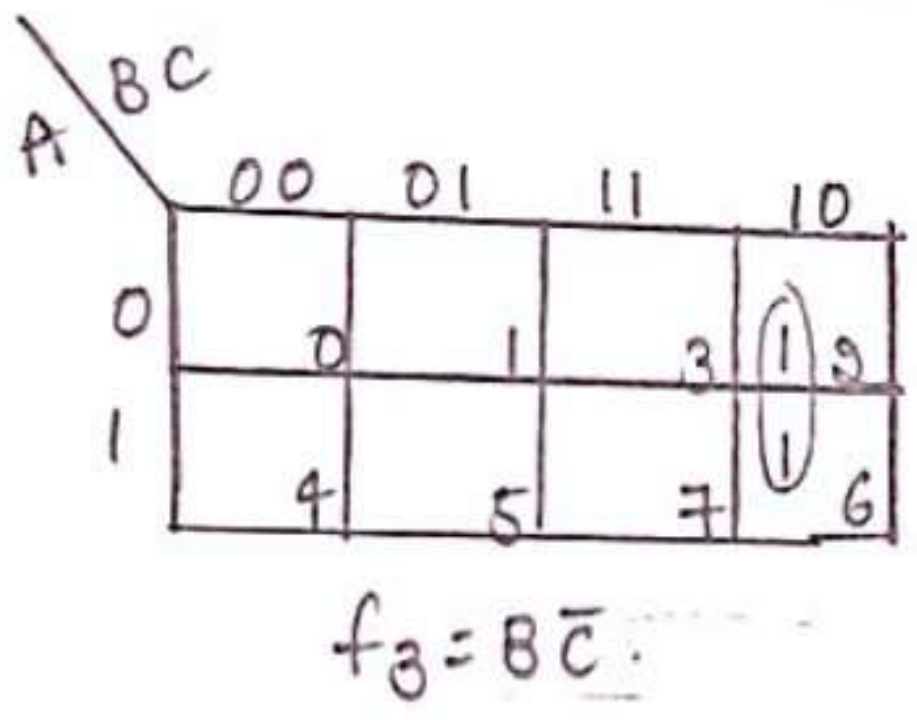
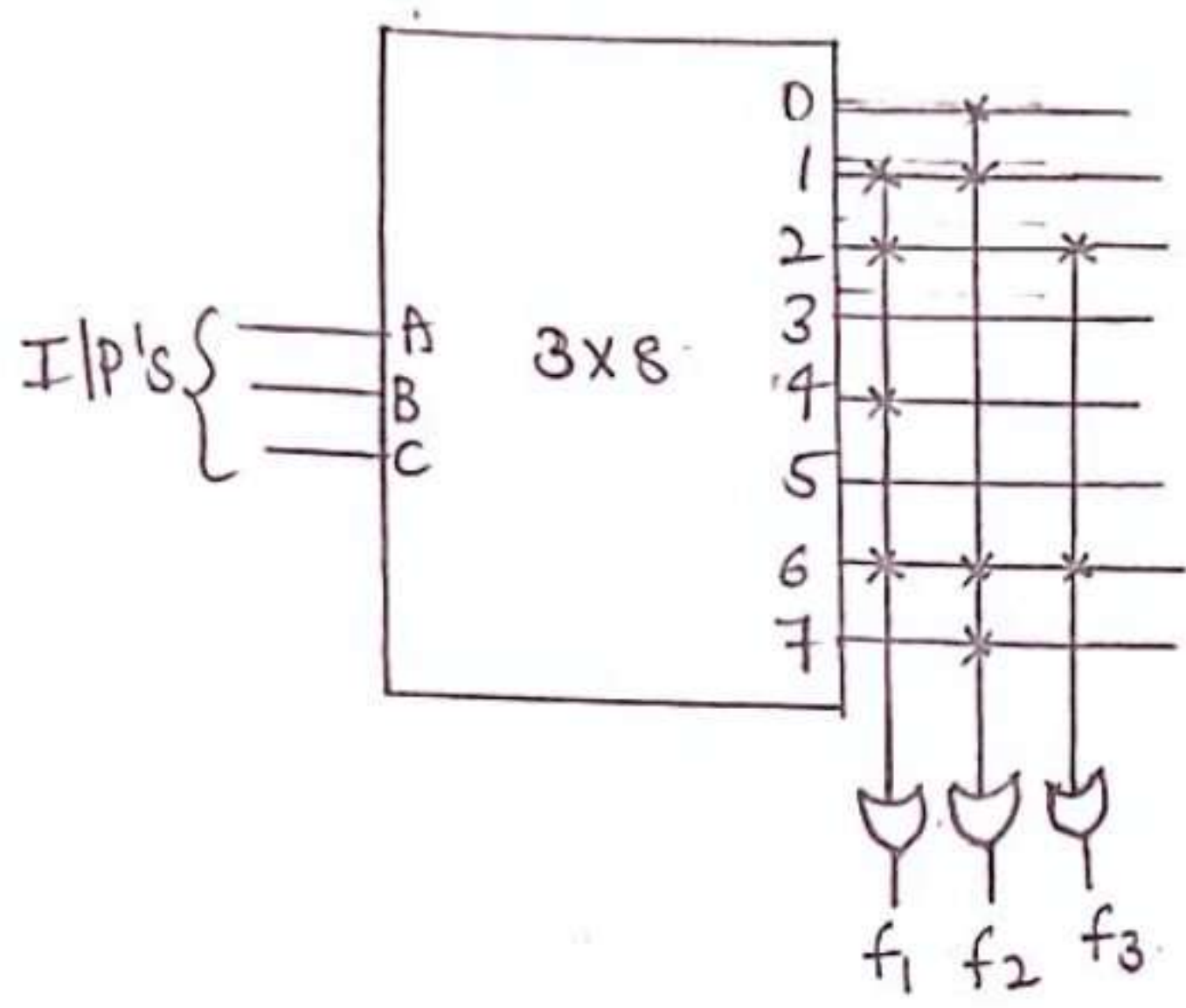
* Implement the following Boolean function by using PROM, PLA, PAL.

$$f_1 = \sum m(1, 2, 4, 6)$$

$$f_2 = \sum m(0, 1, 6, 7)$$

$$f_3 = \sum m(2, 6)$$

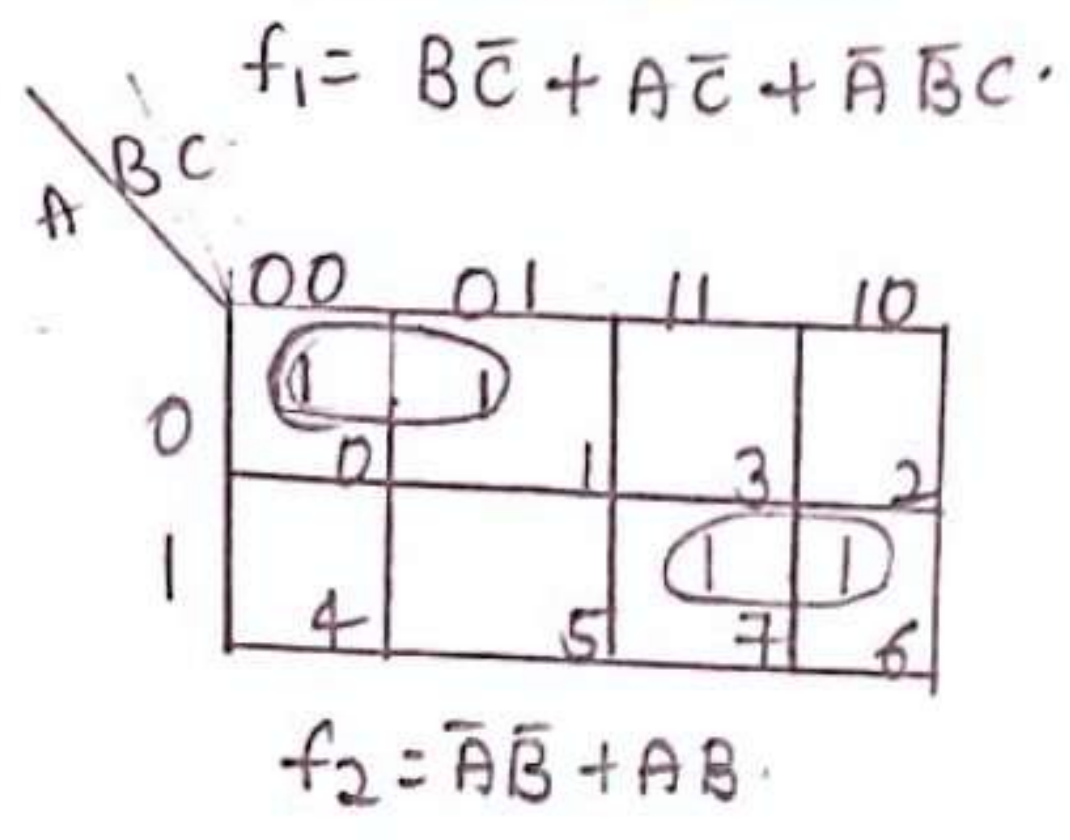
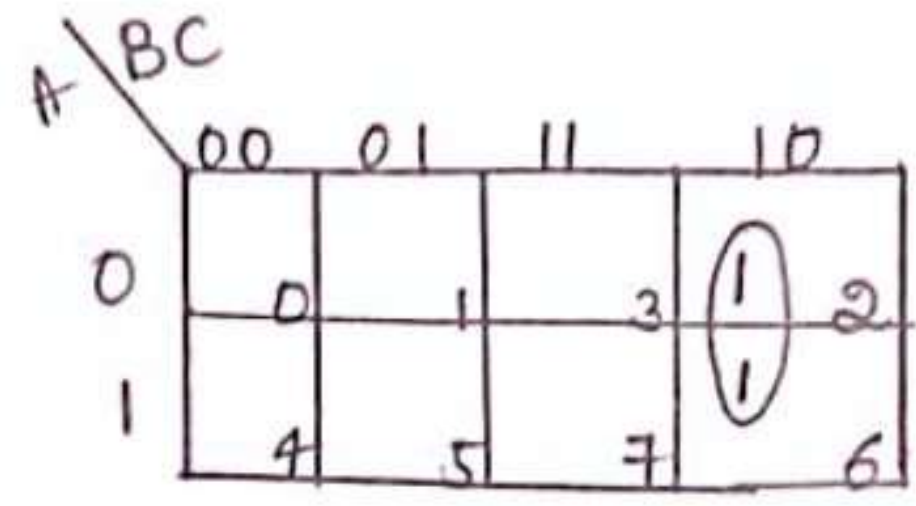
(1) PROM:-



Step 1:- Program table:-

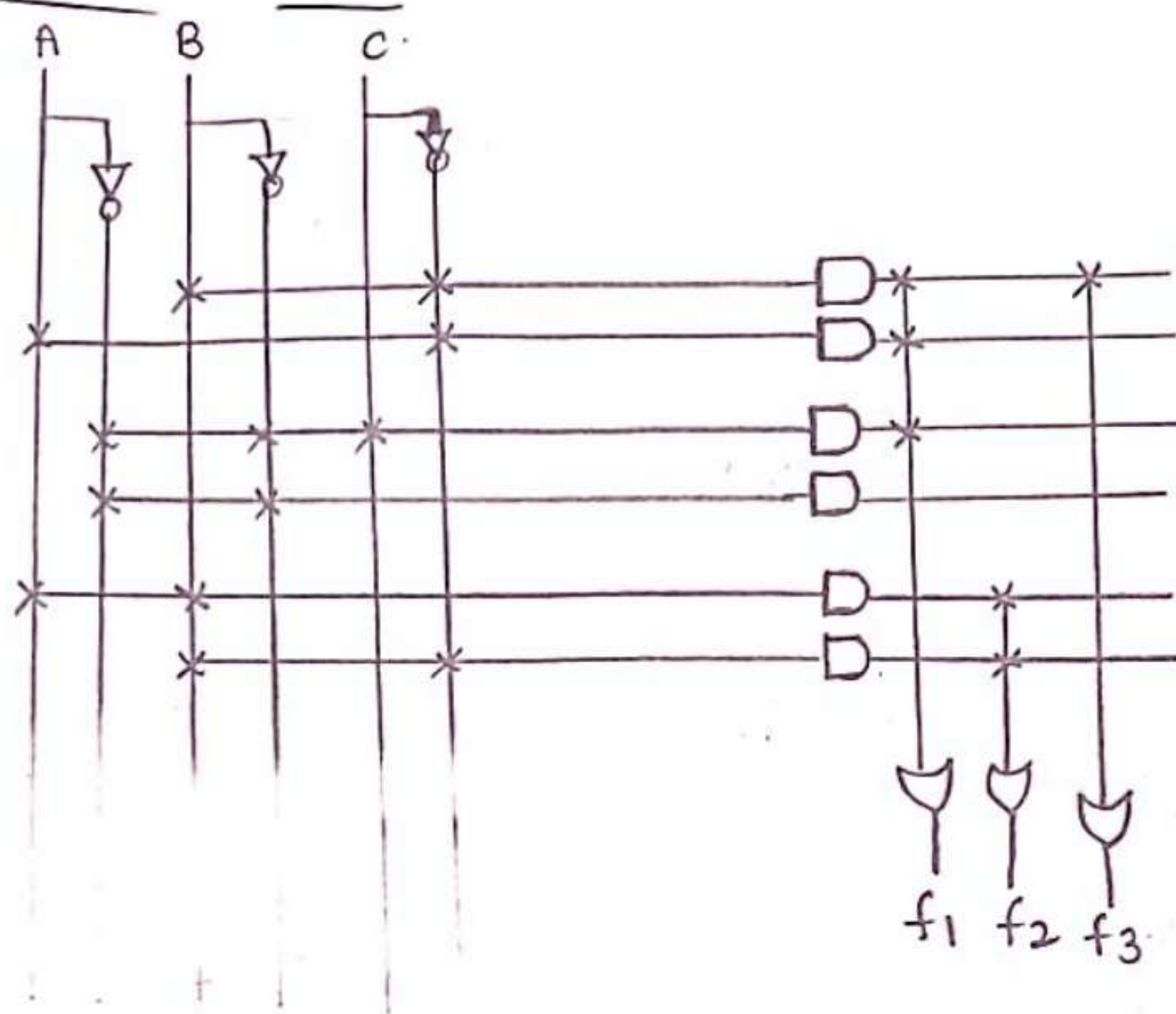
Product term	I/p's			O/p's		
	A	B	C	f_1	f_2	f_3
$\bar{B}\bar{C}$	-	1	0	1	-	1
$A\bar{C}$	1	-	0	1	-	-
$\bar{A}\bar{B}C$	0	0	1	1	-	-
$\bar{A}\bar{B}$	0	0	-	-	1	-
AB	1	1	-	-	1	-

(2) PLA:- ②

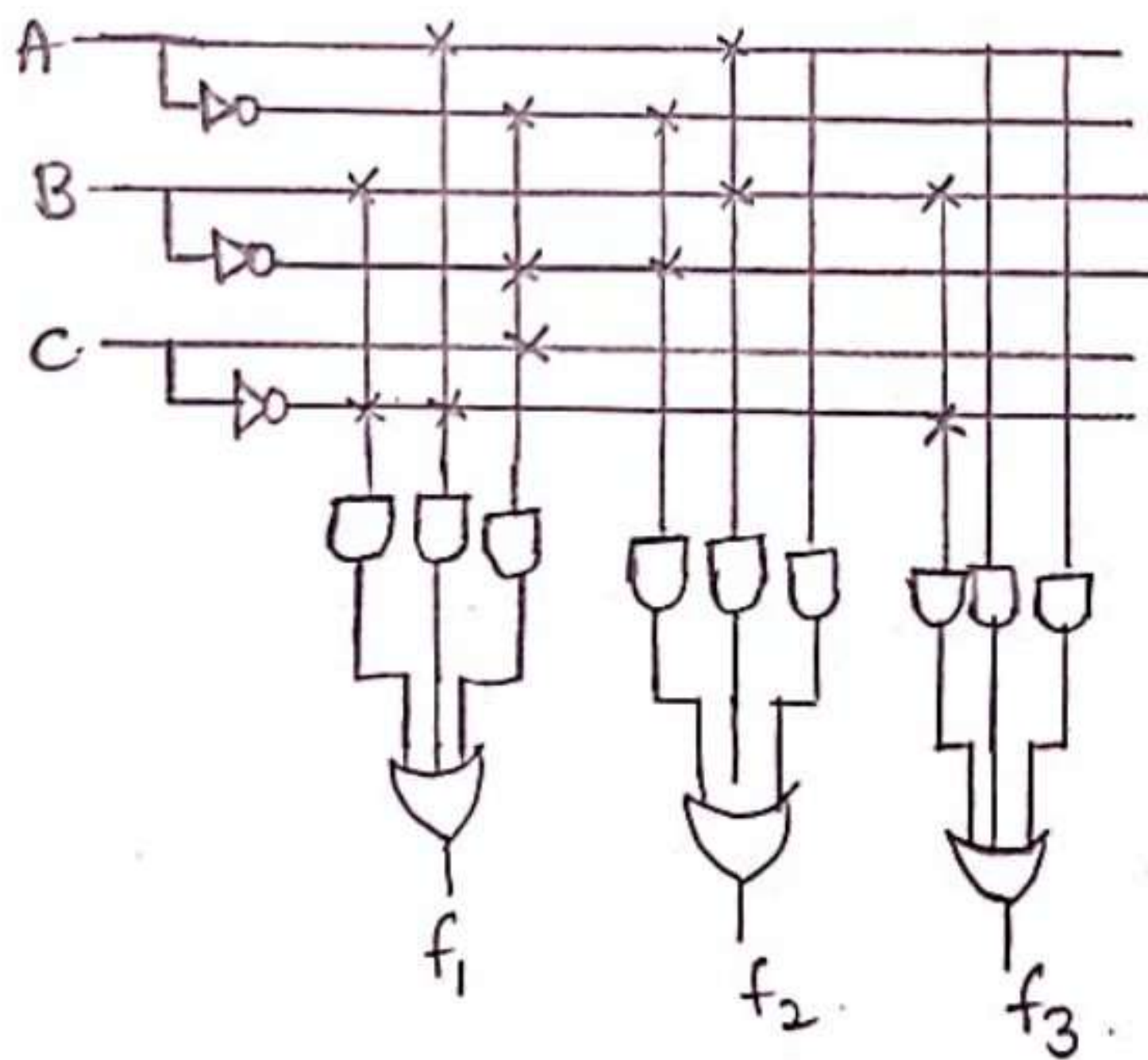


110110

Logic Diagram:-



(3) PAL:-



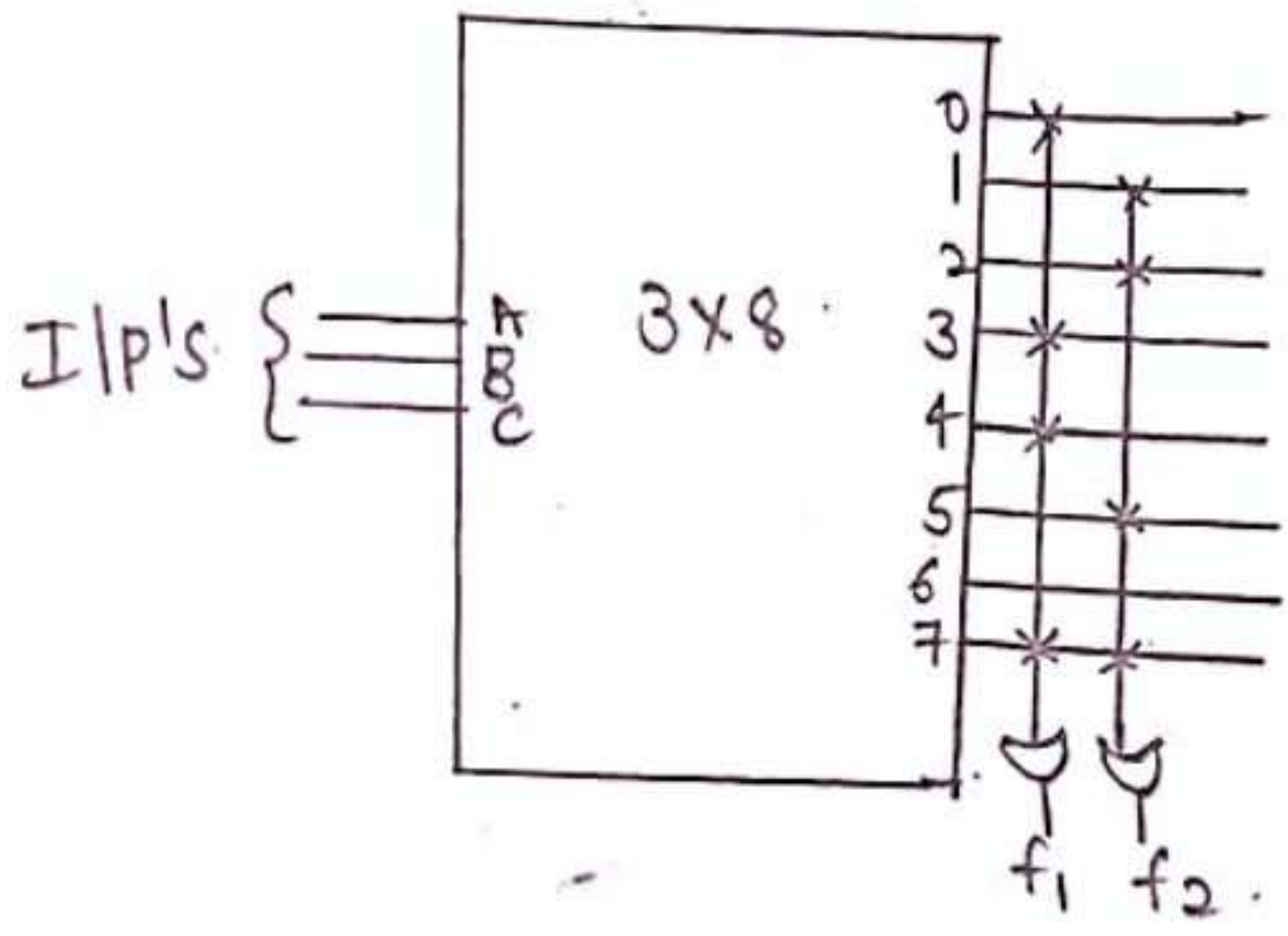
(2) Implement the following Boolean function By using PROM, PLA, PAL

3

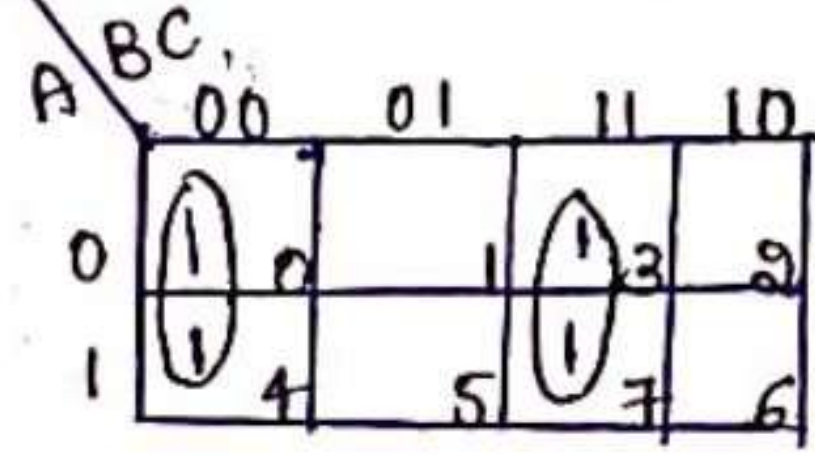
$$f_1 = \sum m(0, 3, 4, 7)$$

$$f_2 = \sum m(1, 2, 5, 7)$$

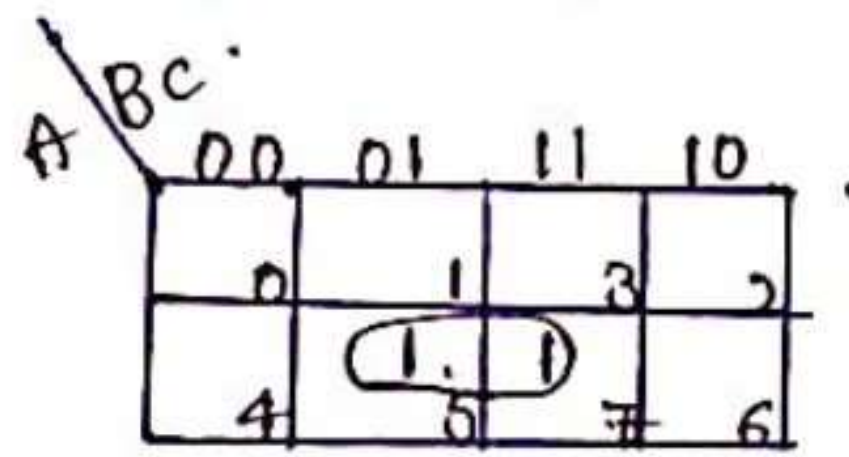
(1) PROM:-



(2) PLA:-



$$f_1 = \bar{B}C + BC$$

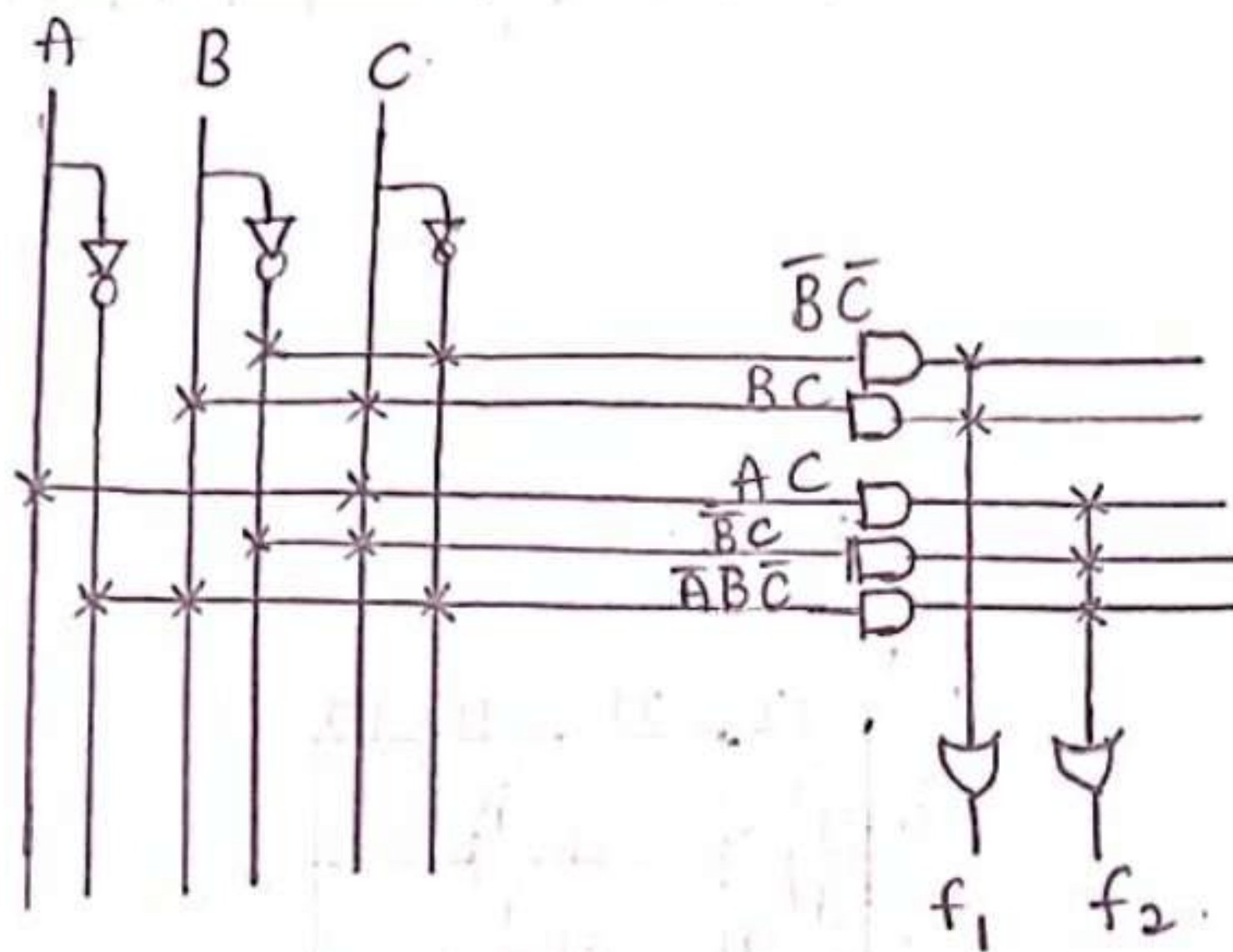


$$f_2 = AC + \bar{B}C + \bar{A}\bar{B}\bar{C}$$

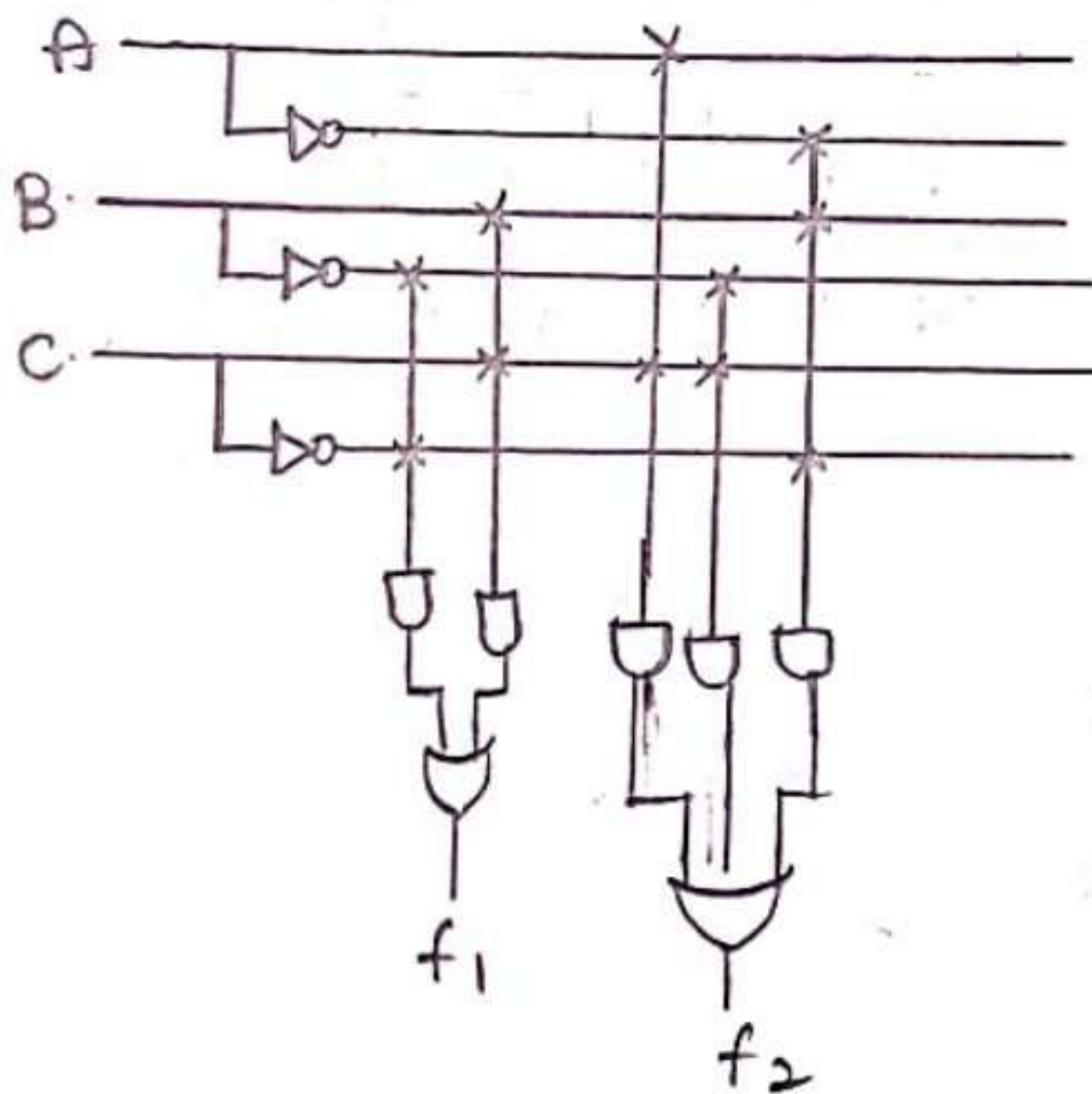
Step 1:- Program table:-

Product term	I/P's		O/P's	
	A	BC	f1	f2
$\bar{B}\bar{C}$	-	00	1	-
BC	-	11	1	-
AC	1	-1	-	1
$\bar{B}C$	-	01	-	1
$\bar{A}\bar{B}\bar{C}$	0	10	-	1

Logic Diagram:-



(3) PAL:-



Differences between PROM, PAL and PLA.

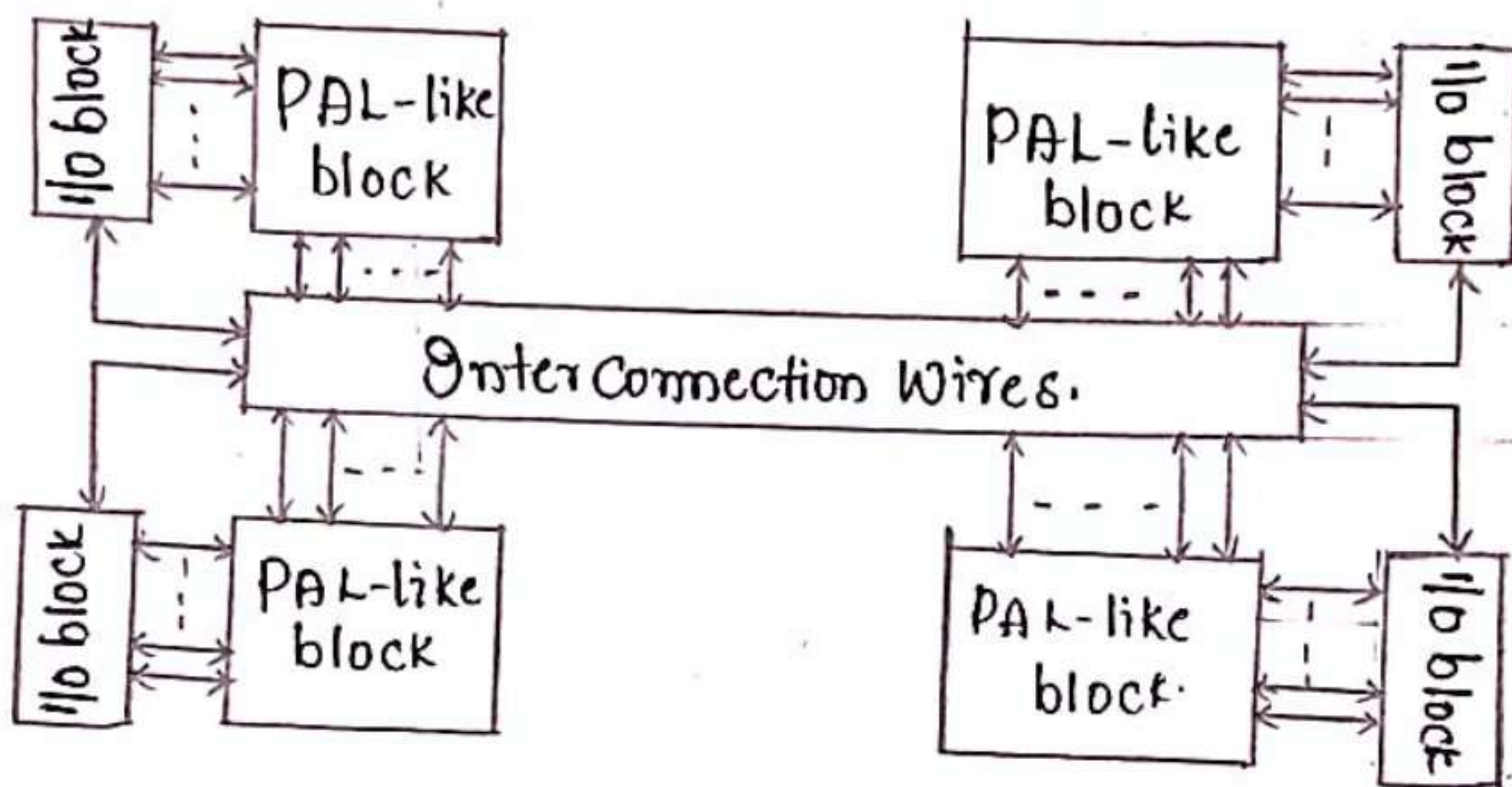
PROM	PAL	PLA.
(1) PROM Stands for Programmable Read Only Memory.	(1) PAL Stands for Programmable Array Logic.	(1) PLA Stands for Programmable Logic Array.
(2) PROM having fixed AND plane and programmable OR plane.	(2) PAL having programmable AND plane & fixed OR plane.	(2) PLA having programmable AND plane and programmable OR plane.
(3) With the help of PROM device we can implement limited no. of SOP terms.	(3) With the help of PAL device, we can implement any type of SOP terms.	(3) With the help of PLA we can implement any type of SOP terms.
(4) PROM device is a less expensive device.	(4) PAL device is also less expensive device compared to PLA.	(4) PLA is more expensive than the PAL and PROM.
(5) PROM having less flexibility.	(5) PAL having less flexibility compared to PLA.	(5) PLA is more expensive flexibility than PROM and PAL.
(6) PROM is a medium scale integrated circuit.	(6) PAL is a medium scale integrated circuit.	(6) PLA is a large scale integrated circuit.
(7) PROM device having storage capability.	(7) PAL doesn't have storage capability.	(7) PLA doesn't have storage capability.
(8) Block diagrams.	(8) Block diagram.	(8) Block diagram.

Basic Structure of CPLD and FPGA:-

5

CPLD:-

Complex Programmable Logic Device:-



- (1) PAL/PLA
- (2) Inter Connection between devices is programmable.
- (3) Memory element- F.F.

Why CPLD?

- (1) The simple programmable logic devices i.e., PLA, PAL have limited no. of inputs, outputs and product terms.
- (2) It supports only 32 input and output terms.
- (3) For implementation of circuits that require more inputs and outputs than that are available in a single SPLD chip, either multiple SPLD chips can be employed.

(4) To overcome these limitations, more sophisticated chips like "CPLD" needed to design the digital circuits.

(1) Function Blocks - PAL/PLA like blocks:-

→ Each block usually contains about "16 macro cells".

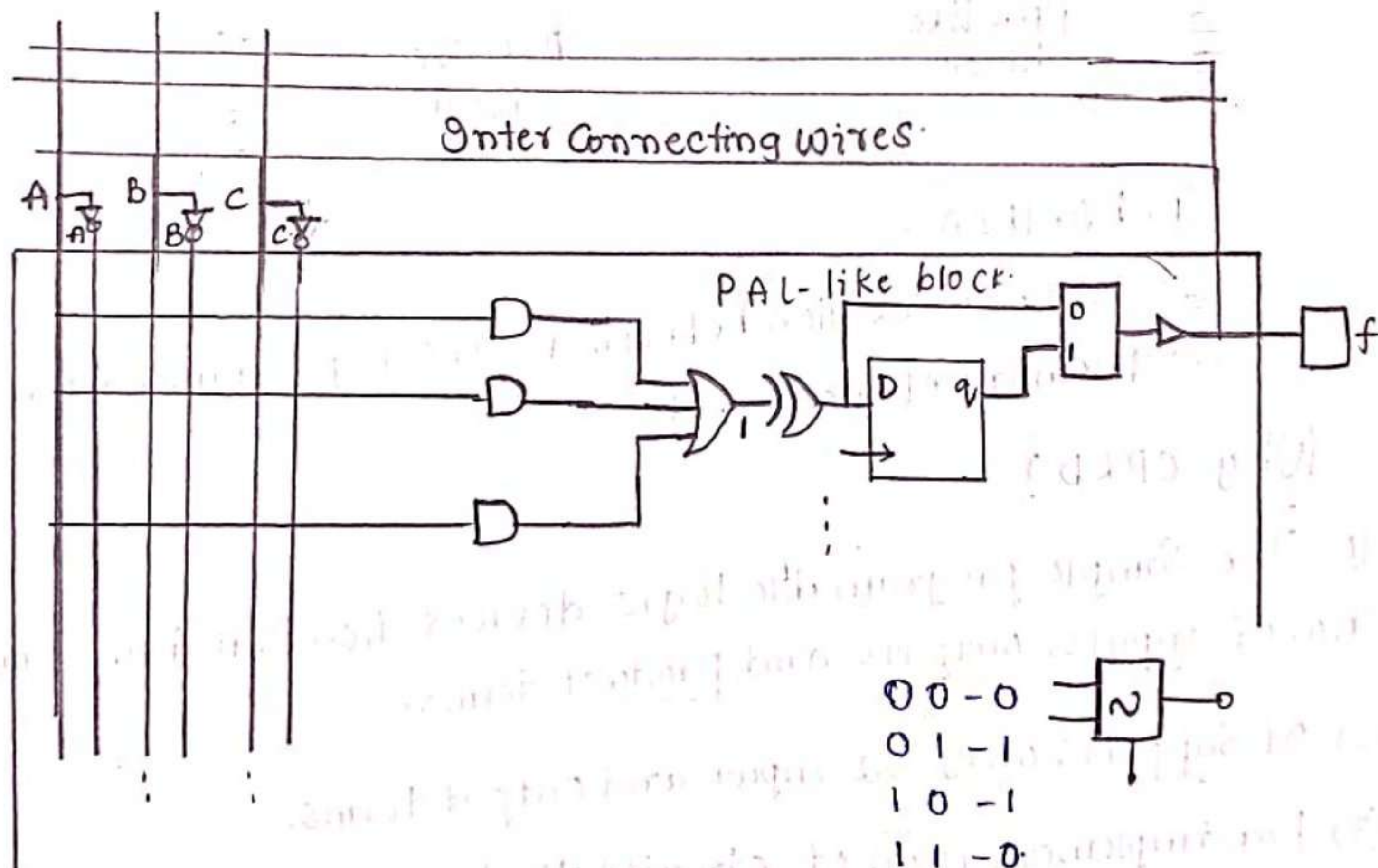
→ Each macro cell contains:

↳ (1) AND-OR Configuration.

(2) Ex-OR gate

(3) flip-flop

(4) A multiplexer.



→ Each AND-OR has 5-20 AND gates and OR gate with 5 to 20 inputs.

→ An Ex-OR gate is used to obtain the output of OR gate in an inverted (or) non-inverted form depending upon its other inputs being 1 to 0 respectively.

→ A D flip-flop stores the Ex-OR gate output.

→ Multiplexer selects either the output of Ex-OR gate or D-Flip flop depending upon select lines.

→ A Tri-State buffer acts as a Switch, which enables the chip's PM to be used as output or as an input.

CPLD Advantages:-

- (1) It is easy to design.
- (2) It has a low development cost.
- (3) It helps to reduce the board area.
- (4) It offers high reliability.

Disadvantages:

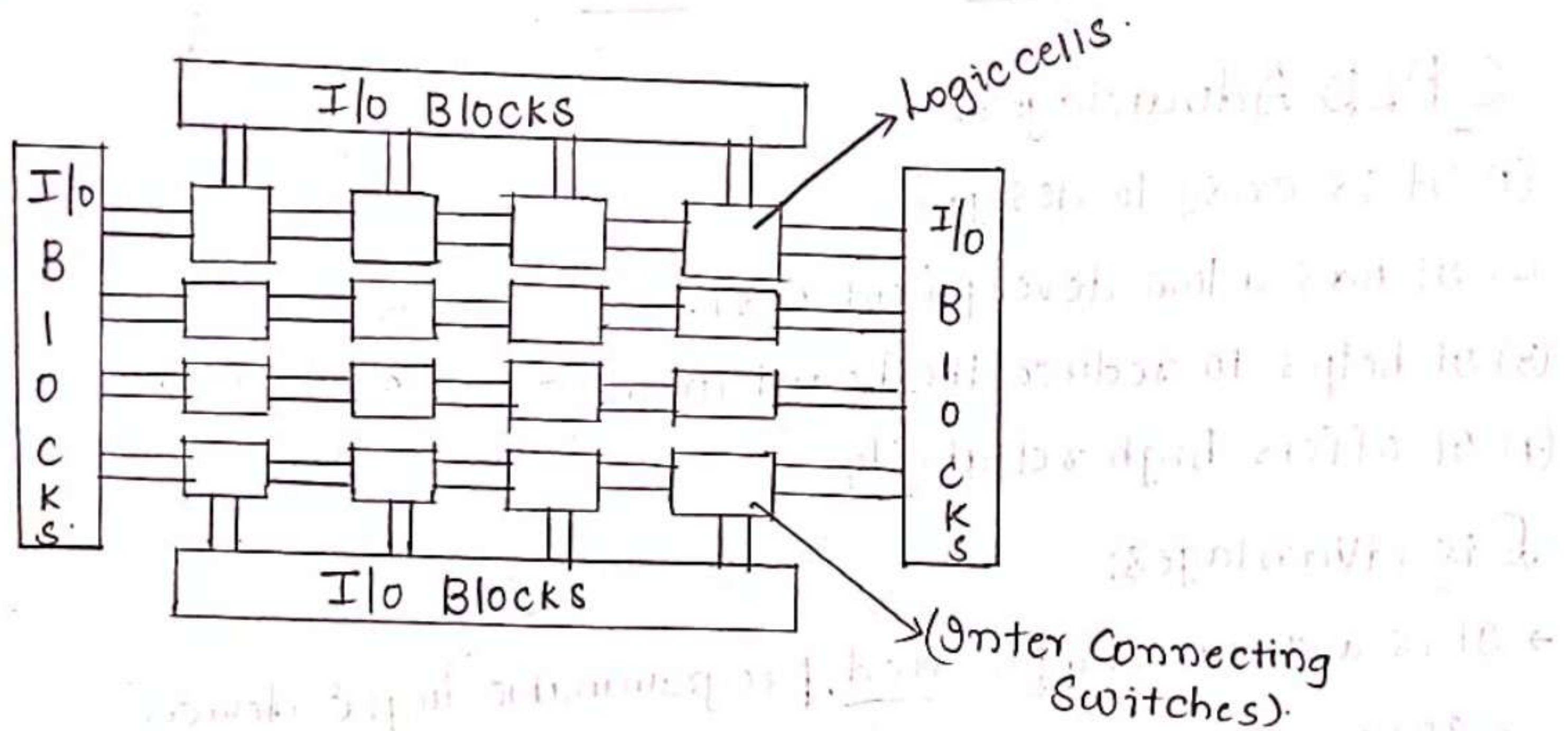
- It is a more Complicated programmable logic device.
- SPLD as

FPGA:- (Field Programmable Gate Array):-

Why FPGA?

- The SPLDs and CPLDs are based on the same architecture PLA, PAL.
- Over the years as PLDs are grown in Complexity, Size with highly Configurable "macrocell" addition.
- Hence to increase Size and to add more functionality in a single PLD, FPGA are introduced.

Architecture:-



→ VLSI (It is VLSI circuit)

- (2) More 1000 no. of gates ^{we} can implement single IC.
- (3) Logic called inter connecting between them is programmable.

→ It doesn't contain any AND, OR, planes.

→ It contains logic blocks to implement logic functions:

FPGA contains

- Configurable logic Blocks (CLBs)
- Configurable I/O blocks
- Programmable inter connection Paths &
- These blocks are arranged as array of rows & columns.
- The logic blocks are connected to the I/O blocks through Programmable inter connects.
- The logic blocks contain a large no. of logic modules (LM).

Advantages of FPGA :-

- (1) FPGA is Fast
- (2) FPGA is suitable for micro processor.
- (3) FPGA is suitable for parallel data processing
- (4) FPGA is used for "Real Time Applications"

Disadvantages:-

- (1) FPGA is slightly costly.
- (2) It's program is not easier.
- (3) FPGA can't handle complex designs.

Examples:-

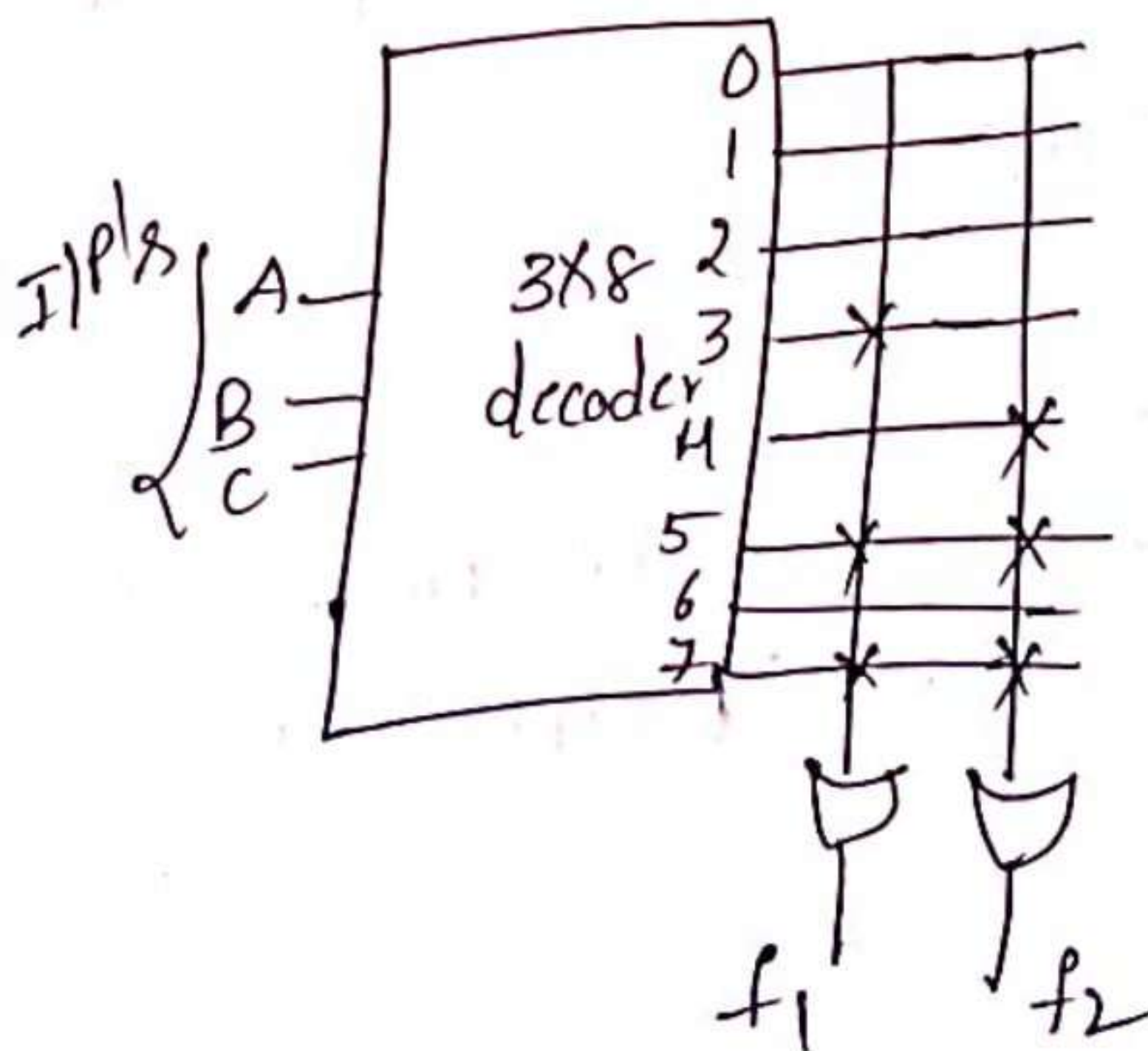
1) Implement the following Boolean function by using PROM, PAL, or PLA

$$f_1 = \sum m(3, 5, 7)$$

$$f_2 = \sum m(4, 5, 7)$$

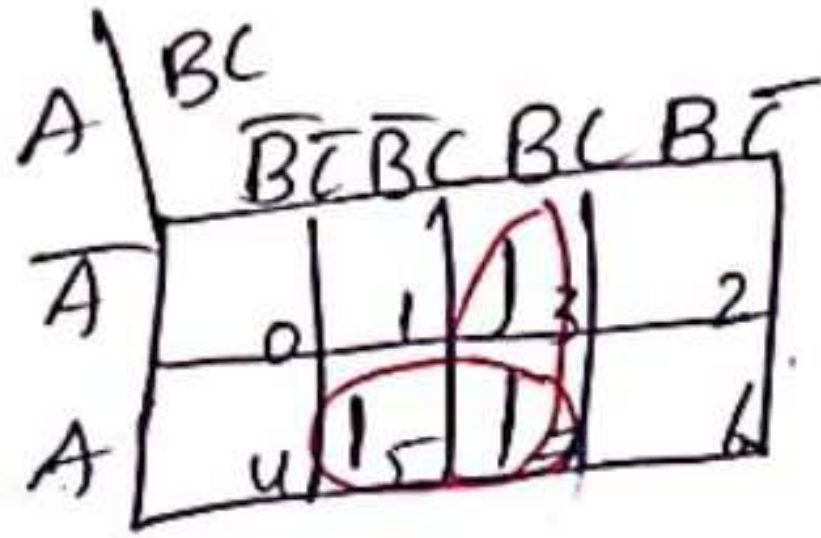
Sol:- PROM:-

* By using 3x8 decoder I/Ps or 3 o/p or 8



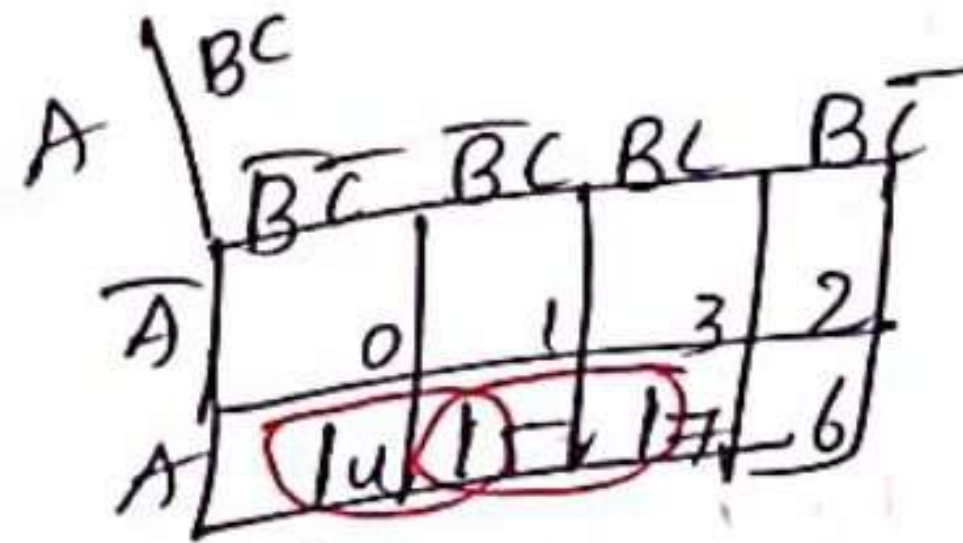
2) PLA:-

$$f_1 = \text{Em}(3, 5, 7)$$



$$f_1 = AC + BC$$

$$f_2 = \text{Em}(4, 5, 7)$$

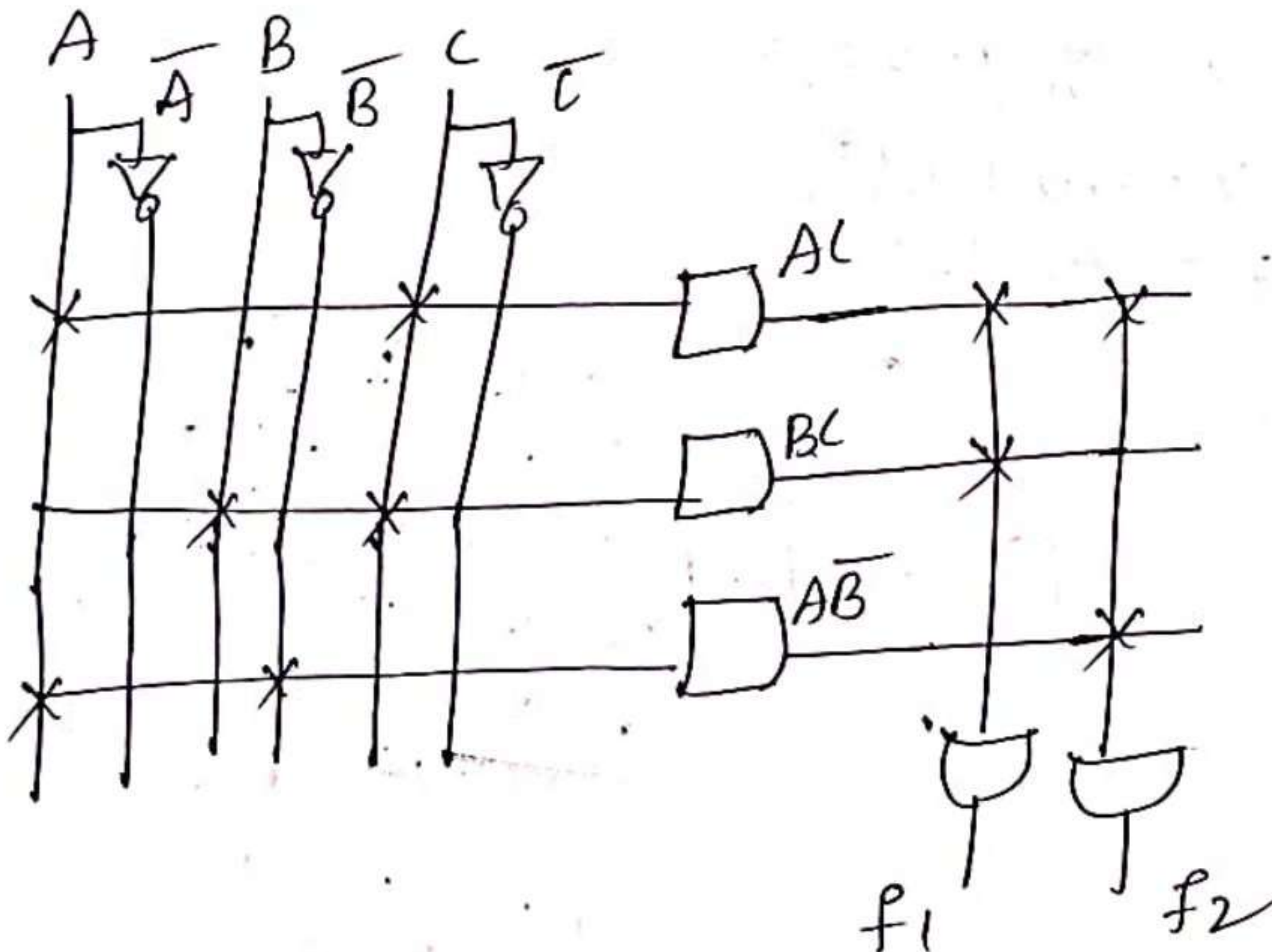


$$f_2 = A\bar{B} + AC$$

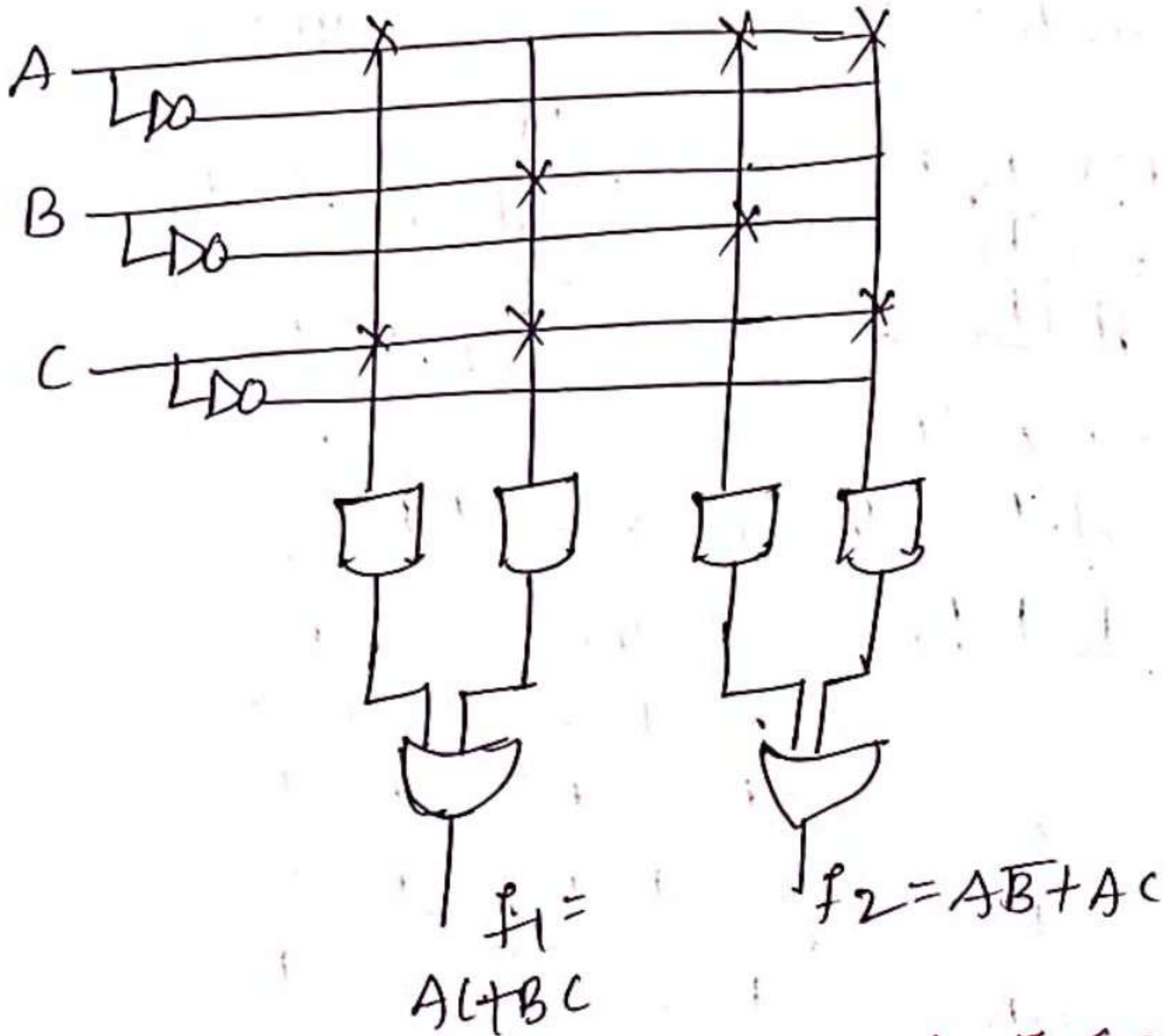
Step 1:- Write the program table

Product term	I/P's			O/P's	
	A	B	C	f_1	f_2
AC	1	-	1	1	1
BC	-	1	1	1	-
$A\bar{B}$	1	0	-	-	1

Step 2:- logic diagram:-



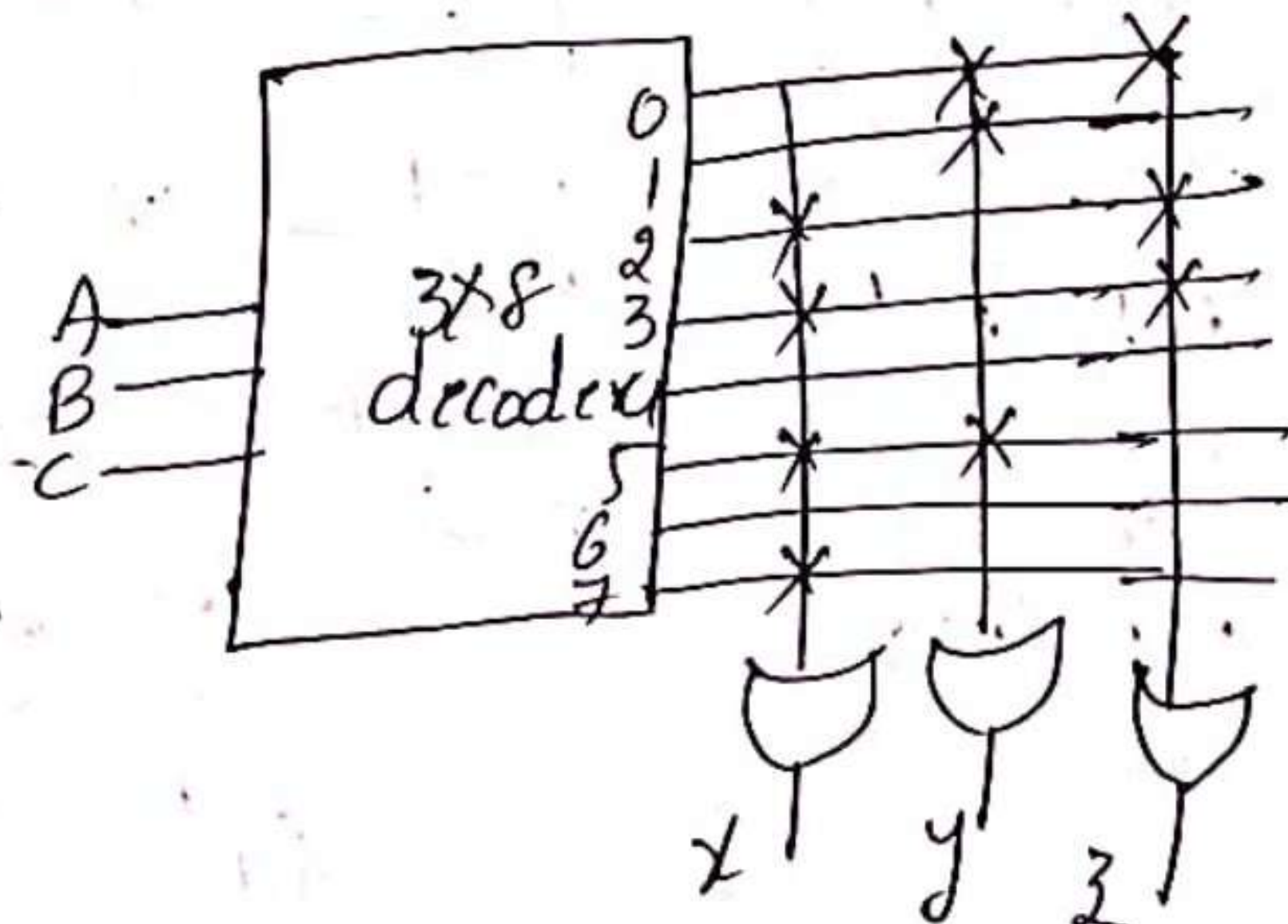
PAL:-



2) ~~P~~ $X = \text{Em}(2,3,5,7)$, $Y = \text{Em}(0,1,5)$, $Z = \text{Em}(0,2,3)$
by using PAL, PLA, PROM.

A) $X = \text{Em}(2,3,5,7)$
 $Y = \text{Em}(0,1,5)$
 $Z = \text{Em}(0,2,3)$

PROM:-



PLA :-

$$X = \Sigma m(2, 3, 5, 7)$$

A \ BC	$\overline{B}\overline{C}$	$\overline{B}C$	$B\overline{C}$	BC
$\overline{A}$	0	1	1	3
A	4	5	6	7

$$X = AC + \overline{A}B$$

$$Y = \Sigma m(0, 1, 5)$$

A \ BC	$\overline{B}\overline{C}$	$\overline{B}C$	$B\overline{C}$	BC
$\overline{A}$	1	0	1	3
A	4	5	6	7

$$Y = \overline{A}\overline{B} + \overline{B}C$$

$$Z = \Sigma m(0, 2, 3)$$

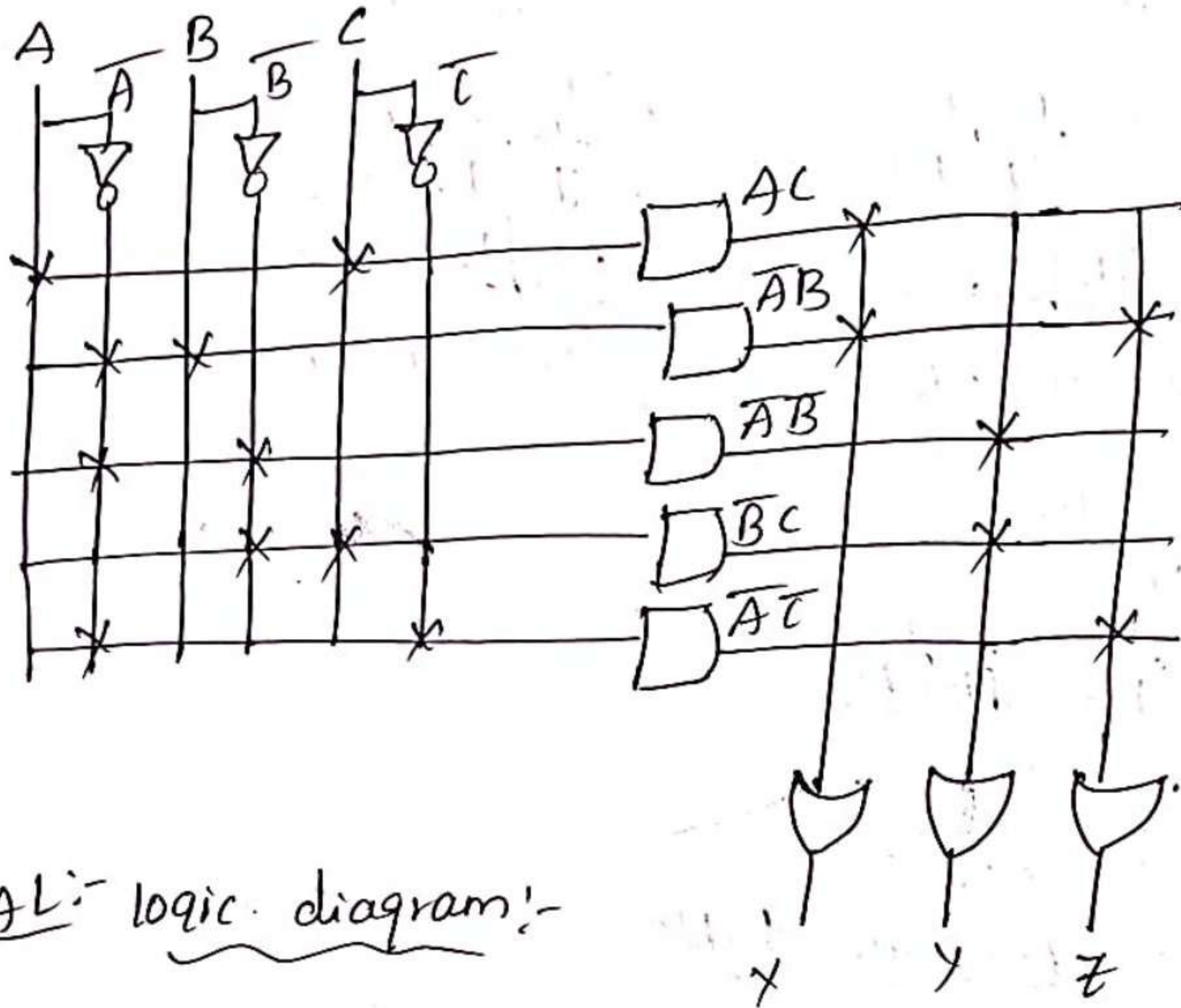
A \ BC	$\overline{B}\overline{C}$	$\overline{B}C$	$B\overline{C}$	BC
$\overline{A}$	1	0	1	3
A	4	5	6	7

$$Z = \overline{A}B + \overline{A}\overline{C}$$

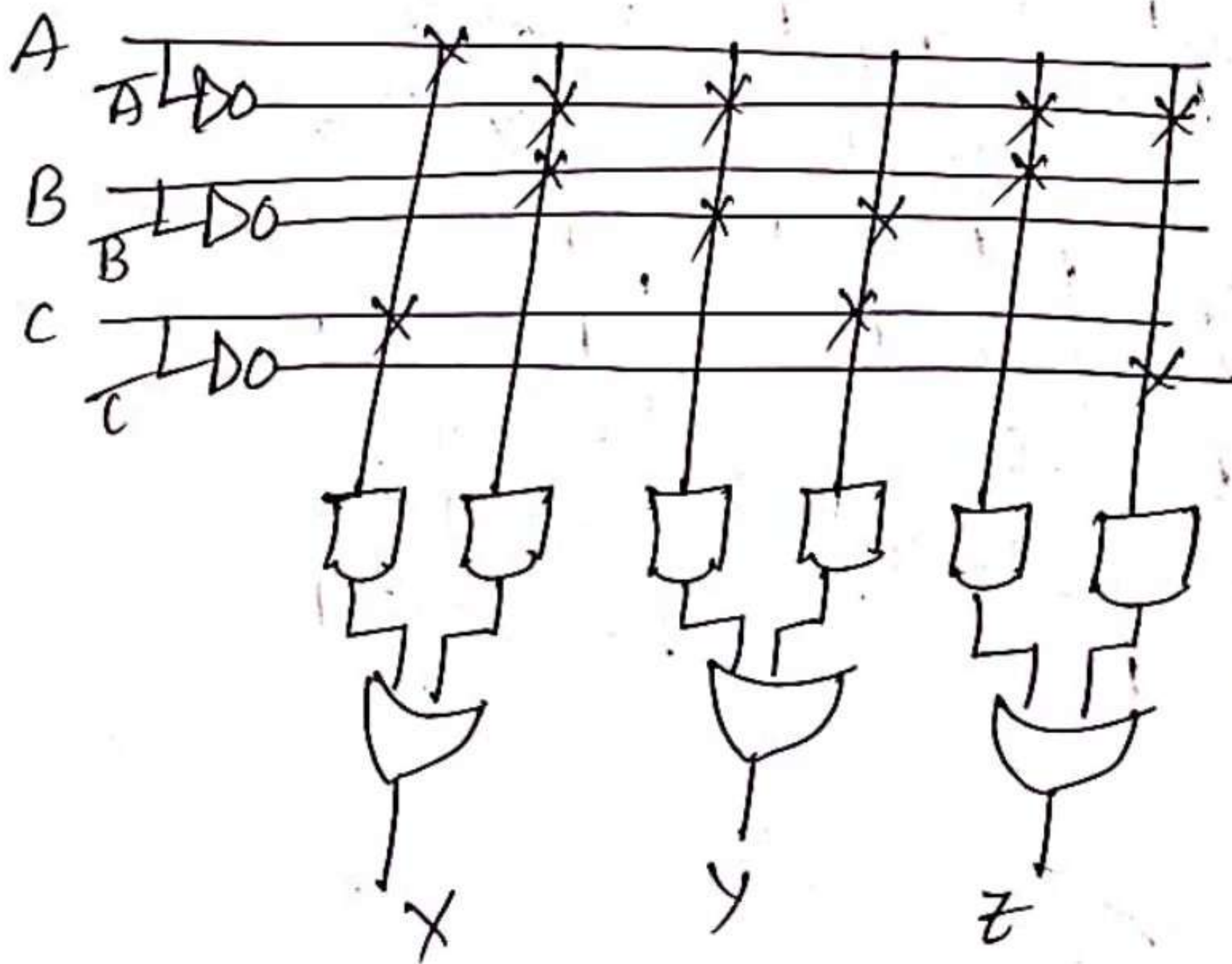
Step 1:- Program table:-

Product term	Inputs			Outputs		
	A	B	C	X	Y	Z
AC	1	-	1	1	-	-
$\overline{A}B$	0	1	-	1	-	1
$\overline{A}\overline{B}$	0	0	-	-	1	-
$\overline{B}C$	-	0	1	-	1	1
$\overline{A}\overline{C}$	0	-	0	-	-	1

step 3:- logic diagram



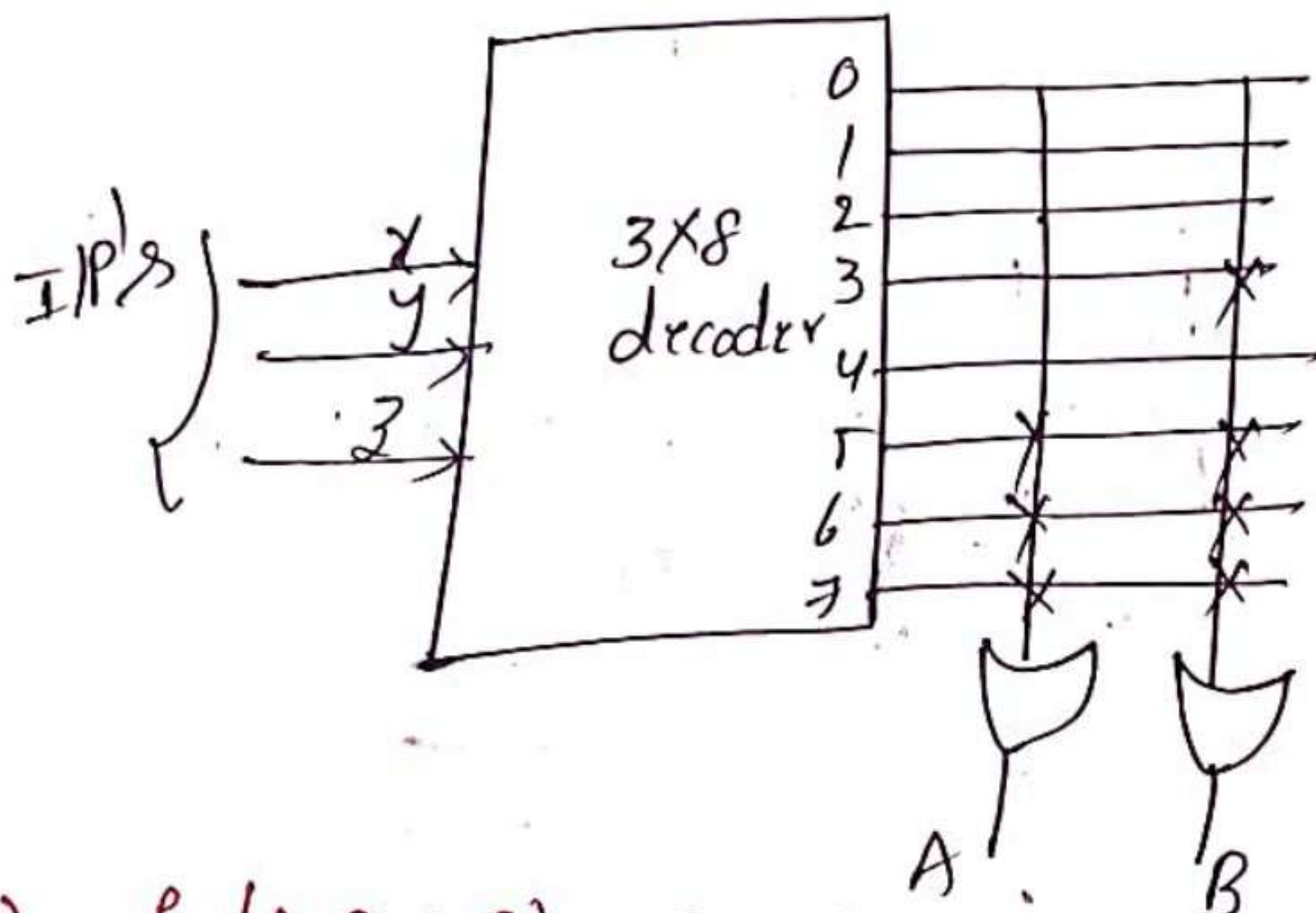
PAL:- logic diagram:-



*H) Implement by using PROM
 5M $A(x,y,z) = \sum m(5,6,7)$
 $B(x,y,z) = \sum m(3,5,6,7)$

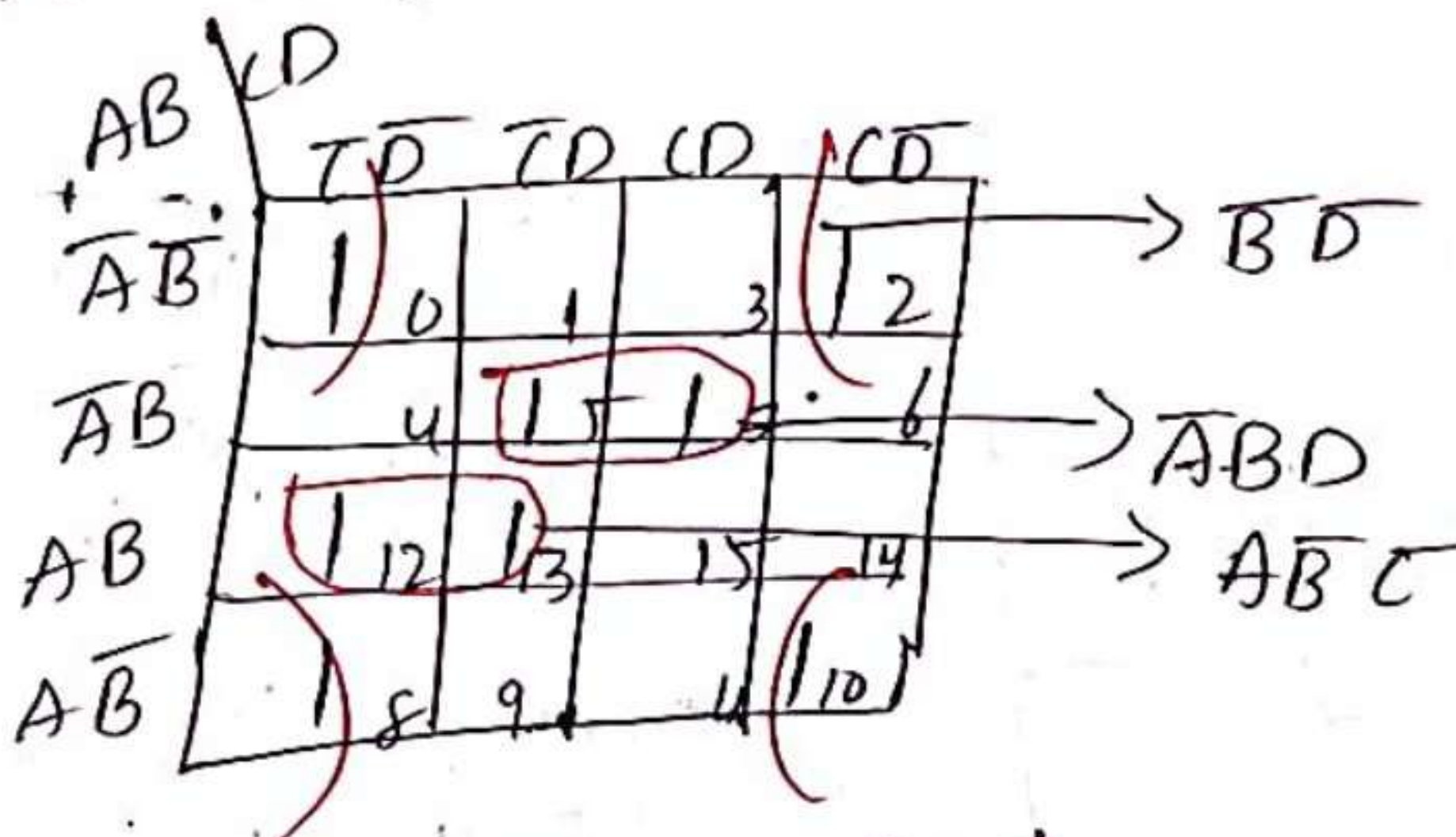
(10)

Sol:- PROM:-



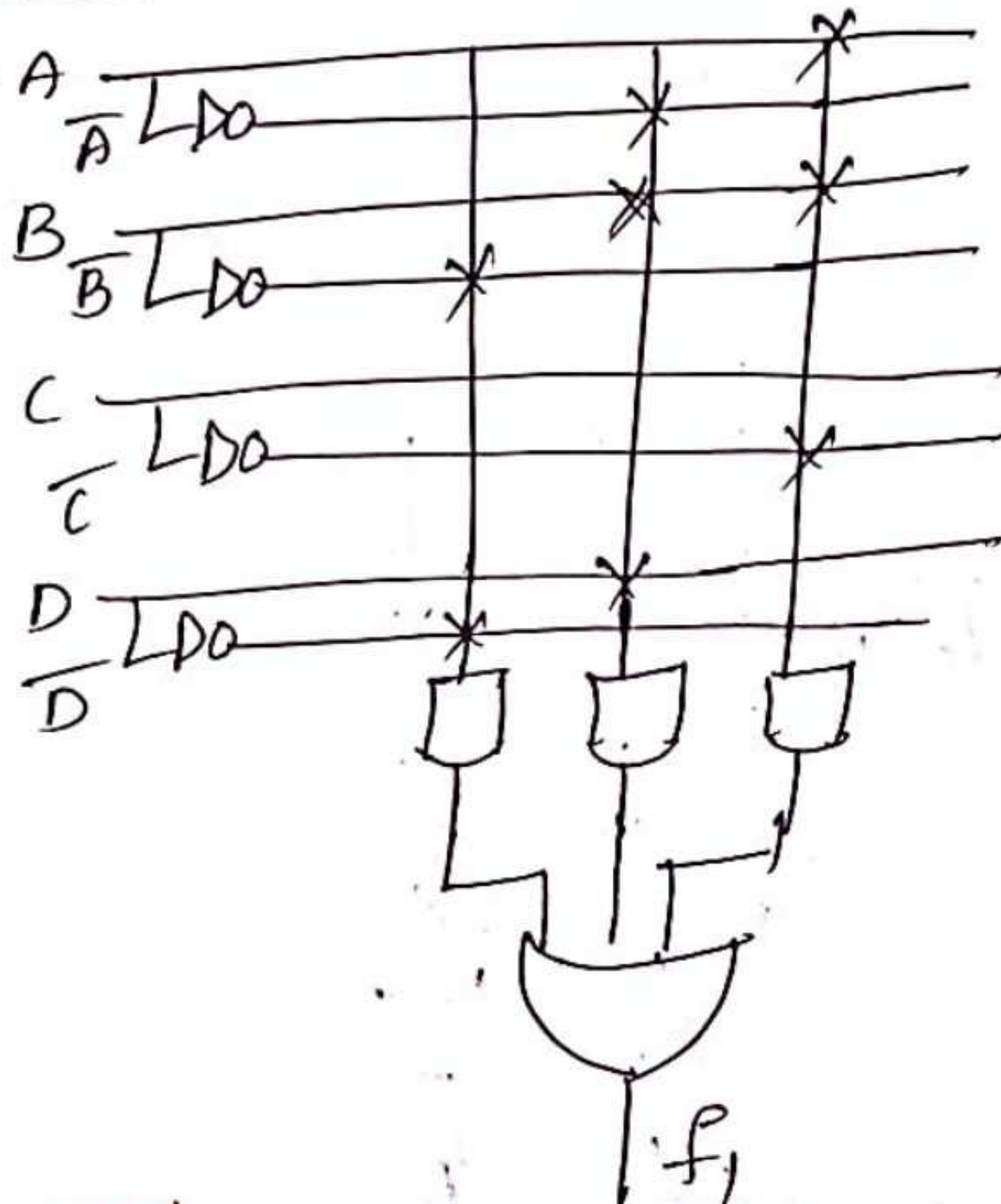
5) $f_1(A,B,C,D) = \sum m(0,2,5,7,8,10,12,13)$ by using PAL

Sol:- $f_1(A,B,C,D) = \sum m(0,2,5,7,8,10,12,13)$



$$f_1 = \overline{B}D + \overline{A}BD + ABC$$

Logic diagram:-

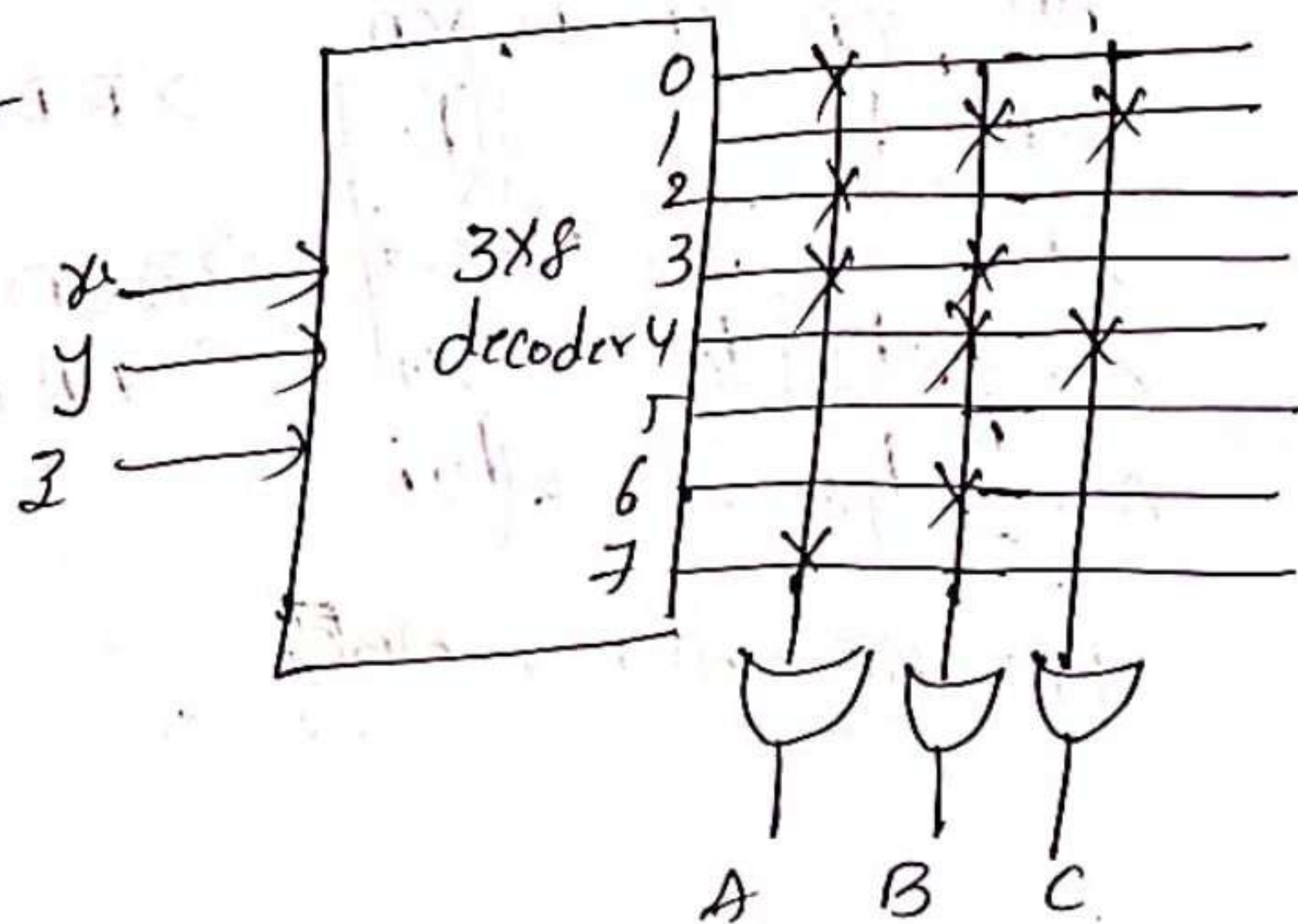


5) $A(x, y, z) = \sum m(0, 2, 3, 7)$

$B(x, y, z) = \sum m(1, 3, 4, 6)$

$C(x, y, z) = \sum m(1, 4)$ Implement by using PROM, PAL, PLA

Sol:- PROM:-



PLA:-

(11)

K-map for A

$x \backslash yz$	$\overline{y}z$	$y\overline{z}$	yz	$\overline{y}\overline{z}$
$\overline{x}$	1	0	1	3
x	4	7	5	6

$$A = \overline{x}\overline{z} + yz$$

K-map for B

$x \backslash yz$	$\overline{y}z$	$y\overline{z}$	yz	$\overline{y}\overline{z}$
$\overline{x}$	0	1	1	3
x	1	7	5	6

$$B = \overline{x}z + x\overline{z}$$

K-map for C

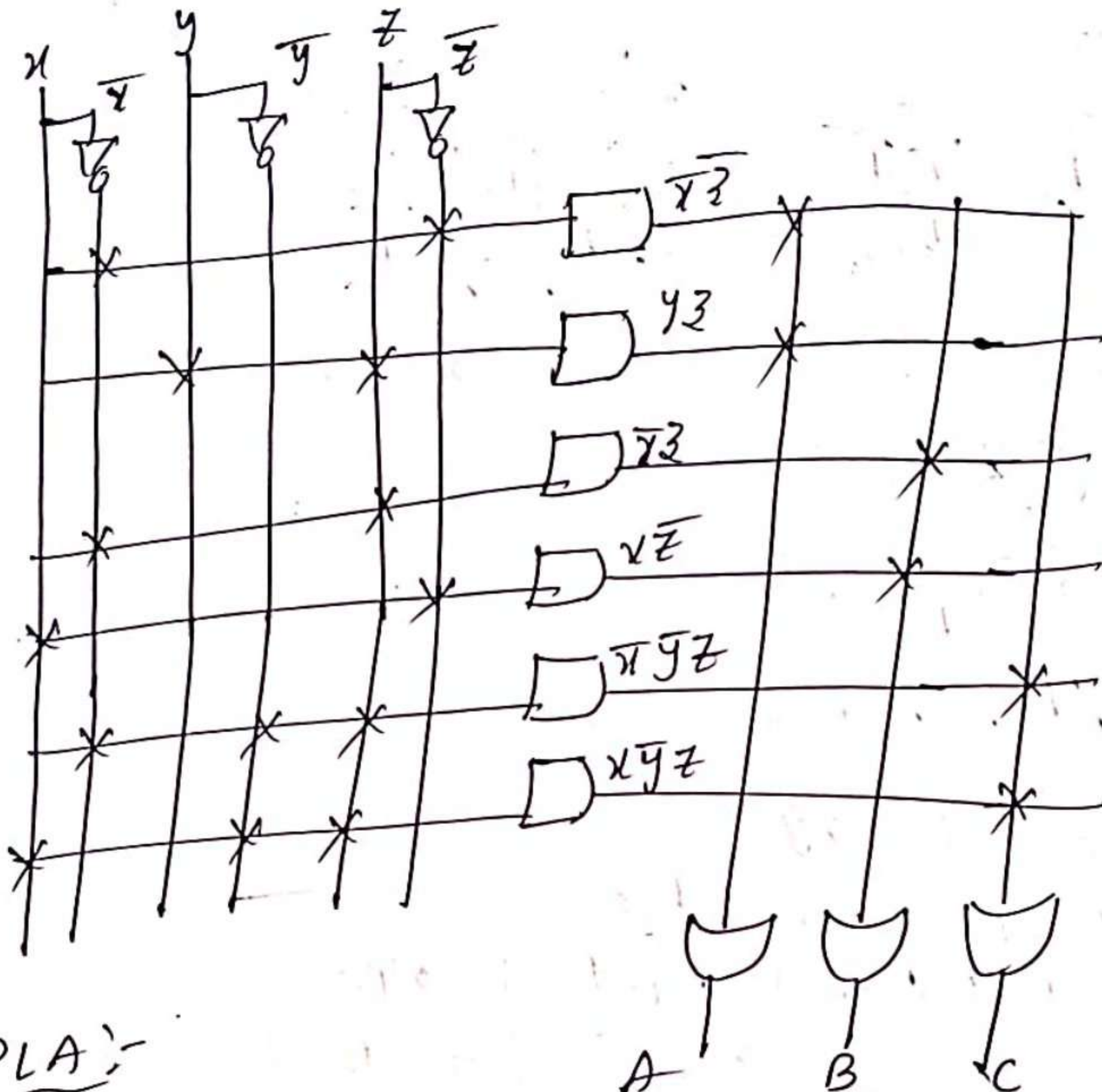
$x \backslash yz$	$\overline{y}z$	$y\overline{z}$	yz	$\overline{y}\overline{z}$
$\overline{x}$	0	1	3	2
x	1	7	5	6

$$C = \overline{x}y\overline{z} + x\overline{y}z$$

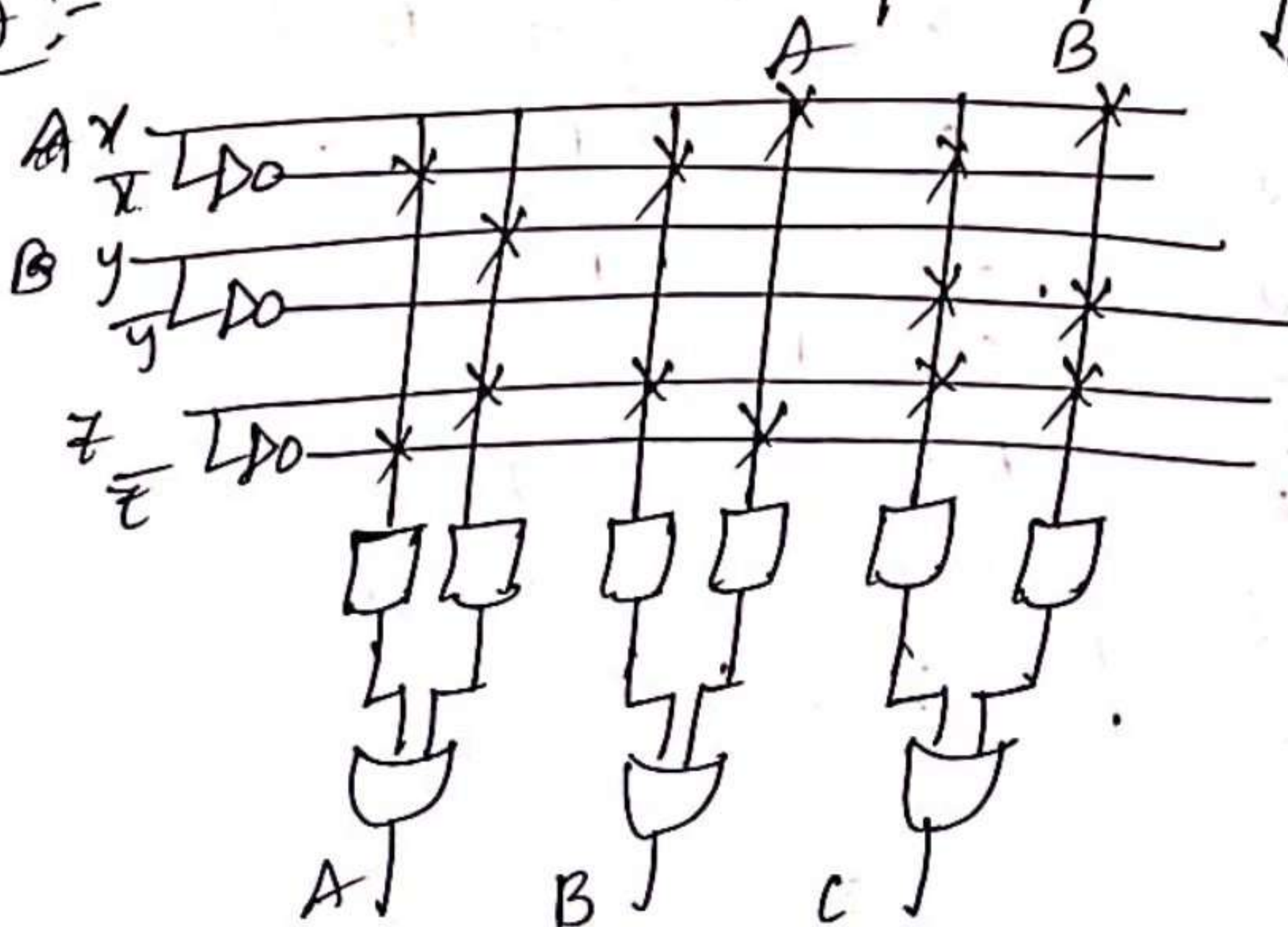
Step 1:- Program Table

Product-term	Inputs			Outputs		
	x	y	z	A	B	C
$\overline{x}\overline{z}$	0	-	0	1	-	-
$y\overline{z}$	-	1	1	1	-	-
$\overline{x}z$	0	-	1	-	1	-
$x\overline{z}$	1	-	0	-	1	-
$\overline{x}y\overline{z}$	0	0	1	-	-	1
$x\overline{y}z$	1	0	0	-	-	1

Logic diagram:-



PLA:-



b) $f_1 = \sum m(3, 5, 6, 7)$, $f_2 = \sum m(0, 1, 6, 7)$ by (12)
using PROM, PAL or PLA

7) $w(A, B, C, D) = \sum m(2, 12, 13)$

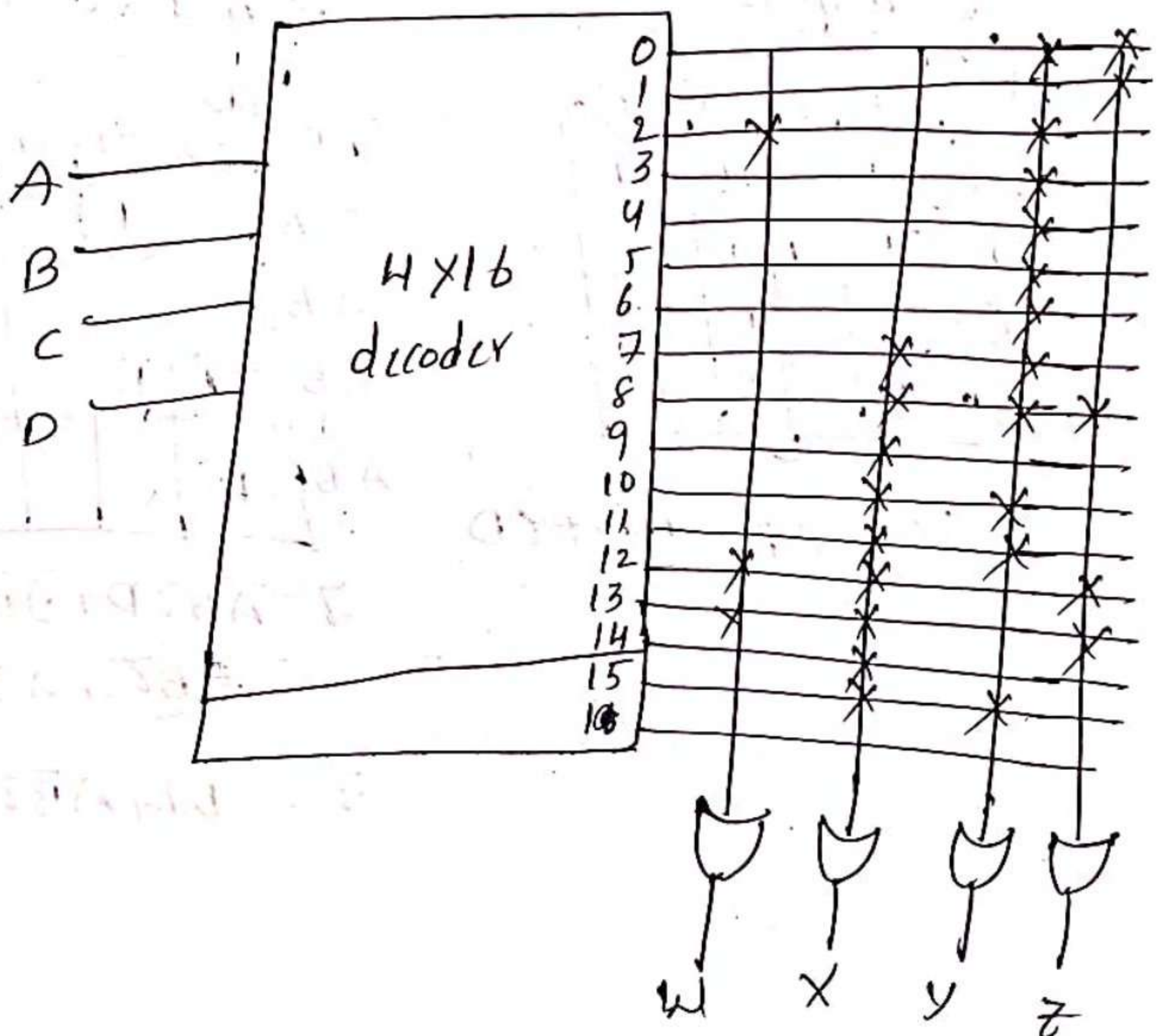
$x(A, B, C, D) = \sum m(7, 8, 9, 10, 11, 12, 13, 14, 15)$

$y(A, B, C, D) = \sum m(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)$

$z(A, B, C, D) = \sum m(1, 2, 8, 12, 13)$ Implement

sol:- by using PROM, ~~PAL~~, PAL

sol:- PROM



PLA - PAL:-

K-map for W

AB \ CD	$\overline{C}\overline{D}$	$\overline{C}D$	$C\overline{D}$	CD
$\overline{A}\overline{B}$				1
$\overline{A}B$				
$A\overline{B}$	1	1		
AB				

$$W = A\overline{B}\overline{C} + \overline{A}\overline{B}CD$$

K-map for X

AB \ CD	$\overline{C}\overline{D}$	$\overline{C}D$	$C\overline{D}$	CD
$\overline{A}\overline{B}$				
$\overline{A}B$			1	
$A\overline{B}$	1	1	1	1
AB	1	1	1	1

$$X = A + BCD$$

K-map for Y

AB \ CD	$\overline{C}\overline{D}$	$\overline{C}D$	$C\overline{D}$	CD
$\overline{A}\overline{B}$		1		1
$\overline{A}B$	1	1	1	1
$A\overline{B}$			1	
AB	1		1	1

$$Y = \overline{A}B + \overline{B}\overline{D} + CD$$

K-map for Z

AB \ CD	$\overline{C}\overline{D}$	$\overline{C}D$	$C\overline{D}$	CD
$\overline{A}\overline{B}$		1		1
$\overline{A}B$				
$A\overline{B}$	1	1		
AB	1			

$$Z = \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}CD + \underline{A\overline{B}\overline{C}} + A\overline{C}\overline{D}$$

$$Z = W + A\overline{B}\overline{C}\overline{D} + A\overline{C}\overline{D}$$

Logic diagram

(13)

