Date :
Page :

5 variable K-map (A, B, C, D, E)

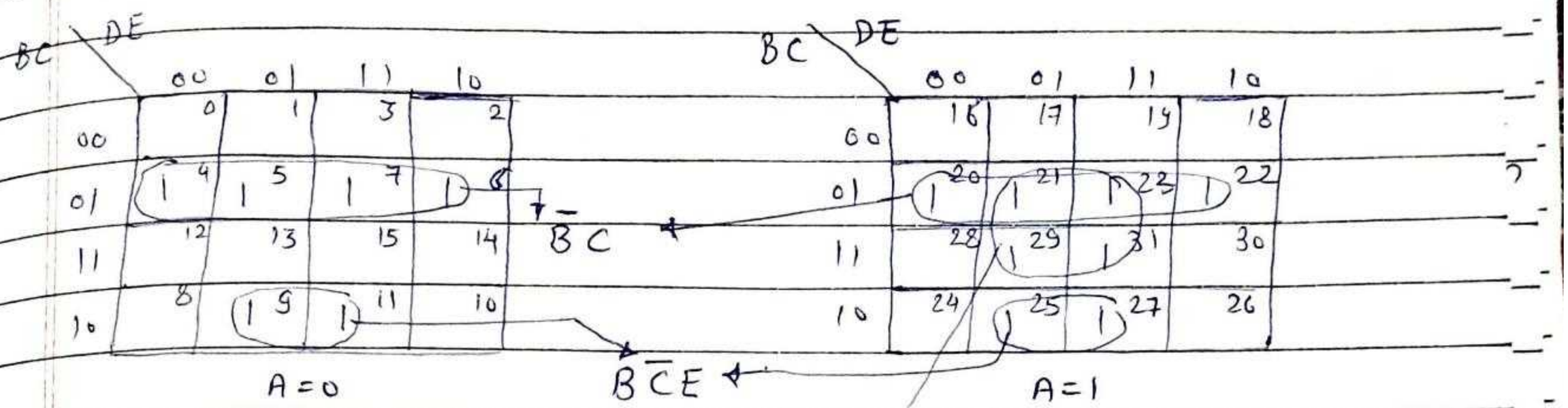
BC \ DE	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

A = 0

BC \ DE	00	01	11	10
00	16	17	19	18
01	20	21	23	22
11	28	29	31	30
10	24	25	27	26

A = 1

$\Rightarrow f(A, B, C, D, E) = \Sigma m(4, 5, 6, 7, 9, 11, 20, 21, 22, 23, 25, 27, 29, 31)$



$$f = ACE + \bar{B}C + B\bar{C}E$$

6 variable K Map (A, B, C, D, E, F)

AB
00
01
10
11

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

A = 0, B = 0

CD \ EF	00	01	11	10
00	16	17	19	18
01	20	21	23	22
11	28	29	31	30
10	24	25	27	26

A = 0, B = 1

CD \ EF	00	01	11	10
00	32	33	35	34
01	36	37	39	38
11	44	45	47	46
10	40	41	43	42

A = 1, B = 0

CD \ EF	00	01	11	10
00	48	49	51	50
01	52	53	55	54
11	60	61	63	62
10	56	57	59	58

A = 1, B = 1

"UNIT-3"

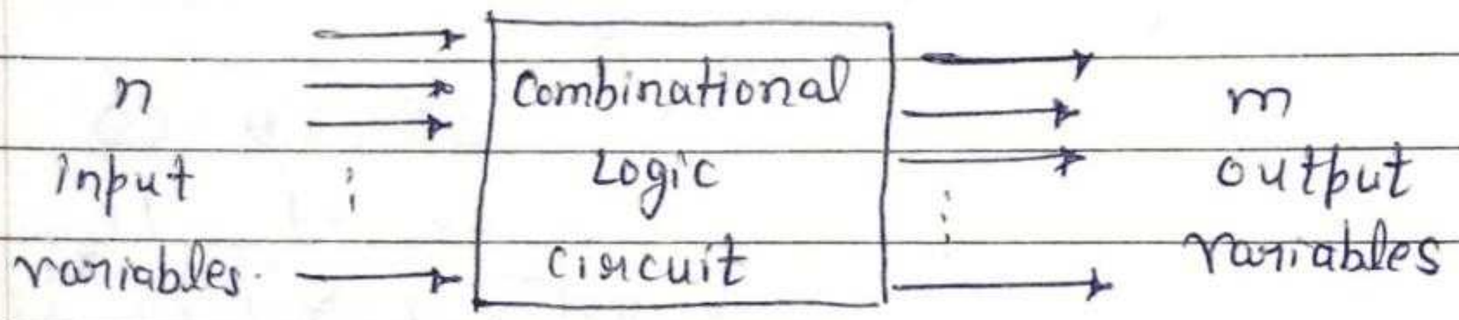
"COMBINATIONAL CIRCUITS"

Combinational circuits →

→ A combinational circuit is one in which the present state of the combination of the logic inputs decides the output.

→ The term combination logic mean combining of two or more logic gates to form a required function where the output at a given time depends only on the input.

→ The required output data is obtained from this process by transforming the binary information given at the input.



→ A combinational circuit operates in three steps-

(a) It accepts n- different inputs.

(b) The combination of gates operates on the inputs.

(c) "m" different output are produced as per requirment.

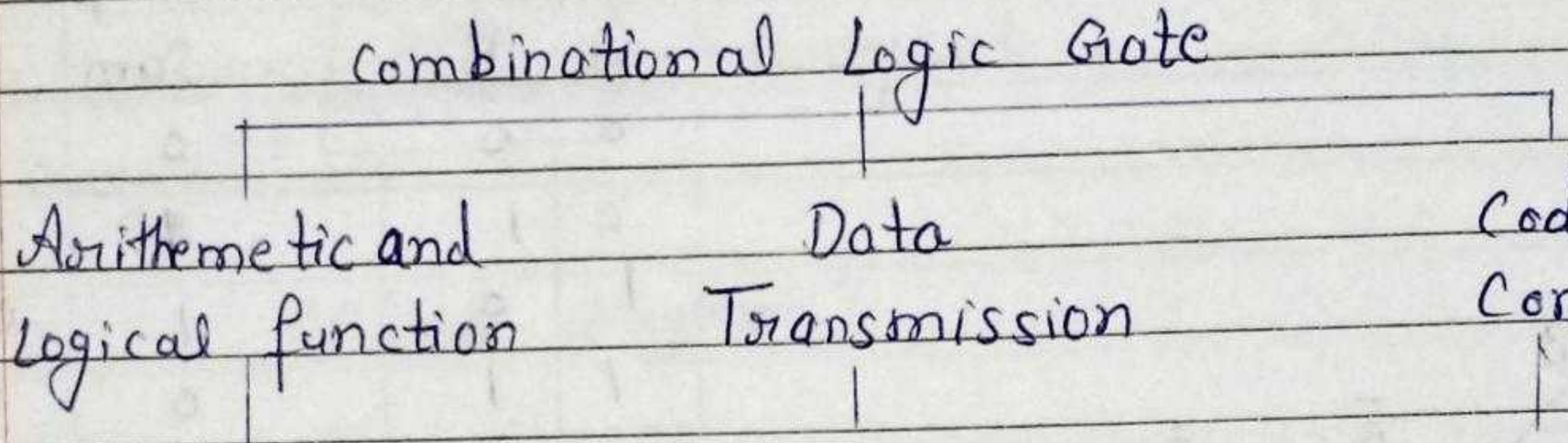
characteristics →

1. The output of combinational circuit at any instant of time, depends only on the level present at input terminals.

2. The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.

3. A combinational circuit can have an n number of inputs and m number of outputs.

Classifications →



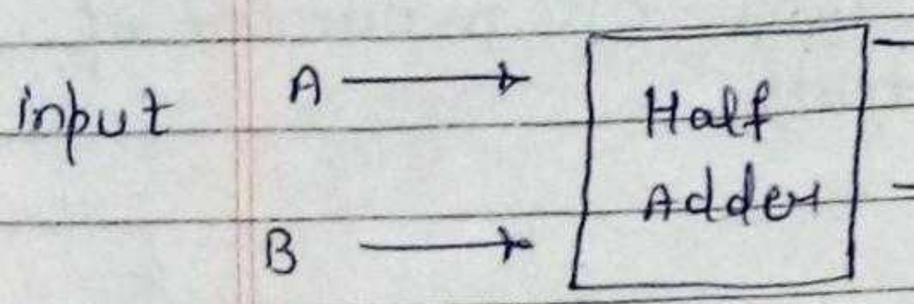
"Adder" → An adder is a digitial circuit that Performs addition of two numbers-

1. Half Adder
2. Full Adder

"Half Adder"

Half adder is a combinational logic circuit with two inputs and two outputs. The half adder is designed to add two single bit binary number A and B.

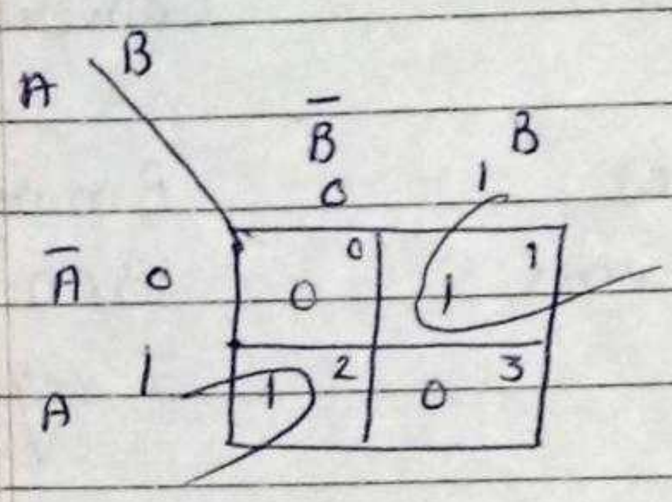
It is the basic building block for addition of two single bit numbers.
This circuit has two outputs carry and sum.



(Block diagram)

Truth Table →

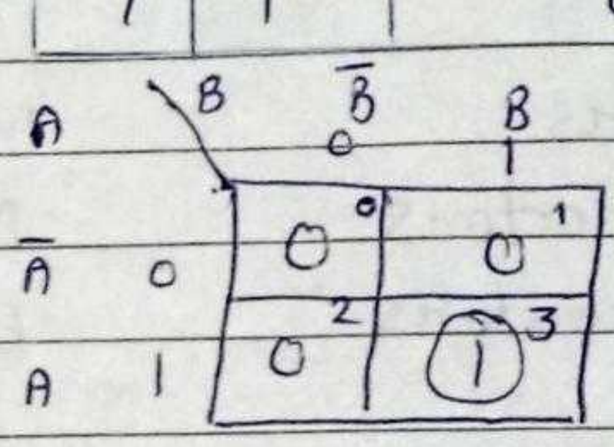
Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



K-map for Sum output

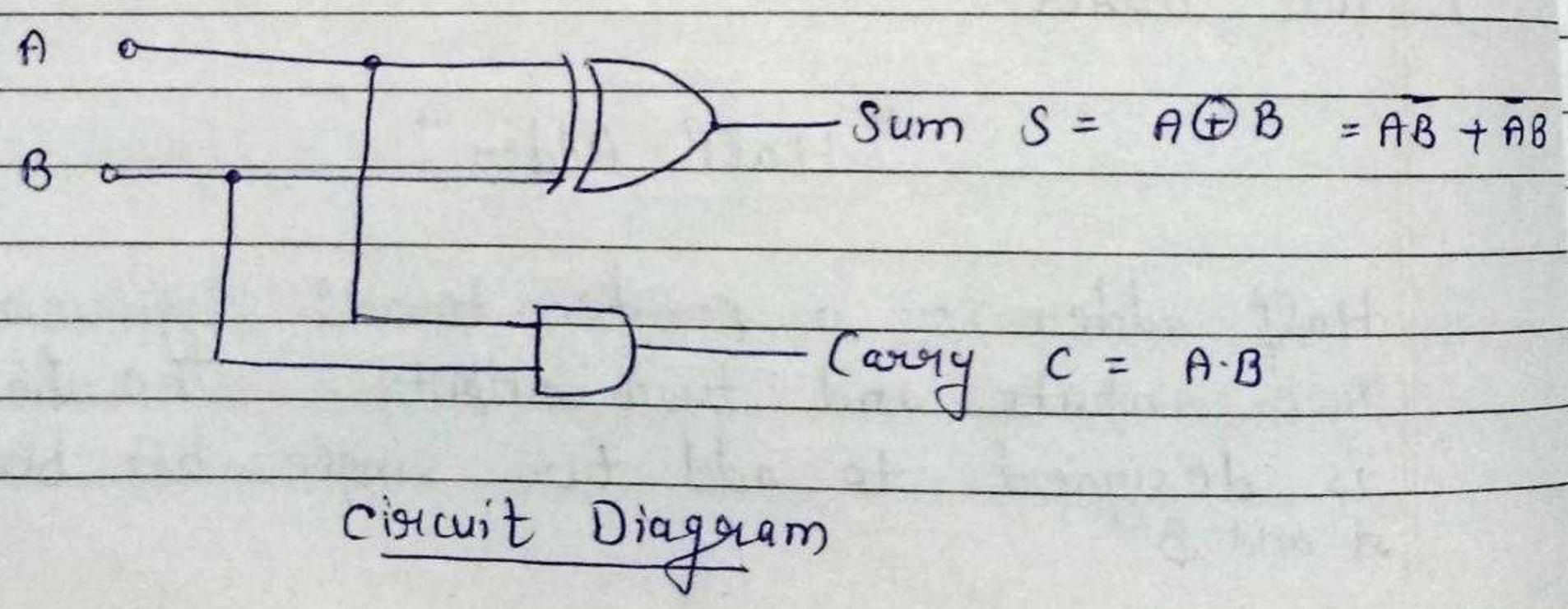
Sum = $A\bar{B} + \bar{A}B$

= $A \oplus B$



K-map for carry output

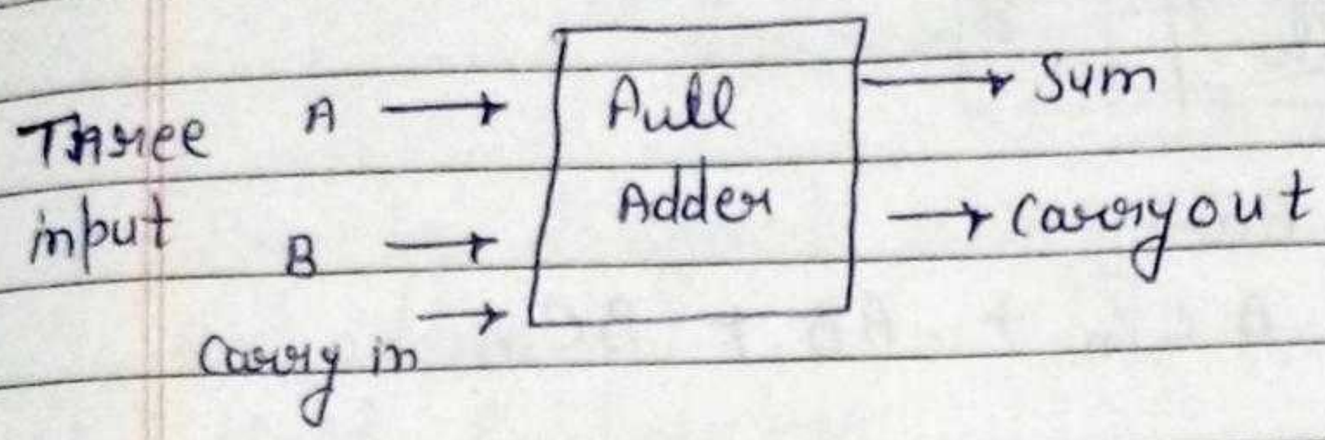
Carry = AB



Circuit Diagram

"Full Adder"

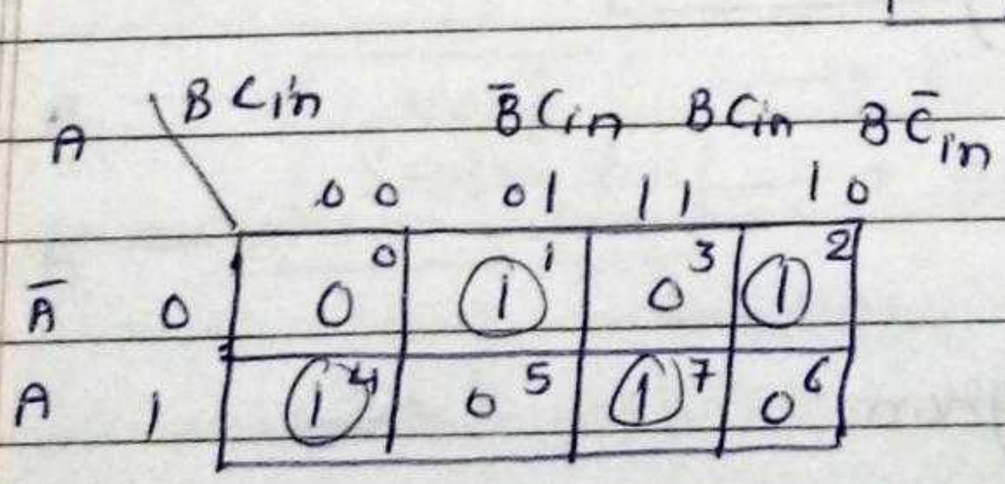
To overcome the drawback of Half adder circuit a 3 single bit adder circuit called Full Adder is developed.
It can add two one-bit numbers A and B and Carry C_{in}. The full adder is a three input and two output combinational circuit.



(Block diagram)

Truth Table -

Inputs			Outputs	
A	B	C_{in}	S	C_o
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



K-map for Sum →

$$Sum = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + AB C_{in} + A\bar{B}\bar{C}_{in}$$

$$= \bar{B}(A\bar{C}_{in} + \bar{A}C_{in}) + B(AC_{in} + \bar{A}\bar{C}_{in})$$

$$= \bar{B}(A \oplus C_{in}) + B(\overline{A \odot C_{in}})$$

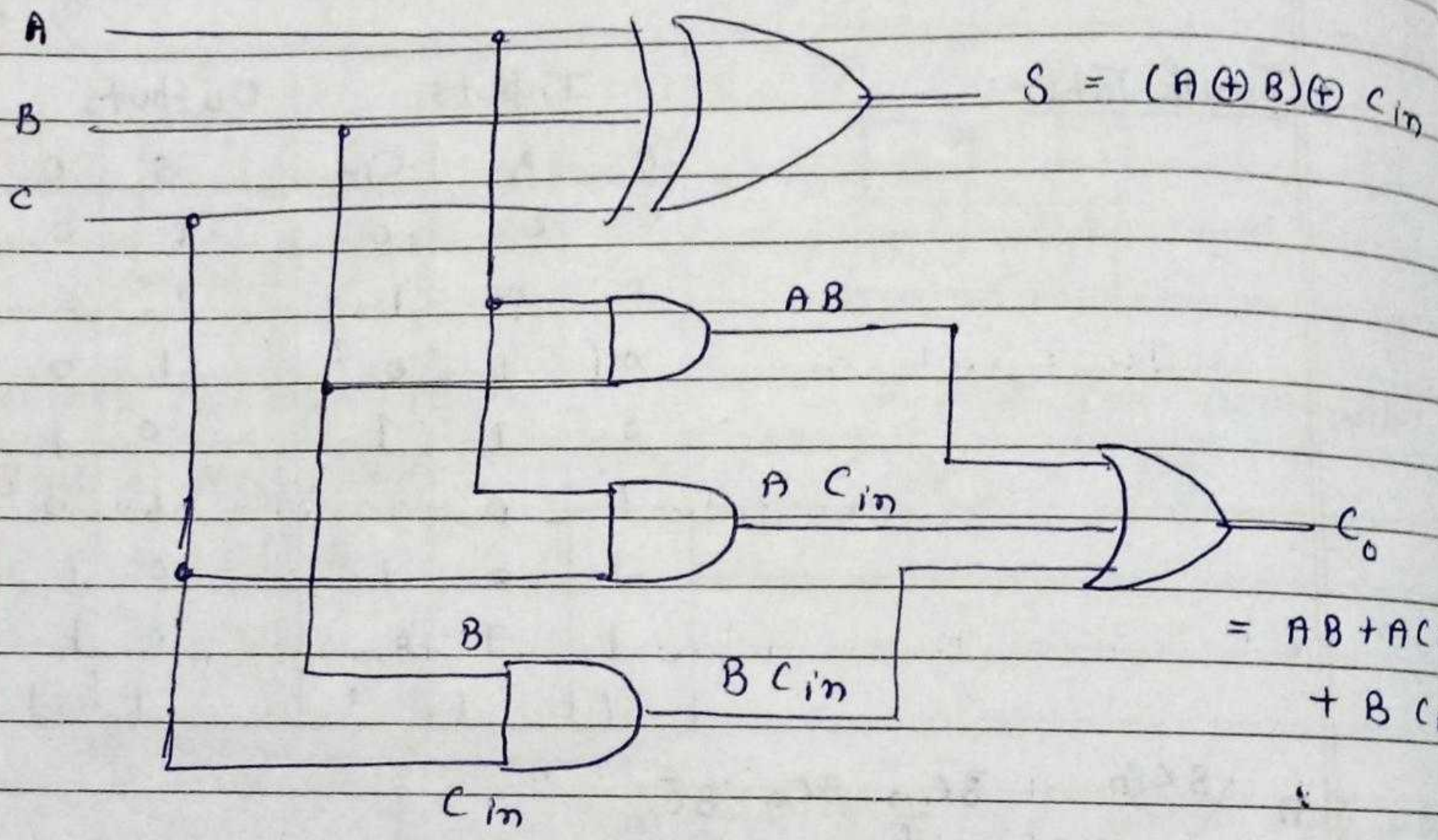
$$= \bar{B}(A \oplus C_{in}) + B(A \oplus C_{in})$$

$$= A \oplus B \oplus C_{in}$$

A \ B	00	01	11	10
$\bar{A}$ 0	0 (0)	0 (1)	1 (3)	0 (2)
A 1	0 (4)	1 (5)	1 (7)	1 (6)

K - map for Carry

$$Carry = AC_{in} + AB + BC_{in}$$



Logic Diagram

Application of Full Adder :-

1. The full adder acts as the basic building block of the 4 bit / 8 bit binary / BCD adder ICs such as 7483.
2. It is used in the digital diary.
3. It is used in digital computers.
4. It is used in arithmetic logic unit (ALU).

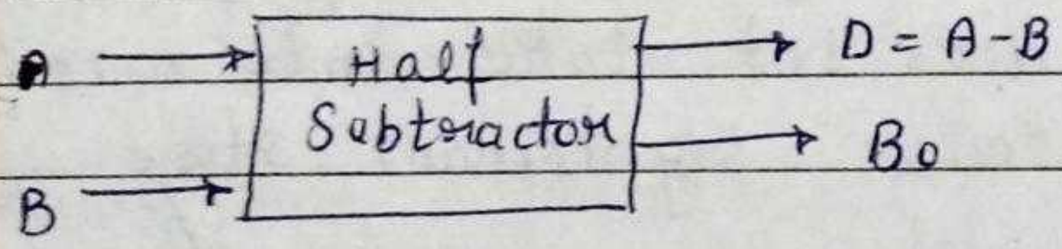
"Subtractor"

In electronic, a subtractor can be designed using the same approach as that of an adder.

1. Half Subtractor
2. Full Subtractor.

"Half Subtractor"

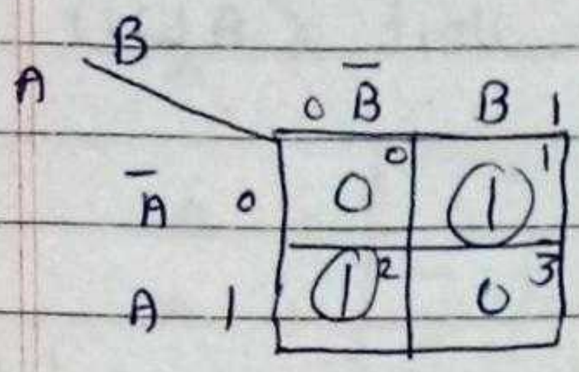
1. Half subtractor is a combination circuit with two inputs and two output (difference and borrow).
2. It produces the difference blw the two binary bits at the input and also produces a output (Borrow) to indicate if a 1 has been borrowed In the subtraction (A-B), A is called as Minuend bit and B is called Subtrahend bit.



(Block diagram)

Truth table -

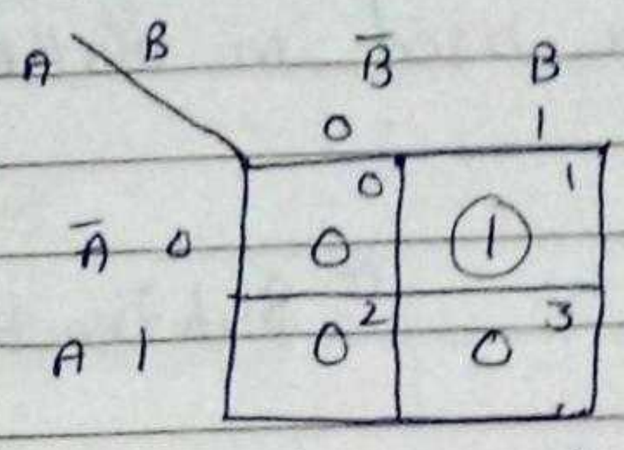
Inputs		Outputs	
A	B	Difference D	Borrow B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



K-map for difference -

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

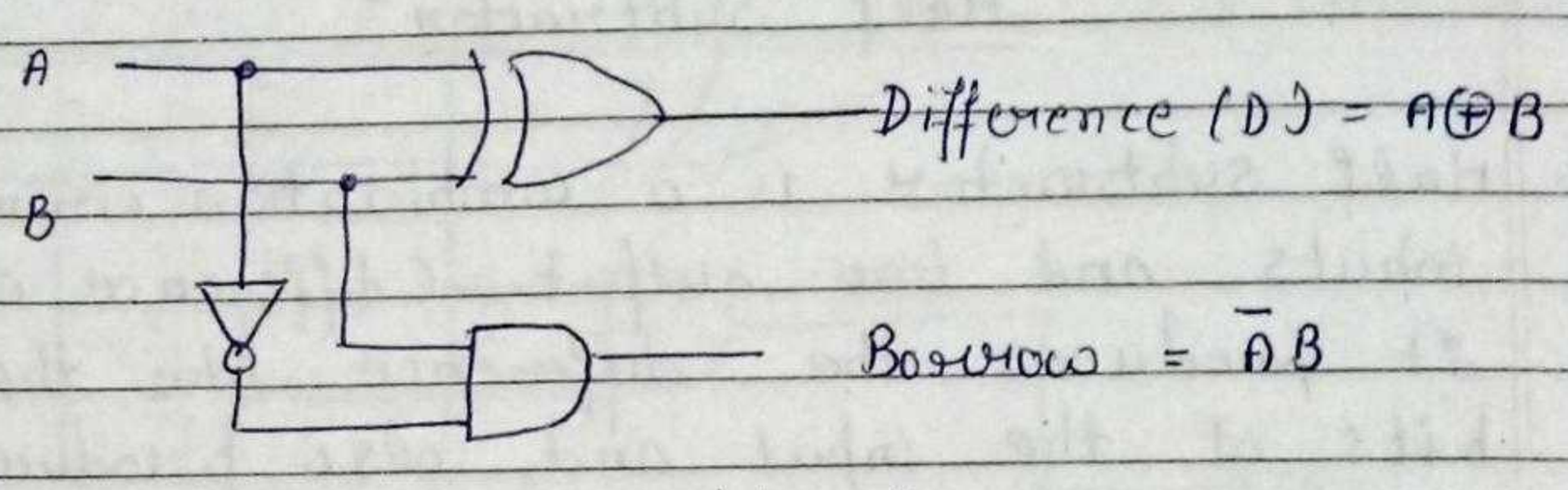
= $A \oplus B$



K-map for Borrow.

Borrow = $\bar{A}B$

Logic Diagram -



Disadvantage of Half subtractor -

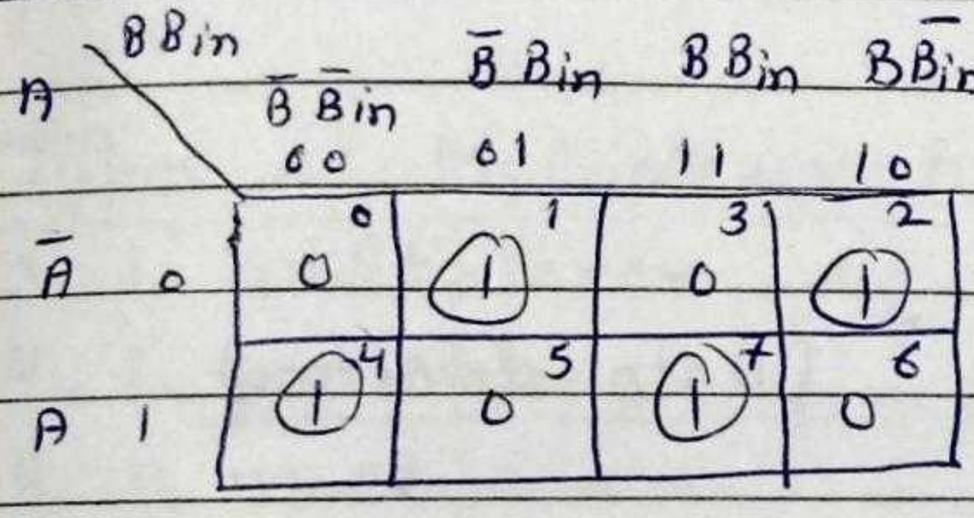
Half subtractor can only perform the subtraction of two binary bits. But while performing the substraction, it does not take into account the borrow of the lower significant stage.

"Full subtractor"

The disadvantage of a half subtractor is overcome by full subtractor.
The full subtractor is a combinational circuit with three inputs A, B, C and two output D and C'. A is the minuend, B is subtrahend, C is the borrow produced by the previous stage, D is the difference output and C' is the borrow output.

Truth Table -

Inputs			Outputs	
A	B	Bin	(A - B - Bin)	B_o
(minuend)	(subtrahend)	Previous borrow		
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 difference =

$\bar{A} B \bar{B}_{in} + A \bar{B} \bar{B}_{in} + \bar{A} \bar{B} B_{in} + AB B_{in}$

$\bar{B} (A \bar{B}_{in} + \bar{A} B_{in}) + B (AB_{in} + \bar{A}\bar{B}_{in})$

$\bar{B} (A \oplus B_{in}) + B (\overline{A \odot B_{in}})$

$\bar{B} (A \oplus B_{in}) + B (A \oplus B_{in})$

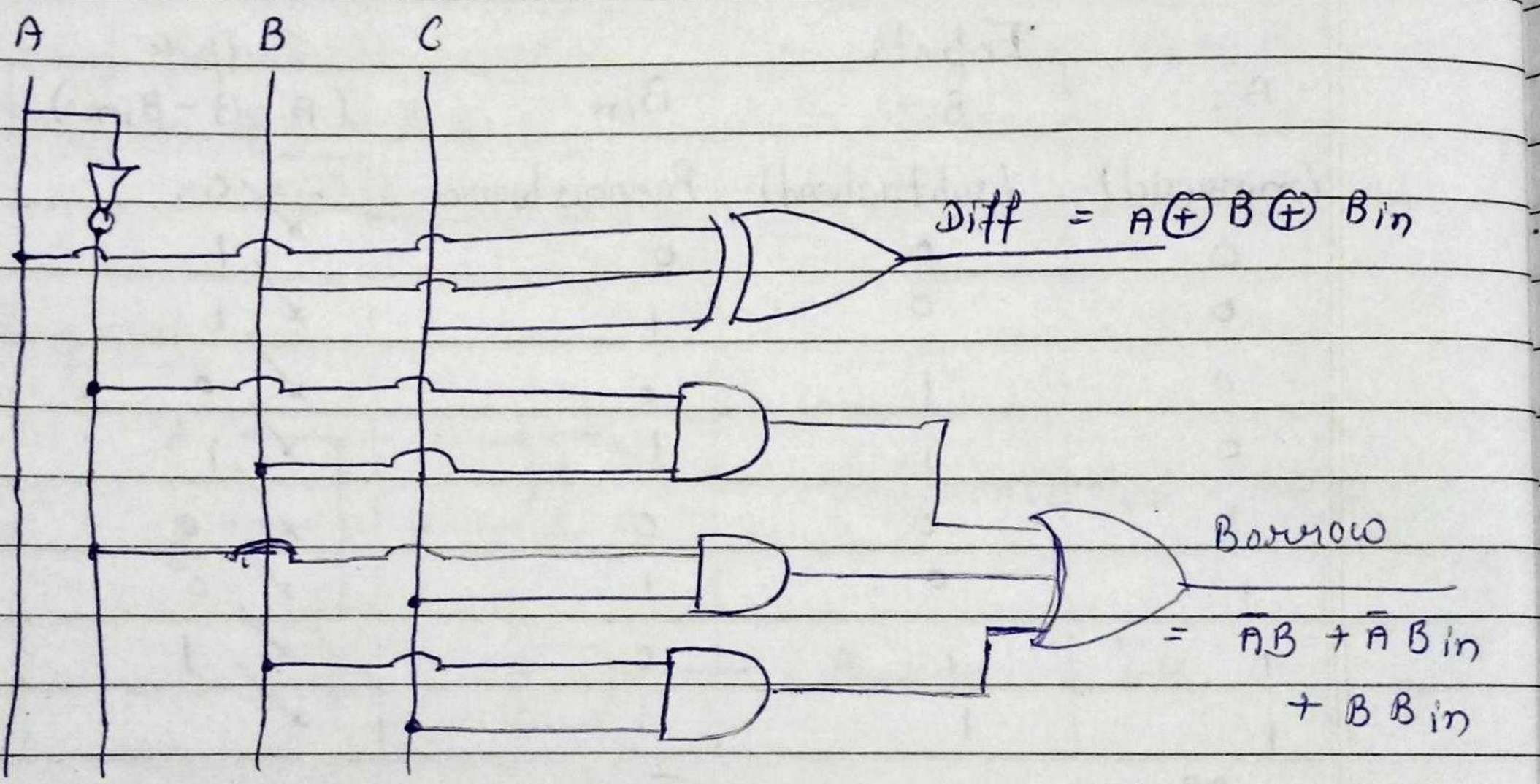
$A \oplus B \oplus B_{in}$

A \ $B B_{in}$	$\bar{B}\bar{B}_{in}$	$\bar{B}B_{in}$	BB_{in}	$B\bar{B}_{in}$
	00	01	11	10
$\bar{A}$ 0	0 (0)	1 (1)	1 (3)	1 (2)
A 1	0 (4)	0 (5)	1 (7)	0 (6)

K-map for Borrow -

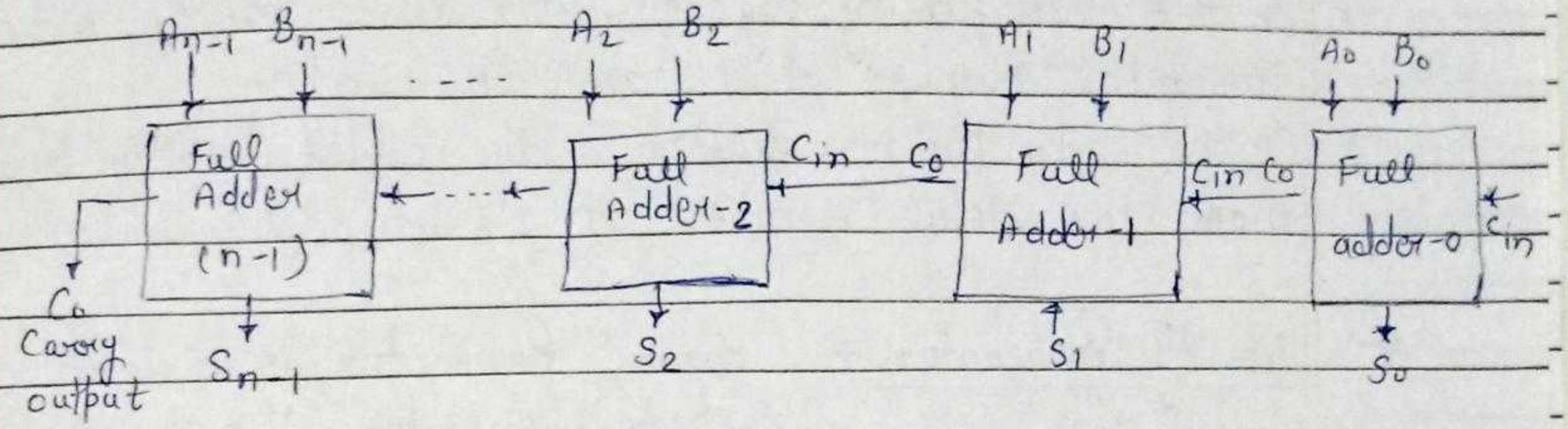
$$\text{Borrow} = \bar{A}B_{in} + \bar{A}B + BB_{in}$$

Logic Diagram →



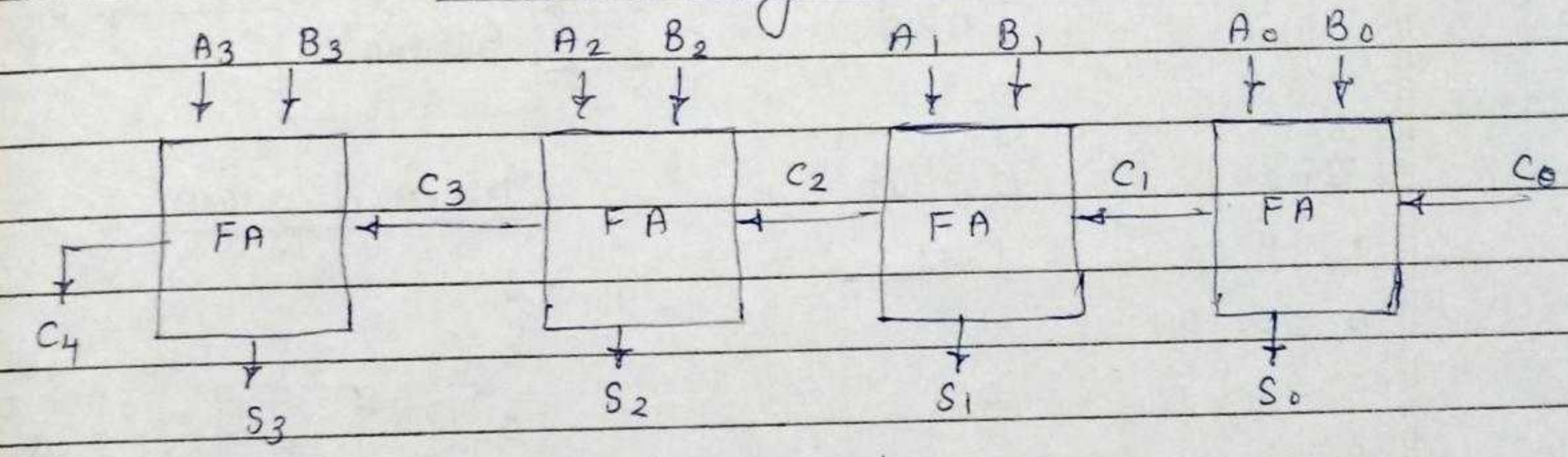
"n-bit Parallel Adder"

* The full adder is capable of adding only two single digit binary numbers along with a carray output
→ To add two n-bit binary numbers we need to use the n-bit parallel adder.
→ It uses a number of full adders in cascade. The carry output of the previous full adder is connected to carry input of the next full adder



Block diagram of n-bit parallel adder

"4-bit Binary Parallel Adder"



4-bit Adder

$C_0 = 0$

→ A_0 and B_0 represent the LSB of the four bit words A and B.

Binary Adder

This is also called Ripple Carry adder, because of the construction with full adders are connected in cascade.

Subscript i :	3	2	1	0	
Input carry	0	0	1	0	C_i
	1	0	1	1	A_i
	0	0	1	1	B_i
Sum	1	1	1	0	S_i
output carry	0	0	1	1	C_{i+1}

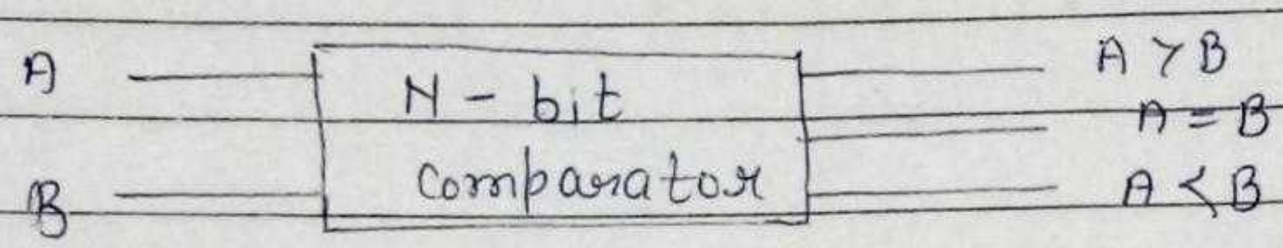
Magnitude Comparator

It is a combinational circuit that compares to numbers and determine their relative magnitude.
The output of Comparator is usually 3 binary variables indicating

$A > B$

$A = B$

$A < B$



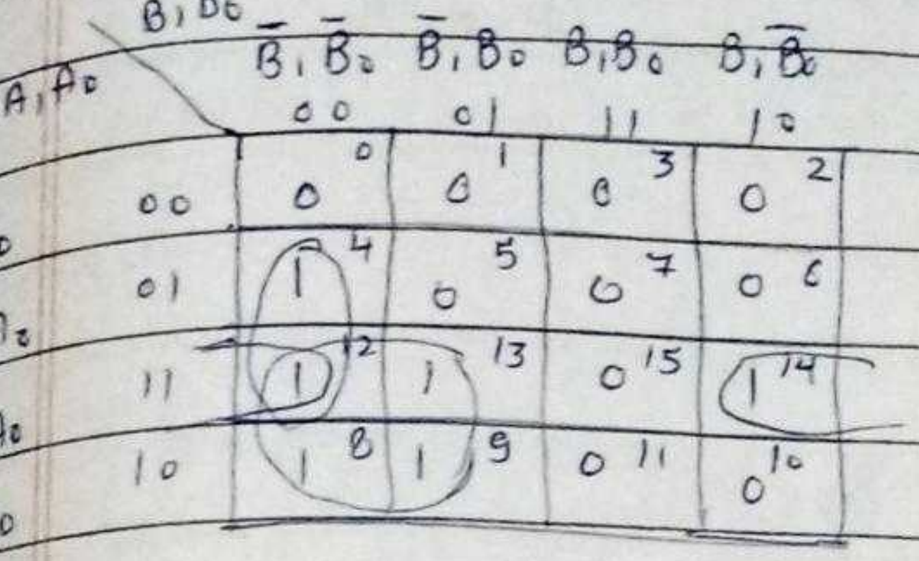
2-Bit Magnitude Comparator

A bit comparator used to compare two binary numbers each of two bits is called 2-bit magnitude comparator. It consists of four inputs and three outputs to generate less than, equal to and greater than Between two binary numbers.

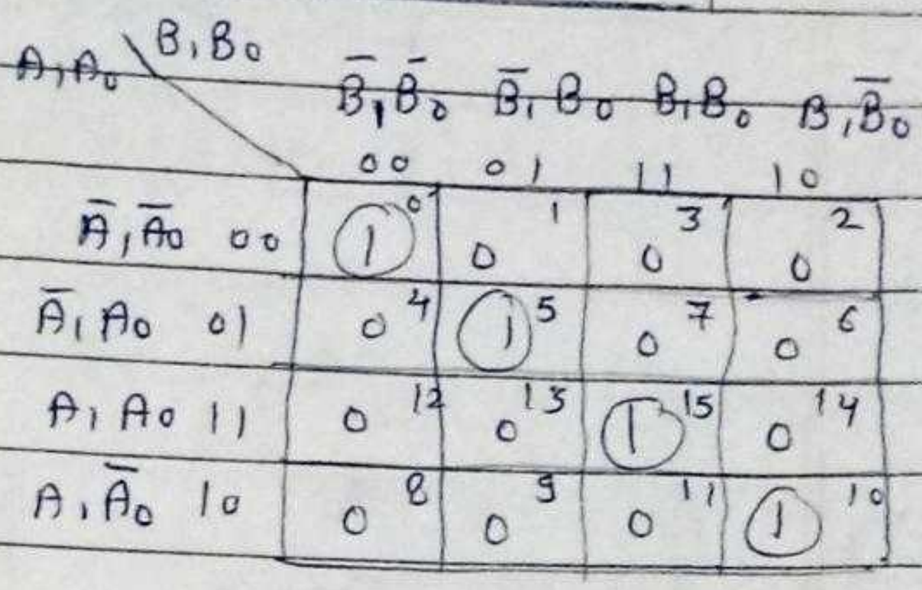
2-Bit Magnitude comparator ($2^4 = 16$)

	Input				output			
	A_0	A_0	B_1	B_0	$A > B$	$A = B$	$A < B$	
0	0	0	0	0	0	1	0	$00 = 00$
1	0	0	0	1	0	0	1	$00 < 01$
2	0	0	1	0	0	0	1	$00 < 10$
3	0	0	1	1	0	0	1	$00 < 11$
4	0	1	0	0	1	0	0	$01 > 00$
5	0	1	0	1	0	1	0	$01 = 01$
6	0	1	1	0	0	0	1	$01 < 10$
7	0	1	1	1	0	0	1	$01 < 11$
8	1	0	0	0	1	0	0	$10 > 00$

	A_1	A_0	B_1	B_0	$A>B$	$A=B$	$A<B$	
9	1	0	0	1	1	0	0	$10 > 01$
10	1	0	1	0	0	1	0	$10 = 10$
11	1	0	1	1	0	0	1	$10 < 11$
12	1	1	0	0	1	0	0	$11 > 00$
13	1	1	0	1	1	0	0	$11 > 01$
14	1	1	1	0	1	0	0	$11 > 10$
15	1	1	1	1	0	1	0	$11 = 11$



$Y_1(A>B) = A_1\bar{B}_1 + A_0\bar{B}_1\bar{B}_0 + A_1A_0\bar{B}_0$

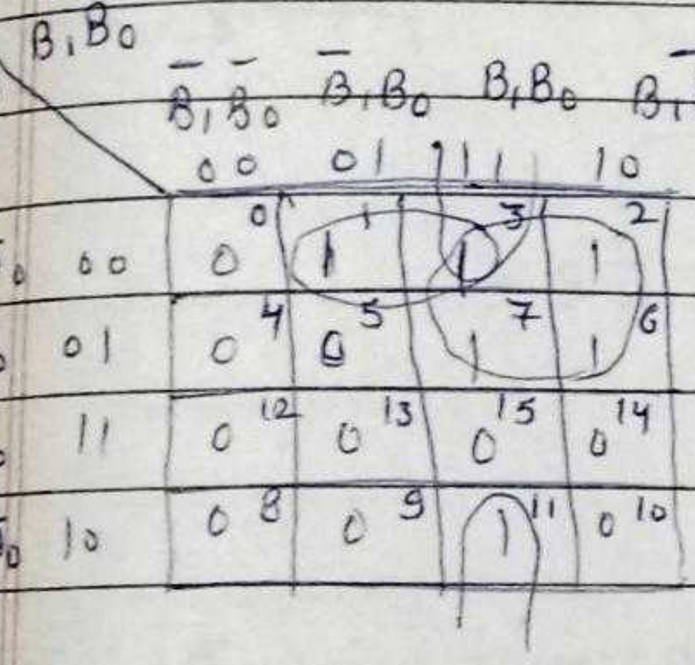


$Y_2(A=B) = \bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0\bar{B}_1B_0 + A_1A_0B_1B_0 + A_1\bar{A}_0B_1\bar{B}_0$

$= \bar{A}_1\bar{B}_1(\bar{A}_0\bar{B}_0 + A_0B_0) + A_1B_1(A_0B_0 + \bar{A}_0\bar{B}_0)$

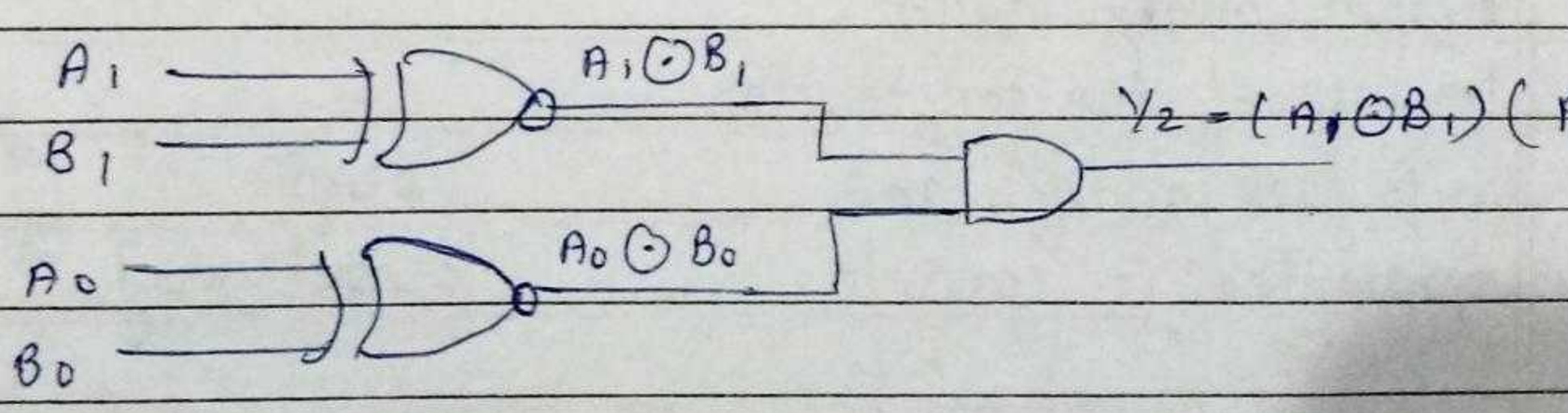
$= (\bar{A}_1\bar{B}_1 + A_1B_1)(\bar{A}_0\bar{B}_0 + A_0B_0)$

$= (A_1 \odot B_1)(A_0 \odot B_0)$



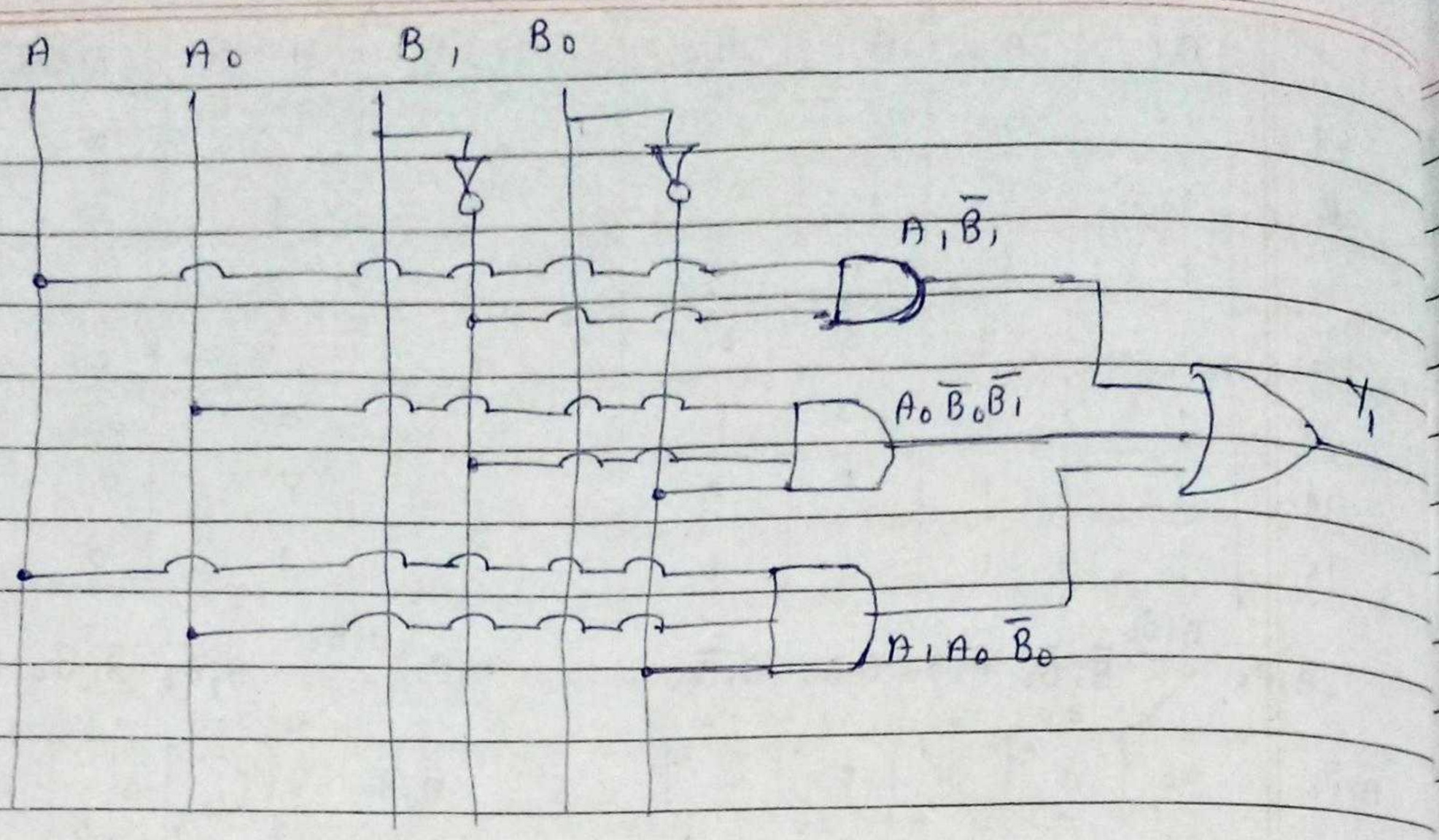
$Y_3(A<B) = \bar{A}_1B_1 + \bar{A}_1\bar{A}B_0 + \bar{A}_0B_1B_0$

For Y_2 →

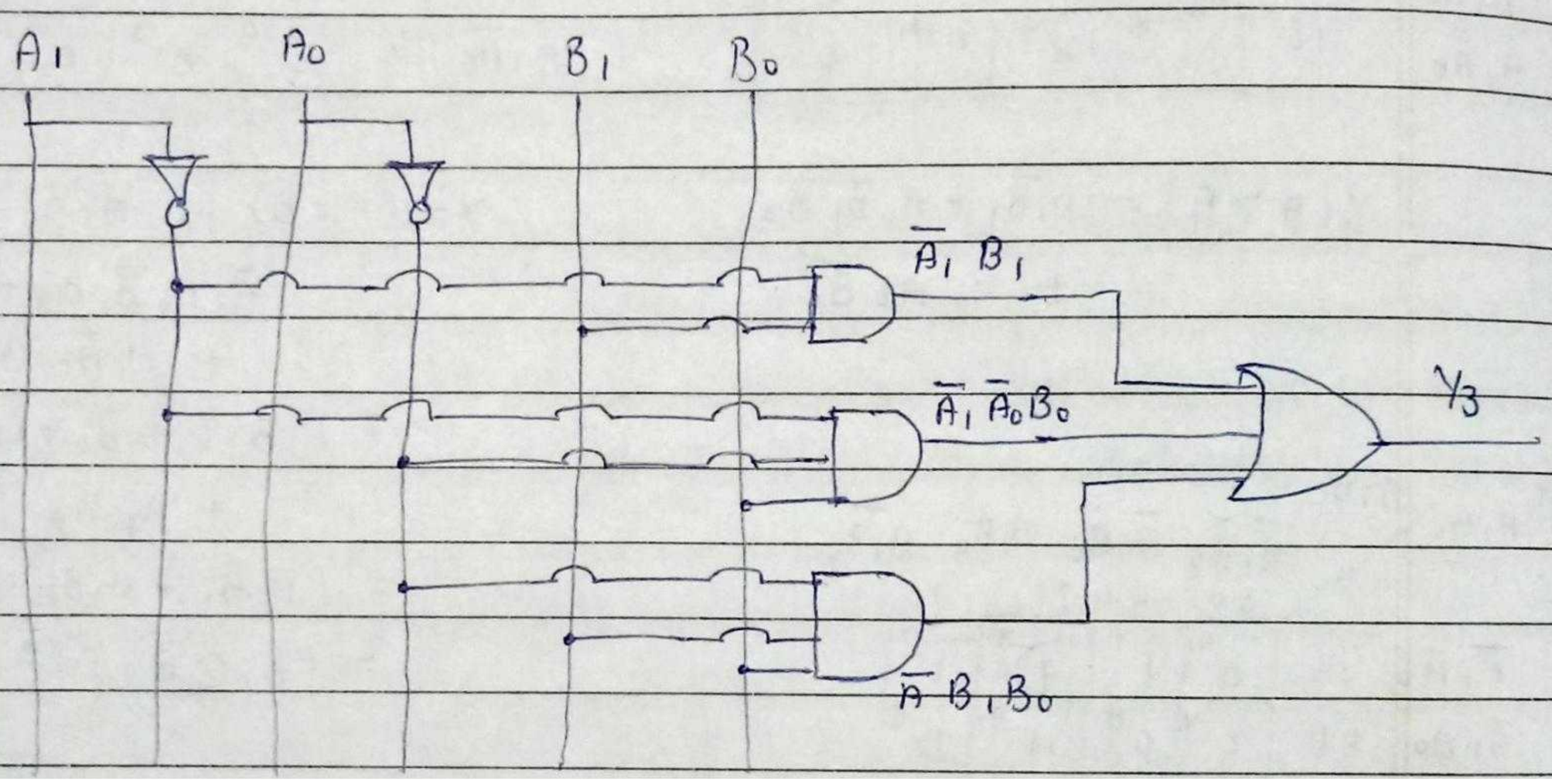


Date :
Page :

For Y_1



For Y_3



"Multiplexer" (Data Selector)

A multiplexer is a digital circuit which selects one of the n data inputs and routes to the output. The selection of one of n inputs is done by the select inputs.

Usually there are 2^n input lines and n selection

lines whose bit combinations determine which input line is selected.

For Ex- for 2 to 1 multiplexer if selection S is zero then I_0 has the path to output and if S is one I_1 has the path to output.

Advantage of Multiplexer -

1. It reduces the number of wires.
2. So it reduces the circuit complexity and cost
3. We can implement many combinational circuit using MUX.
4. It simplifies the logic design.
5. It does not need the Kmap and simplification.

Necessity of Multiplexer +

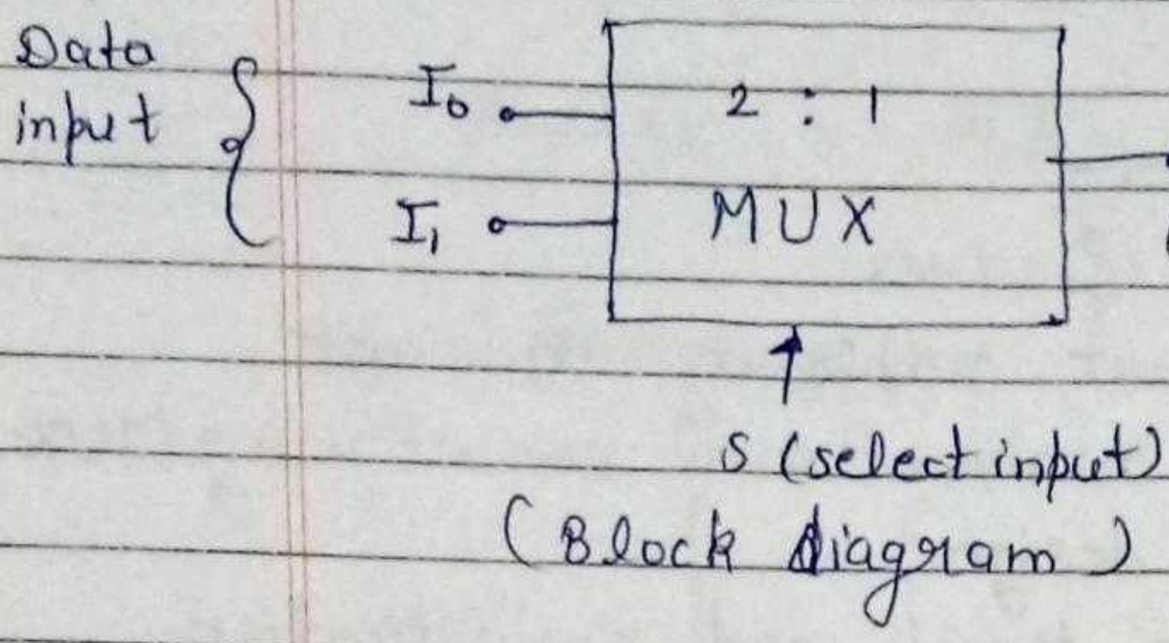
+ In most of the electronic System, the digitial data is available on more than one lines. It is necessary to route this data over a single line.
+ Multiplexer ~~red~~ improves the reliability of the digital circuits because it reduces the number of external wired connections.

Types of Multiplexer +

1. 2:1 multiplexer
2. 4:1 multiplexer
3. 8:1 multiplexer
4. 16:1 multiplexer
5. 32:1 multiplexer

"2:1 multiplexer" (2^1 = 1 select line)

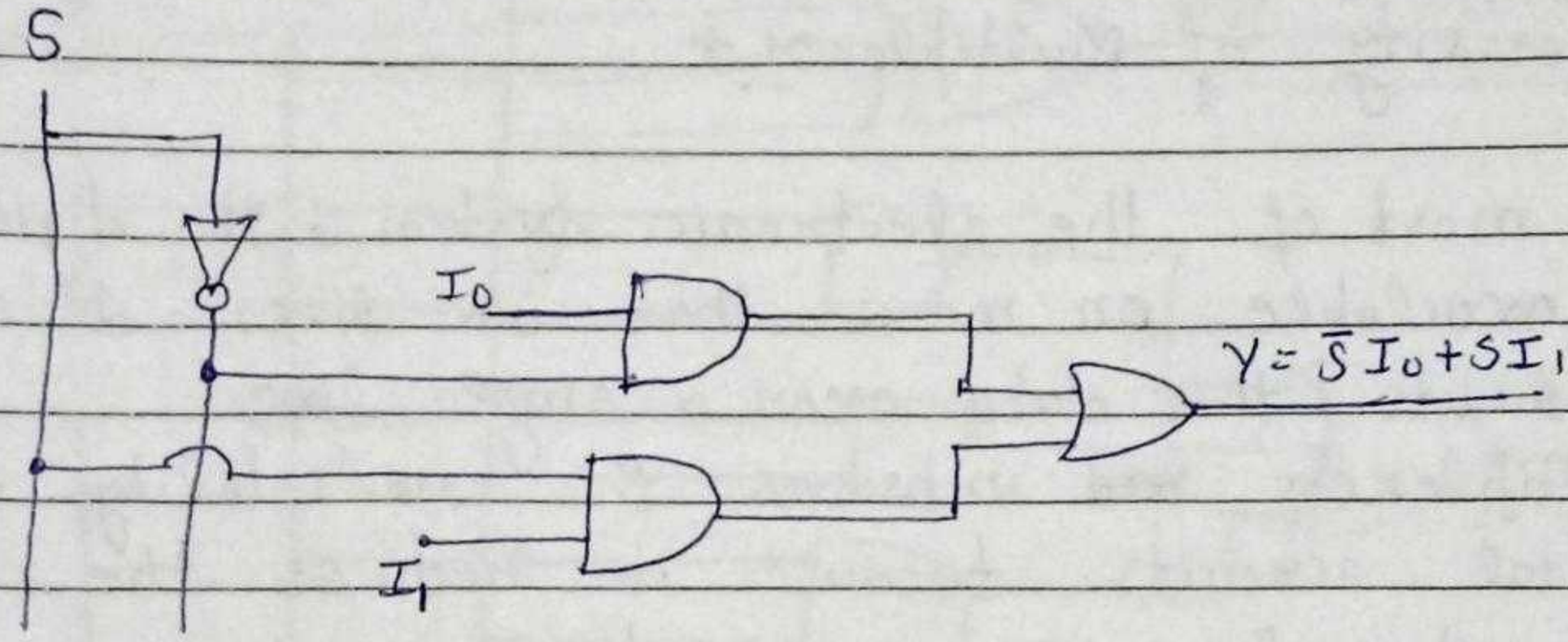
It has two data inputs I_0 and I_1, one select input S and one output.



(Block diagram)

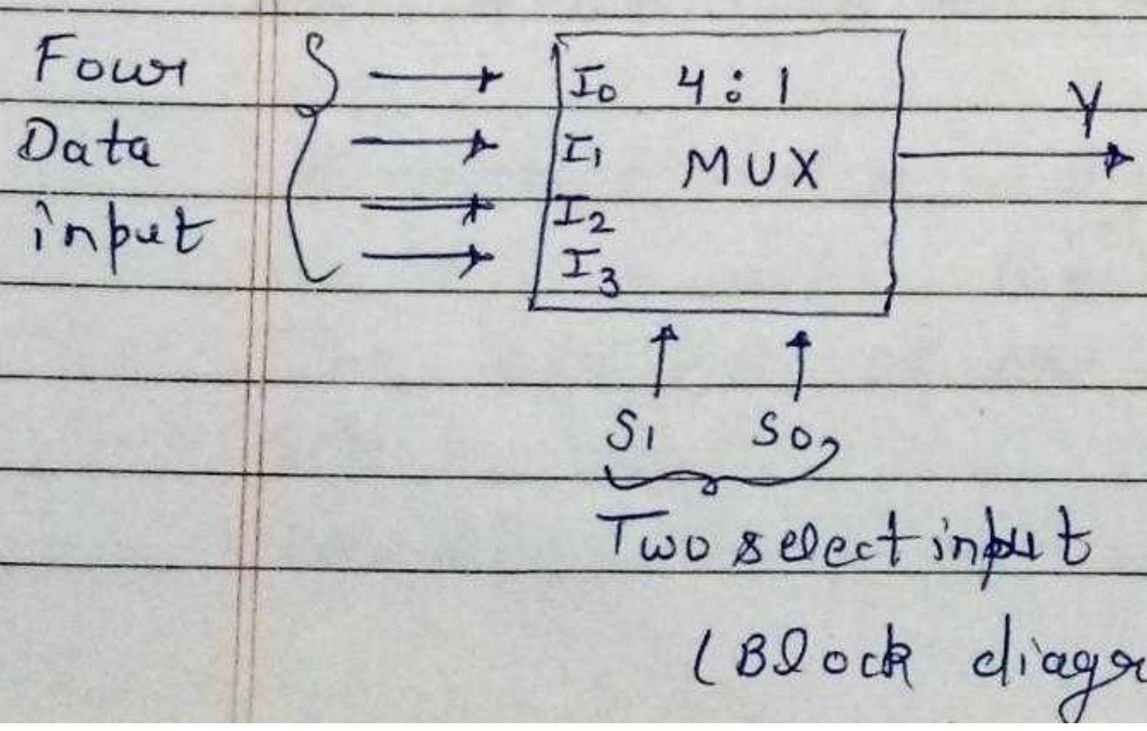
Truth Table	
Select input S	Output Y
0	I_0
1	I_1

$Y = \bar{S} I_0 + S I_1$



Realization of 2:1 MUX using gates.

"4:1 multiplexer" (2^2 = 2 select line)

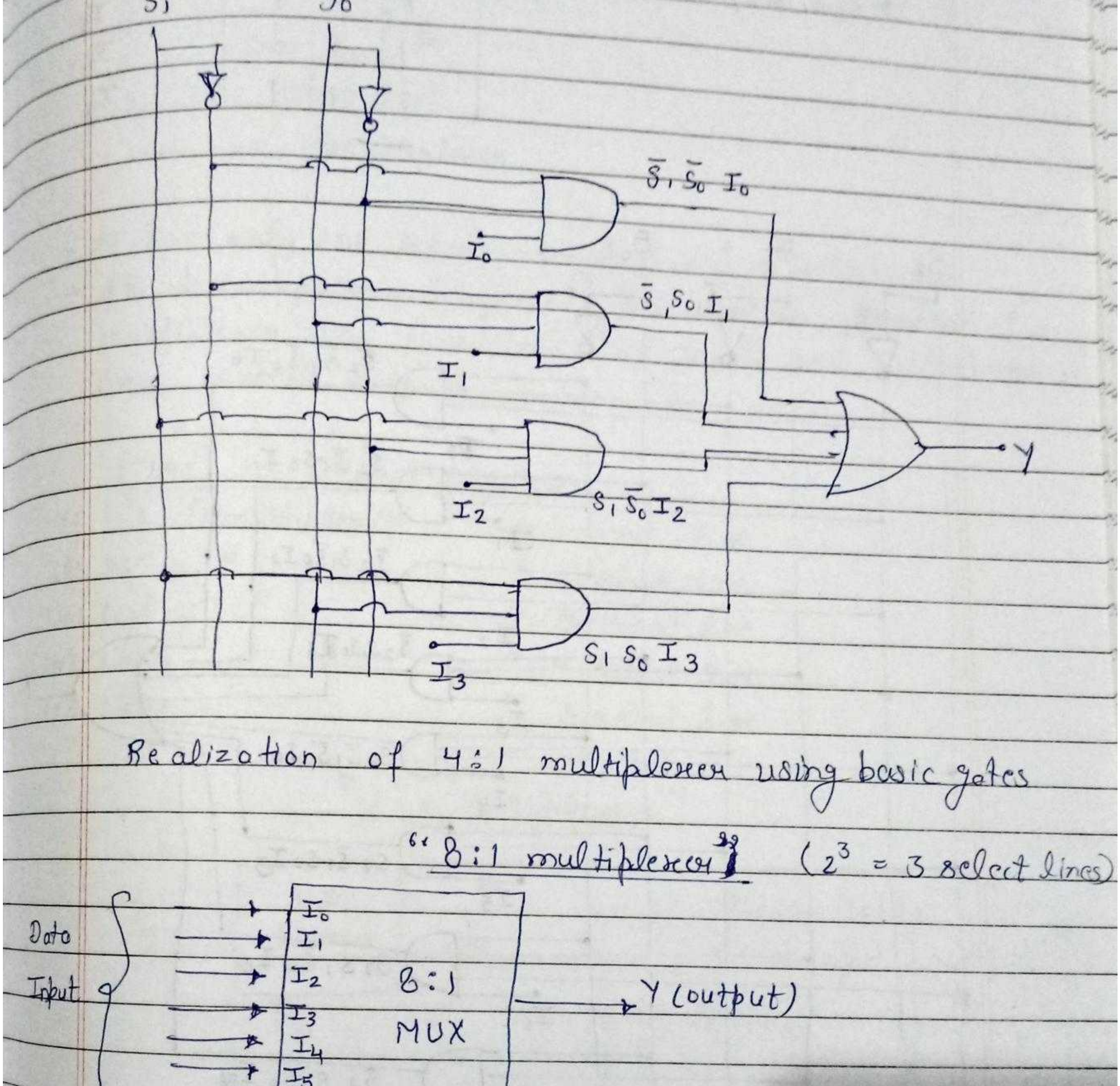


(Block diagram)

Truth Table :-

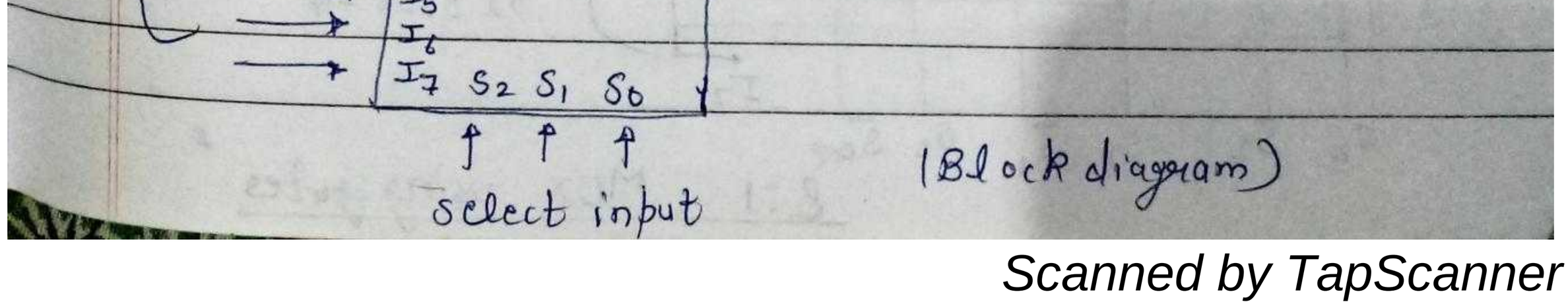
Select input		Output select
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Y = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$$



Realization of 4:1 multiplexer using basic gates

"8:1 multiplexer" (2^3 = 3 select lines)

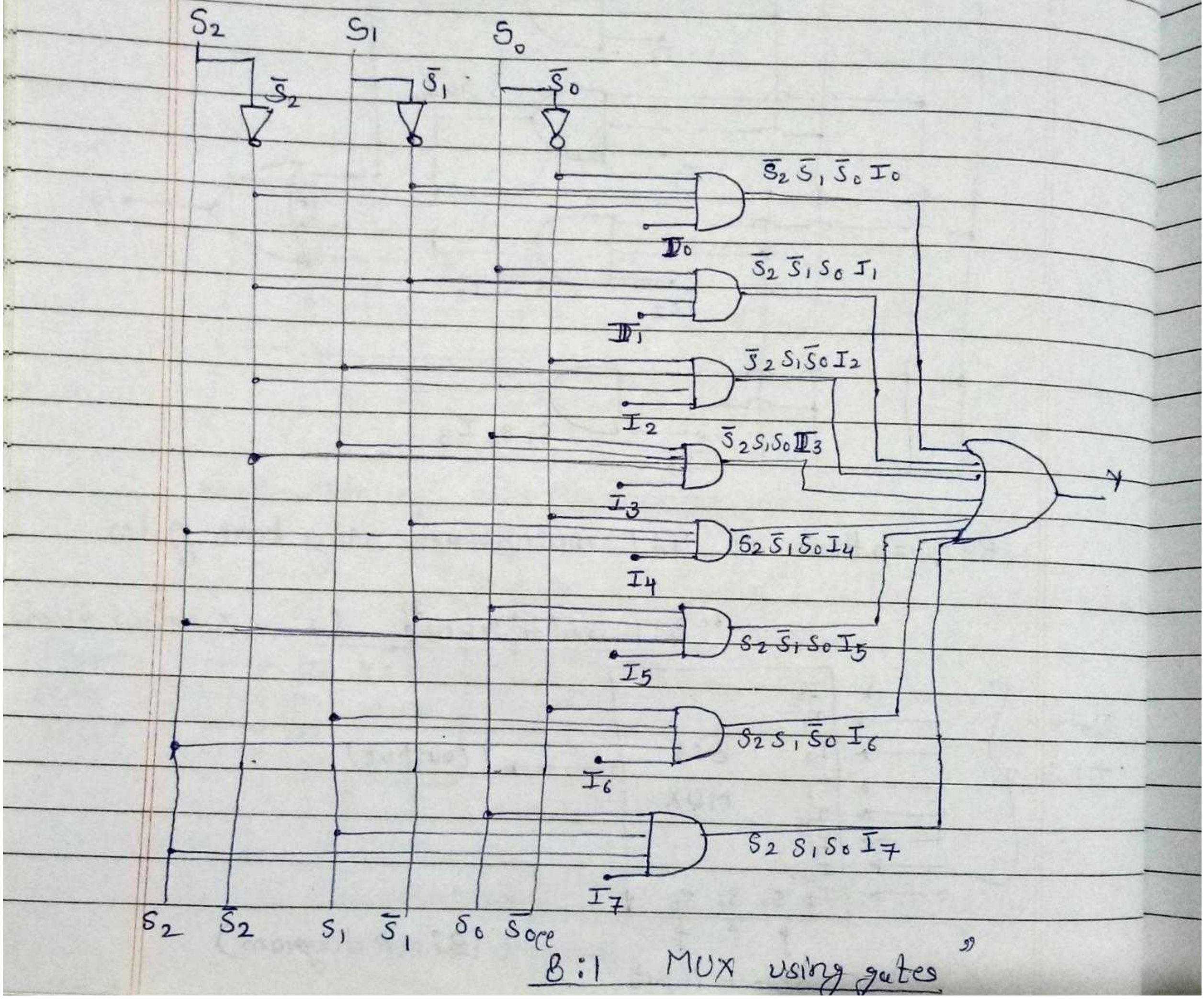


(Block diagram)

Truth Table →

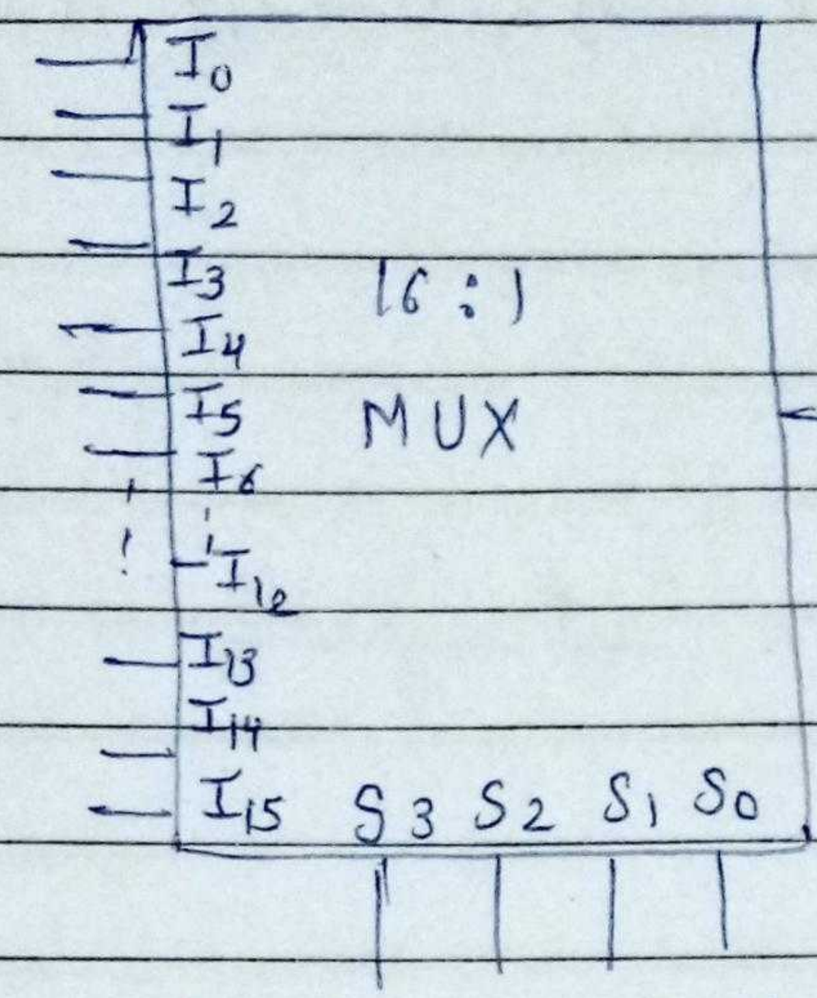
Select input			Output select
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

$$Y = \bar{S}_2\bar{S}_1\bar{S}_0 I_0 + \bar{S}_2\bar{S}_1 S_0 I_1 + \bar{S}_2 S_1\bar{S}_0 I_2 + \bar{S}_2 S_1 S_0 I_3 + S_2\bar{S}_1\bar{S}_0 I_4 + S_2\bar{S}_1 S_0 I_5 + S_2 S_1\bar{S}_0 I_6 + S_2 S_1 S_0 I_7$$



8:1 MUX using gates

"16:1 Multiplexer" $2^4 \rightarrow$ 4 Select lines



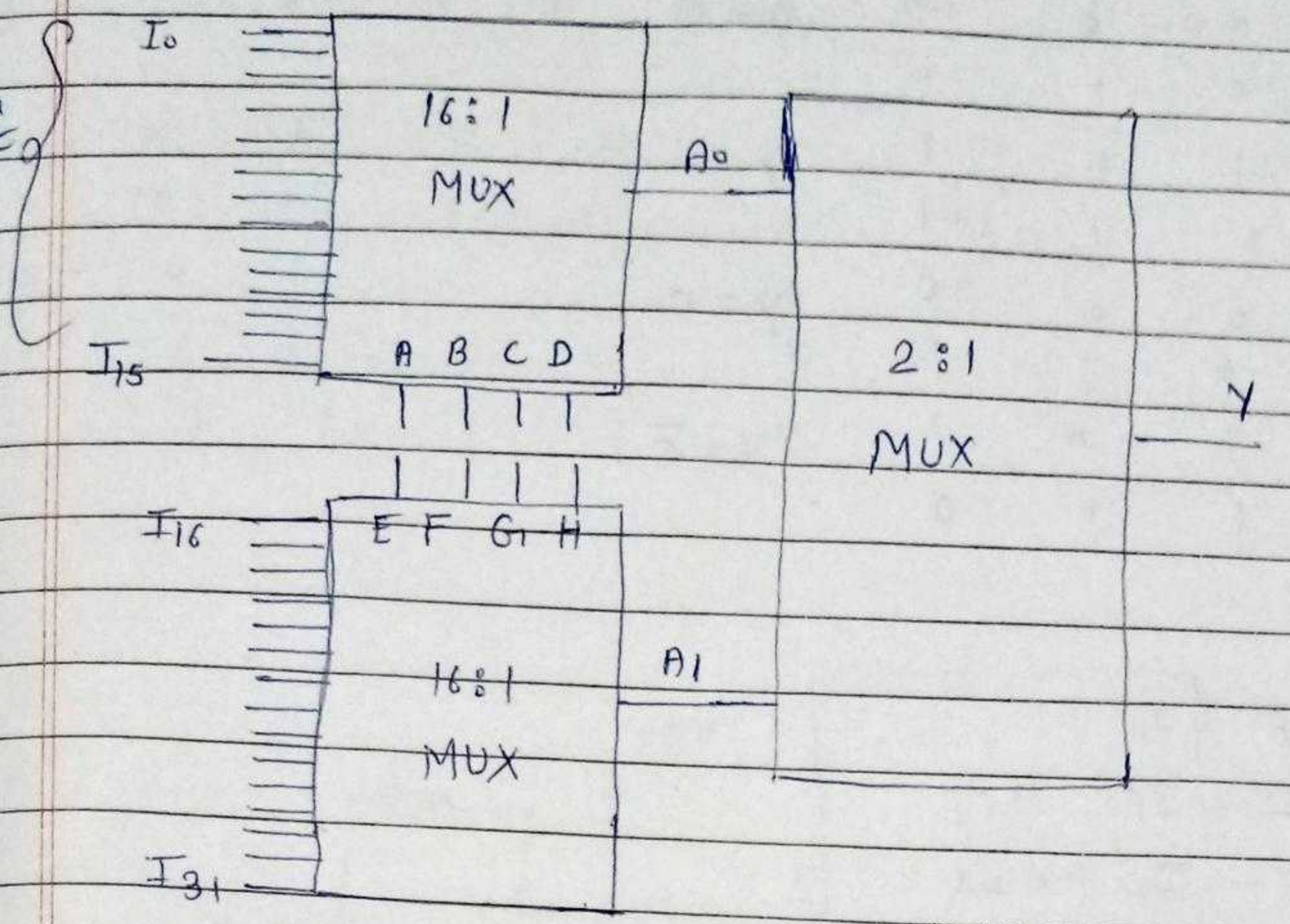
(Block diagram)

select input				output select
S_3	S_2	S_1	S_0	Y
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5
0	1	1	0	I_6
0	1	1	1	I_7
1	0	0	0	I_8
1	0	0	1	I_9
1	0	1	0	I_{10}
1	0	1	1	I_{11}
1	1	0	0	I_{12}
1	1	0	1	I_{13}
1	1	1	0	I_{14}
1	1	1	1	I_{15}

$$Y = \bar{S}_3\bar{S}_2\bar{S}_1\bar{S}_0 I_0 + \bar{S}_3\bar{S}_2\bar{S}_1 S_0 I_1 + \bar{S}_3\bar{S}_2 S_1\bar{S}_0 I_2 + \bar{S}_3\bar{S}_2 S_1 S_0 I_3 + \bar{S}_3 S_2\bar{S}_1\bar{S}_0 I_4 + \bar{S}_3 S_2\bar{S}_1 S_0 I_5 + \bar{S}_3 S_2 S_1\bar{S}_0 I_6 + \bar{S}_3 S_2 S_1 S_0 I_7 + S_3\bar{S}_2\bar{S}_1\bar{S}_0 I_8 + S_3\bar{S}_2\bar{S}_1 S_0 I_9 + S_3\bar{S}_2 S_1\bar{S}_0 I_{10} + S_3\bar{S}_2 S_1 S_0 I_{11} + S_3 S_2\bar{S}_1\bar{S}_0 I_{12} + S_3 S_2\bar{S}_1 S_0 I_{13} + S_3 S_2 S_1\bar{S}_0 I_{14} + S_3 S_2 S_1 S_0 I_{15}$$

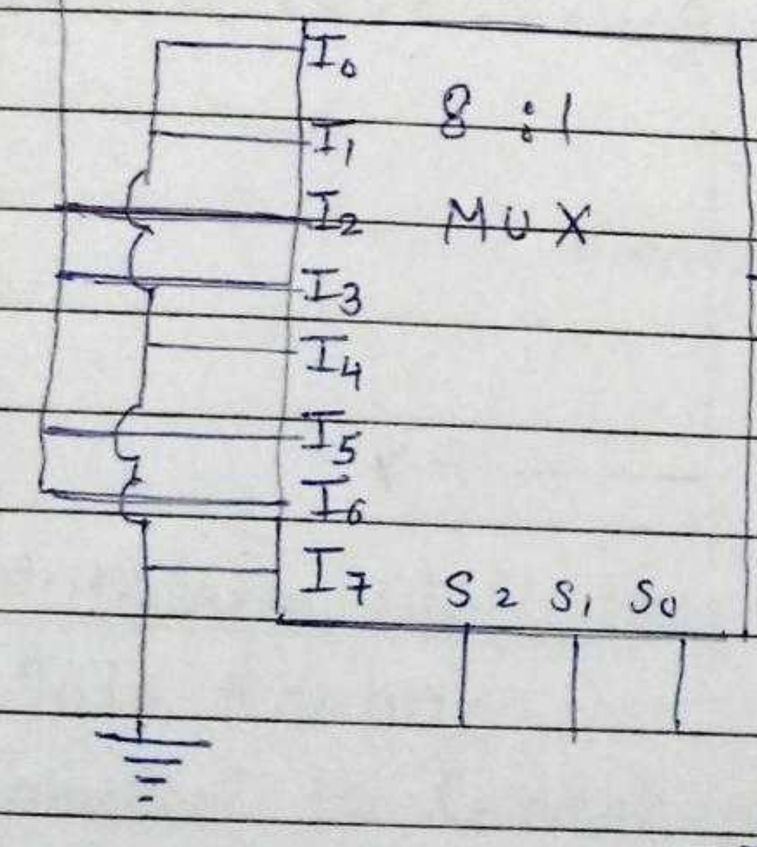
Date:
Page:

Design a 32:1 Multiplexer using 16:1 Multiplexer and 2:1 multiplex



Implement the following using 8:1 , 4:1 multiplexer.

$Y(A, B, C) = \sum m(2, 3, 5, 6)$



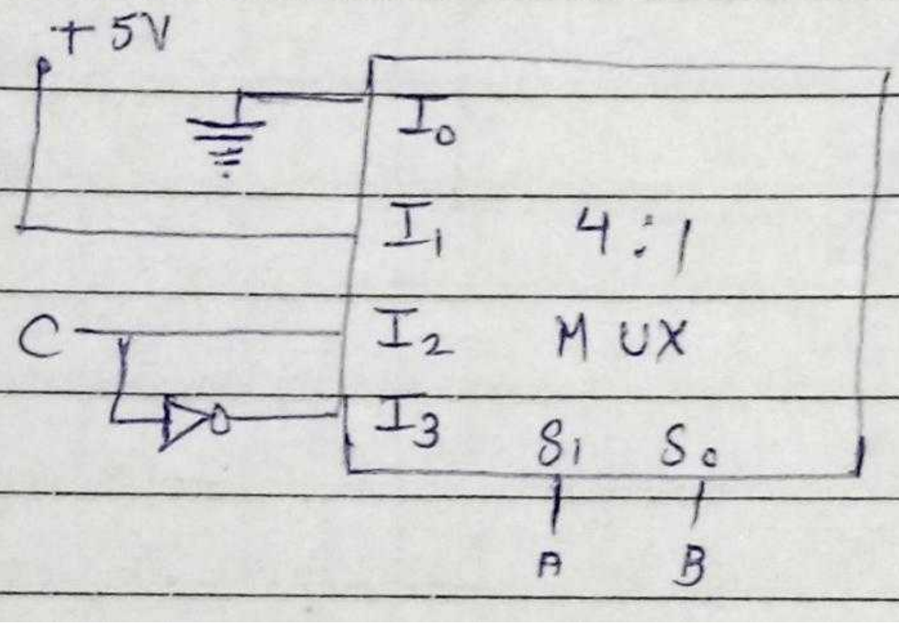
1 = I_2, I_3, I_5, I_6 = +5V

0 = I_0, I_1, I_4, I_7 = ⏚

Using 8:1

Using 4:1

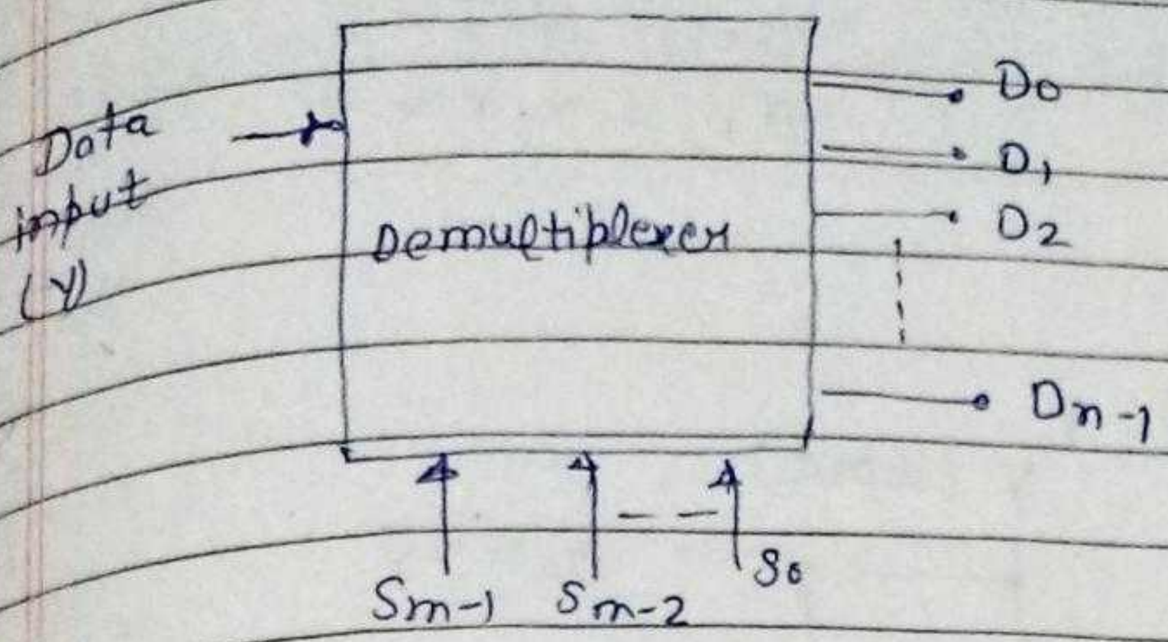
	A	B	C	Y	
0	0	0	0	0	$y = 0$
1	0	0	1	0	
2	0	1	0	1	$y = 1$
3	0	1	1	1	
4	1	0	0	0	$y = c$
5	1	0	1	1	
6	1	1	0	1	$y = \bar{c}$
7	1	1	1	0	



Mutiplexer Application :-

1. Data Routing
2. Parallel to Sereal convertor
   (Many input to one output)
3. Logic Function Generation.

"Demultiplexers"



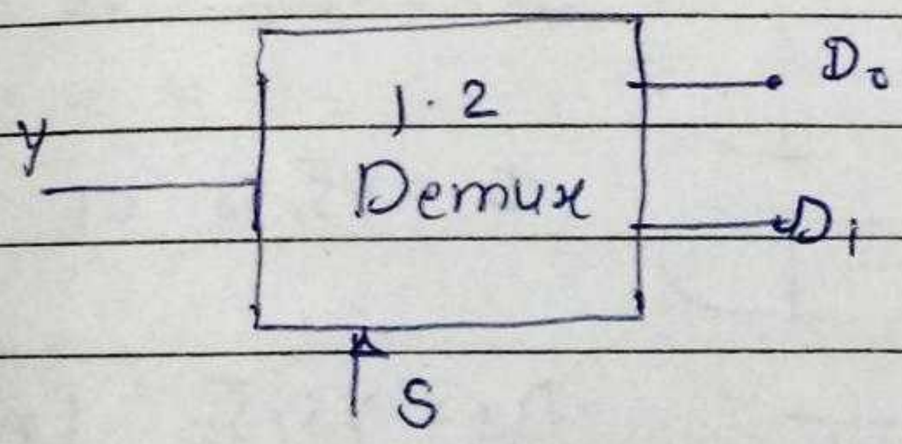
(e) 1 : n demultiplexer

* It has only one input, n output and m select input.
* A demultiplexer performs the reverse operation of a multiplexer i.e. it recives one input and distributes it over several outputs.

Types of Demultiplexers-

(i) 1:2 Demultiplexer → 1 select line
(ii) 1:4 " → 2 select line
(iii) 1:8 " → 3 select line
(iv) 1:16 " → 4 select line
(v) 1:32 " → 5 select line

"1 : 2 Multiplexer"



Truth Table

	Select input	output	
	S	D_1	D_0
0	0	0	1
1	1	1	0

$D_0 = Y\bar{S}$ $D_1 = Y \cdot S$

if $S = 0$

$D_0 = Y \cdot \bar{0} = Y$

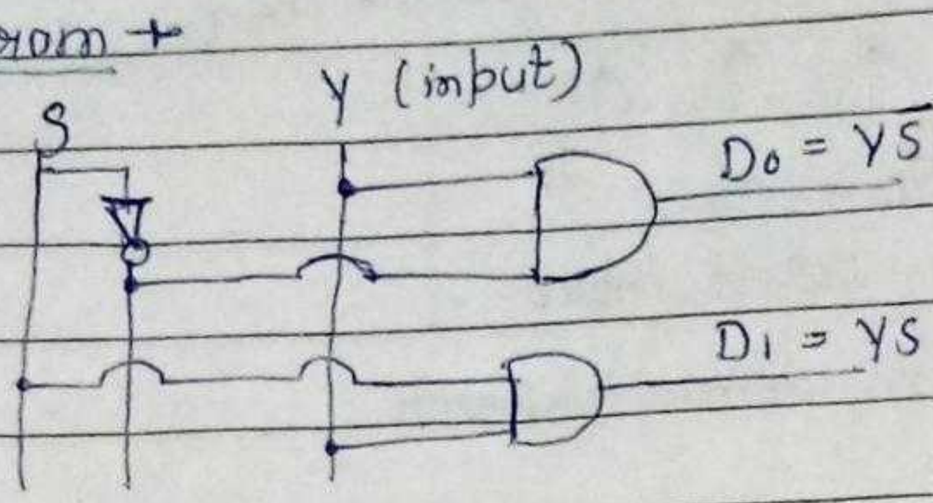
$D_1 = Y \cdot 0 = 0$

if $S = 1$

$D_0 = Y \cdot \bar{1} = 0$

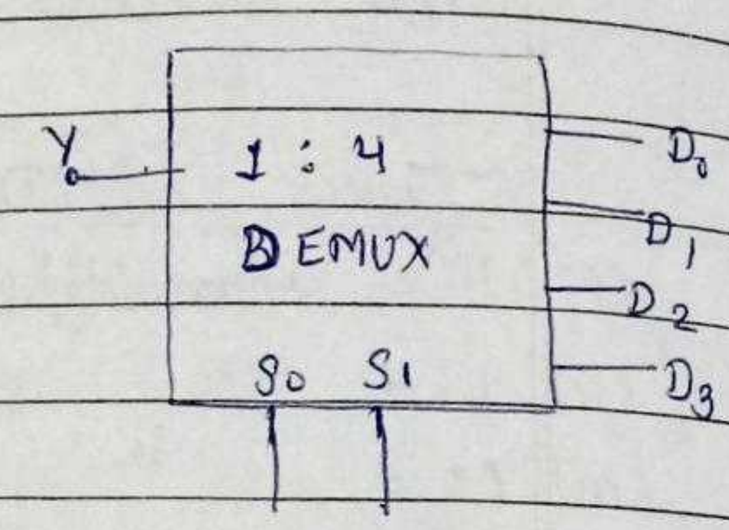
$D_1 = Y \cdot 1 = Y$

combinational Diagram +



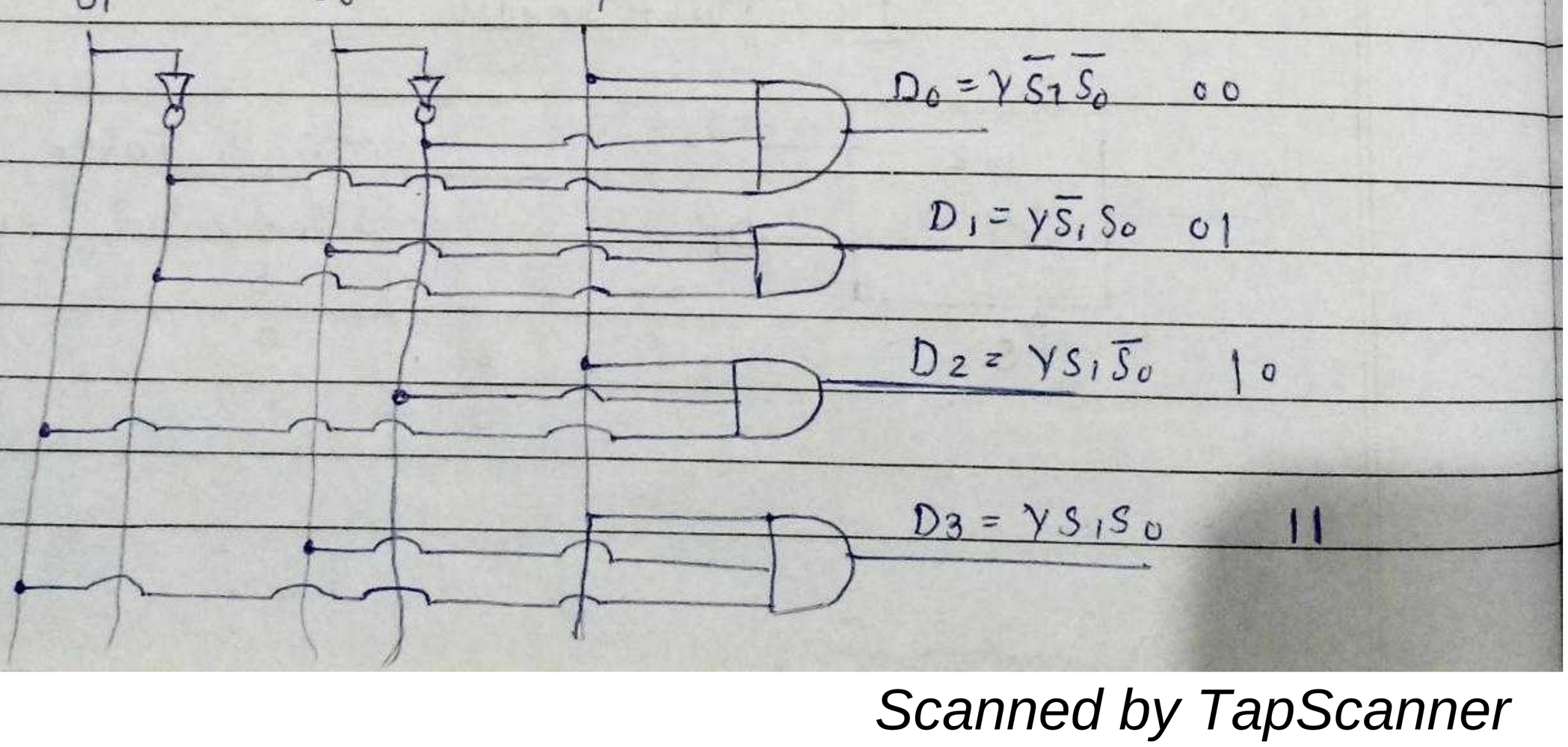
1 : 4 Demultiplexer

select inputs		output			
S_1	S_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
1	0	1	0	0	0
2	1	0	0	1	0
3	1	1	0	0	0



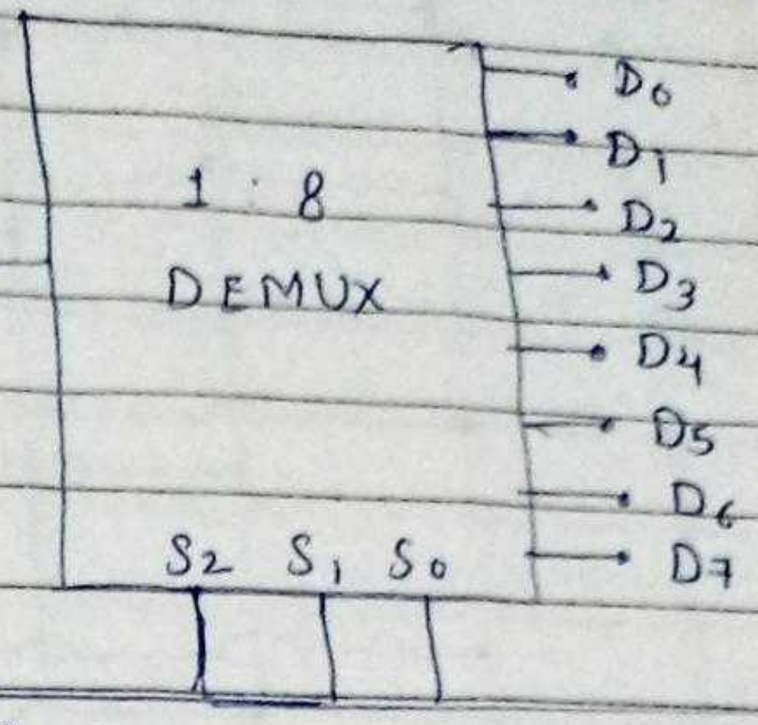
$D_0 = Y\bar{S}_1\bar{S}_0$, $D_1 = Y\bar{S}_1 S_0$, $D_2 = Y S_1\bar{S}_0$, $D_3 = Y S_1 S_0$

combinational Diagram



1:8 Demultiplexer

It has one data input, eight outputs, three select inputs.



select input			output							
S_2	S_1	S_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

$D_0 = Y \bar{S}_2 \bar{S}_1 \bar{S}_0$

$D_1 = Y \bar{S}_2 \bar{S}_1 S_0$

$D_2 = Y \bar{S}_2 S_1 \bar{S}_0$

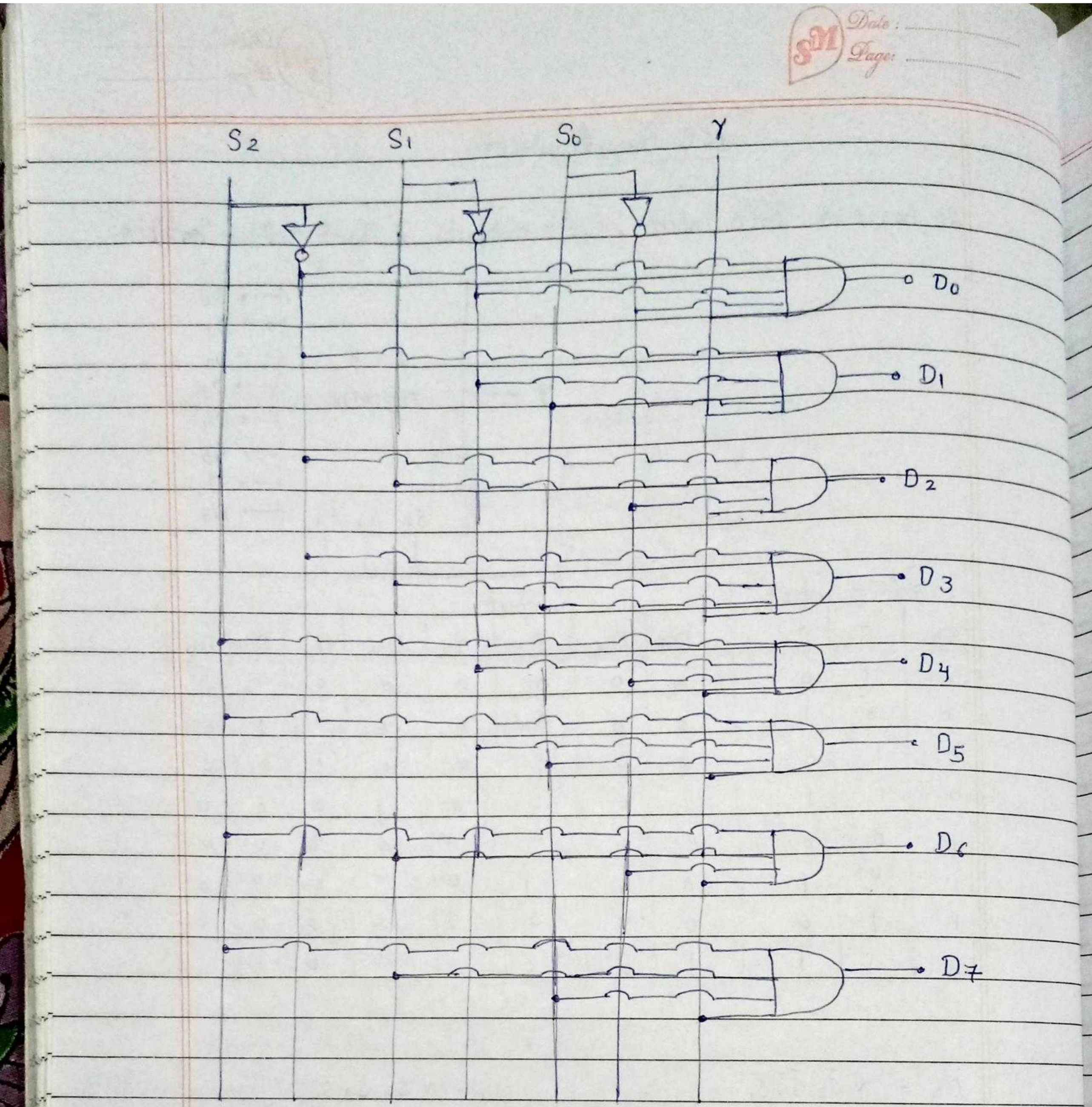
$D_3 = Y \bar{S}_2 S_1 S_0$

$D_4 = Y S_2 \bar{S}_1 \bar{S}_0$

$D_5 = Y S_2 \bar{S}_1 S_0$

$D_6 = Y S_2 S_1 \bar{S}_0$

$D_7 = Y S_2 S_1 S_0$



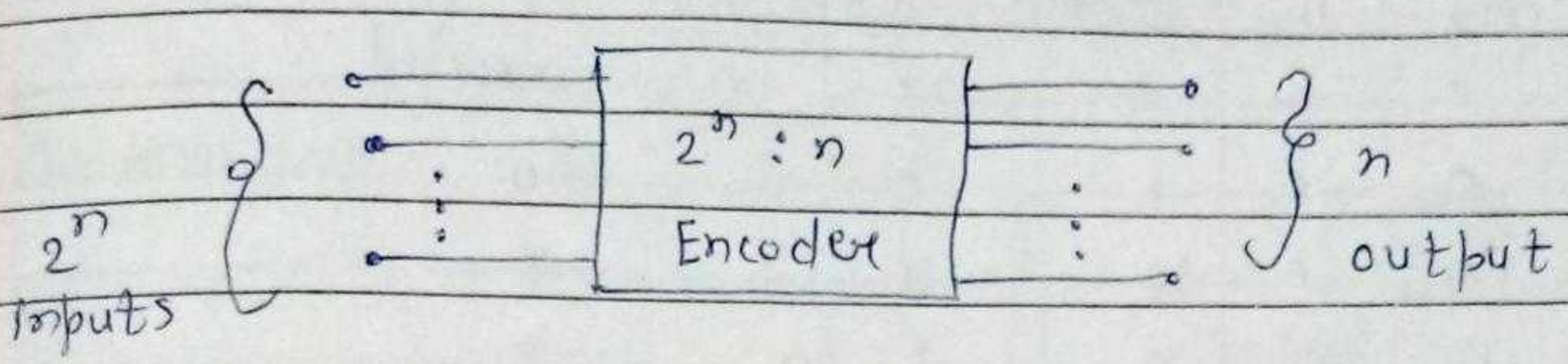
Combinational Diagram.

"Encoders"

An Encoder is a combinational circuit that performs the inverse operation of Decoder. It has maximum 2^n input lines and 'n' output lines. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with 'n' bits.

$2^n : n$

Input output



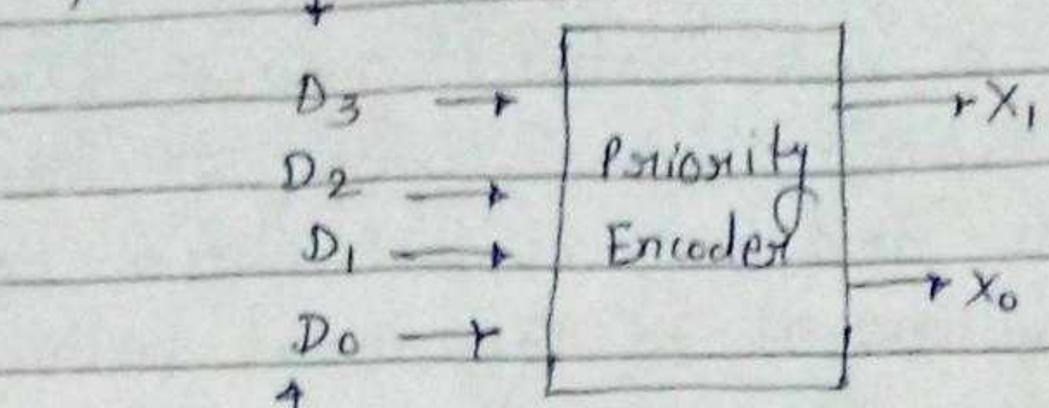
Block diagram of Encoder.

Types of Encoders :-

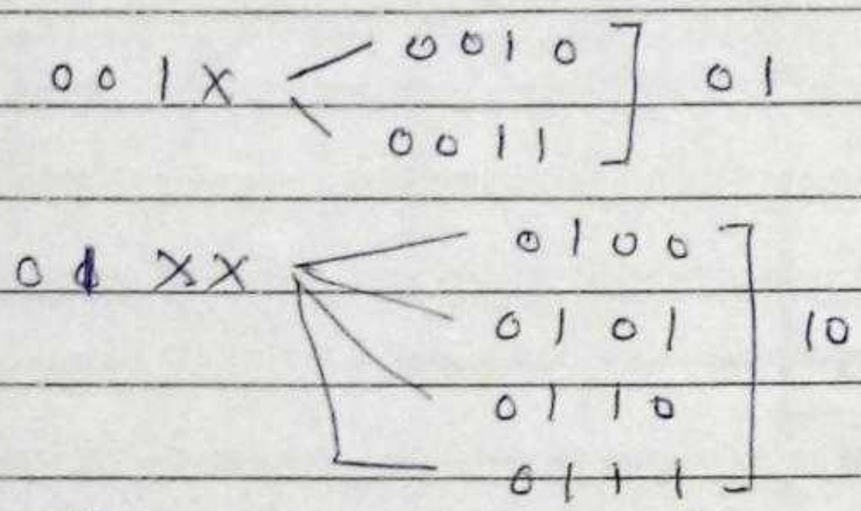
(1) Priority Encoders
(2) Decimal to BCD encoders
(3) Octal to binary encoder
(4) Hexadecimal to binary encoder.

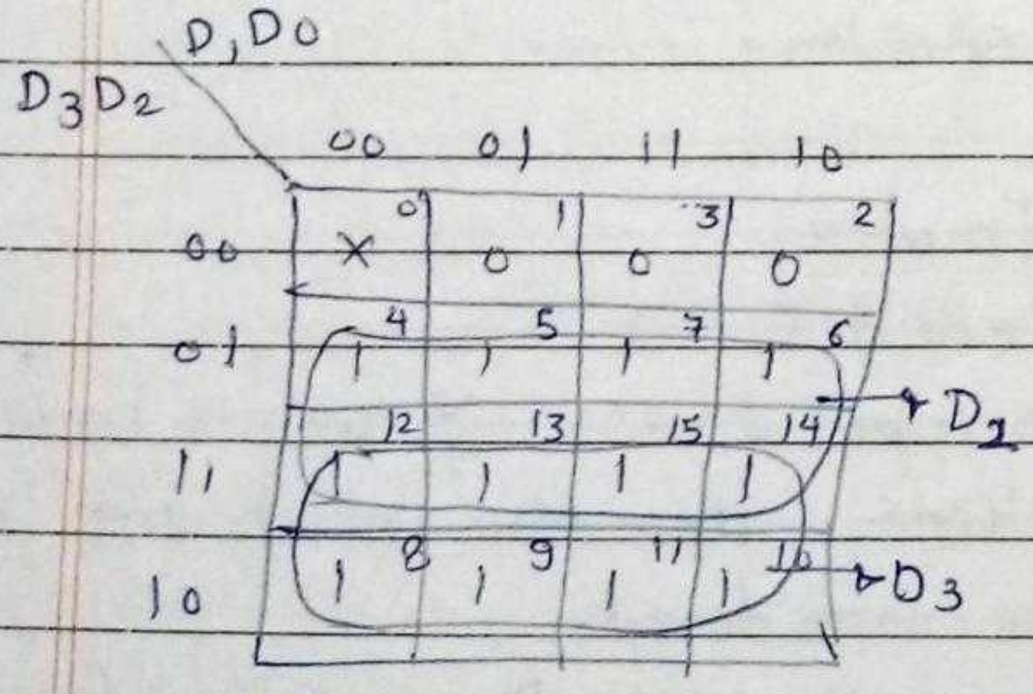
Priority Encoder

- This is a special type of encoder
- Priorities are given to input lines. if two or more input lines are 1 at the same time then the input line with highest priority will be considered.
- There are four inputs D_0 through D_3, and two output x_1 and x_0, out of these D_3 has highest priority and D_0 is lowest priority.



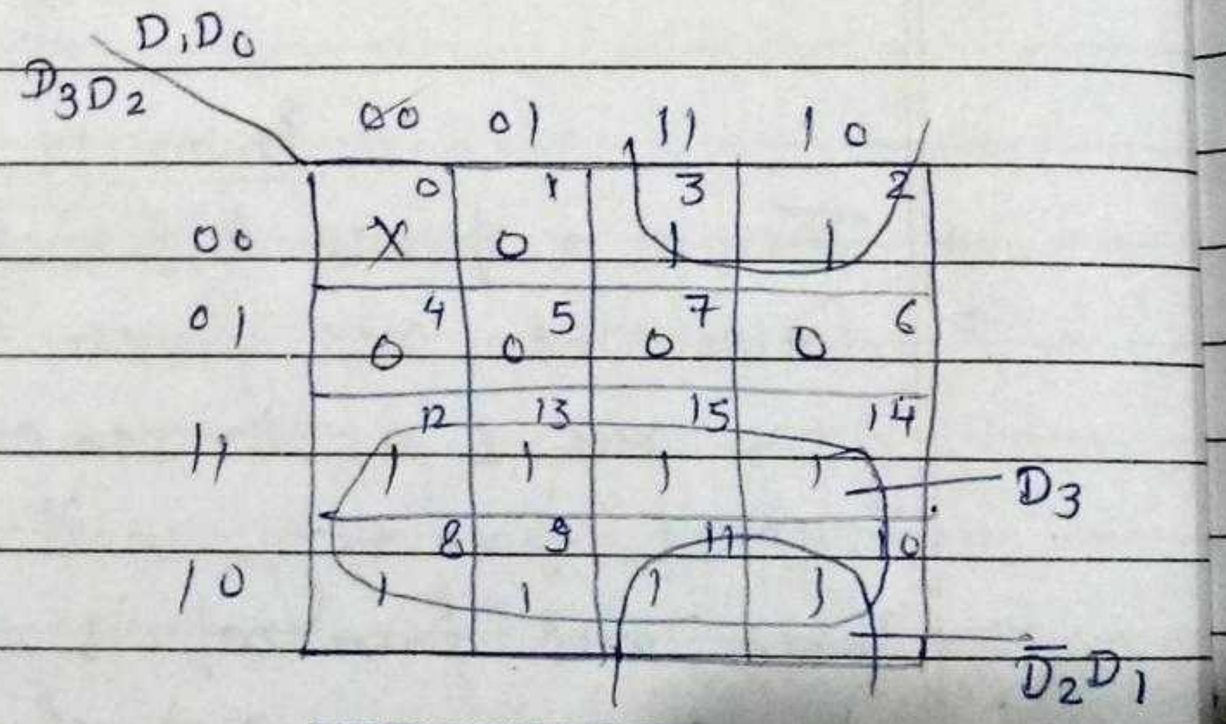
Input				Output		
D_3	D_2	D_1	D_0	X_1	X_0	
0	0	0	0	X	X	→ inverted
0	0	0	1	0	0	→ 0
0	0	1	X	0	1	→ 1
0	1	X	X	1	0	→ 2
1	X	X	X	1	1	→ 3





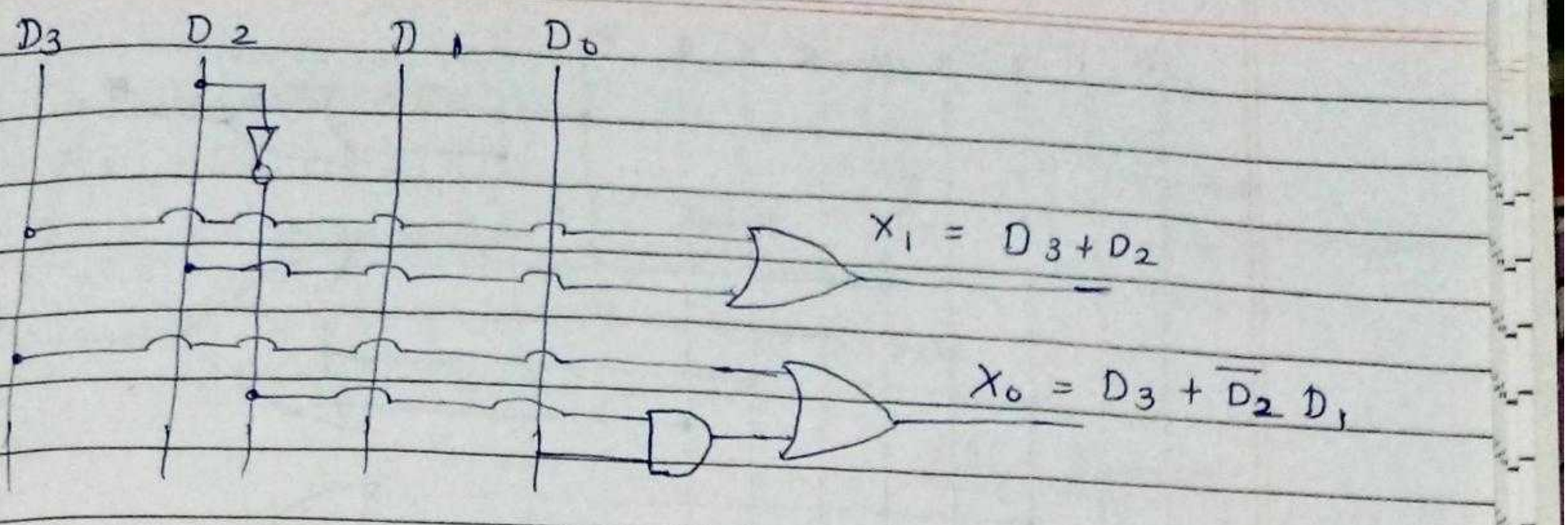
$$X_1 = D_3 + D_2$$

K-map for X_1



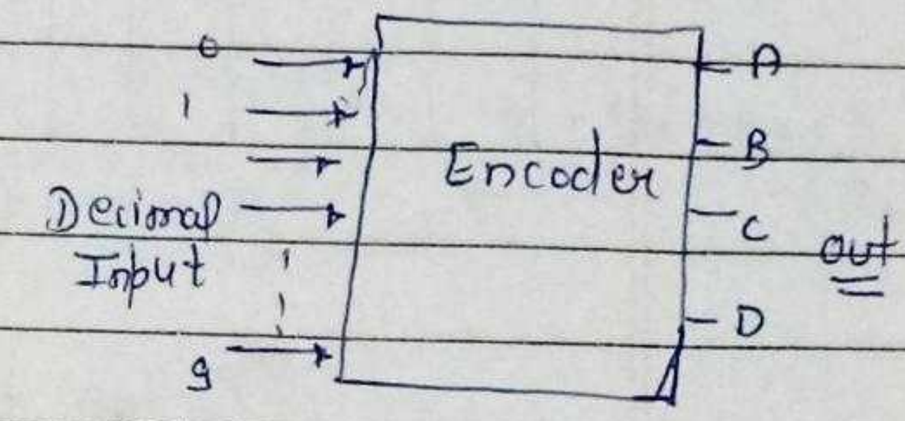
$$X_0 = D_3 + \bar{D}_2 D_1$$

K-map for X_0



"Decimal to BCD Encoder"

Decimal Input	BCD Output			
	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



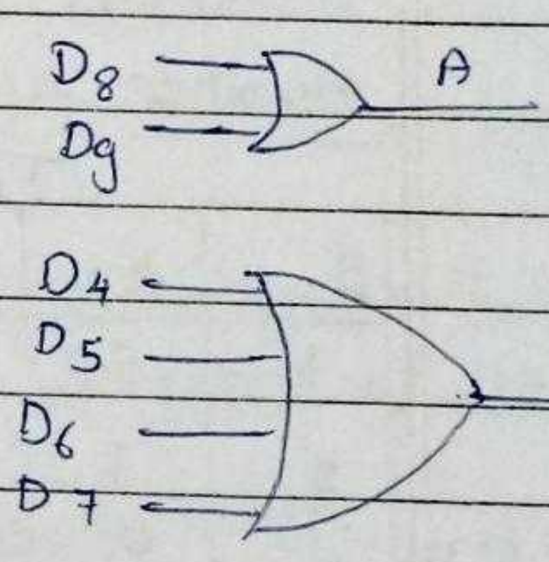
Block diagram

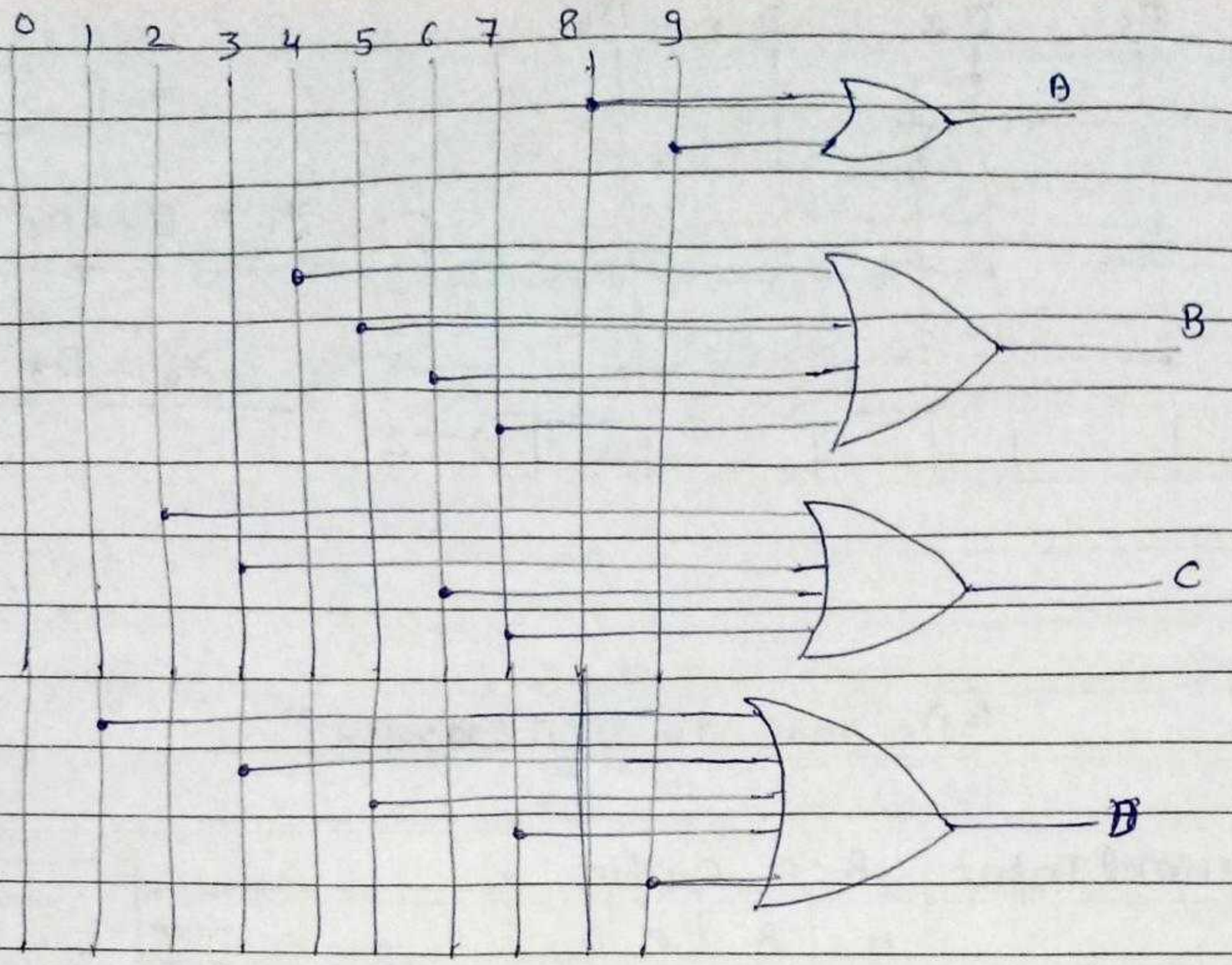
$A = D_8 + D_9$

$B = D_4 + D_5 + D_6 + D_7$

$C = D_2 + D_3 + D_6 + D_7$

$D = D_1 + D_3 + D_5 + D_7 + D_9$





"Octal to Binary Encoder"

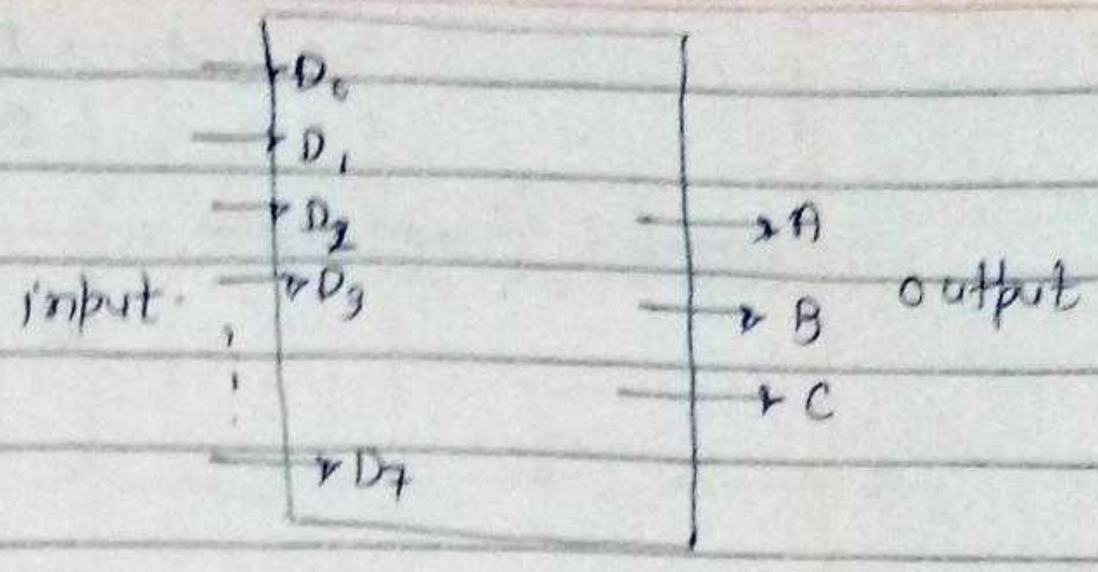
The octal to Binary encoder has 8-input lines and 3-output lines. corresponding to the eight input octal number we get three bit binary output

Input								Output		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

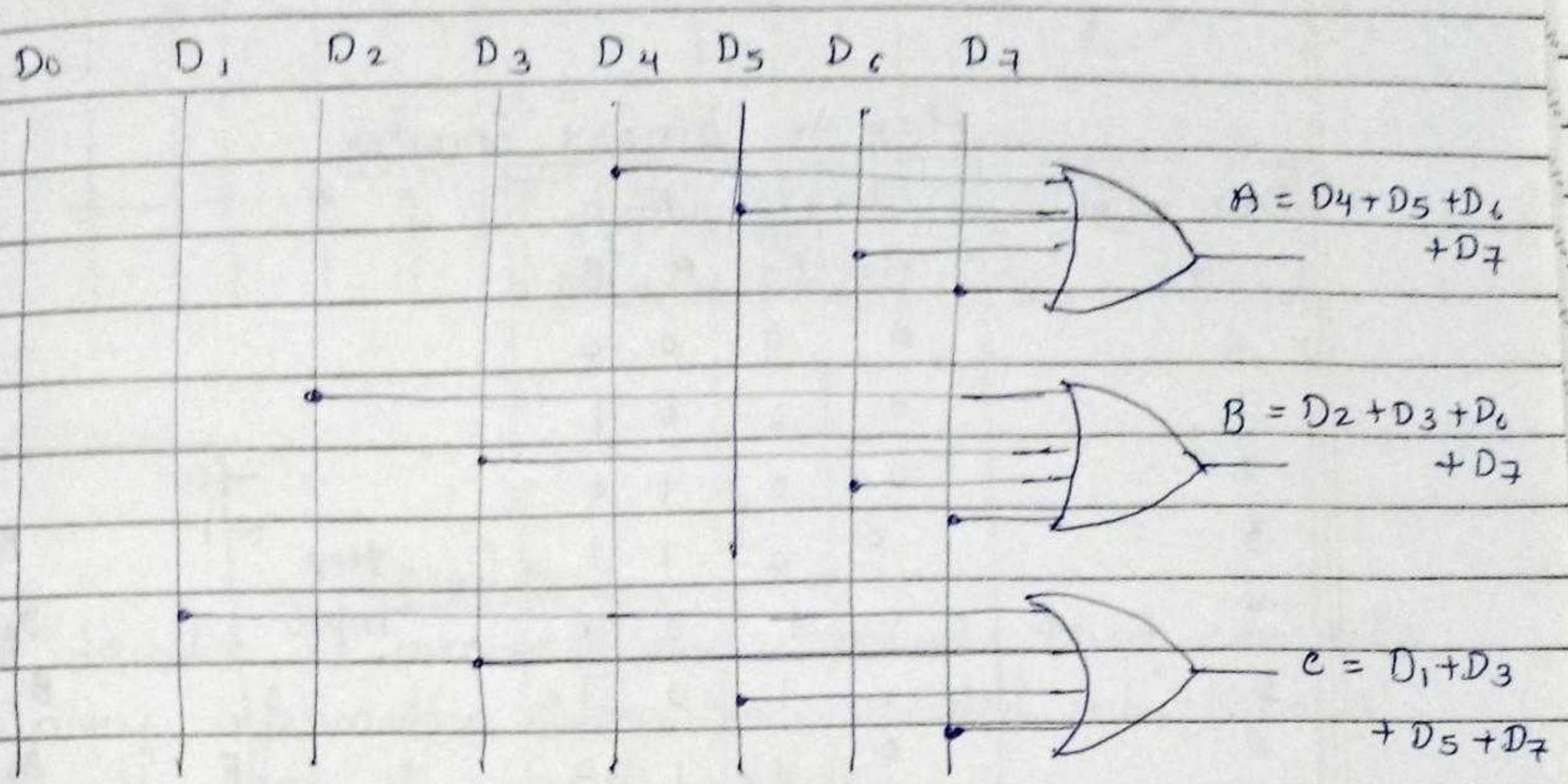
$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$

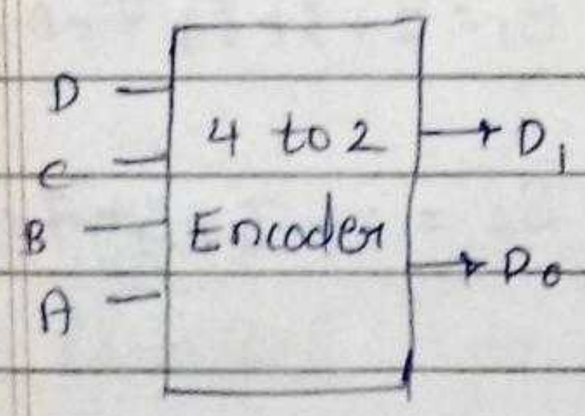


Block diagram



4 to 2 Encoder

Let 4 to 2 Encoder has four inputs, D, C, B, A and two output D_1 and D_0. The block diagram of 4 to 2 Encoder →



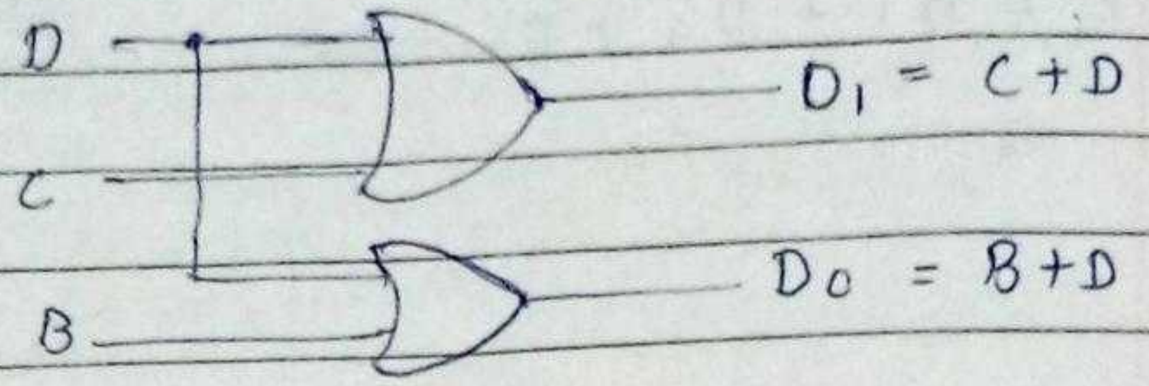
Input				Output	
D	C	B	A	D_1	D_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

at any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output.

So $D_1 = C + D$

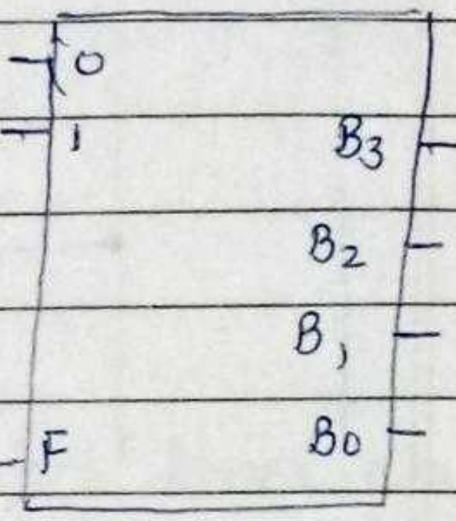
$D_0 = B + D$

Circuit diagram:



Hex to binary Encoder

Hex input H	Binary output B_3	B_2	B_1	B_0
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
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

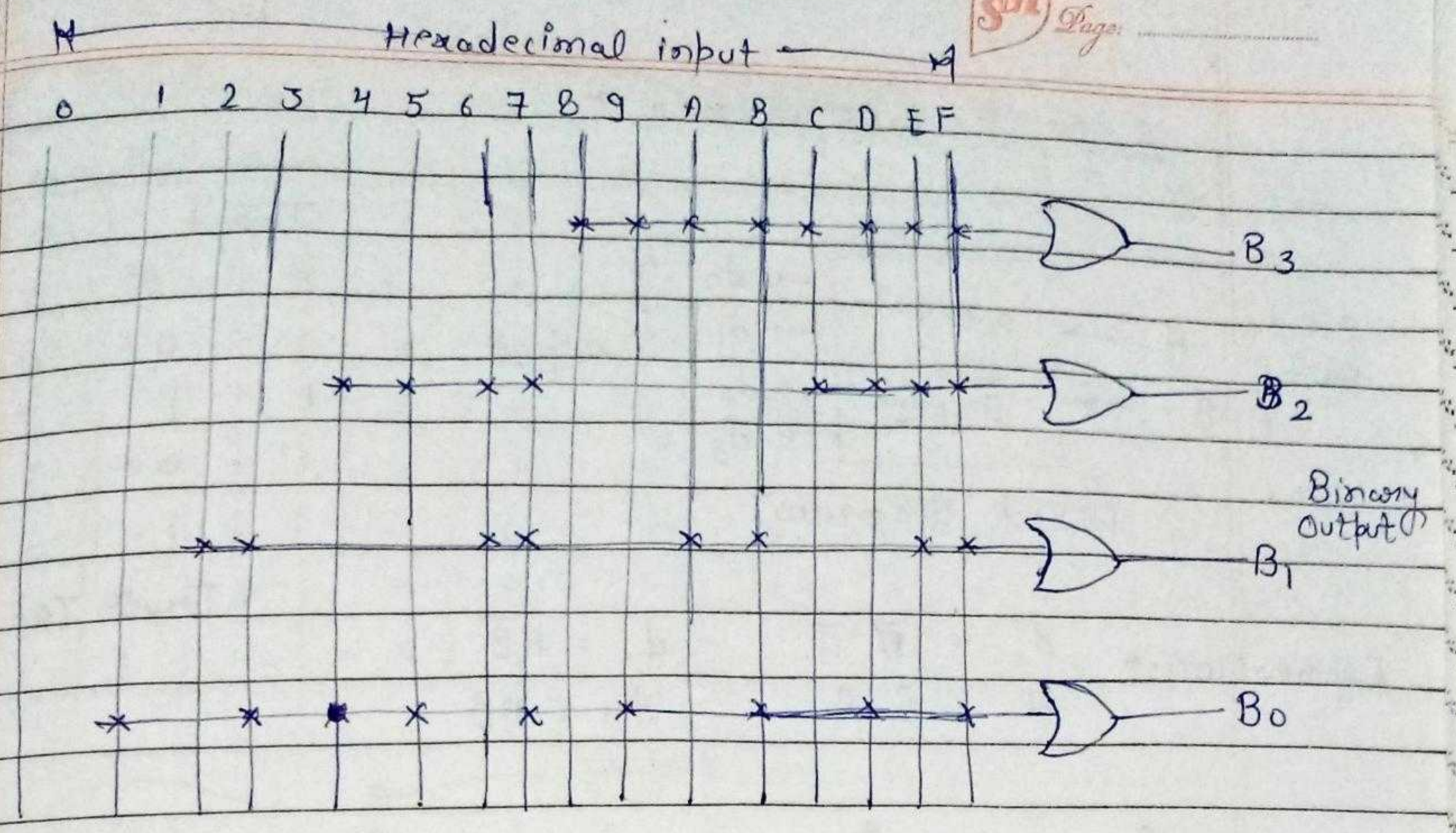


$B_3 = 8 + 9 + A + B + C + D + E + F$

$B_2 = 4 + 5 + 6 + 7 + C + D + E + F$

$B_1 = 2 + 3 + 6 + 7 + A + B + E + F$

$B_0 = 1 + 3 + 5 + 7 + 8 + D + F$

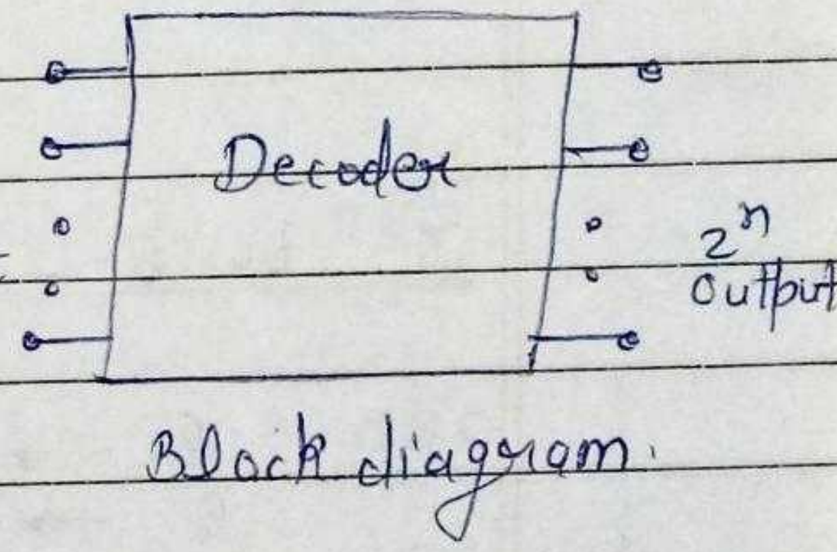


Decoder

A decoder is combinational circuit that converts binary information from the n coded inputs to a maximum of 2^n unique outputs.

Typical applications -

1. Code converters
2. BCD to seven segment decoders
3. Nixie tube decoders
4. Relay actuators

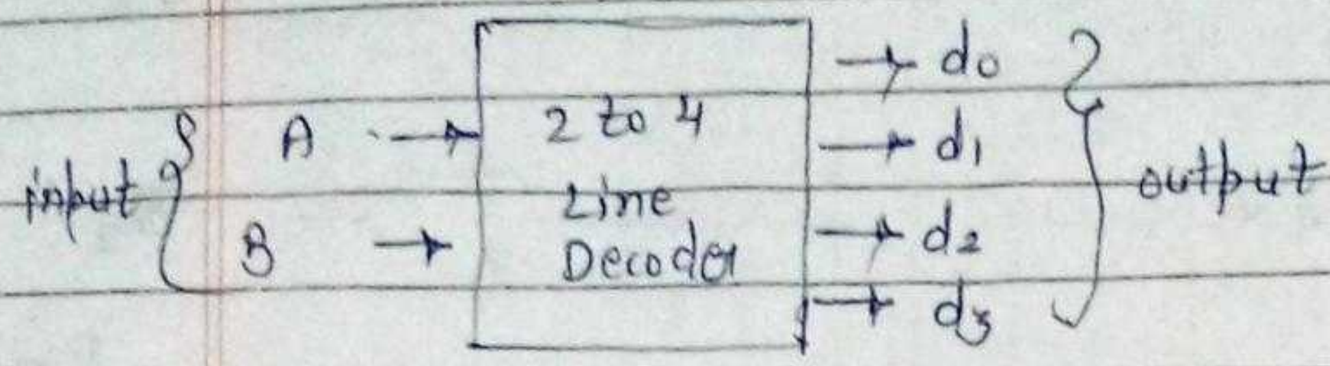


Block diagram.

Types -

1. Parallel → 2 to 4 line, 3 to 8 Line decorder and so on.
2. Tree
3. Balanced

2 to 4 Line decoder →



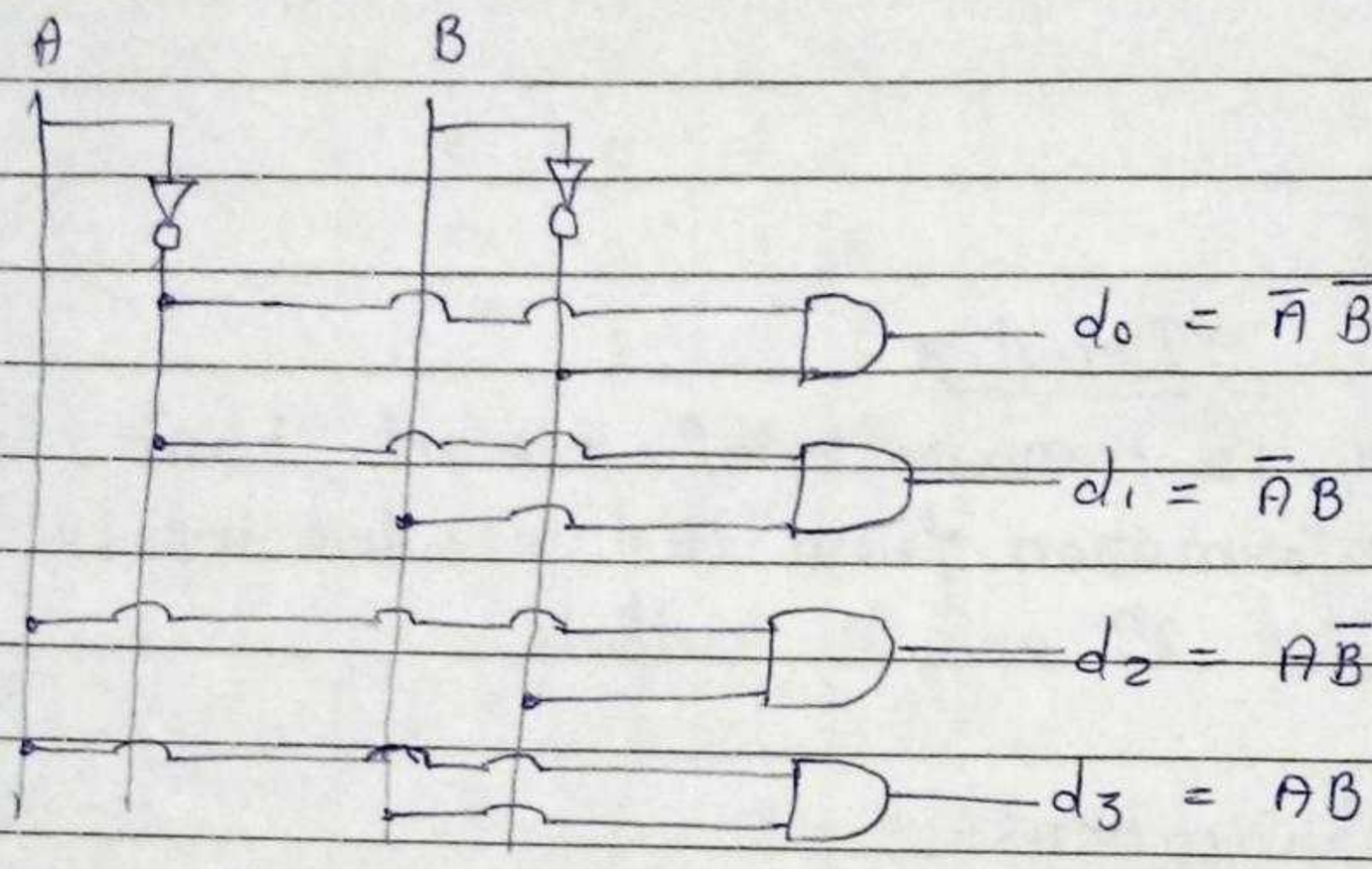
(Block diagram)

Input		Output
A	B	o/p
0	0	d_0
0	1	d_1
1	0	d_2
1	1	d_3

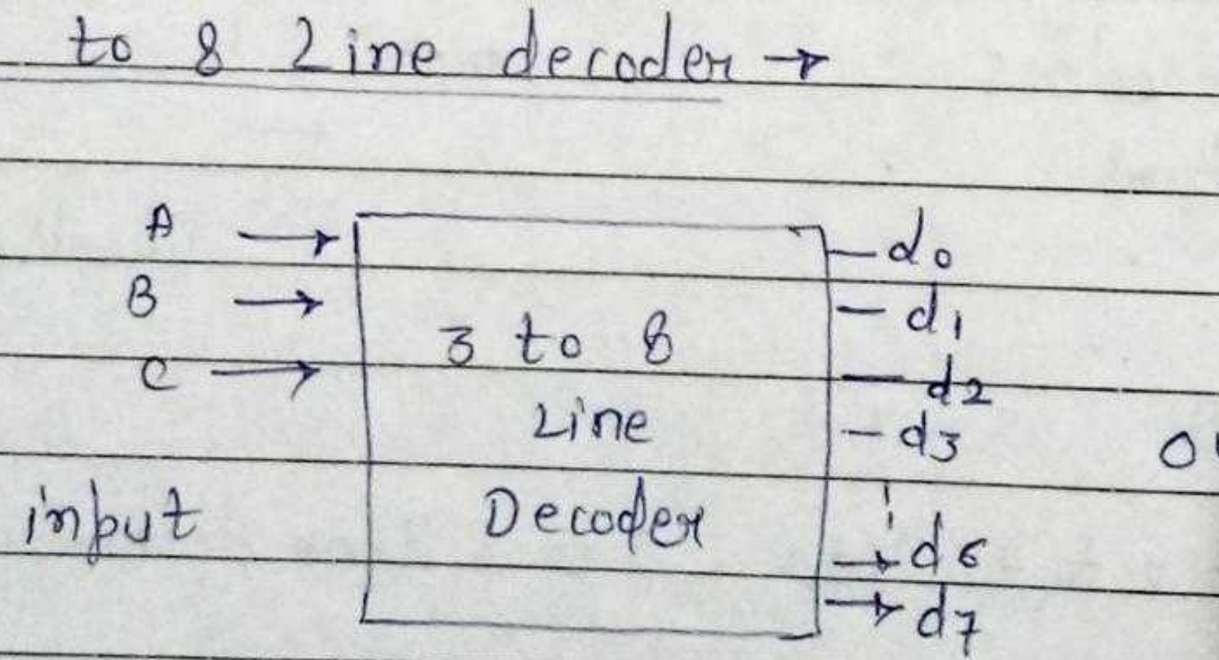
(Truth Table)

Expression → $d_0 = \bar{A}\,\bar{B}$ $\quad d_2 = A\bar{B}$

$d_1 = \bar{A}\,B$ $\quad d_4 = AB$



3 to 8 line decoder →

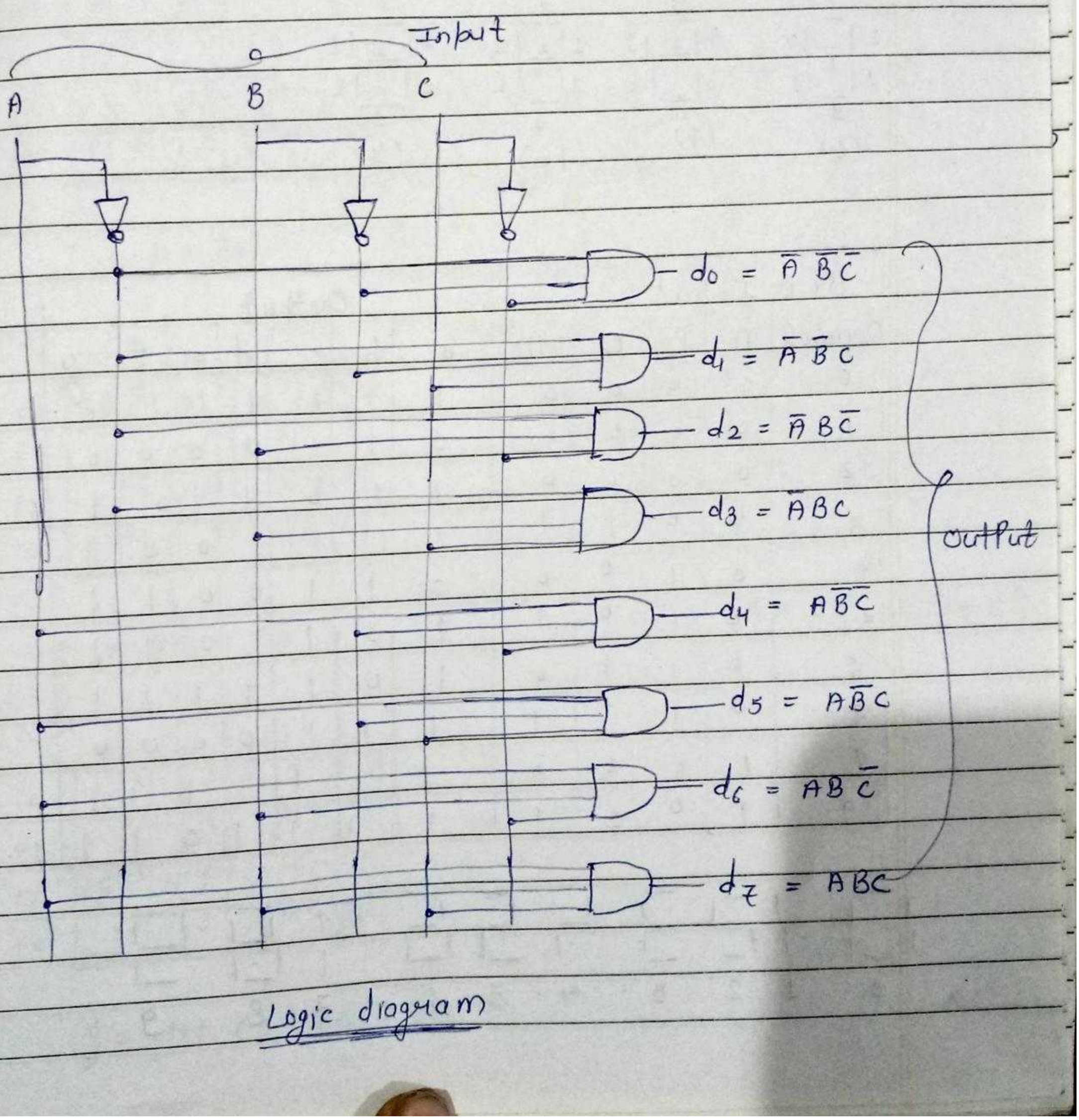


(Block diagram)

Date:
Page:

Truth Table

Input			output								
A	B	C	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	Exp
0	0	0	1	0	0	0	0	0	0	0	$\bar{A}\bar{B}\bar{C}$
0	0	1	0	1	0	0	0	0	0	0	$\bar{A}\bar{B}C$
0	1	0	0	0	1	0	0	0	0	0	$\bar{A}B\bar{C}$
0	1	1	0	0	0	1	0	0	0	0	$\bar{A}BC$
1	0	0	0	0	0	0	1	0	0	0	$A\bar{B}\bar{C}$
1	0	1	0	0	0	0	0	1	0	0	$A\bar{B}C$
1	1	0	0	0	0	0	0	0	1	0	$AB\bar{C}$
1	1	1	0	0	0	0	0	0	0	1	ABC



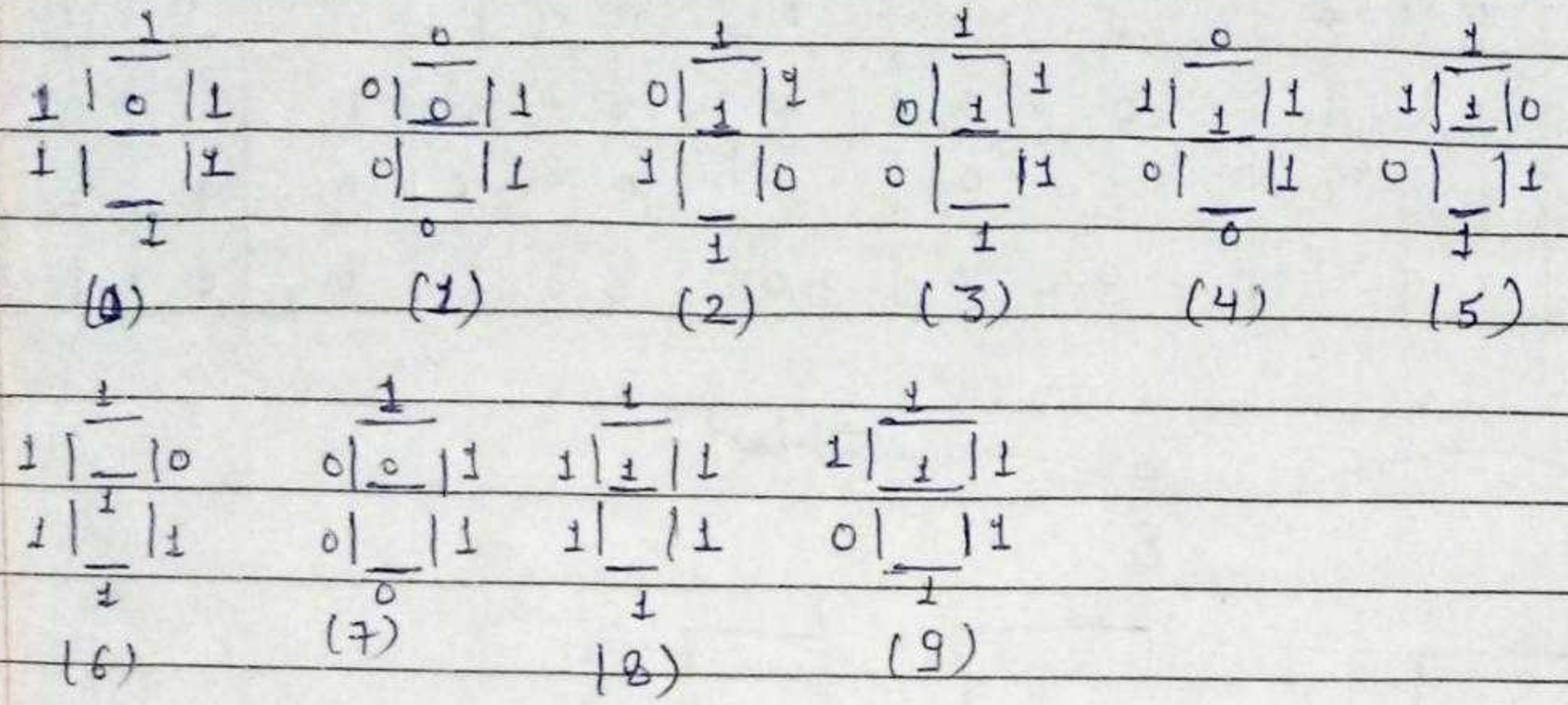
Logic diagram

"BCD to 7-Segment decoder"

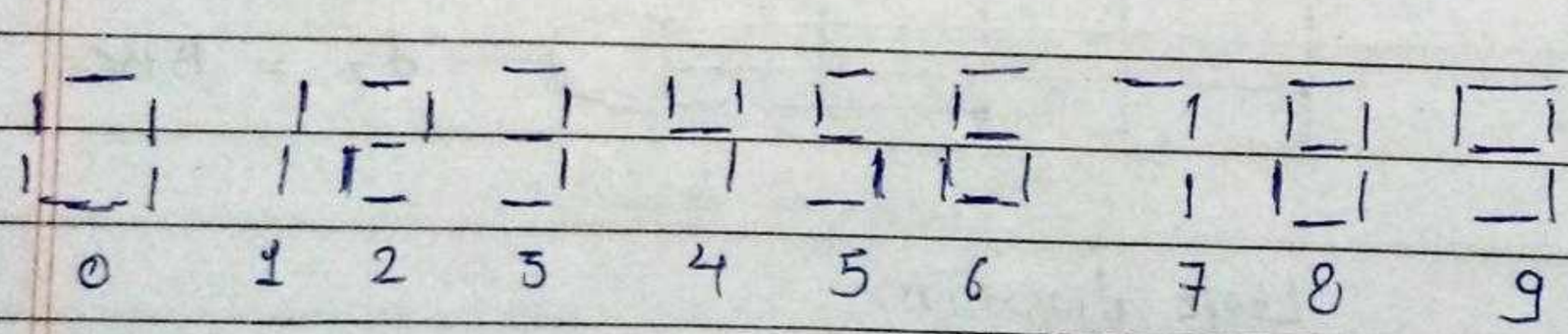
LED GLOW → 1
NOT GLOW → 0

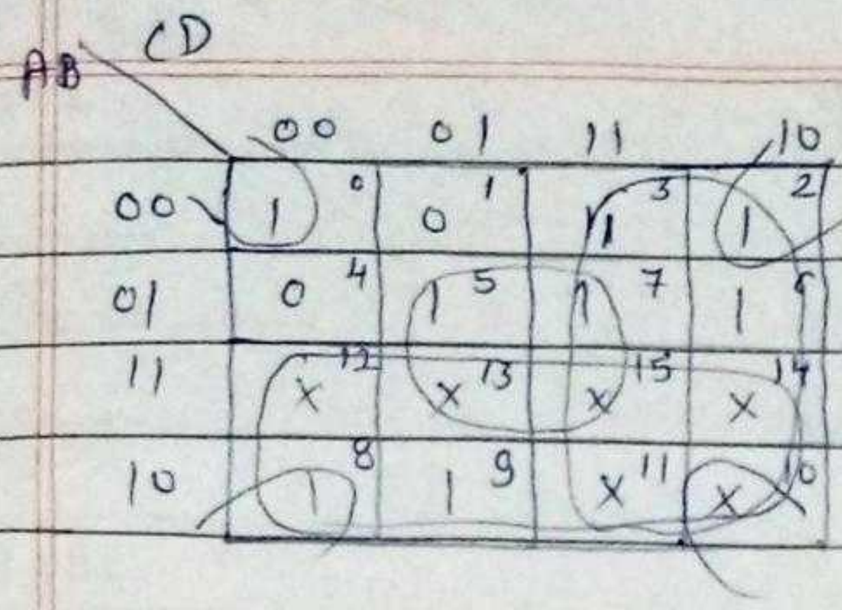
Common cathode (1) Common anode (0)

0-9

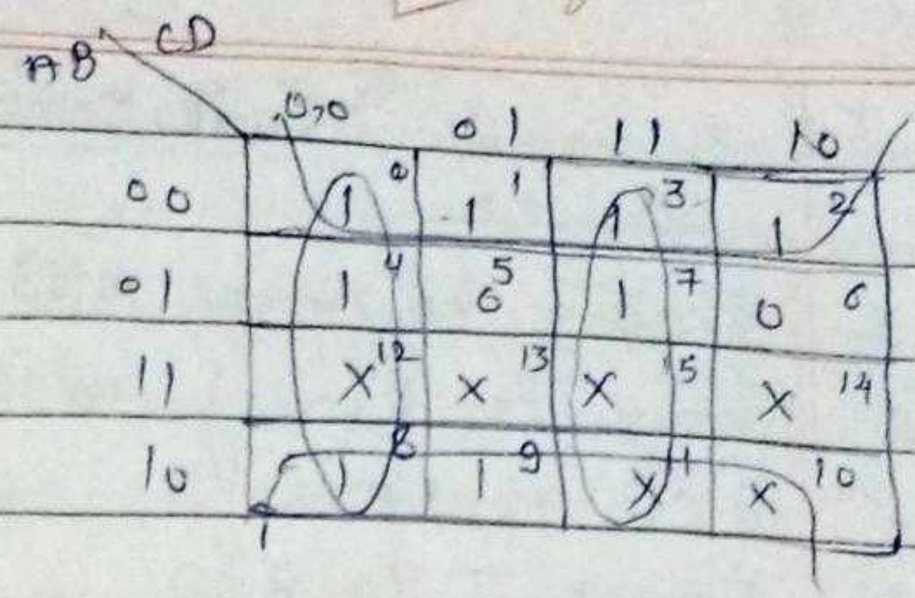


BCD Input					Output						
Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

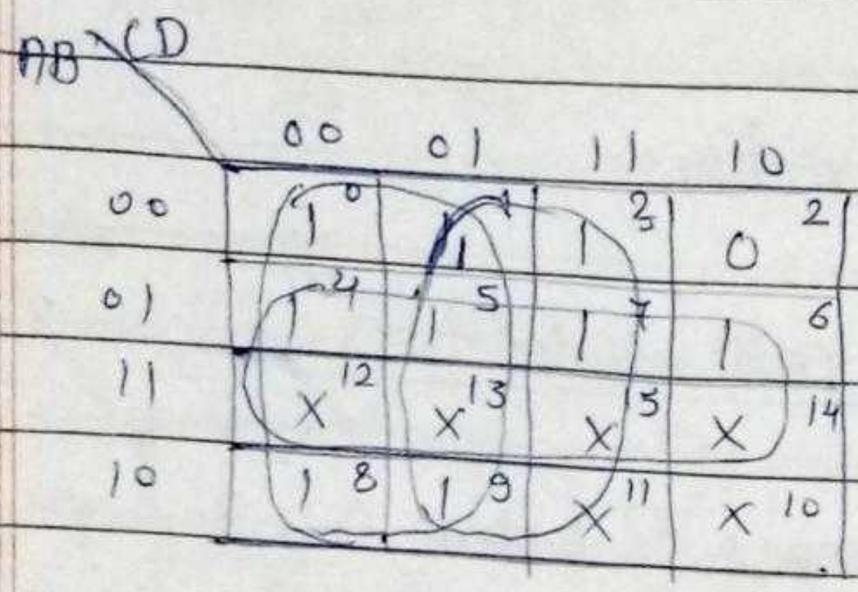




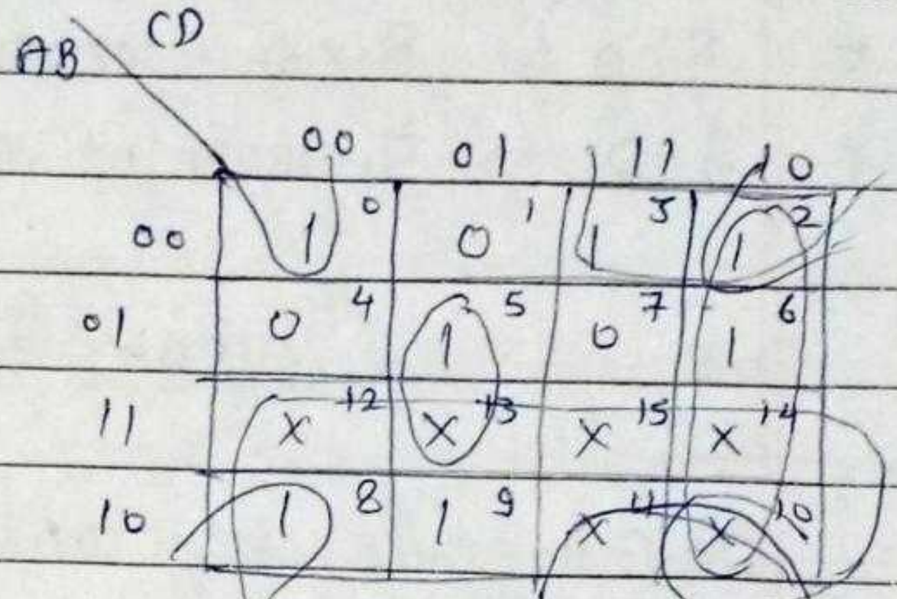
$$a = A + C + BD + \bar{B}\,\bar{D}$$



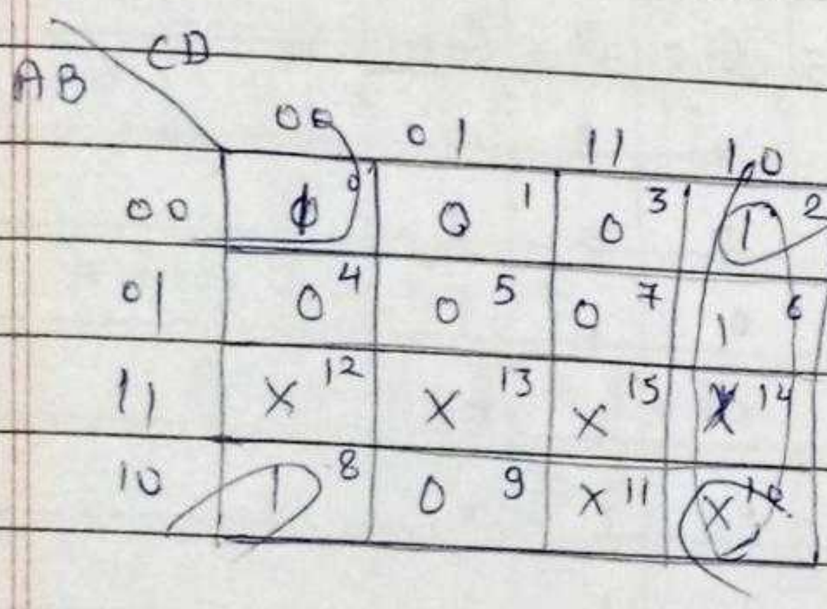
$$b = \bar{B} + \bar{C}\,\bar{D} + CD$$



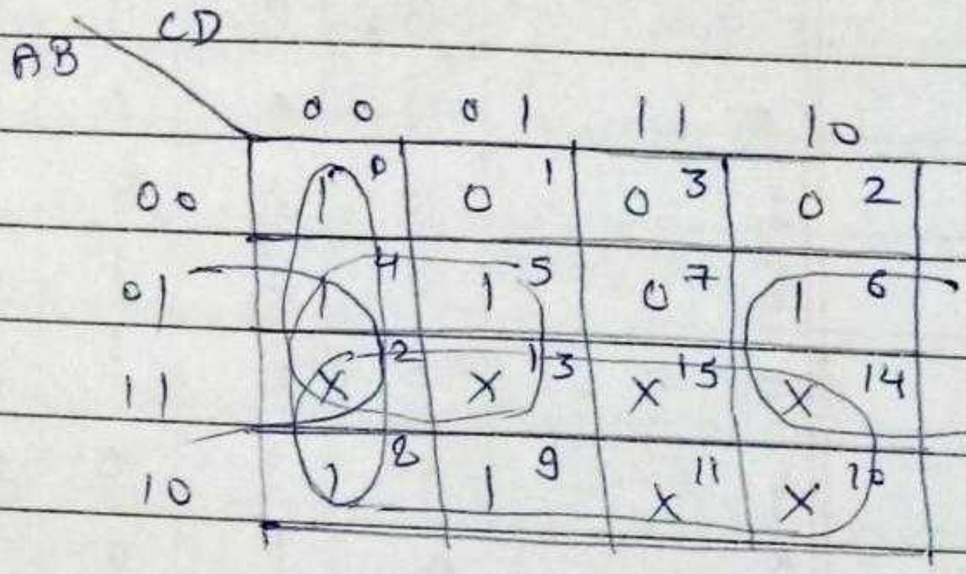
$$C = B + \bar{C} + D$$



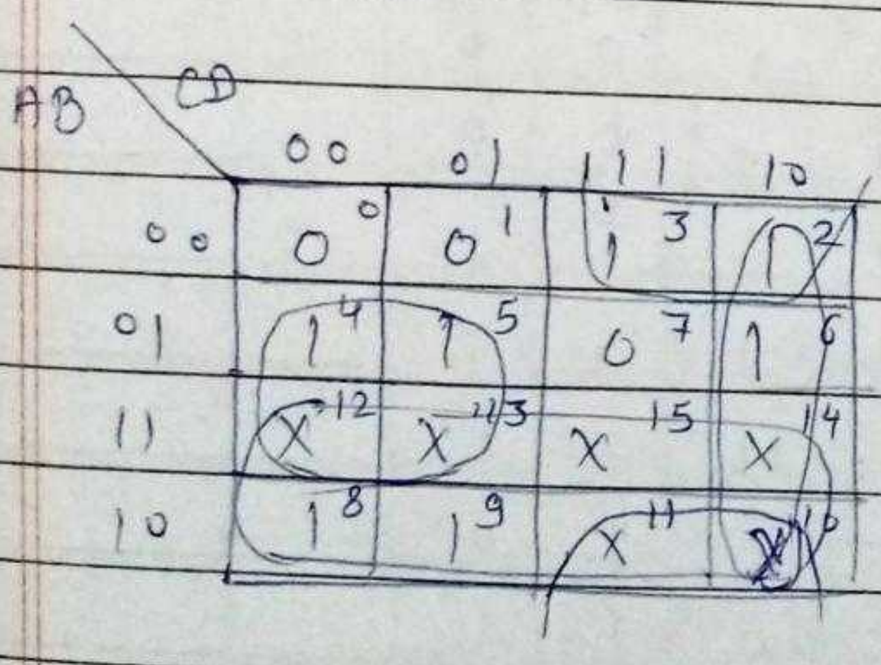
$$d = A + C\bar{D} + \bar{D}C + C\bar{D} + B\,\bar{C}D$$



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



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



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

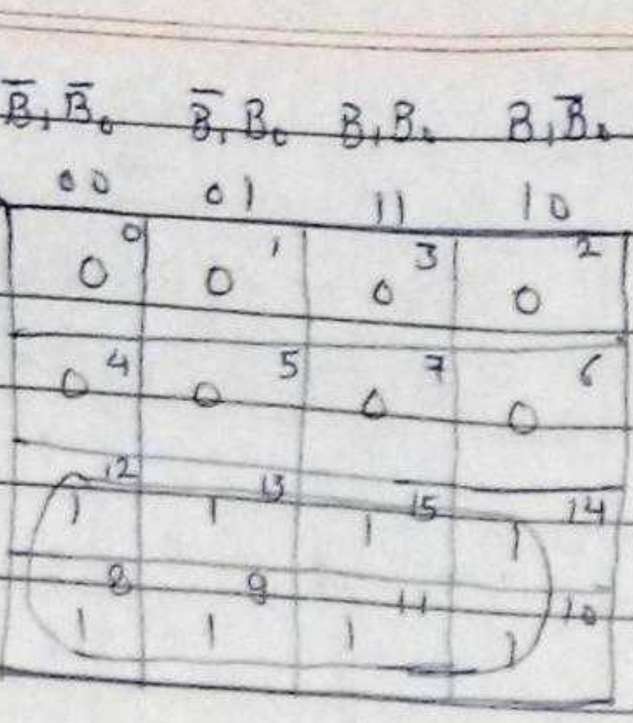
"Code conversion"

1. Binary to Gray code convertor.
2. Gray to Binary code convertor
3. BCD to Excess 3
4. Excess 3 to BCD
5. Grey to Excess 3
6. Binary to BCD convertor
7. BCD to Binary convertor
8. BCD to 7 segment code convertor.

"Binary to Gray code convertor."

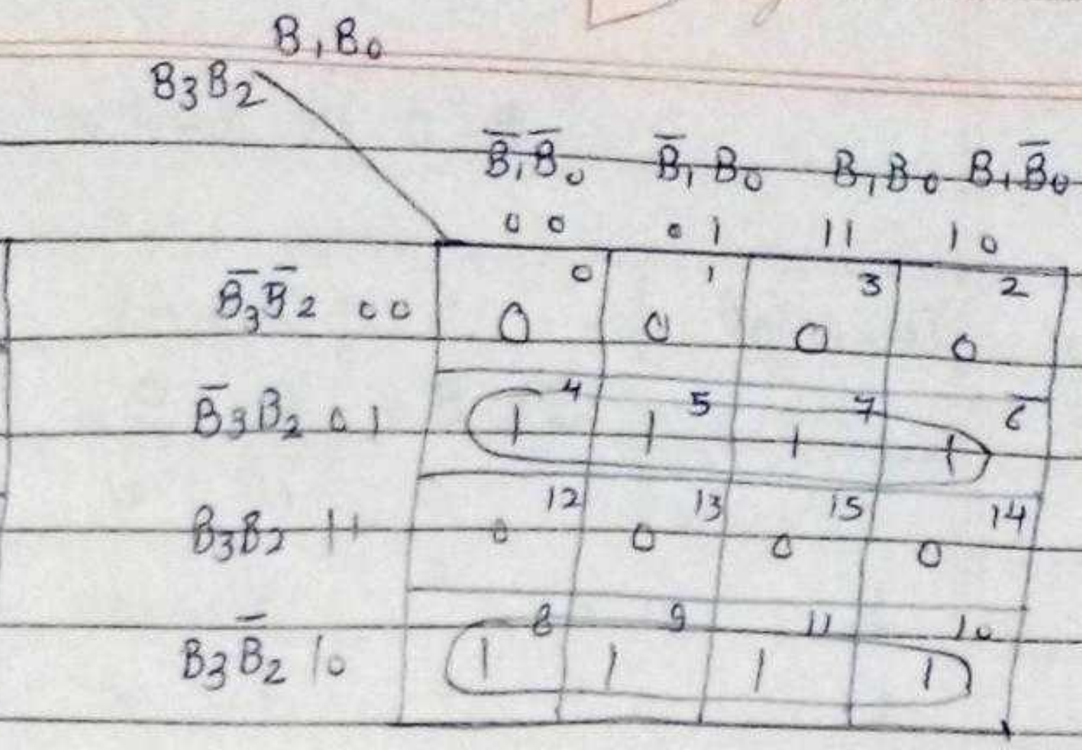
In Gray code only one bit changes at a time.

Decimal	Binary inputs				Gray output			
	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0



K-map for G_3 →

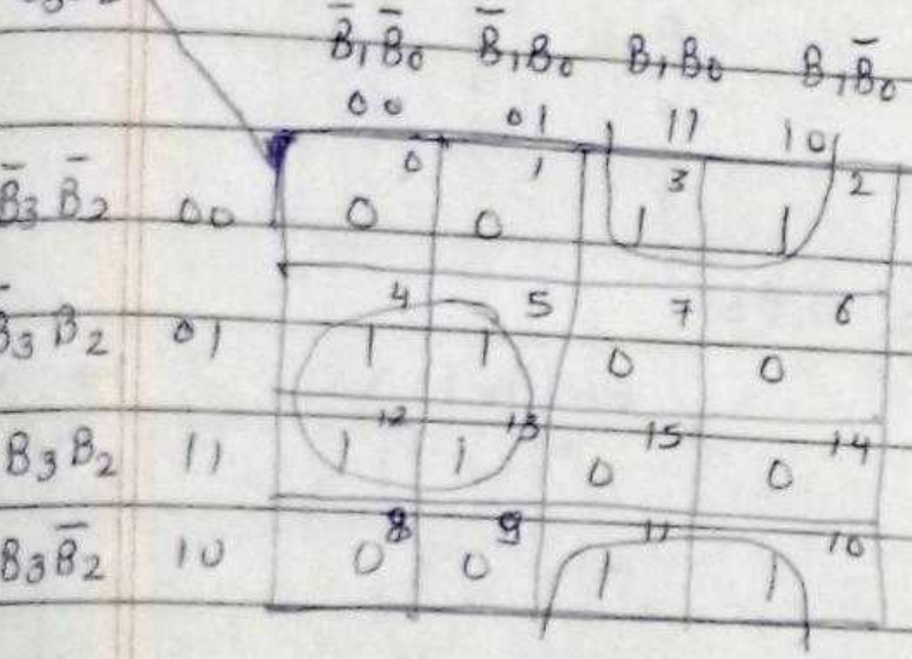
$G_3 = B_3$



K-map for G_2 →

$G_2 = \bar{B}_3 B_2 + B_3 \bar{B}_2$

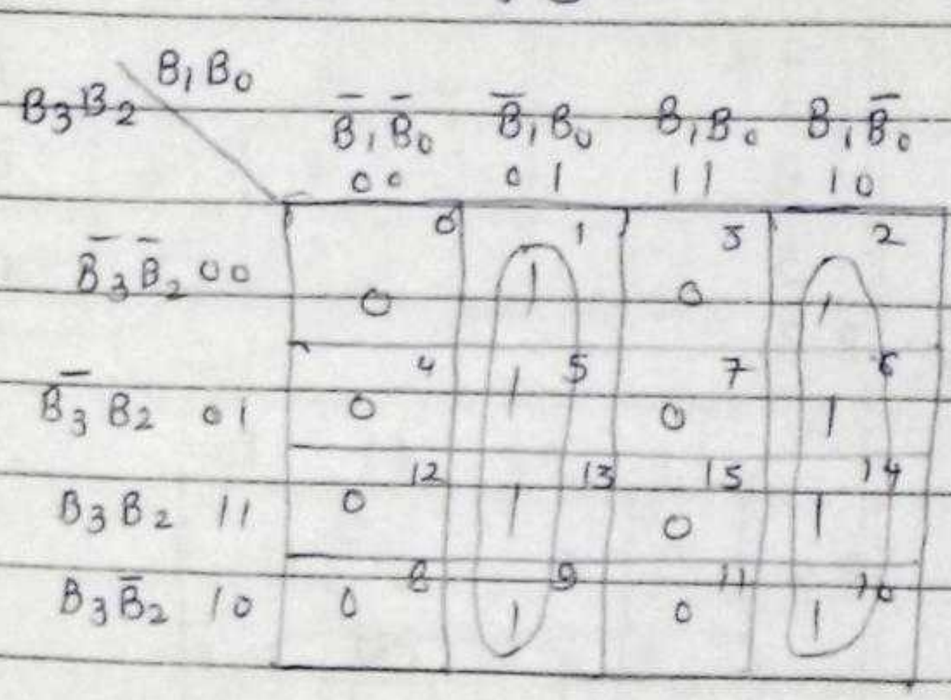
$G_2 = B_3 \oplus B_2$



K-map for G_1 →

$G_1 = B_2 \bar{B}_1 + \bar{B}_2 B_1$

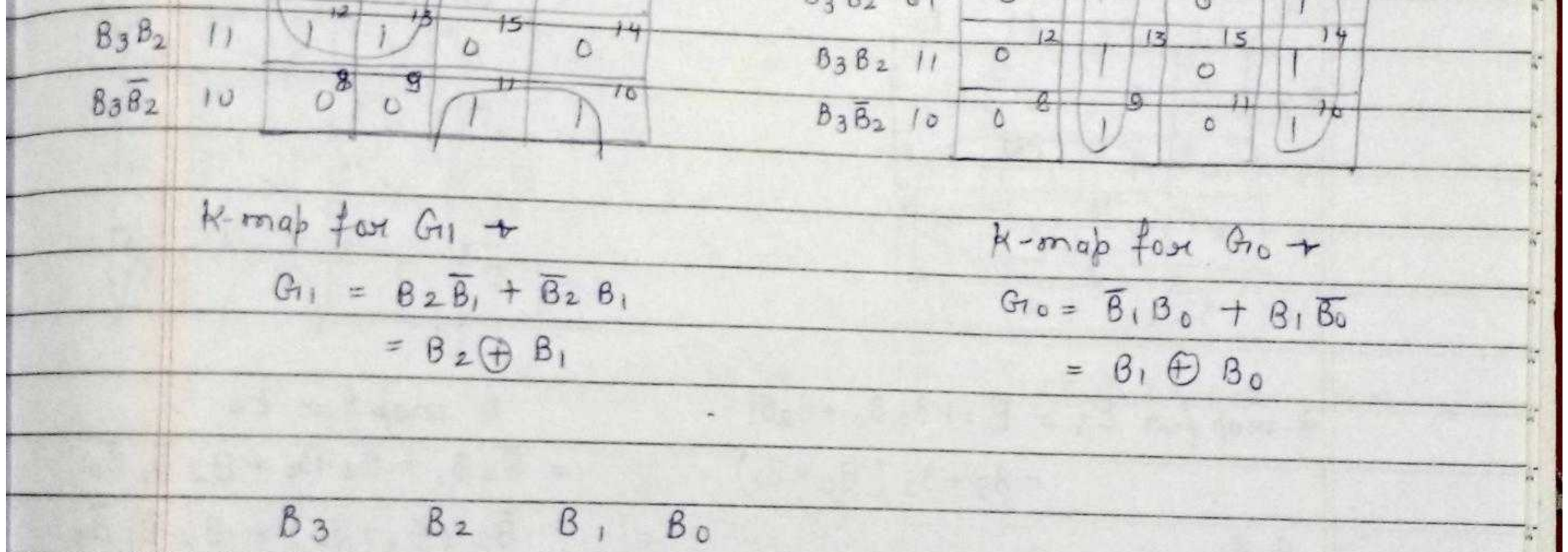
$= B_2 \oplus B_1$



K-map for G_0 →

$G_0 = \bar{B}_1 B_0 + B_1 \bar{B}_0$

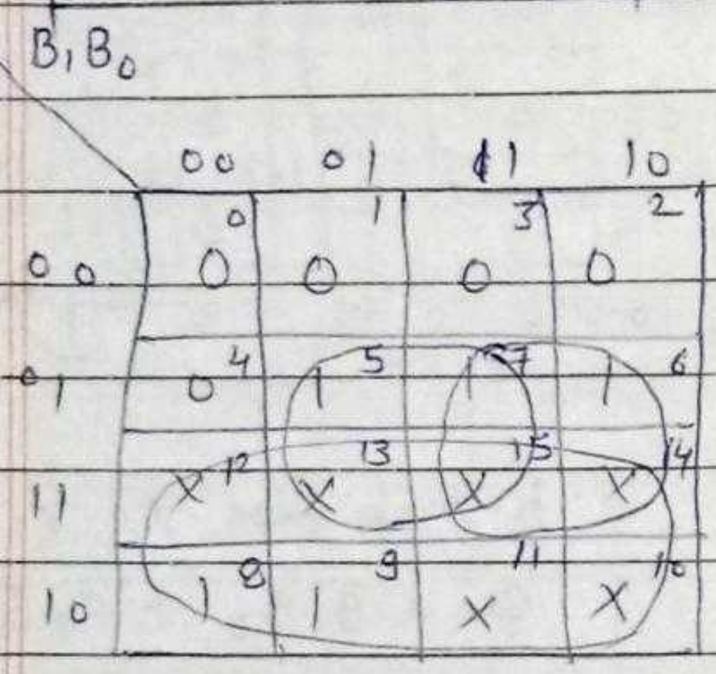
$= B_1 \oplus B_0$

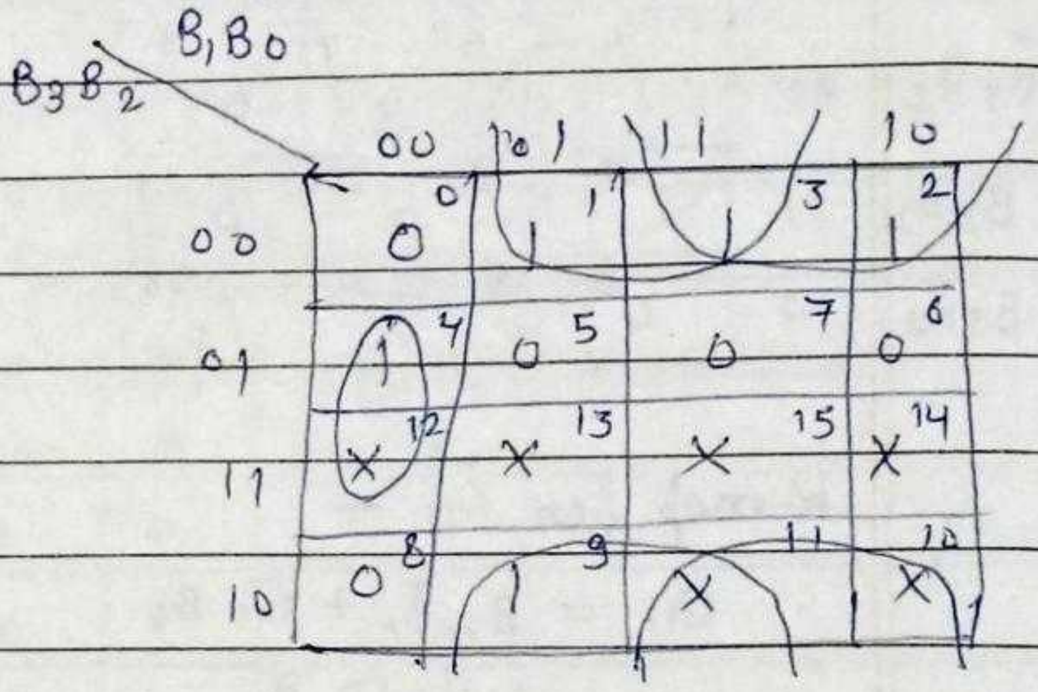


Diagram

"BCD to Excess-3"

Decimal	BCD Inputs				Excess-3 outputs			
	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0





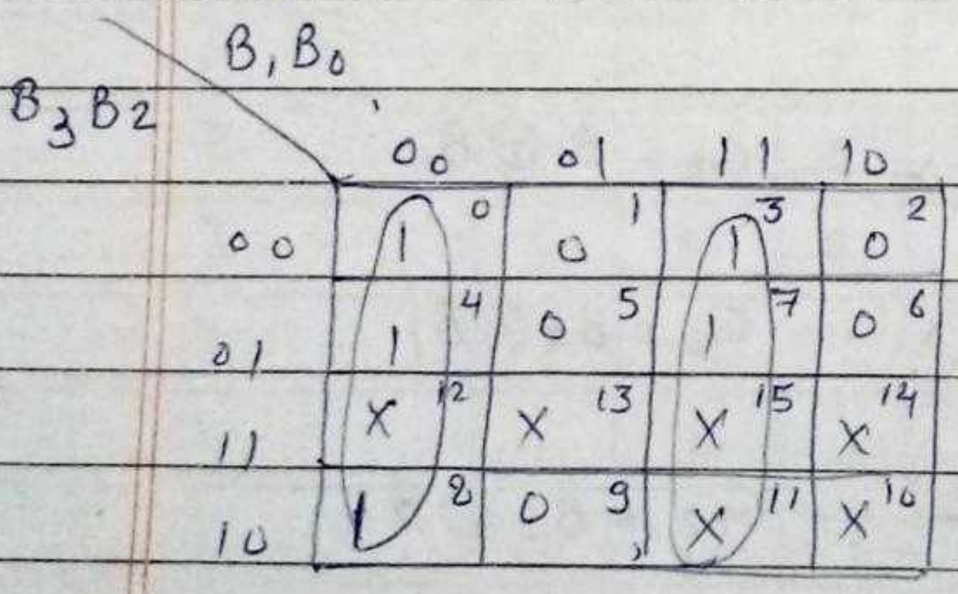
K-map for $E_3 = B_3 + B_2B_0 + B_2B_1$

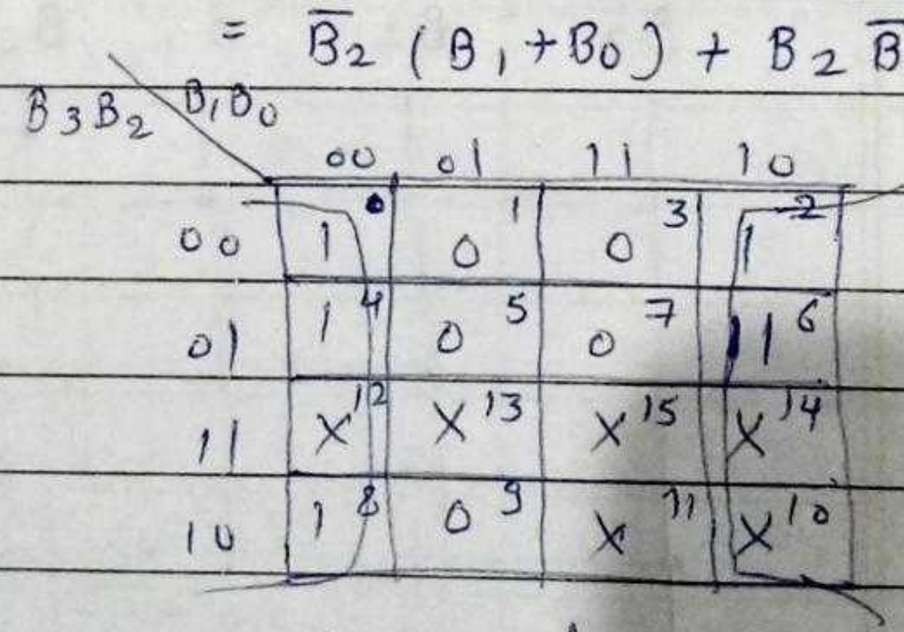
$= B_3 + B_2(B_0 + B_1)$

K-map for $E_2 =$

$= \overline{B_2}B_1 + \overline{B_2}B_0 + B_2\overline{B_1}\,\overline{B_0}$

$= \overline{B_2}(B_1 + B_0) + B_2\overline{B_1}\,\overline{B_0}$





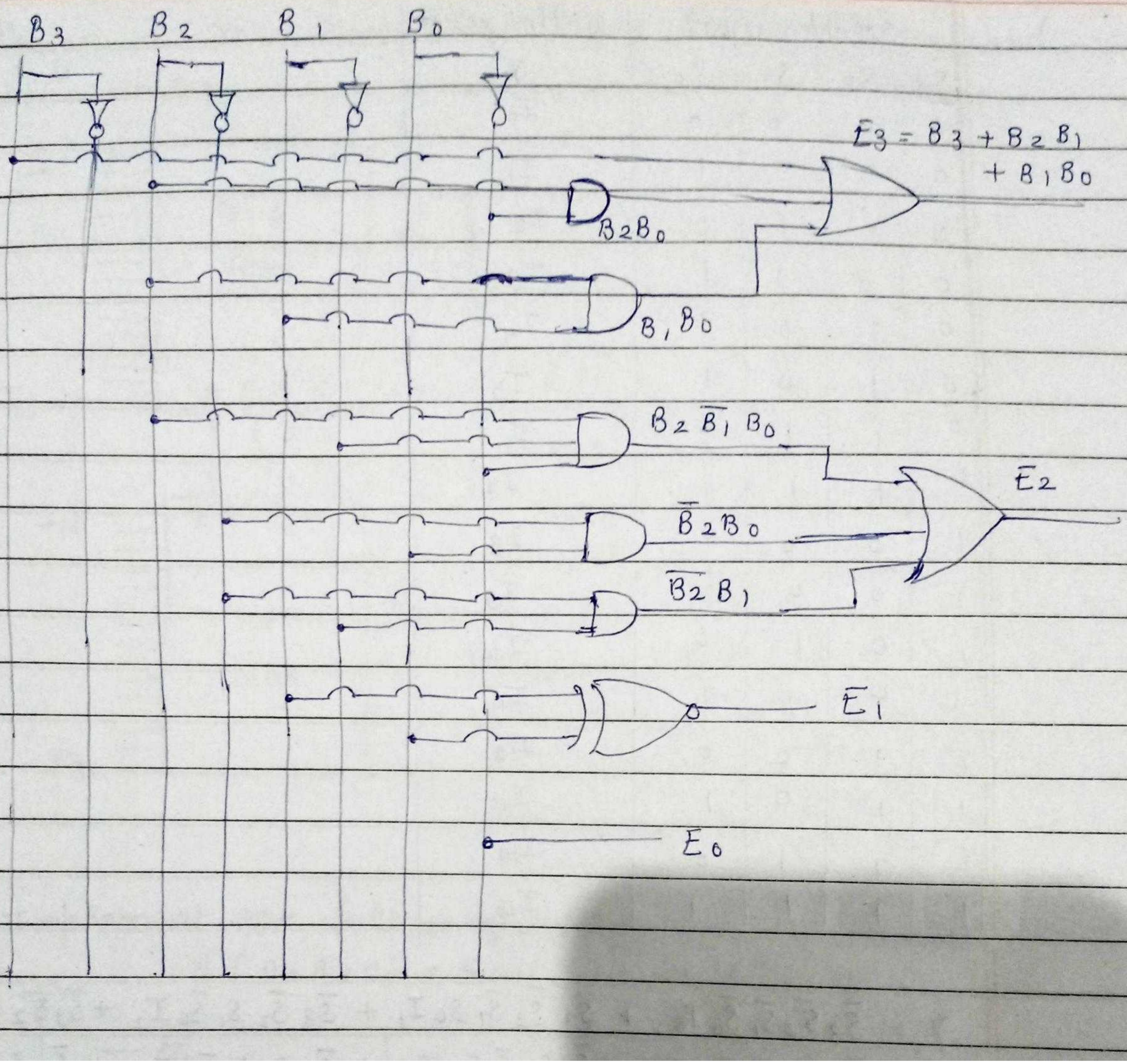
K-map for $E_1 = \overline{B_1}B_0 + B_1B_0$

$= (B_1 \odot B_0)$

K map for E_0

$E_0 = \overline{B_0}$

Date :
Page :



* Parity Generators / checkers "

Parity bit [0|0|1|0|0|0|0]

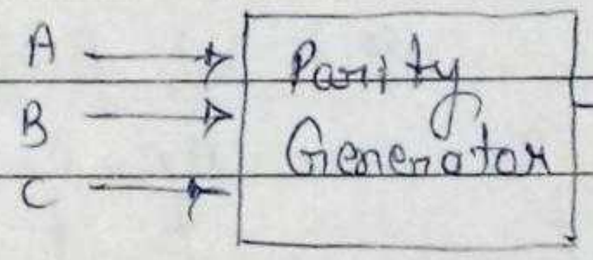
Even Parity [1|0|0|1|0|0|0|0] Number of 1's = 2
↳ Parity bit

Odd Parity [0|0|0|1|0|0|0|0] Number of 1's = 1
↳ Parity bit

→ In the even parity system, the added parity bit will make the total number of 1's an even number.
→ In the odd parity system, the added parity bit will make the total number of 1's an odd number.

* Parity Generator → Parity Generator is a logic circuit which generates the parity bits for even parity or odd parity.

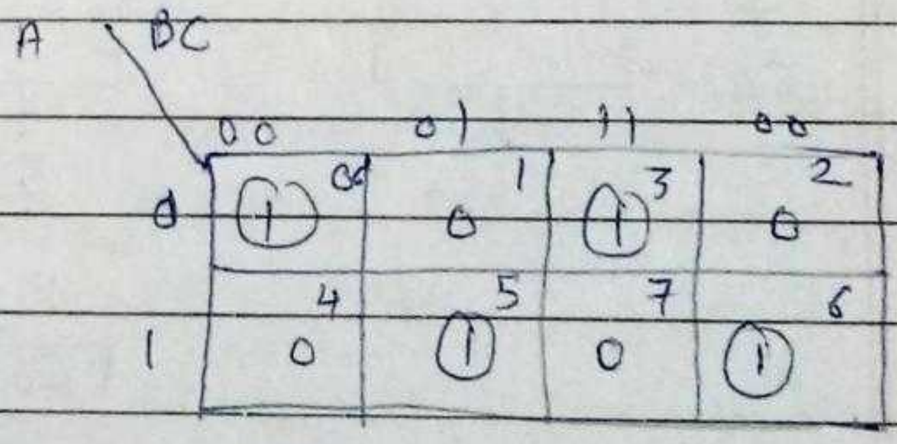
Input



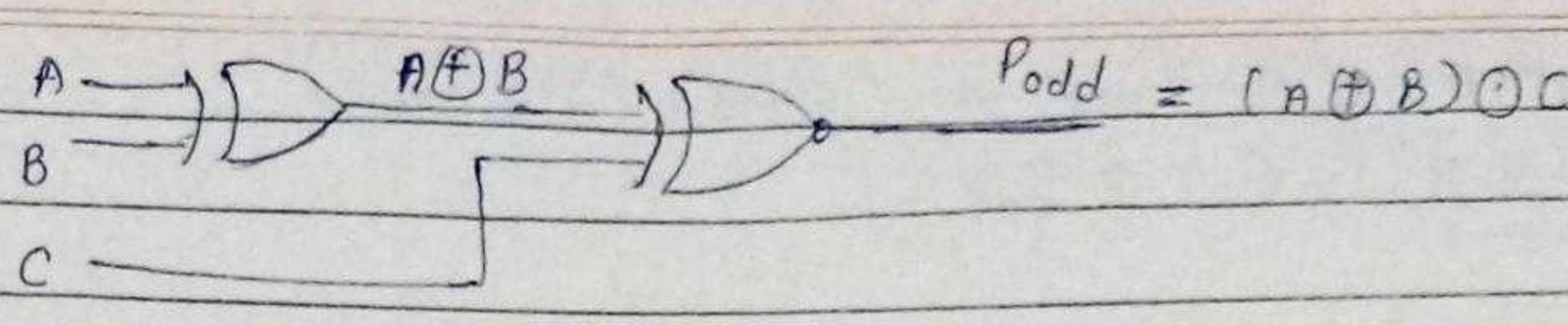
Input			Output
A	B	C	Podd
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Table

Block diagram



$$P_{odd} = \bar{A}\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC + AB\bar{C}$$
$$= \bar{C}(\bar{A}\bar{B} + AB) + C(A\bar{B} + \bar{A}B)$$
$$= \bar{C}(\overline{A \oplus B}) + C(A \oplus B)$$
$$= (A \oplus B) \oplus C$$



odd Parity Generator

Parity checker →

→ The Parity checker circuit is at the reciver.

→ The output of Parity checker circuit is denoted by Parity error output PEO.

PEO=1 --- if error is present i.e. if the received four bit word has an even parity

PEO=0 --- if there is no error i.e. if the received four bit word has an odd parity.

Input				Output
A	B	C	Podd	Pc(odd)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

K-map →

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

$$P_{c(odd)} = \bar{A}\bar{B}\bar{C}\bar{P}_{odd} + \bar{A}\bar{B}C P_{odd} + \bar{A}B\bar{C}P_{odd} + \bar{A}BC\bar{P}_{odd} + AB\bar{C}\bar{P}_{odd} + ABCP_{odd} + A\bar{B}\bar{C}P_{odd} + A\bar{B}C\bar{P}_{odd}$$

$$= \bar{A}\bar{B}(\bar{C}\bar{P}_{odd} + CP_{odd}) + \bar{A}B(\bar{C}P_{odd} + C\bar{P}_{odd}) + AB(\bar{C}\bar{P}_{odd} + CP_{odd}) + A\bar{B}(\bar{C}P_{odd} + C\bar{P}_{odd})$$

$$\bar{x}\bar{y} + xy = x \odot y$$

Date :
Page :

$= (A \odot B)(C \odot P_{odd}) + (A \oplus B)(C \oplus P_{odd})$

$\boxed{P_{e(odd)} = (A \oplus B) \odot (C \oplus P_{odd})}$

Logic diagram -