Chapter = 6

MSI And LSI components in Combinational Logic designs.

Encoder & Decoder

Decoder = A decoder is a combinational ckt that converts binary information from 'n' input lines to a maxm of 2^n o/p lines. There are 'n' input lines and m o/p line where $m \leq 2^n$.

→ $m < 2^n$ whenever there is dont care condition.

→ Size of a decoder = no. of i/p lines X no of o/p lines.
i.e Size = $n \times 2^n$

2:4 decoder

Fig: Block diagram of Decoder.

Table of Decoder

	A	B	Y_3	Y_2	Y_1	Y_0
0)	0	0	0	0	0	1
1)	0	1	0	0	1	0
2)	1	0	0	1	0	0
3)	1	1	1	0	0	0

2 to 4 line decoder

In this decoder '2' input are decoded into '4' output. Each output representing one of the minterms of 2-input variable.

From the table,

$Y_0 = \bar{A}\bar{B}$ $Y_1 = \bar{A}B$ $Y_2 = A\bar{B}$

$Y_3 = AB$.

Realization;

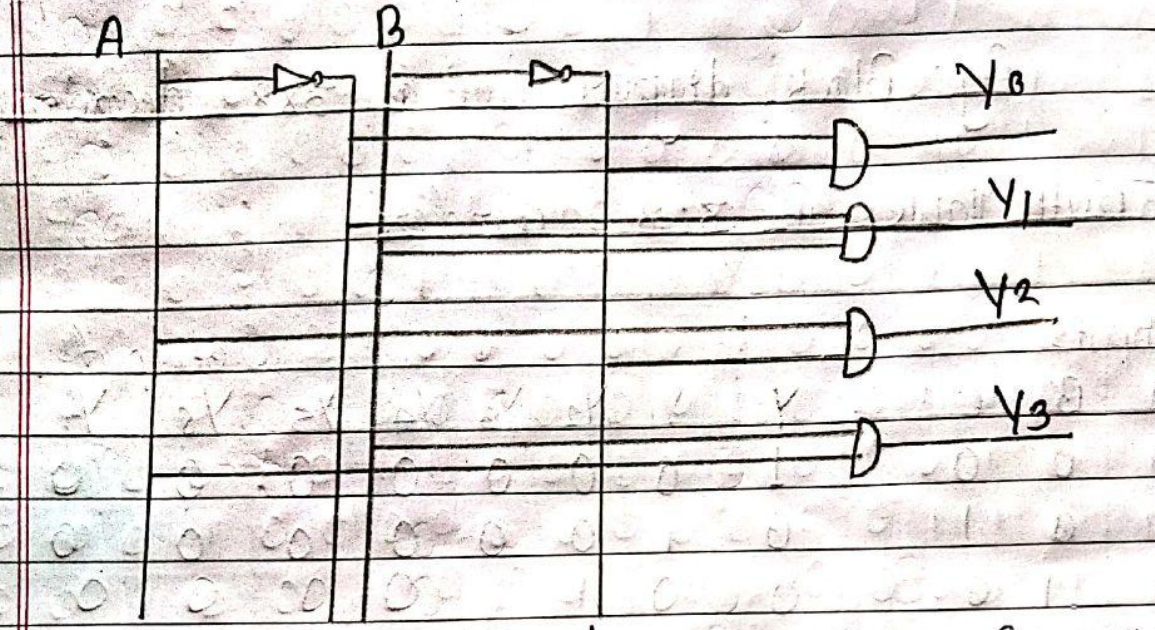


Fig: 2:4 line decorder (2×4)

Application of Decorder

It is used in many types of application. For example, in computer, it is used to select input/output ports. The binary ports address, is decoded and the appropreate peripherals is selected for communication. Each I/O port has unique address.

3X8 Decorder (Binary to octal converter)

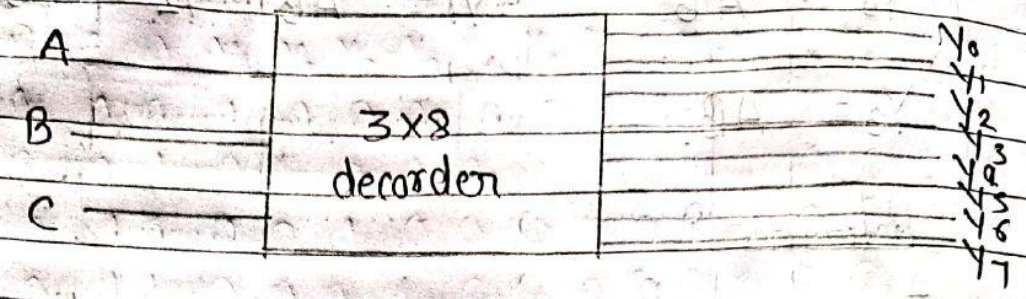


Fig: Block diagram of a 3X8 decorder

Truth table of 3:8 decorder

Inputs			outputs							
A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1		0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1
			0	0	0	0	0	0	0	0

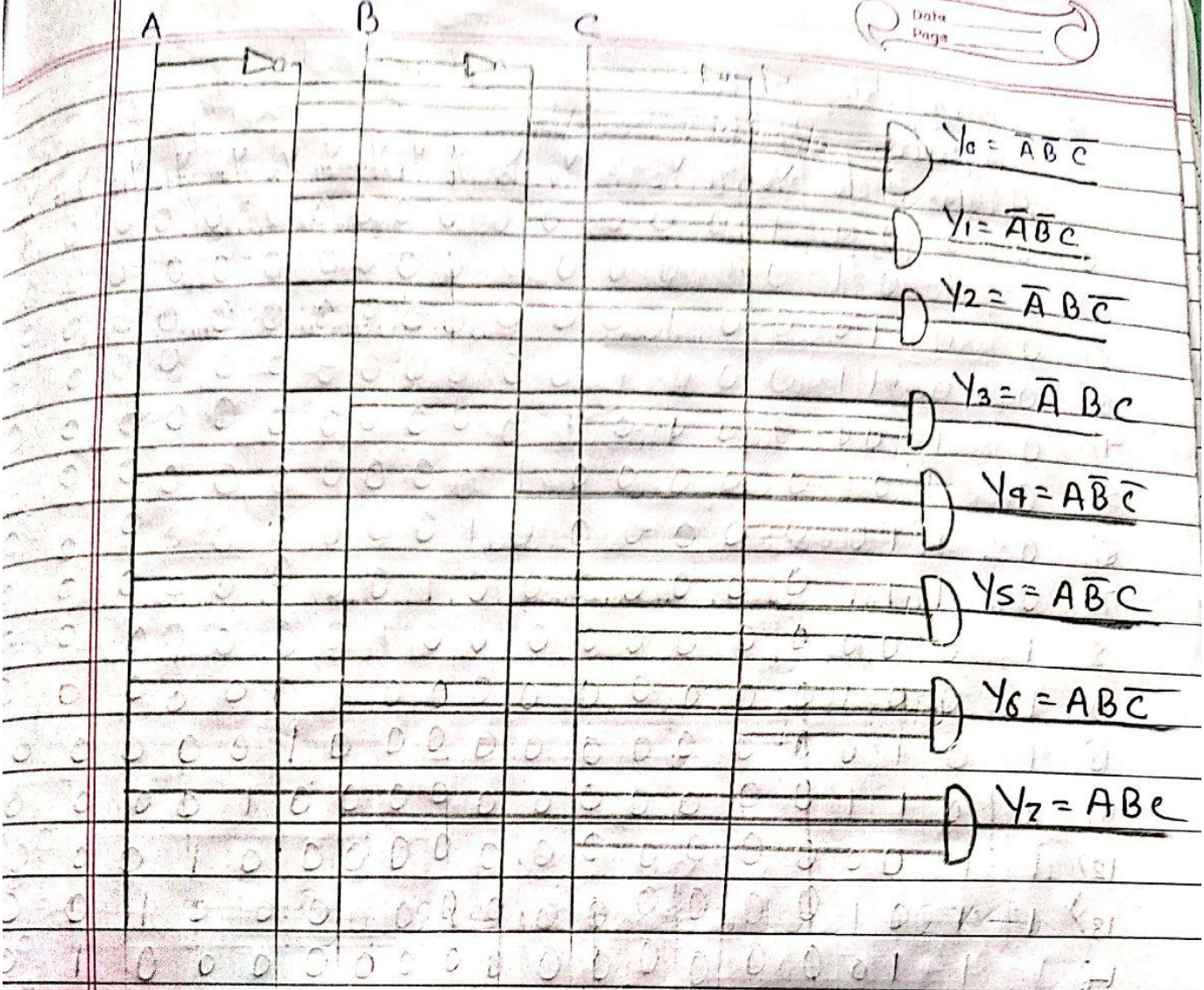
From the table

$Y_0 = \bar{A}\bar{B}\bar{C}$, $Y_1 = \bar{A}\bar{B}C$ $Y_2 = \bar{A}B\bar{C}$

$Y_3 = \bar{A}BC$ $Y_4 = A\bar{B}\bar{C}$ $Y_5 = A\bar{B}C$ $Y_6 = AB\bar{C}$

$Y_7 = ABC$

3×8 decoder using 2×4 decoder



4:16 Line Decoder (Binary to Hexadecimal converter)

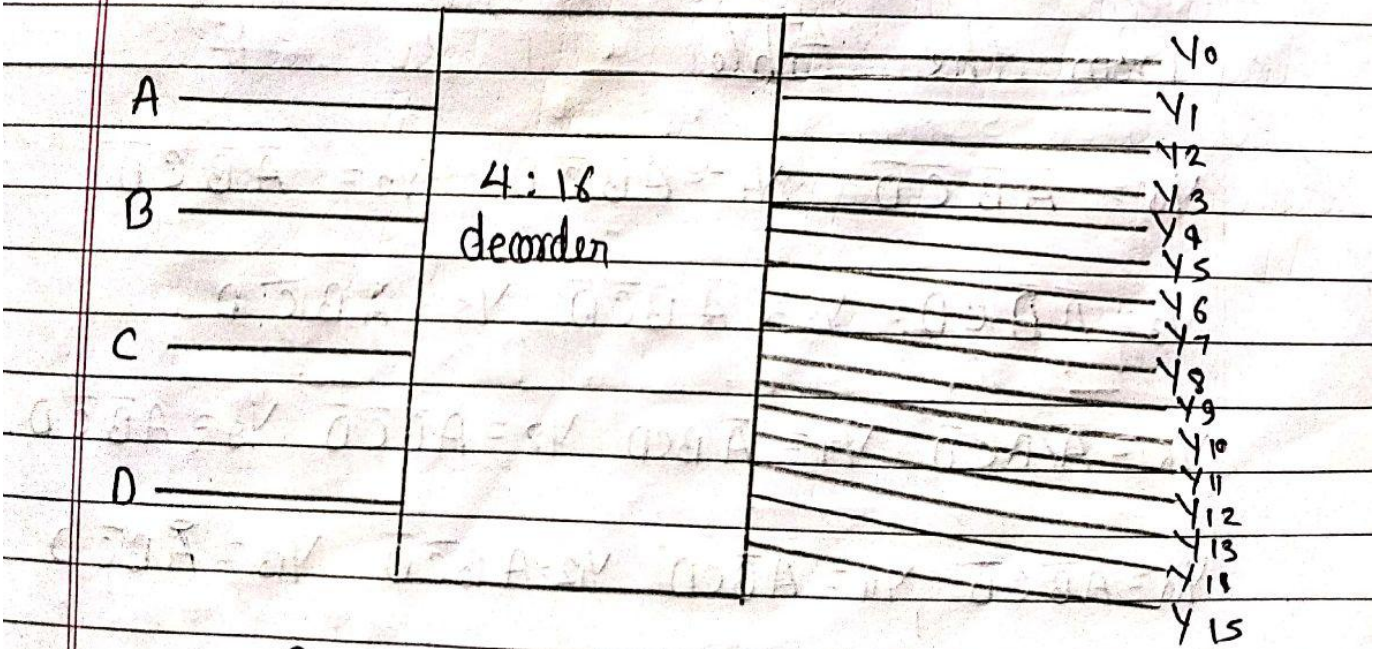


Fig: Block diagram of 4:16 decoder.

Truth table

	Input				Outputs															
	A	B	C	D	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}
0)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1)	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2)	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3)	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
4)	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
5)	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
6)	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7)	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
8)	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
9)	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
10)	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
11)	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
12)	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
13)	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
14)	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15)	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

From the table

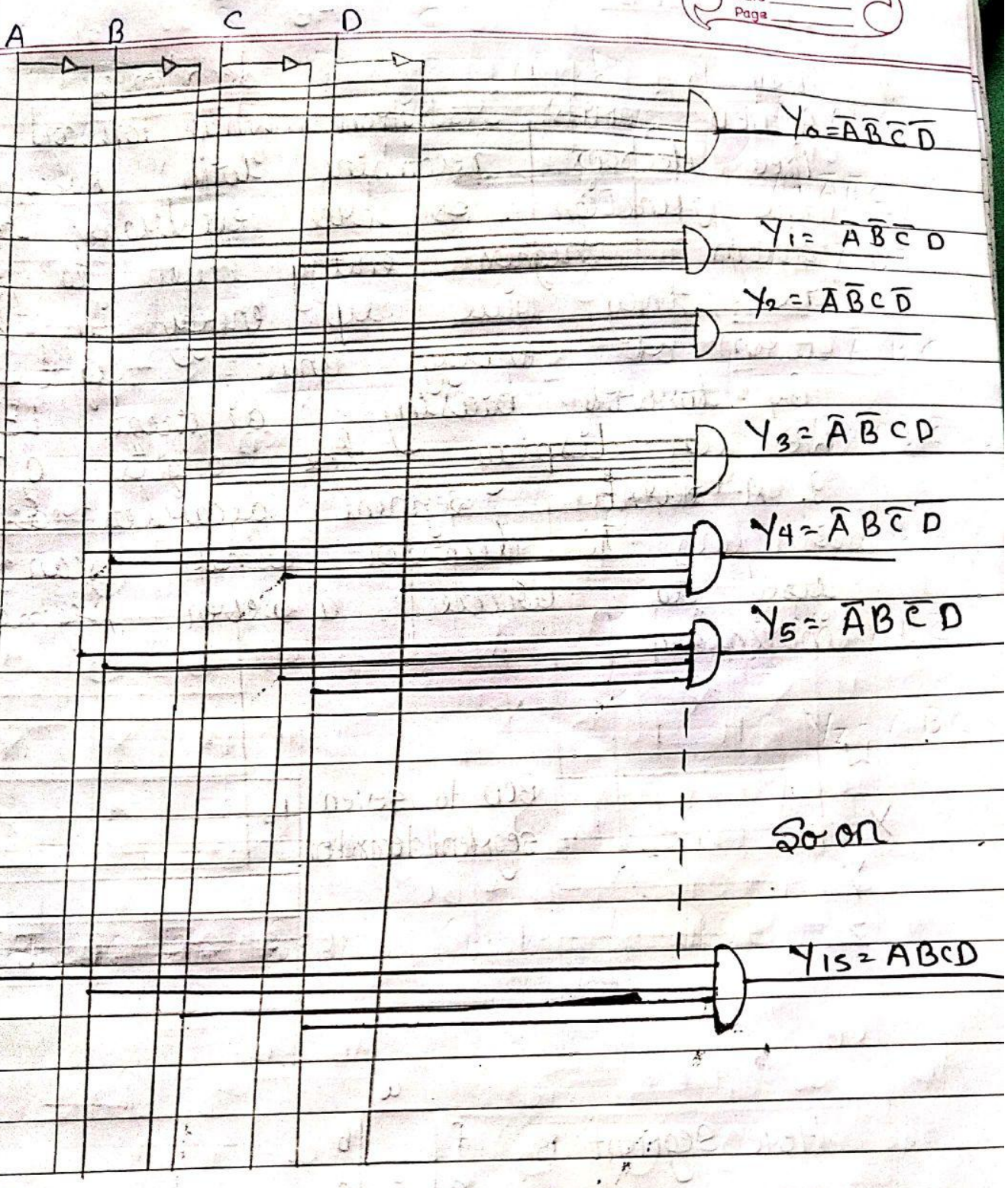
$Y_0 = \bar{A}\bar{B}\bar{C}\bar{D}$ $Y_1 = \bar{A}\bar{B}\bar{C}D$ $Y_2 = \bar{A}\bar{B}C\bar{D}$

$Y_3 = \bar{A}\bar{B}CD$ $Y_4 = \bar{A}B\bar{C}\bar{D}$ $Y_5 = \bar{A}B\bar{C}D$

$Y_6 = \bar{A}BC\bar{D}$ $Y_7 = \bar{A}BCD$ $Y_8 = A\bar{B}\bar{C}\bar{D}$ $Y_9 = A\bar{B}\bar{C}D$

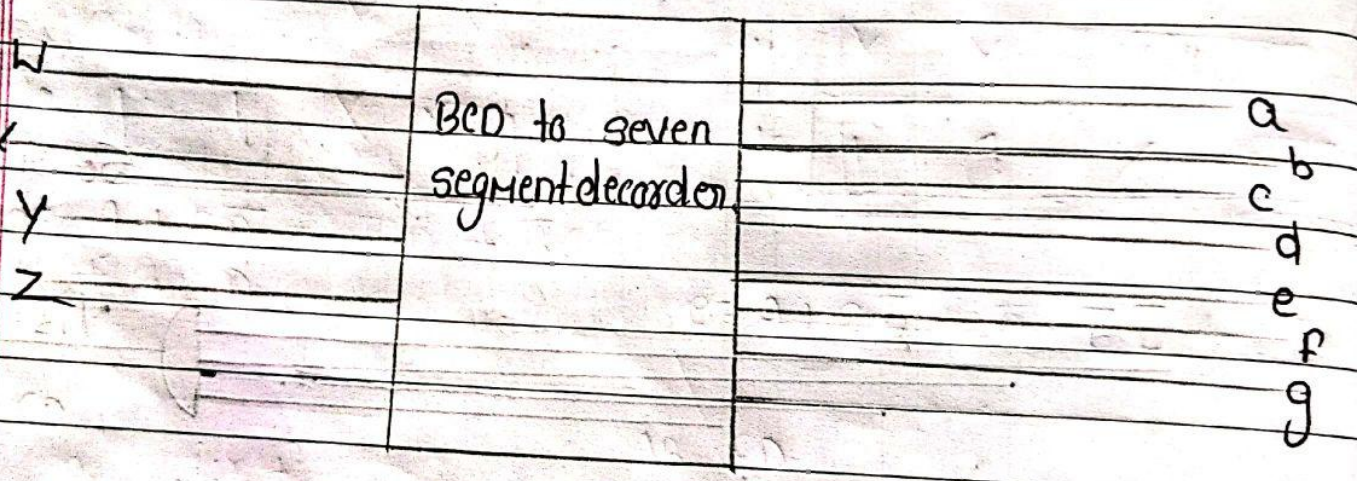
$Y_{10} = A\bar{B}C\bar{D}$ $Y_{11} = A\bar{B}CD$ $Y_{12} = AB\bar{C}\bar{D}$ $Y_{13} = \bar{A}B\bar{C}D$

$Y_{14} = ABC\bar{D}$ $Y_{15} = ABCD$

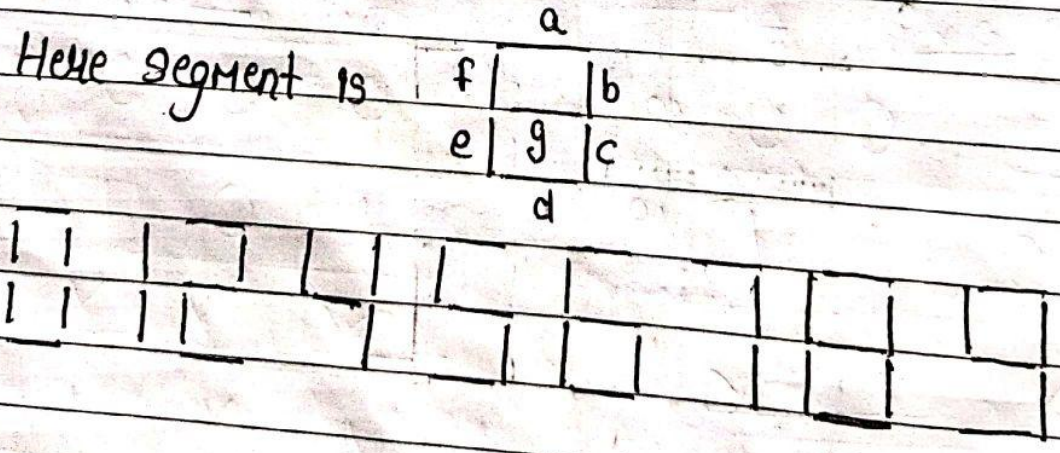


BCD to Seven Segment decoder;

A LED emits radiation when forward biased. Free electrons recombine with holes near the junction. As free electrons fall from higher energy level to lower one, they give up energy in the form of energy and light. By forward biasing different LEDs, we can display the digits 0 through 9. A seven segment decoder driver is an IC decoder that can be used to drive a seven segment indicator.



Here segment is



	A	B	C	D	a	b	c	d	e	f	g
	W	X	Y	Z	Output						
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	1	0	1	0							

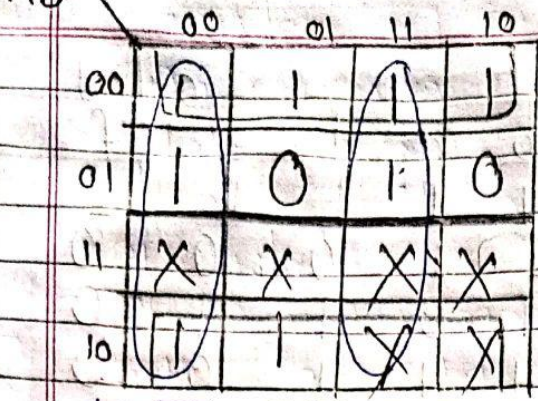
For a,

WX \ YZ	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$$a = W + Y + XZ + \overline{X}\,\overline{Z}$$

$$= W + Y + \overline{X \oplus Z}$$

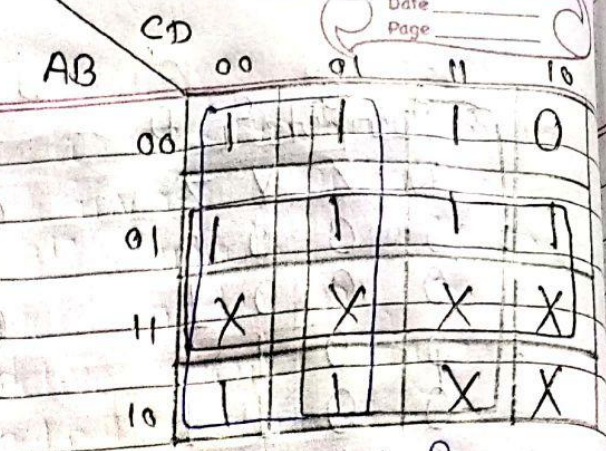
for b



$b = \overline{B} + \overline{C}\,\overline{D} + CD$

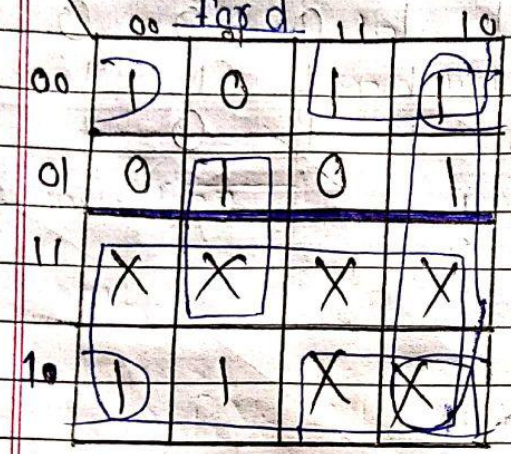
$= \overline{B} + \overline{C \oplus D}$

for c



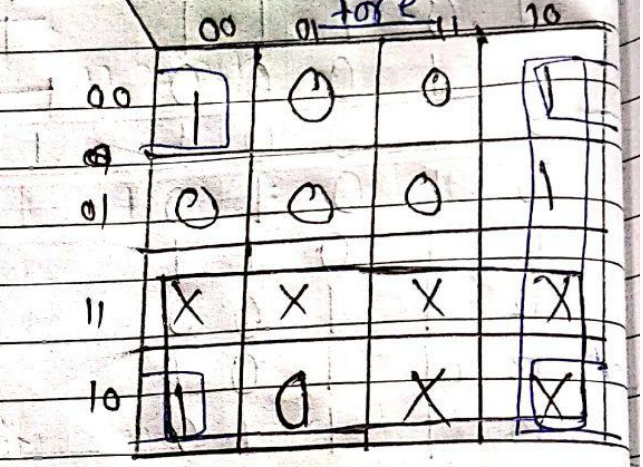
$c = \overline{C} + D + B$

for d



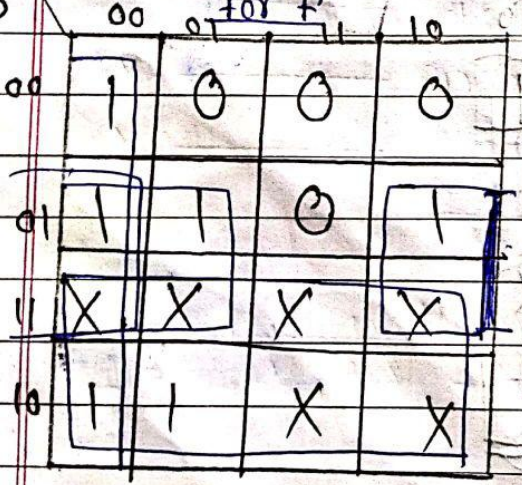
$d = A + C\overline{D} + \overline{B}\,\overline{D} + \overline{B}C + B\overline{C}D$

for e



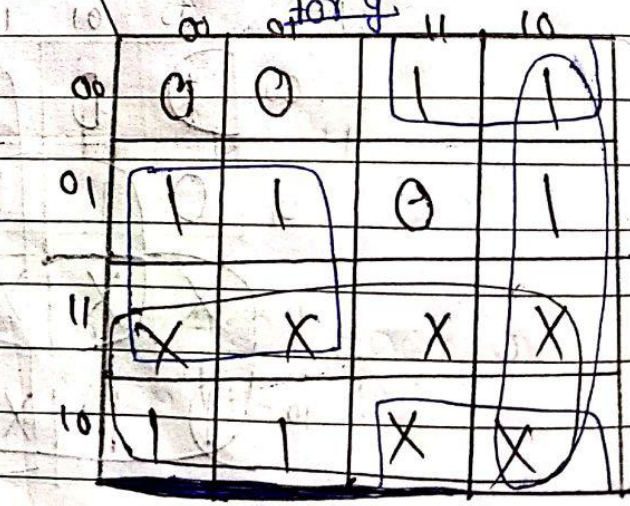
$e = \overline{B}\,\overline{D} + C\overline{D}$

for f



$f = A + \overline{C}\,\overline{D} + B\overline{C} + B\overline{D}$

for g



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

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

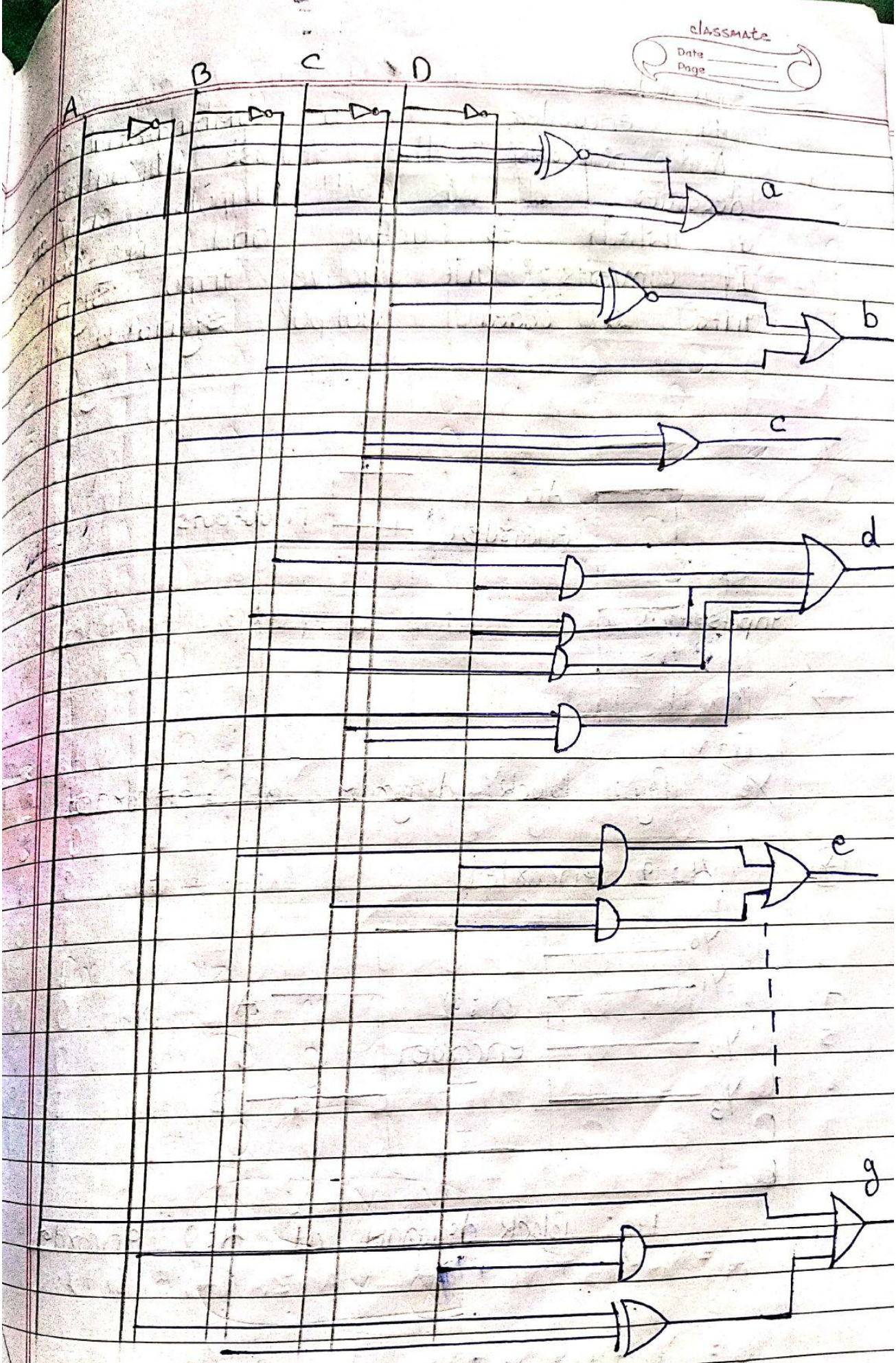


fig: circuit diagram of a seven segment decorder

Encorder

An encorder is a combinational circuit that performs the inverse operation of decorder. It has 2^n inputs only one of which is active and 'n' outputs. It converts an active input signal into a coded output signal.

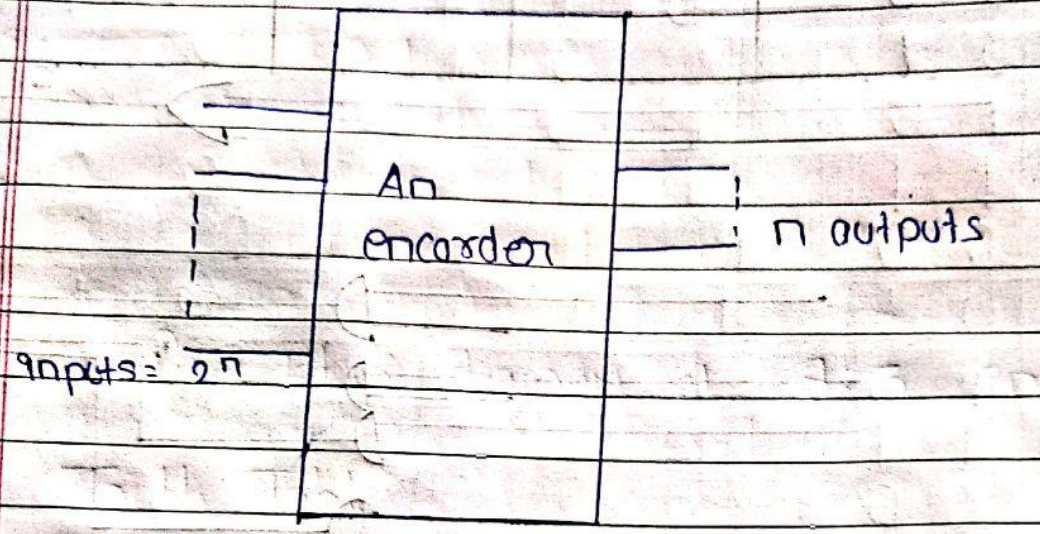


fig: Block diagram of encorder

4:2 encorder

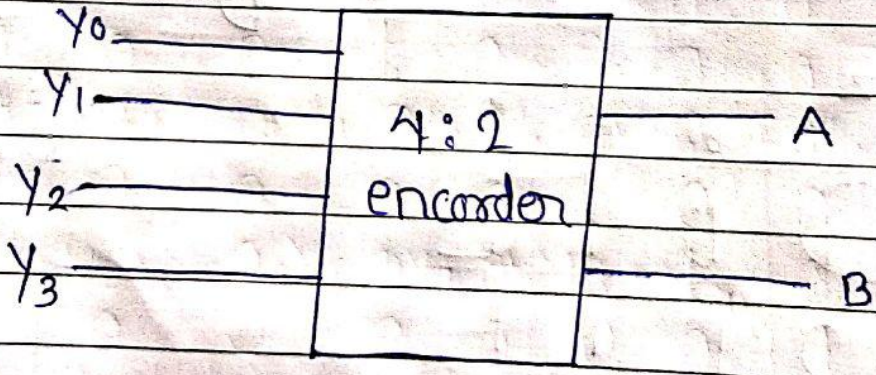


fig: Block diagram of 4:2 encorder

Truth table:

Inputs				outputs	
Y_0	Y_1	Y_2	Y_3	A	B
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

$A = Y_2 + Y_3$

$B = Y_1 + Y_3$

→ Realization of logic gate

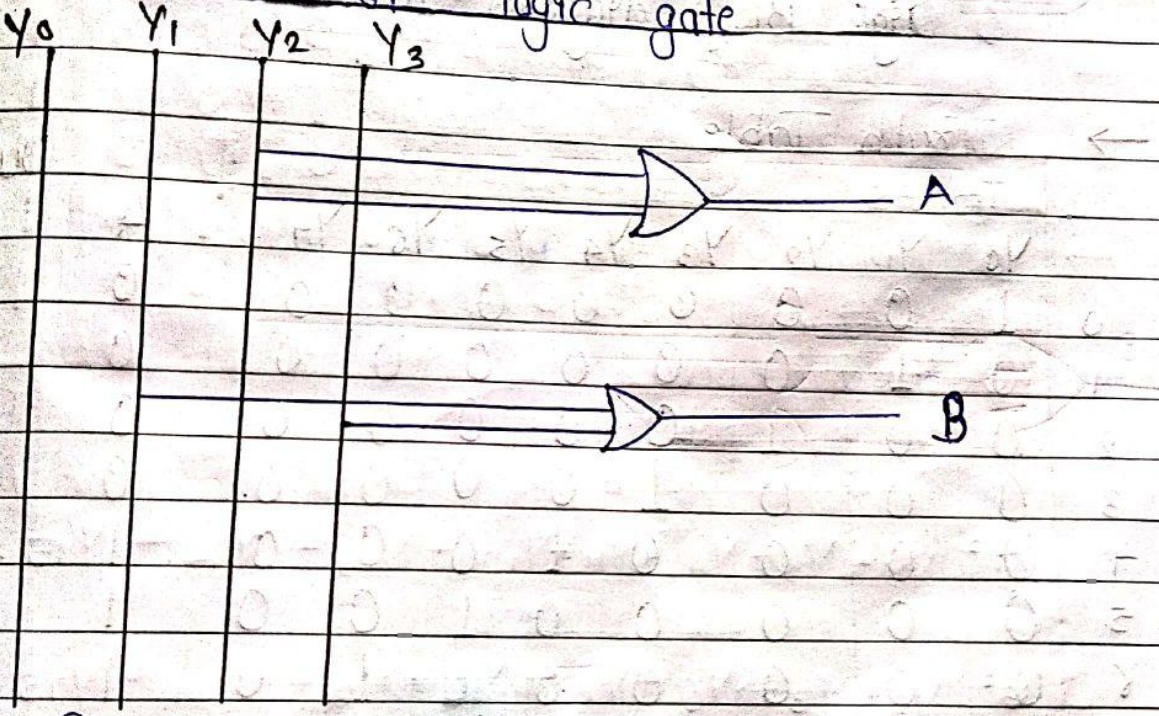


fig: 4x2 encorder

Limitation

Only one input can be enabled at a time. If two inputs are enabled at the same time, then output is undefined.

8:3 Encoder (Octal to Binary Conversion)

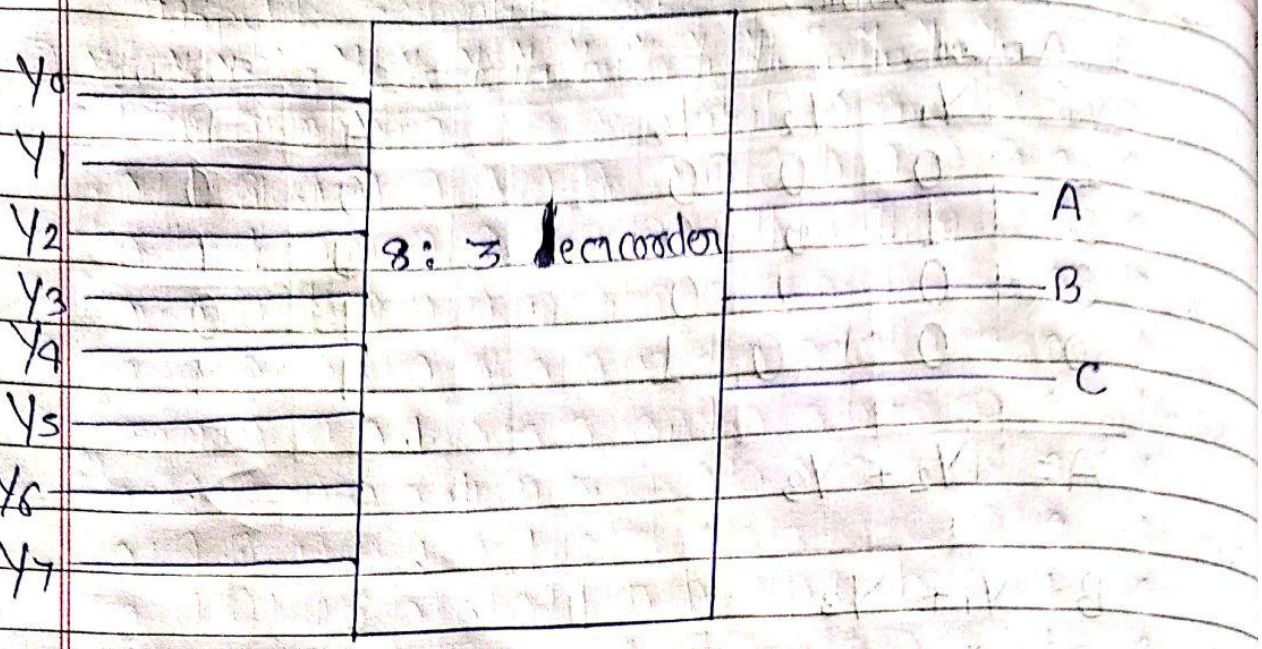


Fig: Block diagram of 8:3 encoder;-

→ Truth Table

	Inputs								Outputs		
	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	A	B	C
0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	1	0	1
6	0	0	0	0	0	0	1	0	1	1	0
7	0	0	0	0	0	0	0	1	1	1	1

$A = Y_4 + Y_5 + Y_6 + Y_7$	$A = \Sigma_M(4, 5, 6, 7)$
$B = Y_2 + Y_3 + Y_6 + Y_7$	$B = \Sigma_M(2, 3, 6, 7)$
$C = Y_1 + Y_3 + Y_5 + Y_7$	$C = \Sigma_M(1, 3, 5, 7)$

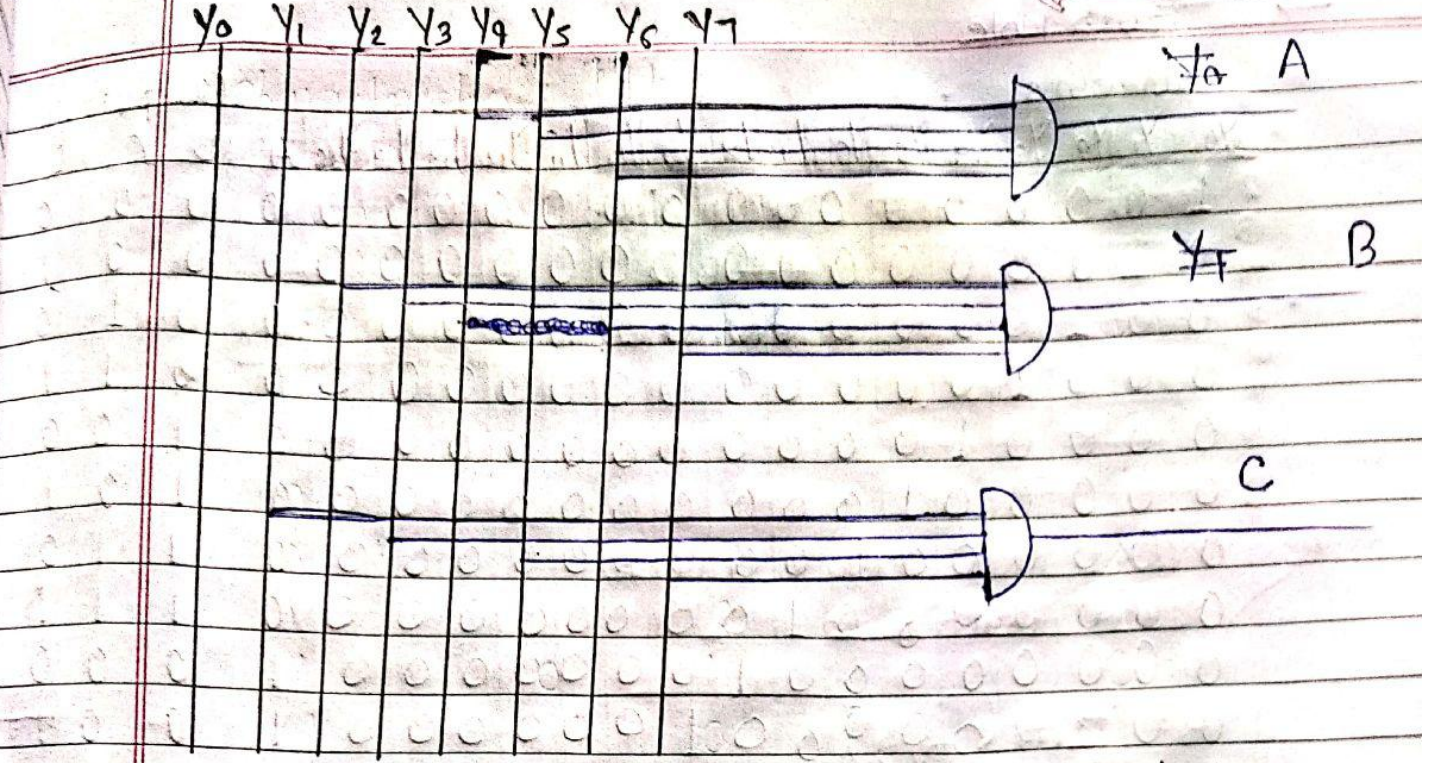


Fig: Circuit diagram of 8x3 decorder.

16:4 encorder

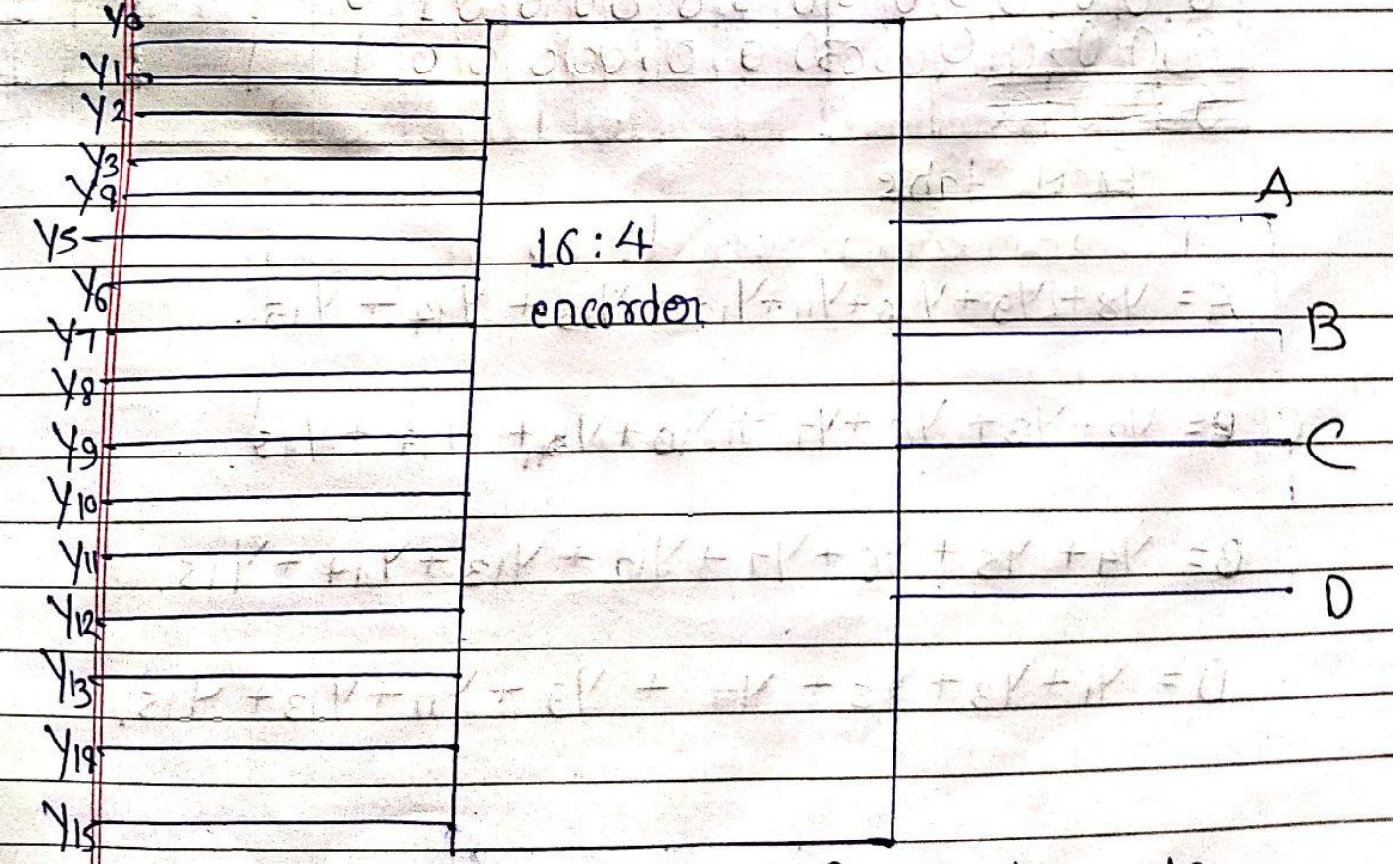


Fig: Block diagram of 16:4 encorder

Truth table

Inputs																Outputs			
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}	A	B	C	D
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	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	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	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	0	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

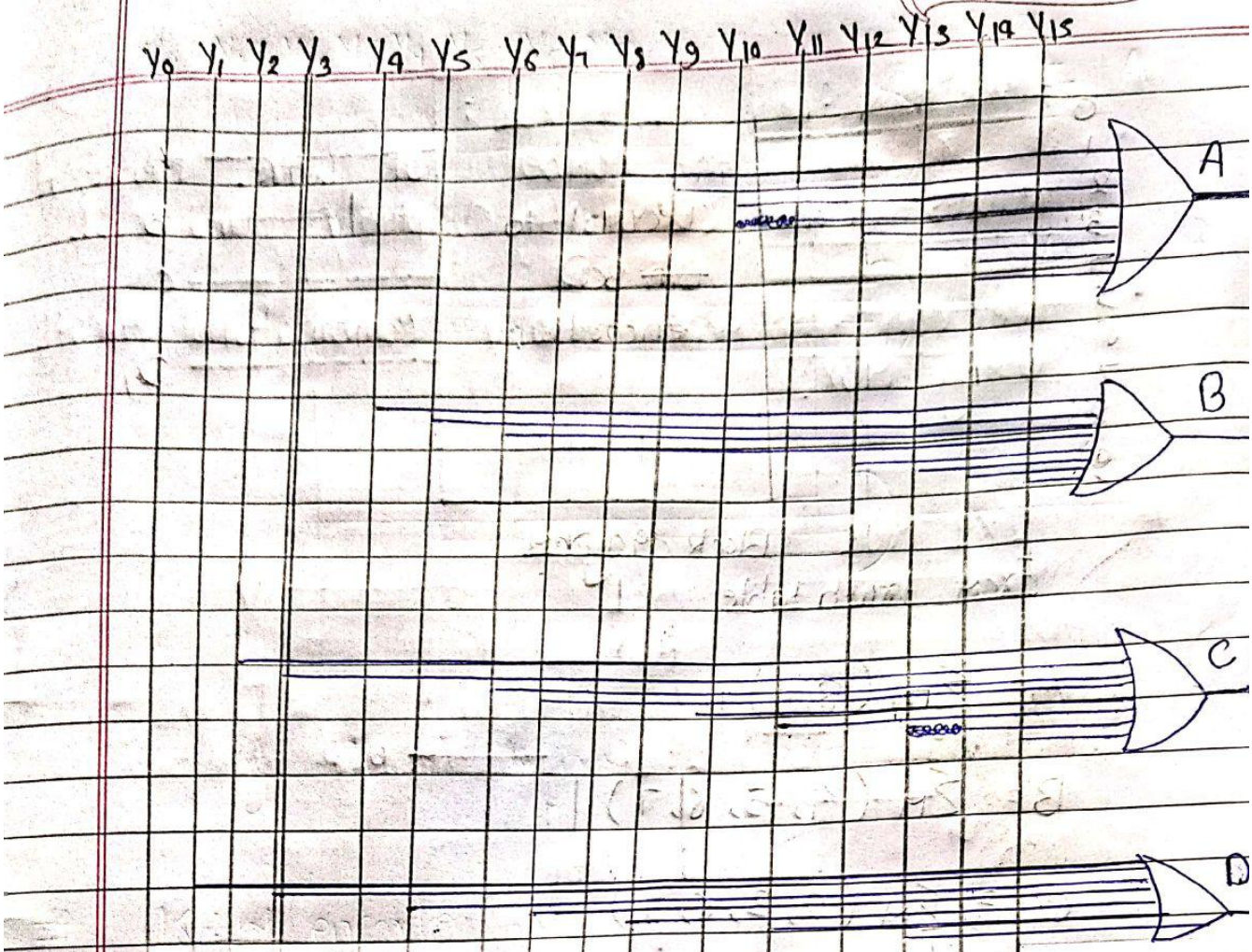
From table

$A = Y_8 + Y_9 + Y_{10} + Y_{11} + Y_{12} + Y_{13} + Y_{14} + Y_{15}$.

$C = Y_2 + Y_3 + Y_6 + Y_7 + Y_{10} + Y_{11} + Y_{14} + Y_{15}$.

$B = Y_4 + Y_5 + Y_6 + Y_7 + Y_{12} + Y_{13} + Y_{14} + Y_{15}$.

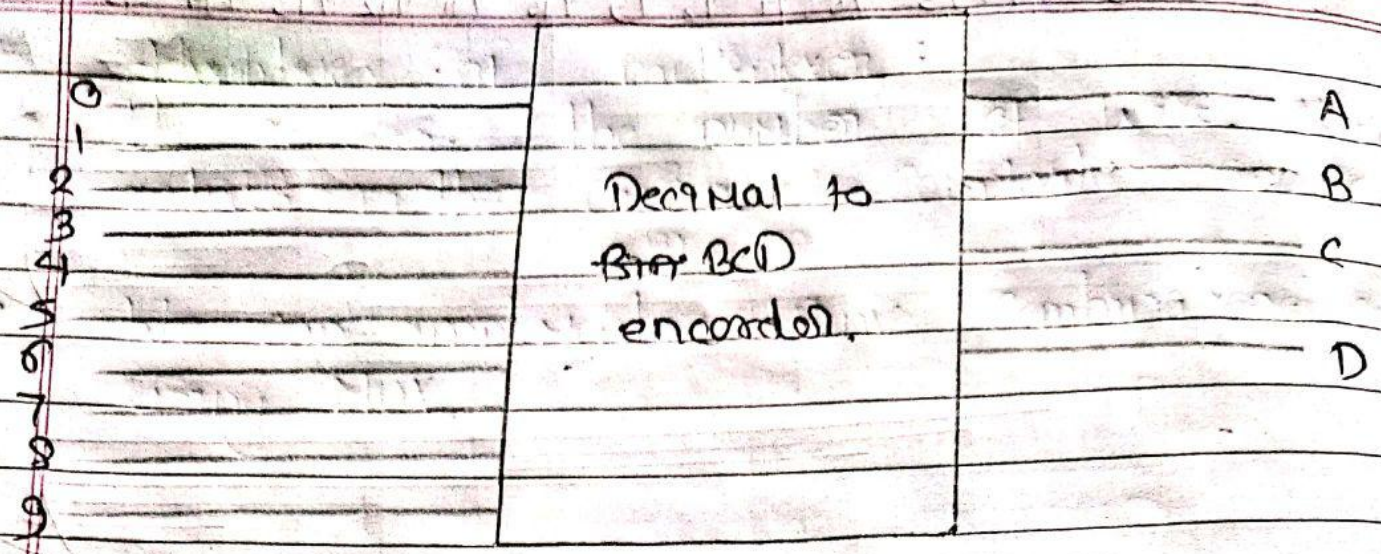
$D = Y_1 + Y_3 + Y_5 + Y_7 + Y_9 + Y_{11} + Y_{13} + Y_{15}$.



Decimal to BCD encoder

Truth table

Decimal	A	B	C	D
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



Block diagram

From truth table

$A = \Sigma_m(8,9)$

$B = \Sigma_M(4,5,6,7)$

$C = \Sigma_M(2,3,6,7)$

$D = \Sigma_m(1,3,5,7,9)$

Do realization yourself

Multiplexer & Demultiplexer.

Multiplexer (Data Selector)

Multiplex means 'many to one'. A multiplexer is a combinational circuit which selects single information from multiple inputs one at a time with the help of selection line. Multiplexing is the process of transmitting a large number of information over a single line. For 'n' inputs, there are 'm' selection line and a single output where $2^m = n$.

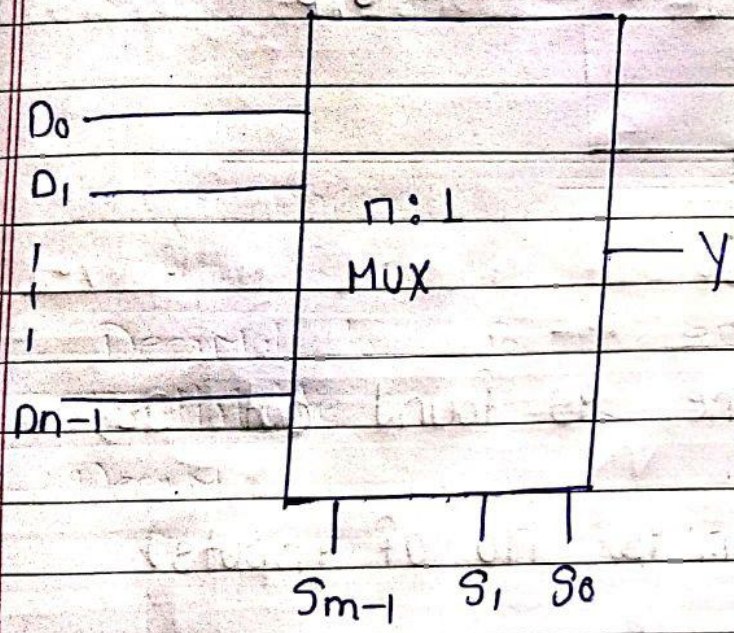


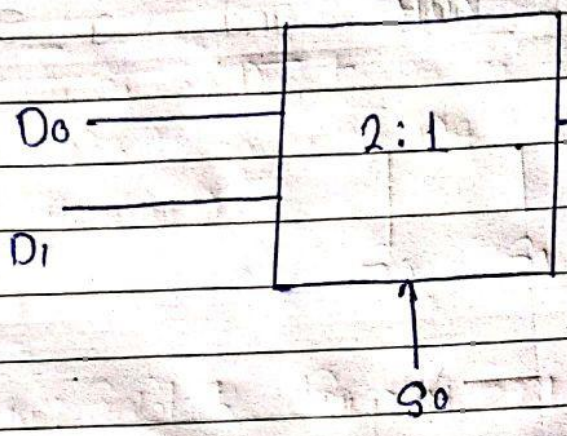
Fig: Block diagram of a multiplexer

Application:
As building block in the CPU.
Used in the telecommunication at the transmitter side.

Advantage of multiplexer;

- It reduces the number of wire. Hence, it reduces the circuit complexity and cost.
- We can implement many combinational circuits using MUX.

2x1 MUX



no of selection line is found out as,

$2^M = 2$

or, $2^m = n$ ('n' is no of inputs)

or $2^M = 2^1$

or $m = 1$

('m' no of enabler)

Function table

S_0	O/P
0	D_0
1	D_1

$$O/P = \overline{S_0} D_0 + S_0 D_1$$

Combination circuit diagram

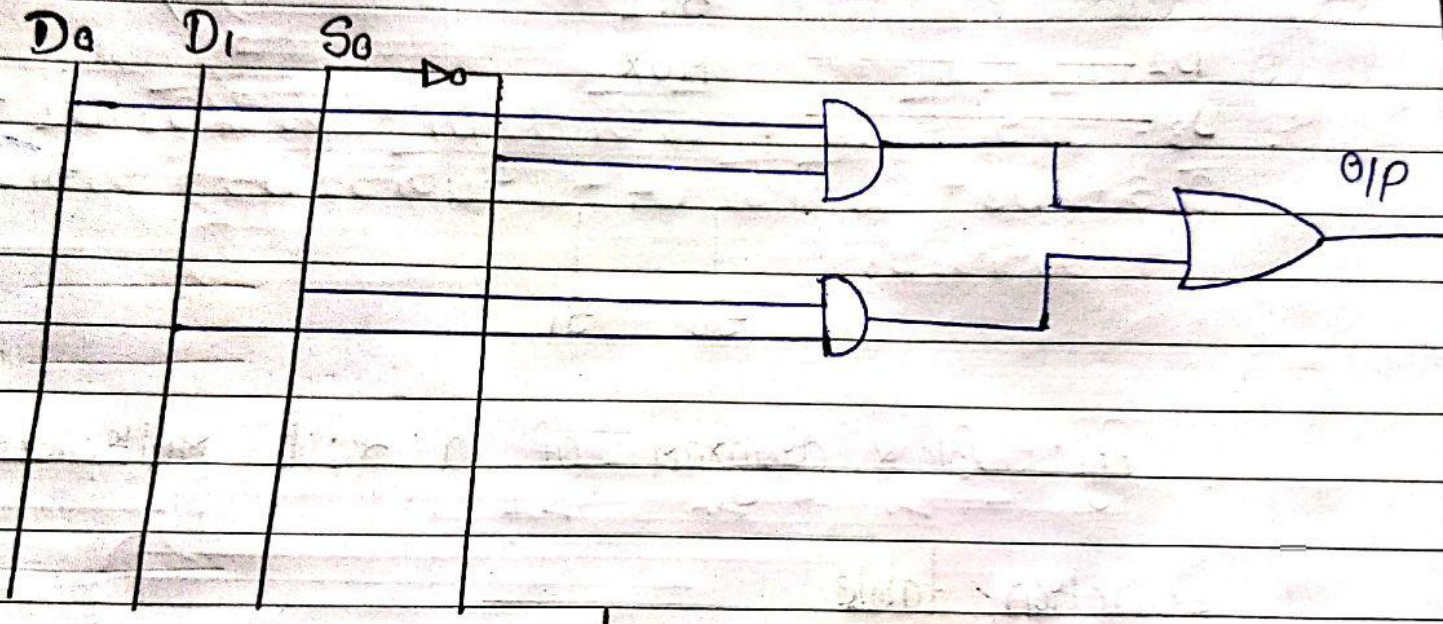


Fig: Logic diagram of 4:1 MUX.

4x1 MUX :

$2^m = n \Rightarrow 2^m = 4$

$2^m = 2^2$

$\therefore m = 2$

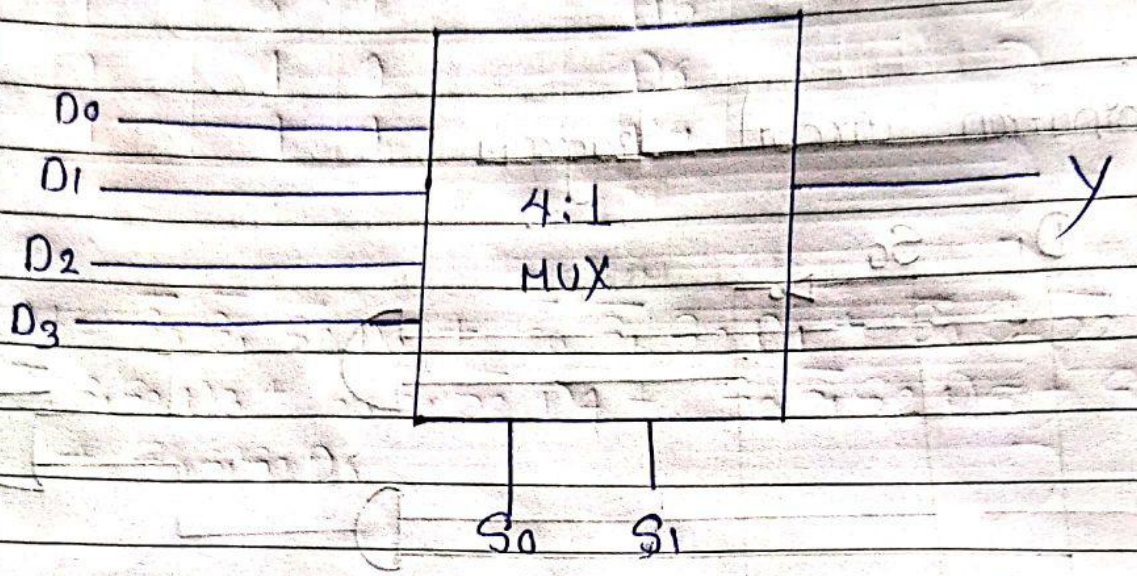


Fig: Block diagram of a 4:1 Multiplexer

Function table

S_0	S_1	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

$$Y = \overline{S_0}\,\overline{S_1} D_0 + \overline{S_0} S_1 D_1 + S_0 \overline{S_1} D_2 + S_0 S_1 D_3$$

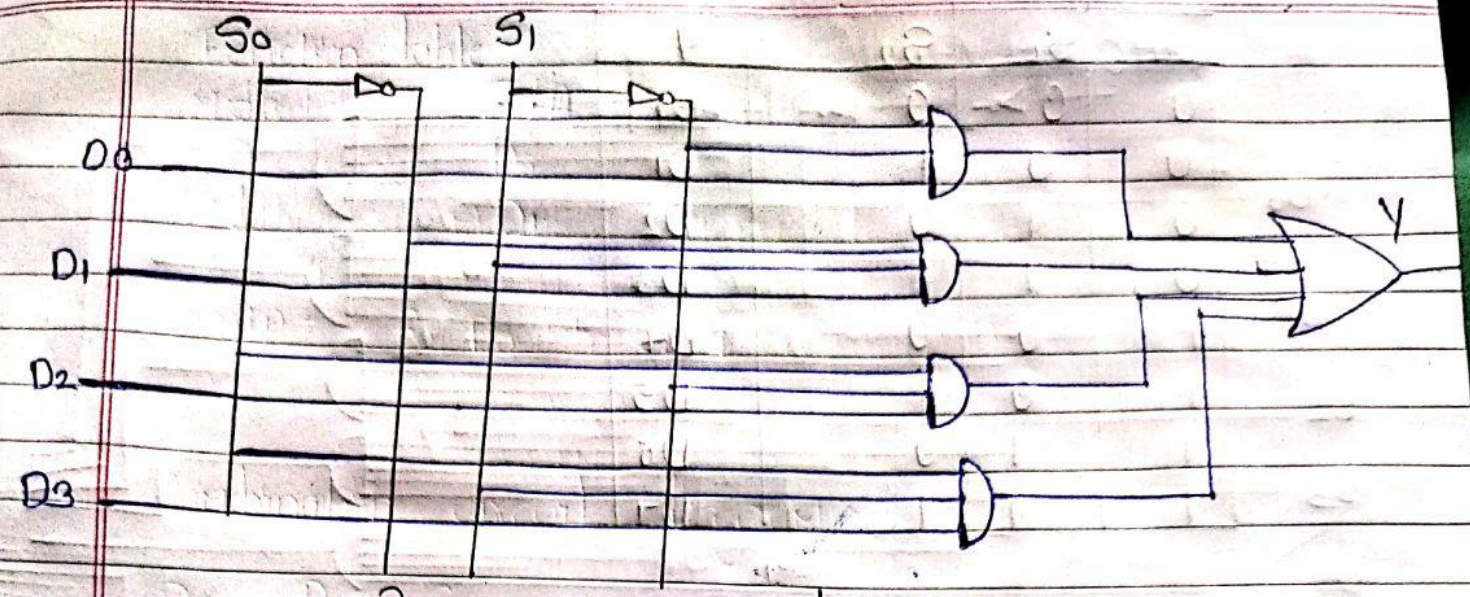


Fig: Logic circuit diagram of 4:1 MUX

8X1 mux:

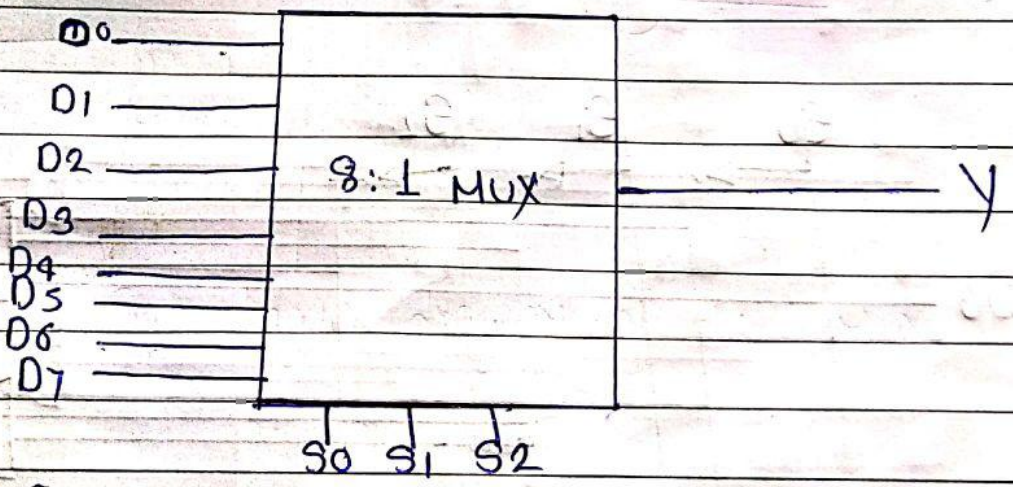


Fig: Block diagram of 8:1 mux

Function table

S_0	S_1	S_2	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

$\therefore Y = \overline{S_0}\,\overline{S_1}\,\overline{S_2}D_0 + S_0\overline{S_1}\,\overline{S_2}D_1 + \overline{S_0}S_1\overline{S_2}D_2 + \overline{S_0}S_1S_2D_3 + S_0\overline{S_1}\,\overline{S_2}D_4 + S_0\overline{S_1}S_2D_5 + S_0S_1\overline{S_2}D_6 + S_0S_1S_2D_6$

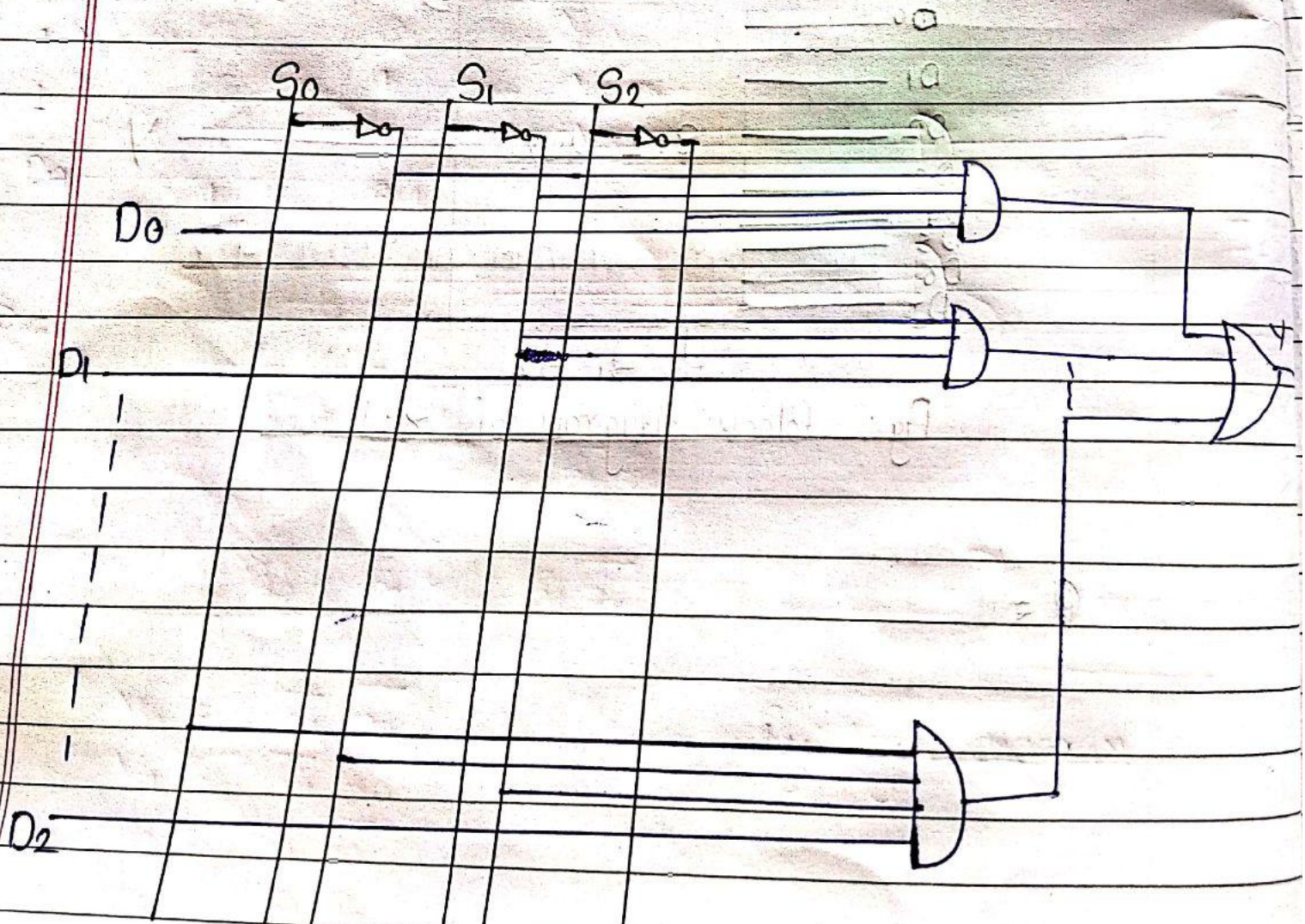


Fig: Logic circuit of 8:1 MUX

16:1 MUX

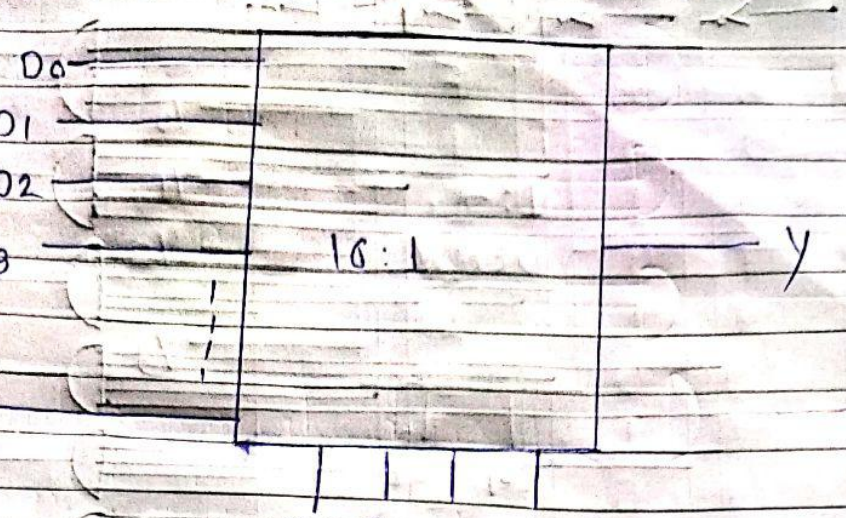


fig: block diagram of 16:1 MUX.

Function table:

S0	S1	S2	S3	y
0	0	0	0	D0
0	0	0	1	D1
0	0	1	0	D2
0	0	1	1	D3
0	1	0	0	D4
0	1	0	1	D5
0	1	1	0	D6
0	1	1	1	D7
1	0	0	0	D8
1	0	0	1	D9
1	0	1	0	D10
1	0	1	1	D11
1	1	0	0	D12
1	1	0	1	D13
1	1	1	0	D14
1	1	1	1	D15

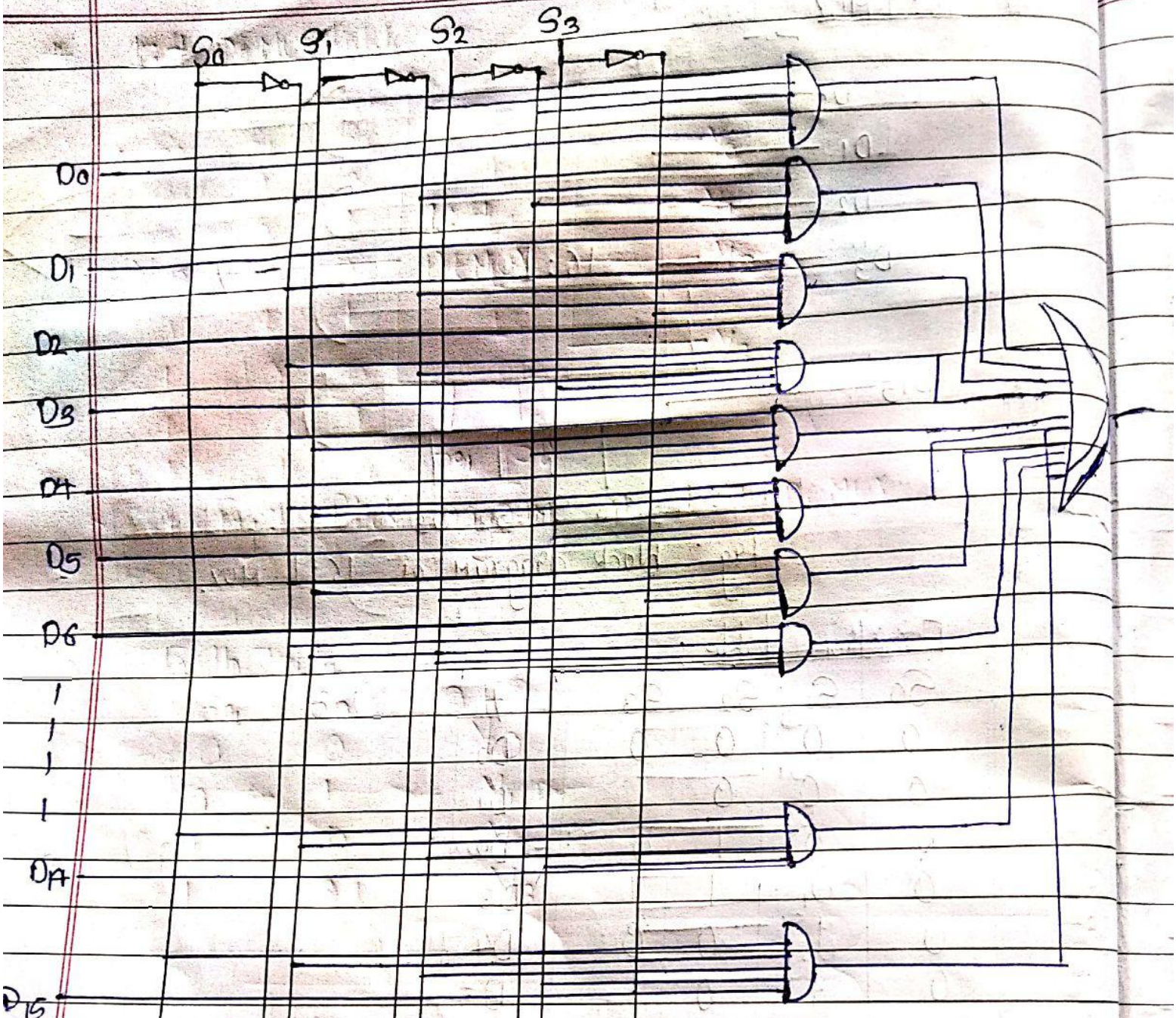


Fig: Logic diagram.

$$Y = \overline{S_0}\,\overline{S_1}\,\overline{S_2}\,D_0 + \overline{S_0}\,\overline{S_1}\,\overline{S_2}\,S_3\,D_1 + \overline{S_0}\,\overline{S_1}\,S_2\,\overline{S_3}\,D_2 + \overline{S_0}\,\overline{S_1}\,S_2\,S_3\,D_3 + \overline{S_0}\,S_1\,\overline{S_2}\,\overline{S_3}\,D_4 + \overline{S_0}\,S_1\,\overline{S_2}\,S_3\,D_5 + \overline{S_0}\,S_1\,S_2\,\overline{S_3}\,D_6 + \overline{S_0}\,S_1\,S_2\,S_3\,D_7 + S_0\,\overline{S_1}\,\overline{S_2}\,\overline{S_3}\,D_8 + S_0\,\overline{S_1}\,\overline{S_2}\,S_3\,D_9 + S_0\,\overline{S_1}\,S_2\,\overline{S_3}\,D_{10} + S_0\,\overline{S_1}\,S_2\,S_3\,D_{11} + S_0\,S_1\,\overline{S_2}\,\overline{S_3}\,D_{12} + S_0\,S_1\,\overline{S_2}\,S_3\,D_{13} + S_0\,S_1\,S_2\,\overline{S_3}\,D_{14} + S_0\,S_1\,S_2\,S_3\,D_{15}.$$

Demultiplexer (Data Divider)

'Demultiplexer' means 'one to many'. A demultiplexer is a combinational circuit that receives information on a single input and transmits the same information over one of the possible output lines. Selection of output lines is controlled by selection line. For 'n' output, there is 'm' selection line and single input where $n = 2^m$

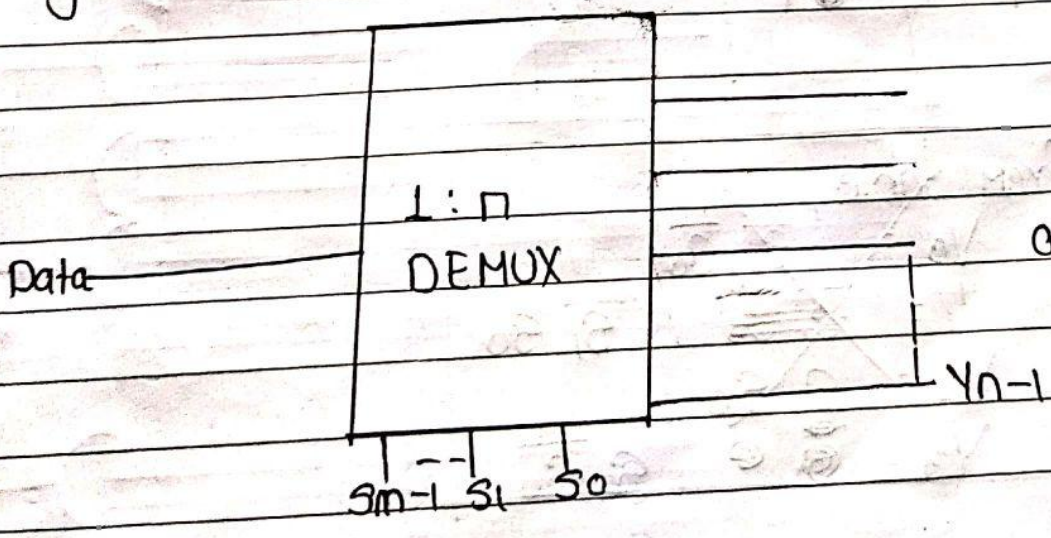


fig: Block diagram of a demultiplexer

1:4 DEMUX

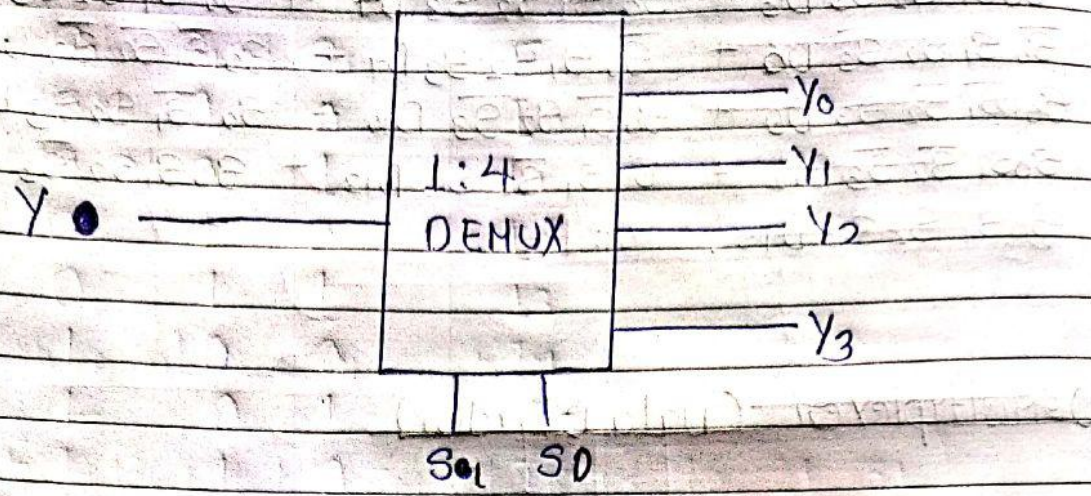


Fig: Block diagram of 1:4 DeMUX.

Truth Table

S_1	S_0	F
0	0	Y_0
0	1	Y_1
1	0	Y_2
1	1	Y_3

From table,

$Y_0 = \overline{S_1}\,\overline{S_0}$

$Y_1 = \overline{S_1}\,S_0$

$Y_2 = S_1\,\overline{S_0}$

$Y_3 = S_1\,S_0$

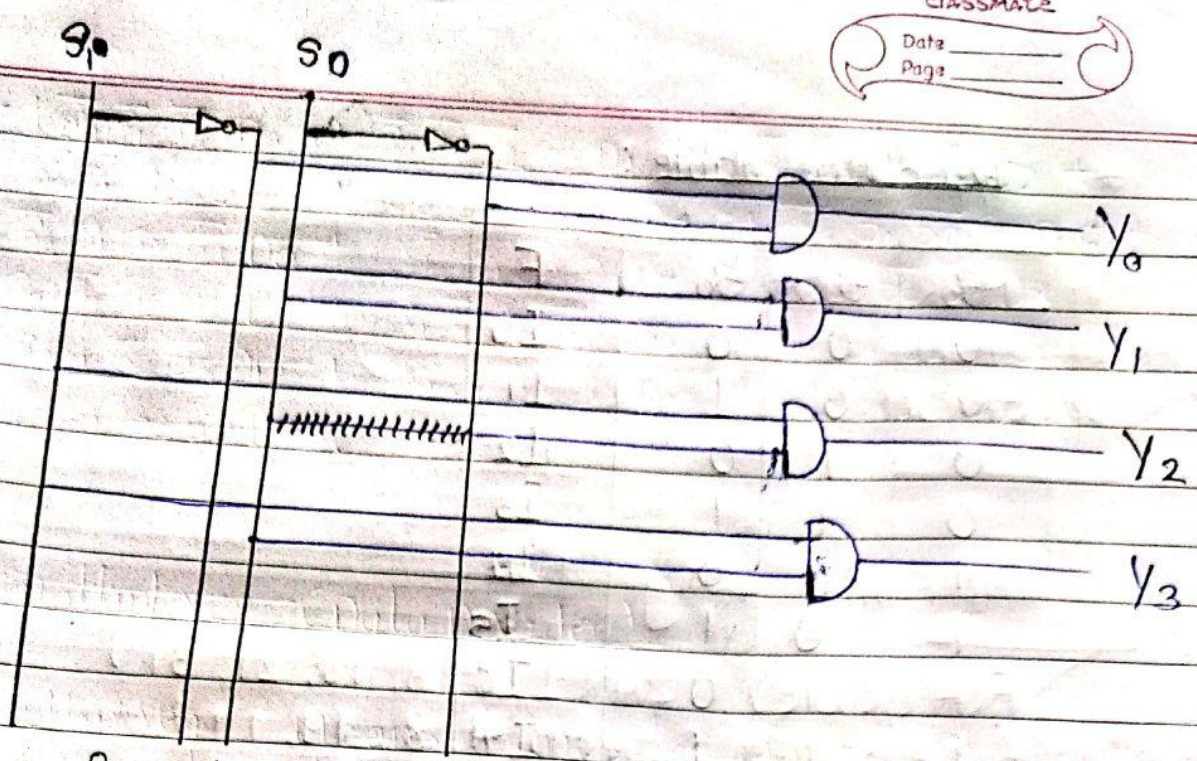


fig: Logic Diagram of 1:4 DEMUX

1:8 De-Multiplexer

→ A 1:8 De-Multiplexer consists of 8 outputs, 3 Select lines and only one inputs.

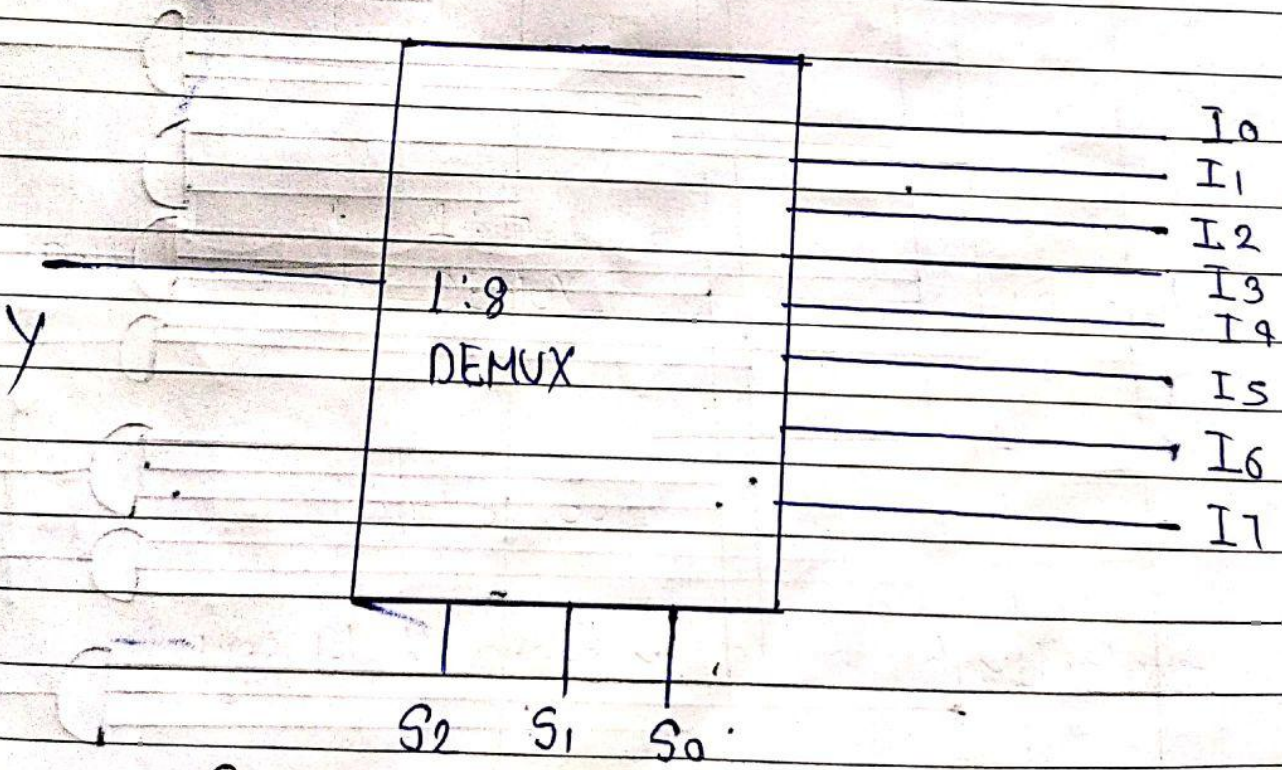
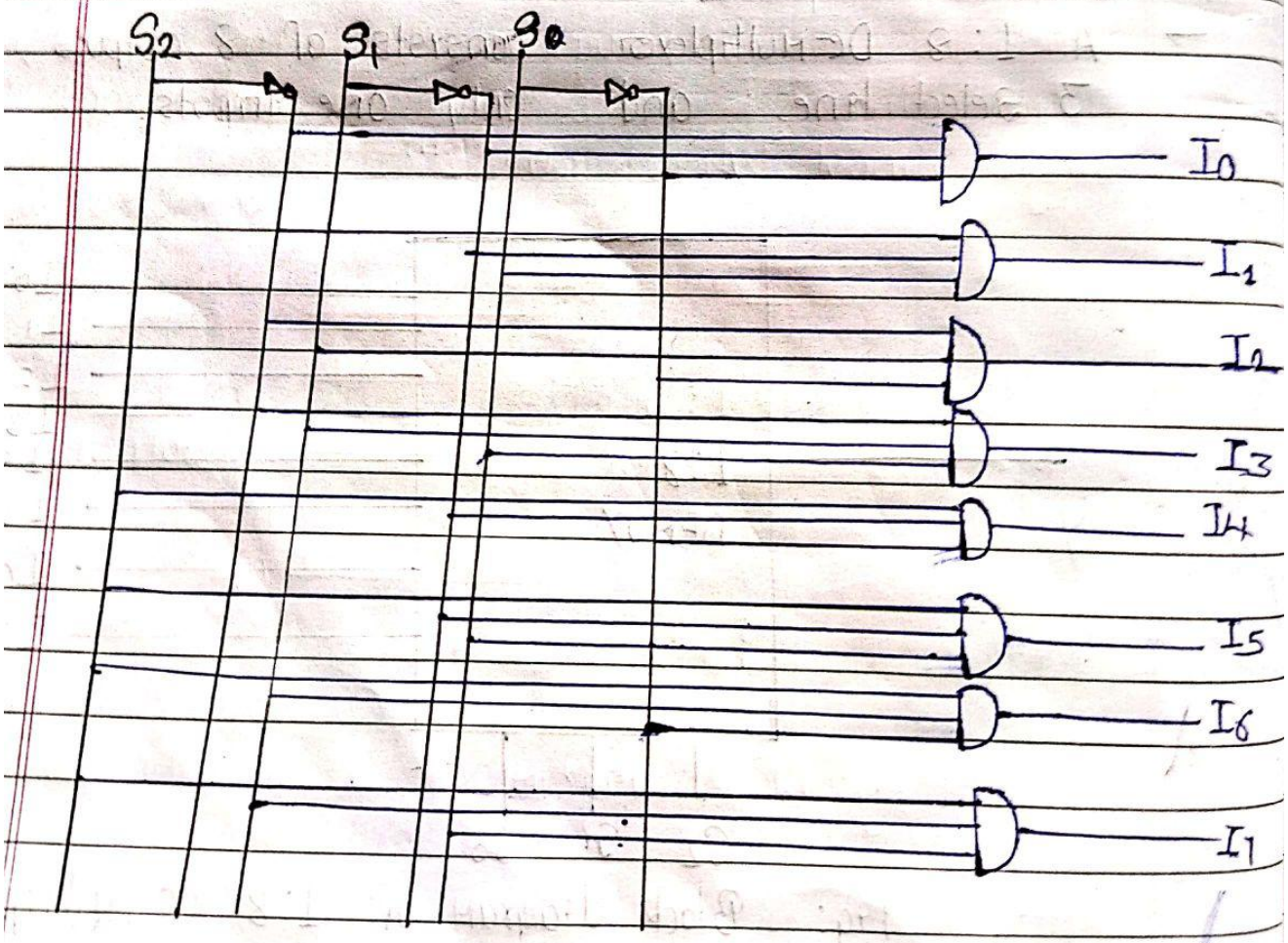


fig: Block diagram of 1:8 de-Multiplexer.

Function table:

S_2	S_1	S_0	F
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Relization



PLD: programming logic Design.

1. ROM (Read only MEMORY)

A ROM is essentially a memory device in which a fixed set of binary information is stored. The binary information must be first specified by the user and then embedded in the unit to form the required interconnection pattern. ROM contains special internal links that can be fused or broken. Once a patter is established for a ROM, it remained fixed even if the power supply to the circuit is switched off and then switched on again i.e. it is non volatile.

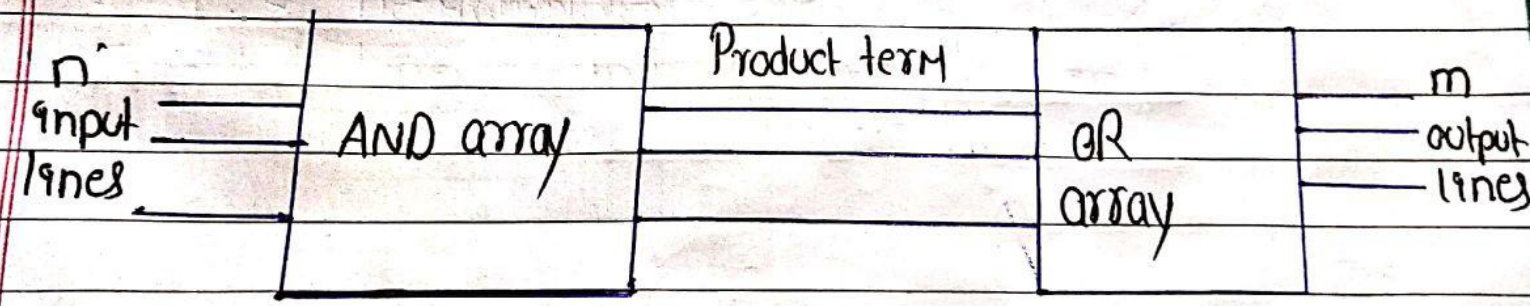


fig: Block diagram of ROM

Types of ROM.

1. Mask ROM
2. PROM
3. EPROM

Implement using ROM;

$F_1(A,B,C,D,E) = \Sigma_M(1, 6, 8, 16, 21, 31)$

$F_2(A,B,C,D,E) = \Sigma_M(0, 4, 8, 19, 17)$

$F_3(A,B,C,D,E) = \Sigma_M(3, 9, 22, 29, 30)$

$F_4(A,B,C,D,E) = \Sigma_M(7, 6, 5, 12, 15, 18)$

$F_5(A,B,C,D,E) = \Sigma_m(16, 20, 21, 28, 27)$

	A	B	C	D	E	F_1	F_2	F_3	F_4	F_5
0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	1	1	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0
3	0	0	0	1	1	0	0	1	0	0
4	0	0	1	0	0	0	1	0	0	0
5	0	0	1	0	1	0	0	0	1	0
6	0	0	1	1	0	1	0	0	1	0
7	0	0	1	1	1	0	0	0	1	0
8	0	1	0	0	0	1	1	0	0	0
9	0	1	0	0	1	0	0	1	0	0
10	0	1	0	1	0	0	0	0	0	0
11	0	1	0	1	1	0	0	0	0	0
12	0	1	1	0	0	0	0	0	1	0
13	0	1	1	0	1	0	0	0	0	0
14	0	1	1	1	0	0	0	0	0	0
15	0	1	1	1	1	0	0	0	1	0
16	1	0	0	0	0	1	0	0	0	1
17	1	0	0	0	1	0	1	0	0	0
18	1	0	0	1	0	0	0	0	1	0
19	1	0	0	1	1	0	1	0	0	0
20	1	0	1	0	0	0	0	0	0	1
21	1	0	1	0	1	1	0	0	0	1
22	1	0	1	1	0	0	0	1	0	0
23	1	0	1	1	1	0	0	0	0	0
24	1	1	0	0	0	0	0	0	0	0
25	1	1	0	0	1	0	0	0	0	0
26	1	1	0	1	0	0	0	0	0	0
27	1	1	0	1	1	0	0	0	0	1
28	1	1	1	0	0	0	0	0	0	1
29	1	1	1	0	1	0	0	1	0	0
30	1	1	1	1	0	0	0	1	0	0
31	1	1	1	1	1	1	0	0	0	0

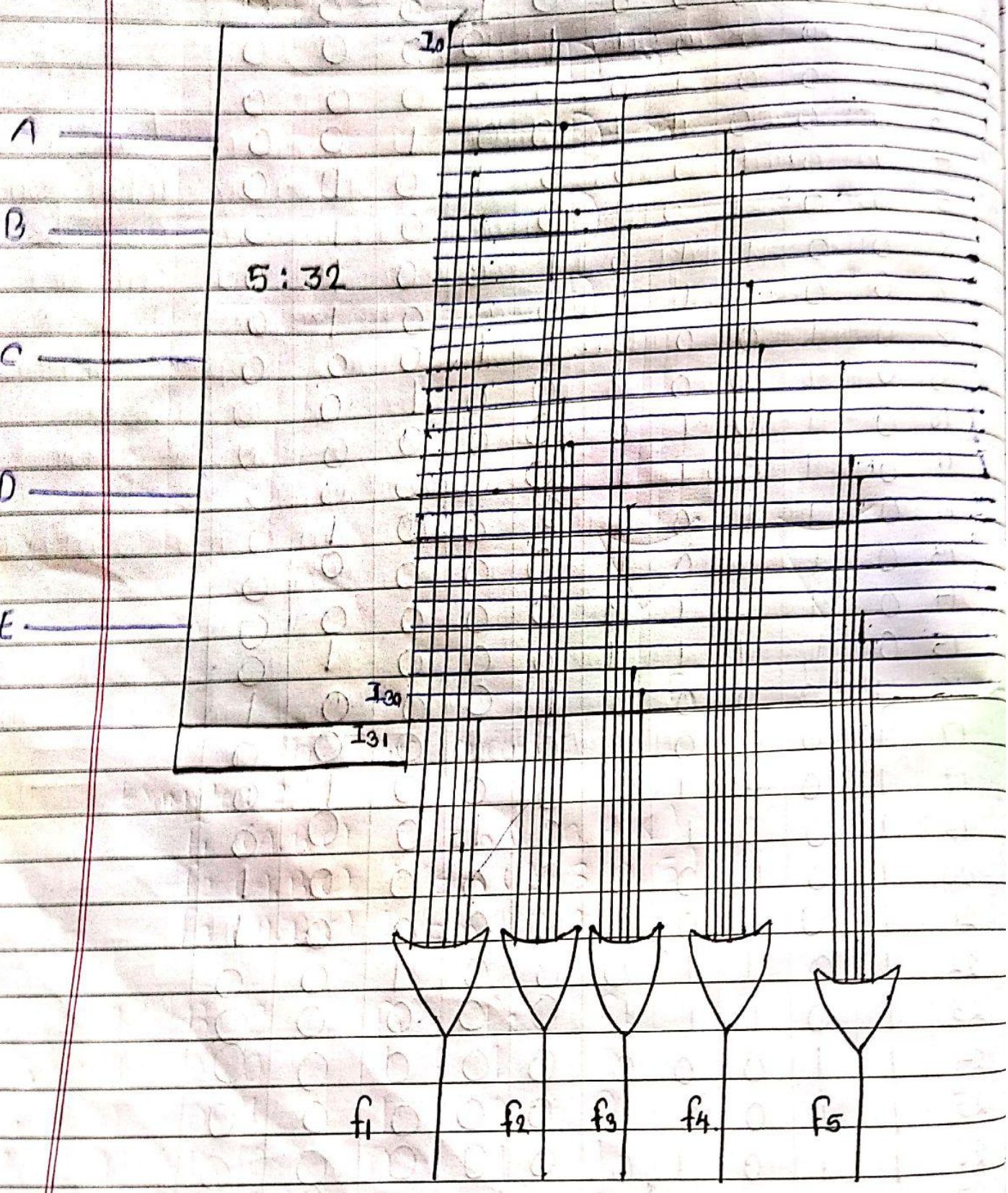


Fig: Logic circuit

PLA (Programming Logic Array)

A PLA consists of a programmable AND array and a programmable OR array. The PLA is also called a field Programmable Logic Array (FPLA) because the user in the field programs it, not the Manufacturer.

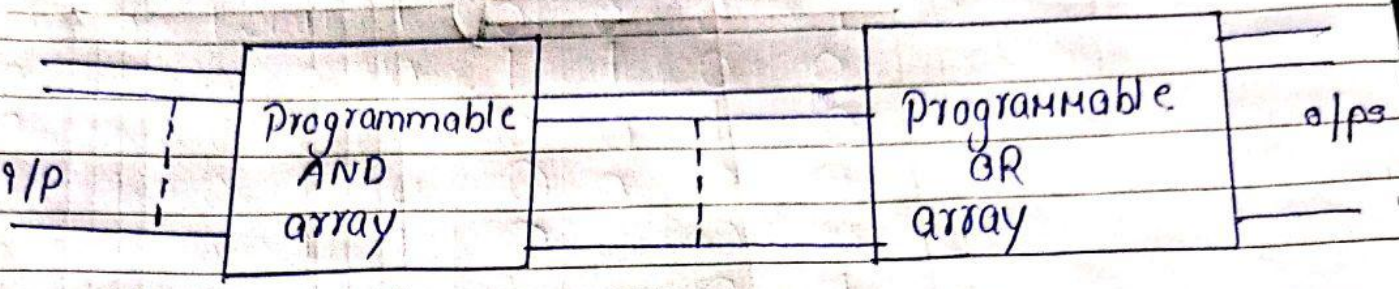


fig: Block diagram of PLA

Example: 1

$F_1(A, B) = \Sigma m(0, 1)$

$F_2(A, B) = \Sigma m(2, 3)$

Here,

A	B	F_1	f_2
0	0	1	0
0	1	1	0
1	0	0	1
1	1	0	1

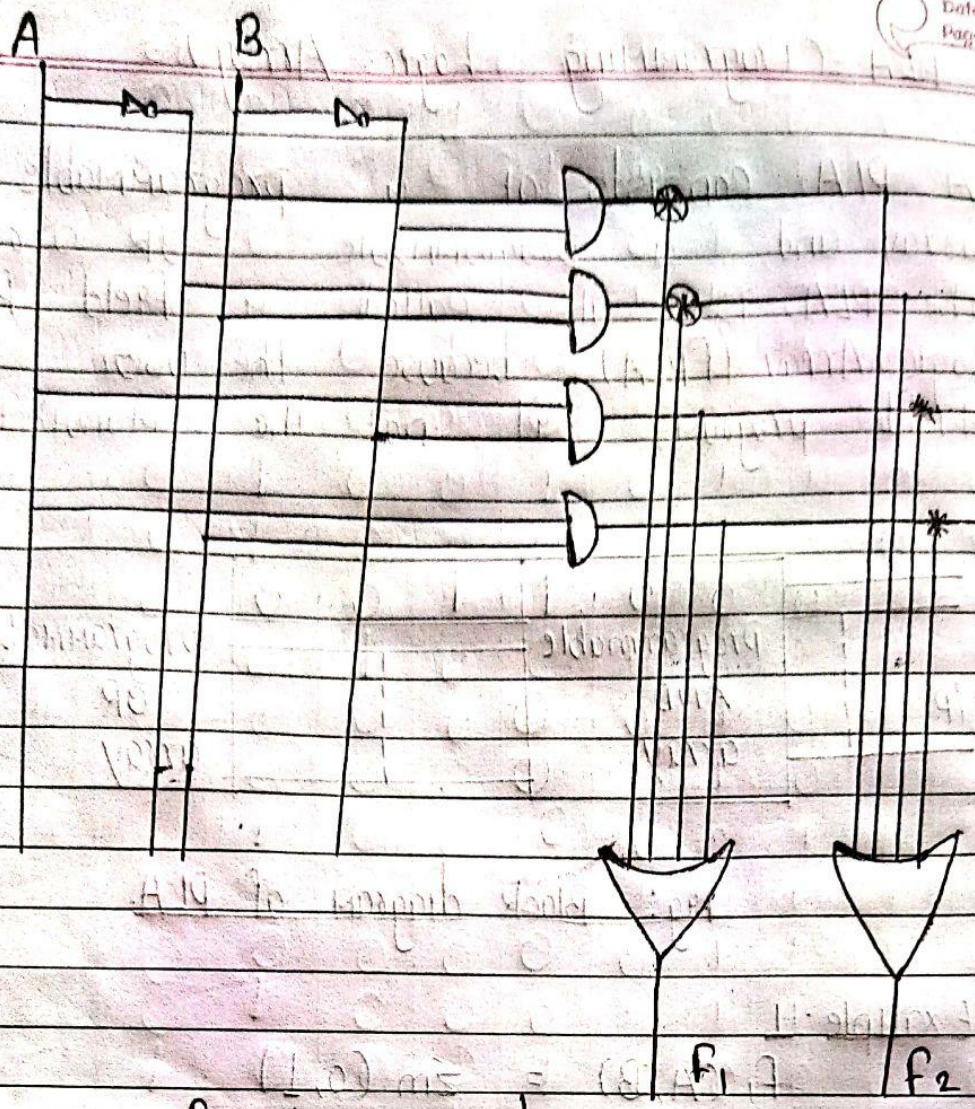


fig: Logic circuit:

Example: 2

$f_1(A,B,C) = \Sigma m(1,0,4,6)$

$f_2(A,B,C) = \Sigma m(2,3,0,5)$

$f_3(A,B,C) = \Sigma m(7,2)$

	A	B	C	f_1	f_2	f_3
0	0	0	0	1	1	0
1	0	0	1	1	0	0
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	0	0
5	1	0	1	0	1	0
6	1	1	0	1	0	0
7	1	1	1	0	0	1

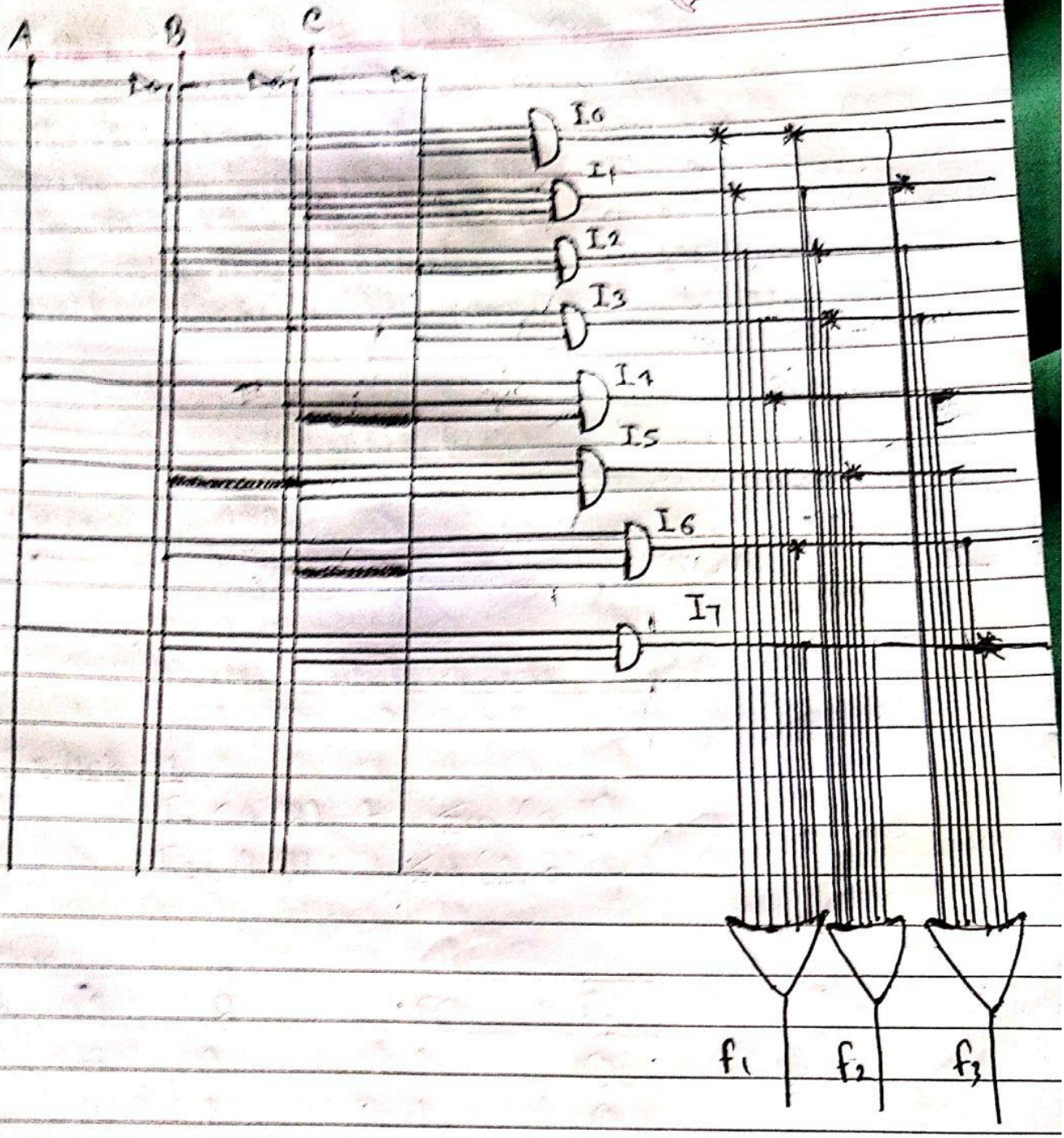


Fig: Logic circuit

Binary Adder / Subtractor

A binary adder-subtractor is a combinational circuit that performs the arithmetic operations of addition and subtraction with binary numbers.

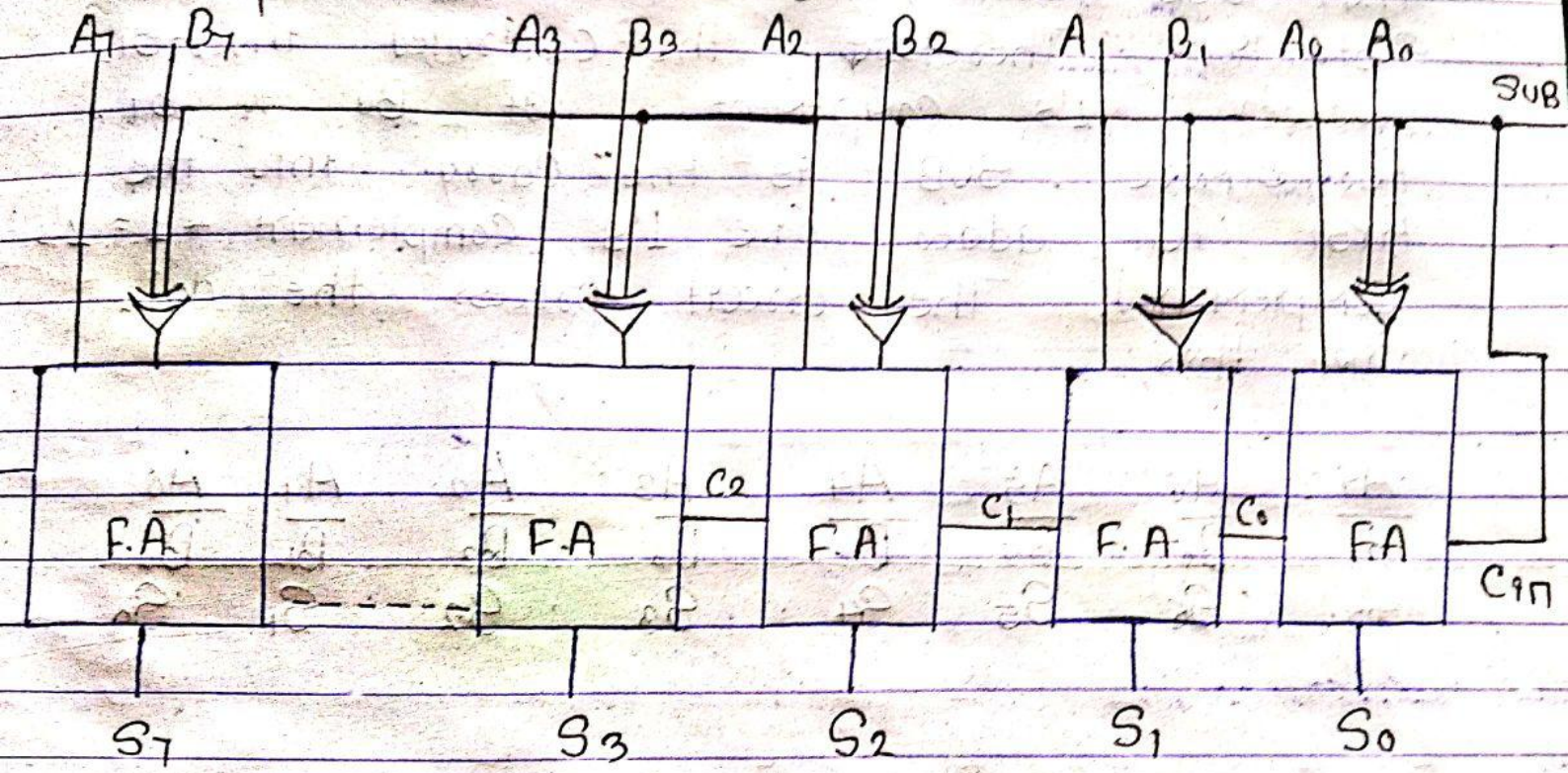


Fig: 8-bit adder/subtractor.

When SUB = LOW, the circuit performs the addition. The binary numbers B7 to B0 passes through the controlled inverter with no change. The full adders then produces the correct output SUM as,

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

When SUB = HIGH, the circuit performs the subtraction. Therefore the controlled inverter produces 1's complement of B_0 to B_7. Furthermore, SUB is the carry into the first full adder (i.e 1's complement + 1 = 2's complement). The circuit process the data like this,

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
$\overline{B_7}$	$\overline{B_6}$	$\overline{B_5}$	$\overline{B_4}$	$\overline{B_3}$	$\overline{B_2}$	$\overline{B_1}$	$\overline{B_0}$
S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

Magnitude Comparator

Magnitude comparator is a combinational circuit, designed to compare the two ~~bit~~ n-bit words applied as its input. The comparator has three outputs namely greater ($A > B$), lesser ($A < B$) and equal ($A = B$). Depending upon the results of comparison, one of these outputs will go high.

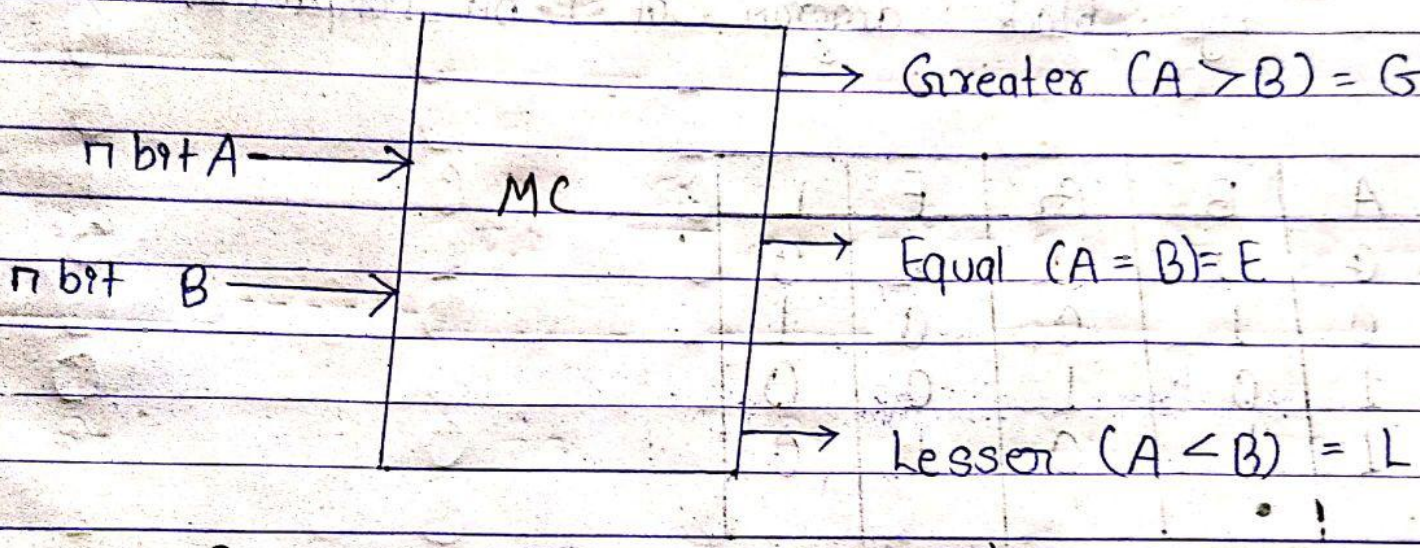


Fig: Block diagram of a n-bit comparator

Design 1-bit magnitude comparator :

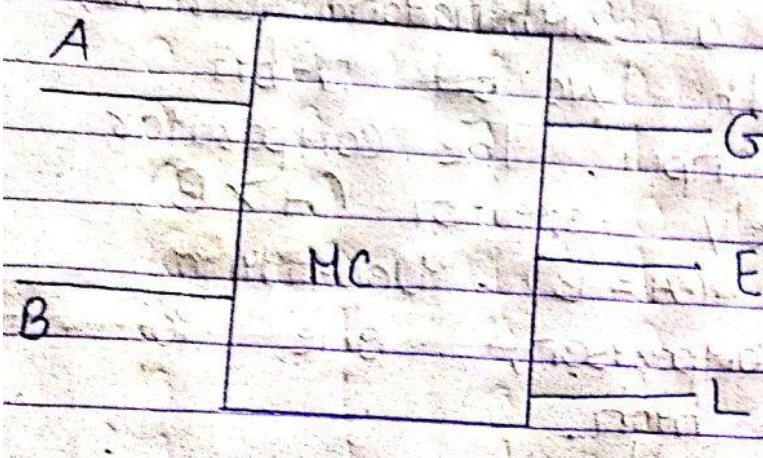


Fig: block diagram of 1-bit comparator

A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$G = A\overline{B}$

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

$= \overline{A \oplus B}$

$L = \overline{A}B$

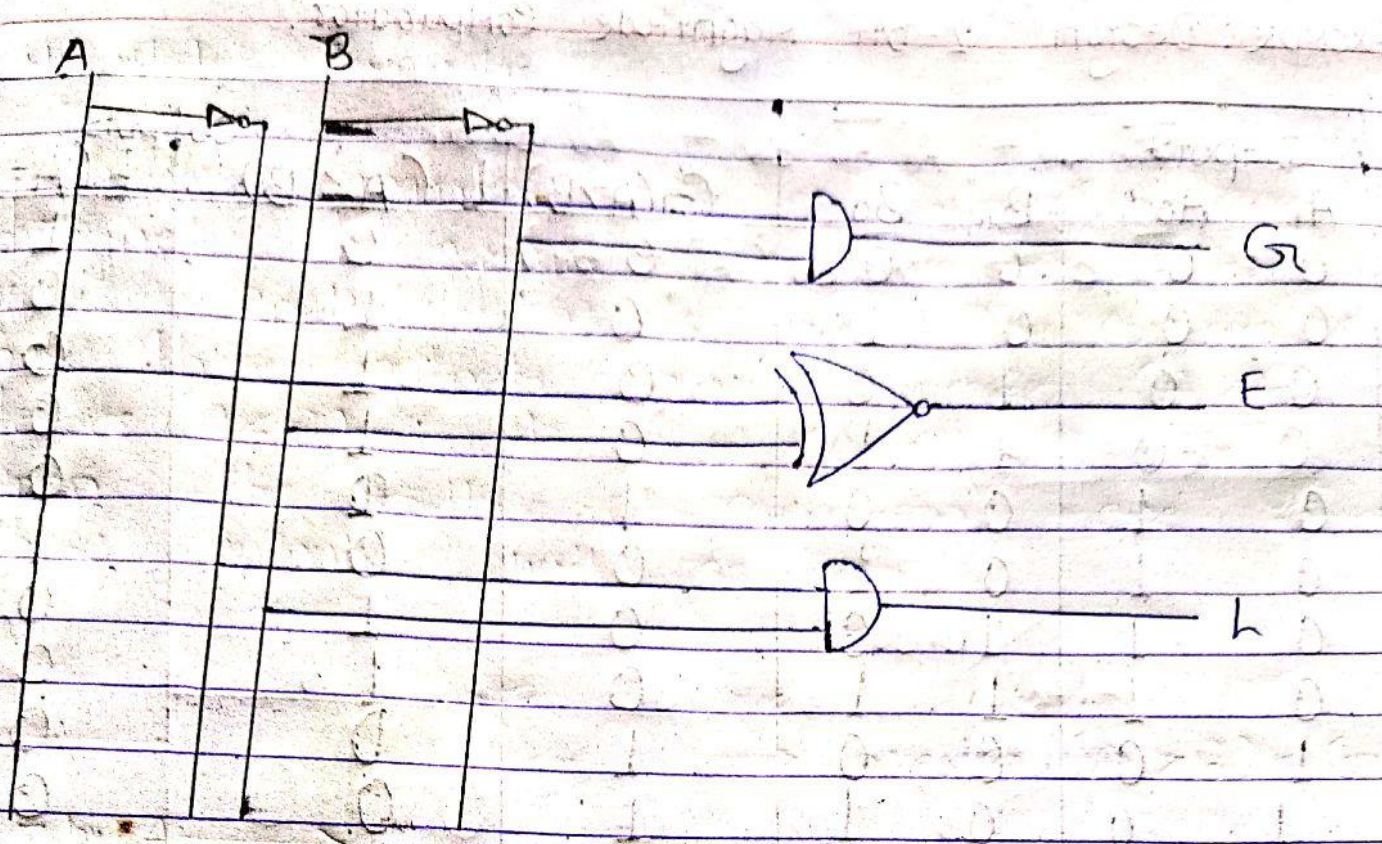


fig: Logic diagram of 1-bit Magnitude Comparator

Example: Design 2-bit Magnitude Comparator

Inputs				outputs		
A_1	A_0	B_1	B_0	G (A>B)	L (A<B)	E (A=B)
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
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	0	1

For G (A > B)

A_1A_0 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

$$G = A_1\overline{B_1} + A_0\overline{B_1}\,\overline{B_0} + A_1A_0\overline{B_0}$$

$$= A_1\overline{B_1} + A_0\overline{B_0}(A_1 + \overline{B_1})$$

For L (A < B)

A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

$$L = \overline{A_1} B_1 + \overline{A_1}\,\overline{A_0} B_0 + \overline{A_0} B_0 B_1$$

$$= \overline{A_1} B_1 + \overline{A_0} B_0 (\overline{A_1} + B_1)$$

For $F(A = B)$

$A_1 A_0$ \ $B_1 B_0$	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

$$E = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1} A_0 \overline{B_1} B_0 + A_1 A_0 B_1 B_0 + A_1 \overline{A_0} B_1 \overline{B_0}$$

$$= \overline{A_1}\,\overline{B_1} (\overline{A_0}\,\overline{B_0} + A_0 B_0) + A_1 B_1 (B_0 A_0 + \overline{A_0}\,\overline{B_0})$$

$= (\bar{A_1}\bar{B_1} + A_1 B_1)(\bar{A_0}\bar{B_0} + A_0 B_0)$

$= \overline{A_1 \oplus B_1} \cdot \overline{A_0 \oplus B_0}$

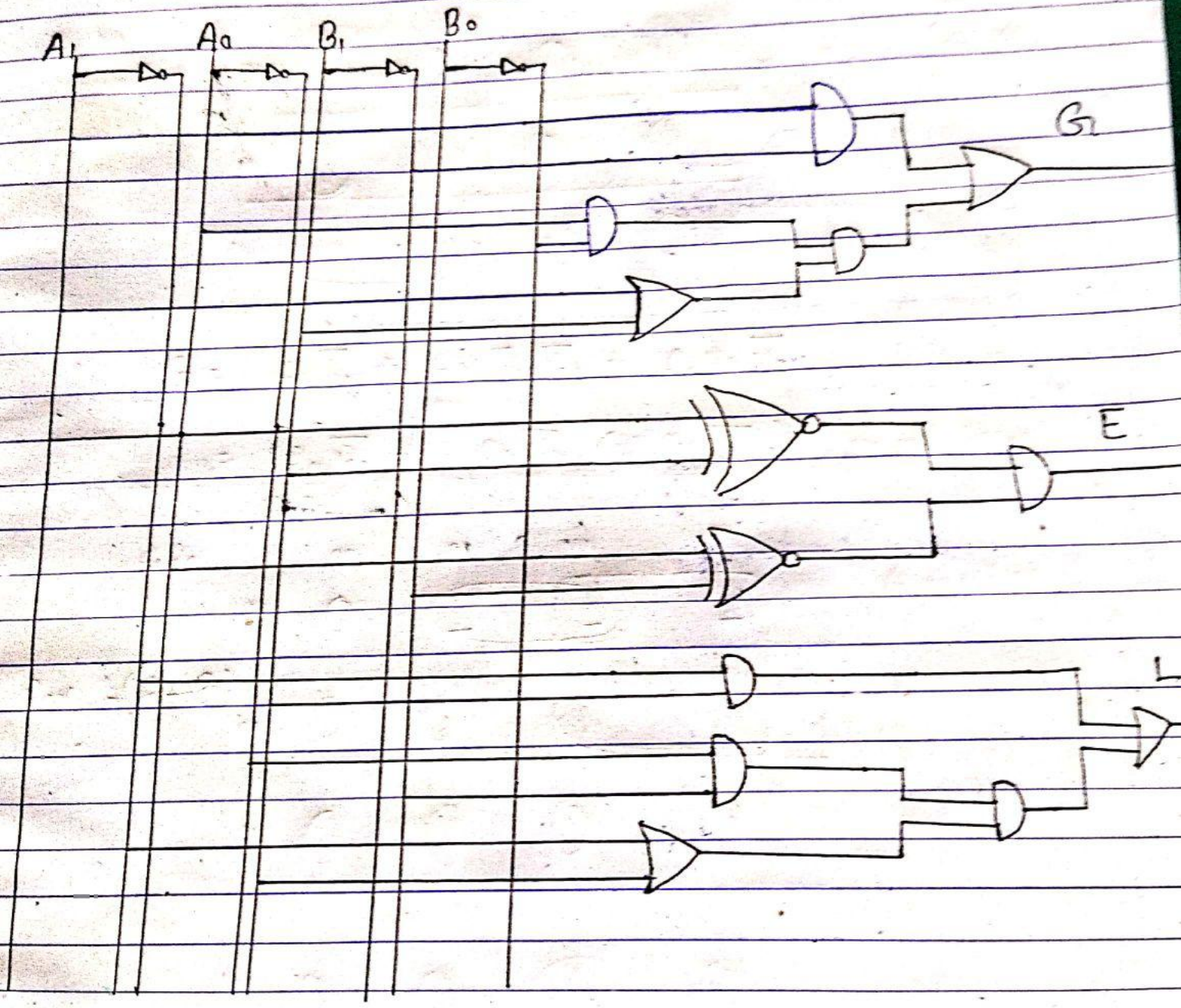


Fig: Circuit diagram of 2-bit Magnitude Comparator