Unit 2

COMBINATIONAL LOGIC DESIGN / CIRCUITS

Introduction to logic Design:

The Boolean theorems and Demorgan's thm's are useful in manipulating the logic expressions. We can then realize the logic expression using gates. The no. of logic gates required for the realization of a logical expression should be reduced to the minimum possible value. This is possible if we can simplify the logical expressions. In this chapter we will discuss one of the simplification technique called Karnaugh map or K-map.

* Classification of Digital Circuits:

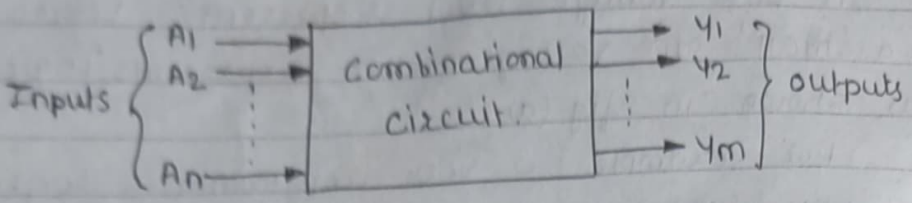
The digital ckts are basically classified into two categories namely:

1) Combinational ckts.
2) Sequential ckts.

1) Combinational circuits:

A combinational ckt is a logic ckt the o/p of which depends only on the combination of i/p's. The o/p does not depend on the past value of i/p's or o/p's. Hence combinational ckt do not require any memory.

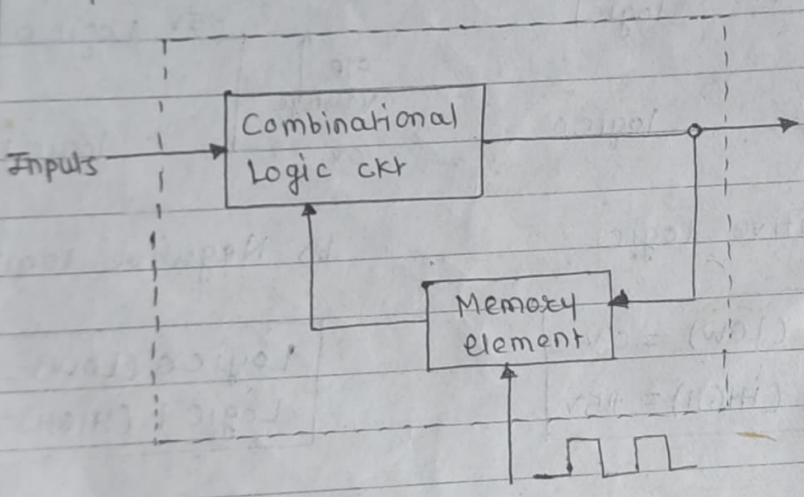
The block diagram is shown in fig.



A combinational ckt can have a no. of i/ps & a no. of o/p's. The ckt of fig. has 'n' i/p's & 'm' o/p's.

2) Sequential circuits:

The o/p of a sequential ckt depends on the present i/p's, the previous o/p & the sequence in which the i/p's are applied.



To provide the previous i/p or o/p a memory element is required to be used. Thus a sequential ckt need a memory element.

Fig. shows the block diagram of a sequential ckt which includes the memory element in the feedback path.

Comparison of Combinational & Sequential ckts:

	Combinational ckts		Sequential ckts.
1)	Output depends on present i/p's at that instant of time.	1)	output depends on present i/p & past inputs / outputs
2)	Memory is not necessary	2)	Memory is necessary
3)	Clock i/p is not necessary	3)	clock i/p is necessary.
4)	eg: Adders, subtractors, Code converters	4)	eg: Flipflops, shift registers, counters.

★ Code Converters:

In this section, we are going to design following converter ckts.

1) Binary to Gray code converter
2) Gray code to binary code converter

• General procedure for code converter design:

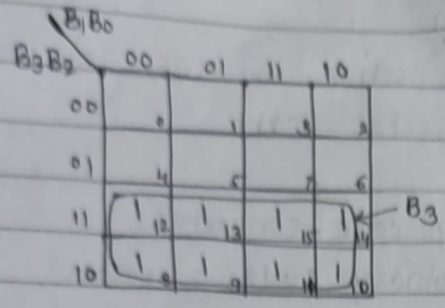
1) Write the truth table
2) Write the K-map & obtain the reduced eqns
3) Draw the logic diagram.

★ Binary to Gray Code Converter:

Decimal	Binary i/p's				Gray o/p's			
	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

8 4 2 1
8 8 4 2
4 4 2 2
4 2 2 2
2 2 2 1

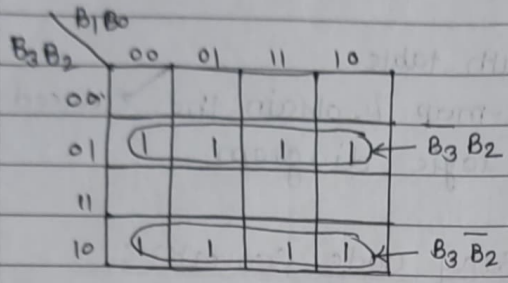
K-map for G_3



o/p eqn:

$\therefore G_3 = B_3$

K-map for G_2

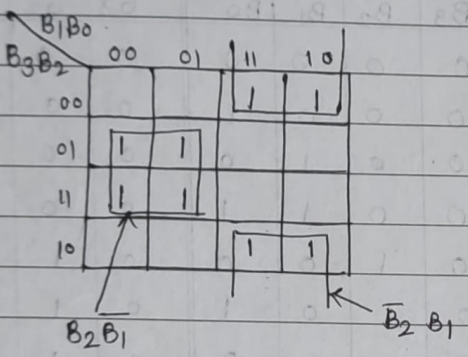


$\therefore$ o/p eqn

$\therefore G_2 = \bar{B_3} B_2 + \bar{B_2} B_3$

$= B_2 \oplus B_3$

K-map for G_1

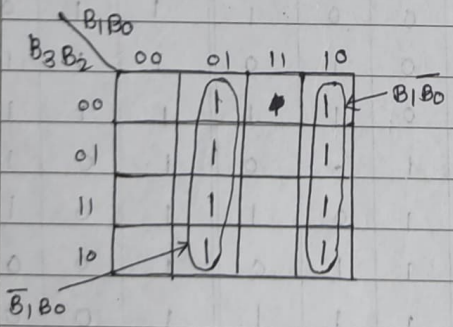


$\therefore$ o/p eqn

$G_1 = \bar{B_1} B_2 + \bar{B_2} B_1$

$= B_1 \oplus B_2$

K-map for G_0

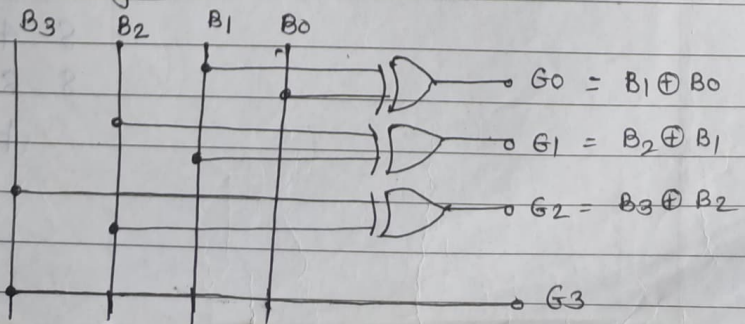


$\therefore$ o/p eqn

$G_0 = B_0 \bar{B_1} + \bar{B_0} B_1$

$= B_0 \oplus B_1$

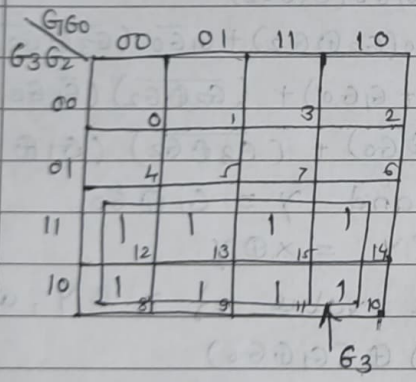
★ Logic diagram



* Gray to binary code converter :

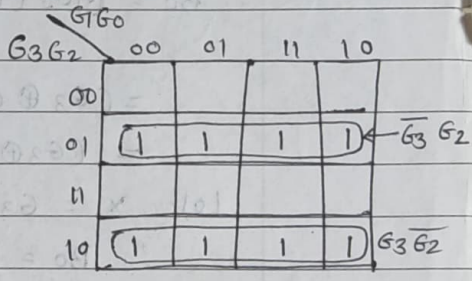
Decimal	Gray code i/p's				Binary o/p's			
	G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	0	0
7	0	1	1	1	0	1	0	1
8	1	0	0	0	1	1	1	1
9	1	0	0	1	1	1	1	0
10	1	0	1	0	1	1	0	0
11	1	0	1	1	1	1	0	1
12	1	1	0	0	1	0	0	0
13	1	1	0	1	1	0	0	1
14	1	1	1	0	1	0	1	1
15	1	1	1	1	1	0	1	0

* K-map for B_3



$\therefore B_3 = G_3$

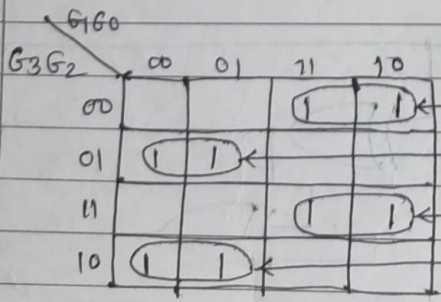
* K-map for B_2



$\therefore B_2 = \bar{G_3} G_2 + G_3 \bar{G_2}$

$= G_3 \oplus G_2$

* K-map for B_1



$$\therefore B_1 = \bar{G_3}\bar{G_2}G_1 + \bar{G_3}G_2\bar{G_1} + G_3G_2G_1 + G_3\bar{G_2}\bar{G_1}$$

$$= \bar{G_1}(\underbrace{\bar{G_3}G_2 + G_3\bar{G_2}}_{\text{EX-OR}}) + G_1(\underbrace{\bar{G_3}\bar{G_2} + G_3G_2}_{\text{EX-NOR}})$$

$$= \bar{G_1}(G_3 \oplus G_2) + G_1(\overline{G_3 \oplus G_2})$$

Let $G_3 \oplus G_2 = X$

$$\therefore B_1 = \bar{G_1}X + G_1\bar{X}$$

$$= G_1 \oplus X$$

$$= G_1 \oplus G_2 \oplus G_3$$

* K-map for B_0

G_3G_2 \ G_1G_0	00	01	11	10	
$\bar{G_3}\bar{G_2}$		(1)		(1)	$\bar{G_3}\bar{G_2}G_1\bar{G_0}$
$\bar{G_3}G_2$	(1)		(1)		$\bar{G_3}G_2G_1G_0$
G_3G_2		(1)		(1)	$G_3G_2G_1G_0$
$G_3\bar{G_2}$	(1)		(1)		$G_3\bar{G_2}G_1G_0$

$$\therefore B_0 = \bar{G_3}\bar{G_2}\bar{G_1}G_0 + \bar{G_3}G_2\bar{G_1}\bar{G_0} + \bar{G_3}\bar{G_2}G_1\bar{G_0} + \bar{G_3}G_2G_1G_0$$
$$+ G_3G_2\bar{G_1}G_0 + G_3G_2G_1\bar{G_0} + G_3\bar{G_2}\bar{G_1}\bar{G_0} + G_3\bar{G_2}G_1G_0$$

$$= \bar{G_1}\bar{G_0}(\underbrace{\bar{G_3}G_2 + G_3\bar{G_2}}_{\text{EX-OR}}) + \bar{G_1}G_0(\underbrace{\bar{G_3}\bar{G_2} + G_3G_2}_{\text{EX-NOR}})$$
$$+ G_1G_0(\underbrace{\bar{G_3}G_2 + G_3\bar{G_2}}_{\text{EX-OR}}) + G_1\bar{G_0}(\underbrace{\bar{G_3}\bar{G_2} + G_3G_2}_{\text{EX-NOR}})$$

$$\therefore B_0 = \bar{G_1}\bar{G_0}(G_3 \oplus G_2) + \bar{G_1}G_0(\overline{G_3 \oplus G_2})$$
$$+ G_1G_0(G_3 \oplus G_2) + G_1\bar{G_0}(\overline{G_3 \oplus G_2})$$

$$= (G_3 \oplus G_2)(\bar{G_1}\bar{G_0} + G_1G_0) + (\overline{G_3 \oplus G_2})(\bar{G_1}G_0 + G_1\bar{G_0})$$

$$= (G_3 \oplus G_2)(\overline{G_1 \oplus G_0}) + (\overline{G_3 \oplus G_2})(G_1 \oplus G_0)$$

Let $x = G_3 \oplus G_2$ and $y = G_1 \oplus G_0$

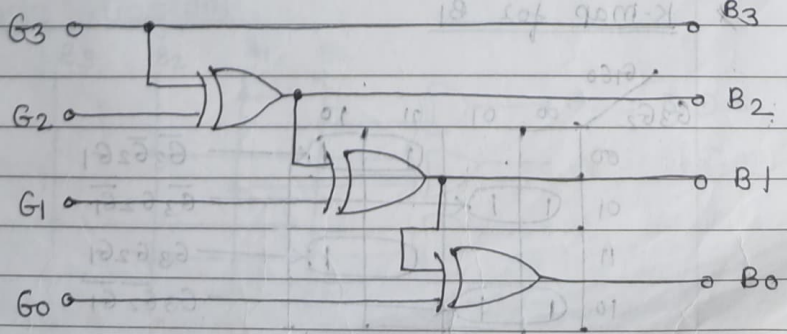
$$\therefore B_0 = x\bar{y} + \bar{x}y = x \oplus y$$

∴ Substituting the value of x & y, we get.

$$B_0 = (G_3 \oplus G_2) \oplus (G_1 \oplus G_0)$$

$$= G_3 \oplus G_2 \oplus G_1 \oplus G_0$$

* Logic dig



* Adders :

Rules :	Sum	carry
1) 0 + 0 =	0	0
2) 0 + 1 =	1	0
3) 1 + 0 =	1	0
4) 1 + 1 =	0	1
5) 1 + 1 + 1 =	1	1

Types of Binary Adders :

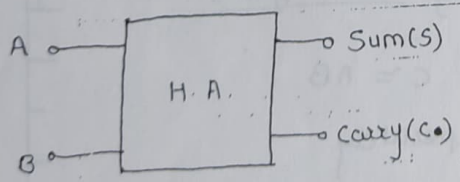
① Half adder ② Full adder.

① Half adder :

Half adder is a combinational ckt with two i/p's & two o/p's. It is the basic building block for addition of two single bit numbers.

This ckt has two i/p's namely A & B and two o/p's namely 'sum' & 'carry'

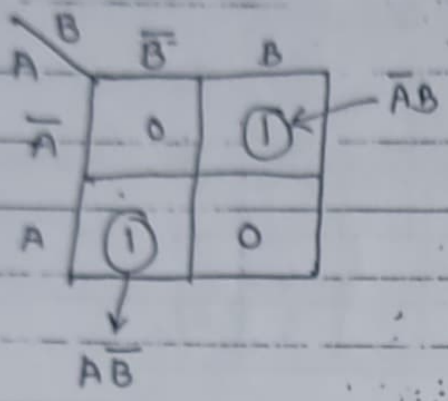
The block dig. & truth table of half adder is given below.



A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-map & simplified exp^{n}s for outputs:

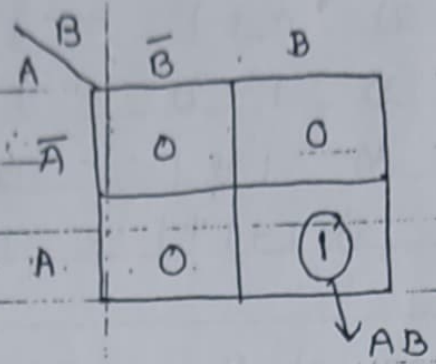
① K-map for sum



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

$S = A \oplus B$

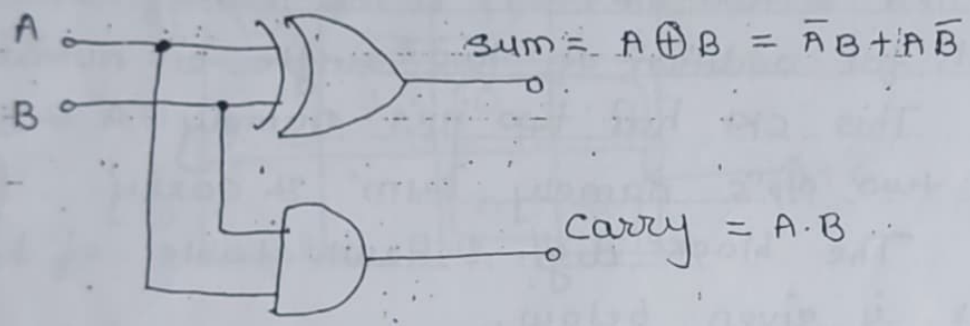
K-map for carry



$\therefore$ carry $= AB$

$C = AB$

Half adder ckt:



Half adder using only NAND gates:

① $S = \bar{A}B + A\bar{B}$ ② $C = AB$

Consider the eq's seperately

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

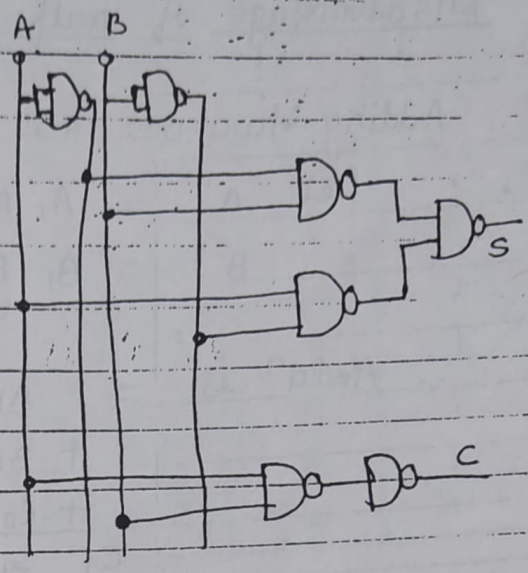
Take double inversion, we get

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

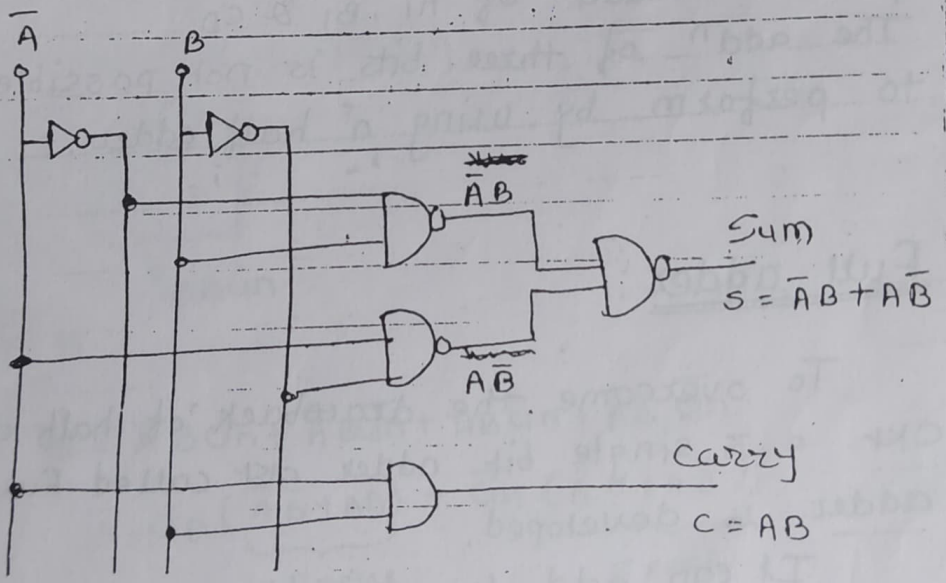
Use demorgan's thm

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

Similarly, $C = AB = \overline{\overline{AB}}$



★ Half adder using basic gates



Disadvantage of half adder:

Adding two nos. A & B,

Let A = A_1 A_0

B = B_1 B_0

∴ Addⁿ is = A_1 A_0

+ B_1 B_0

+ C_0

C_1 S_1 S_0

C_0 ⇒ carry generated from the addⁿ (A_0 + B_0)

- A half adder can Add A_0 & B_0 to produce S_0 & C_0, But the addⁿ of next bits requires the addⁿ of A_1, B_1 & C_0.
- The addⁿ of three bits is not possible to perform by using a half adder.

✶ Full adder:

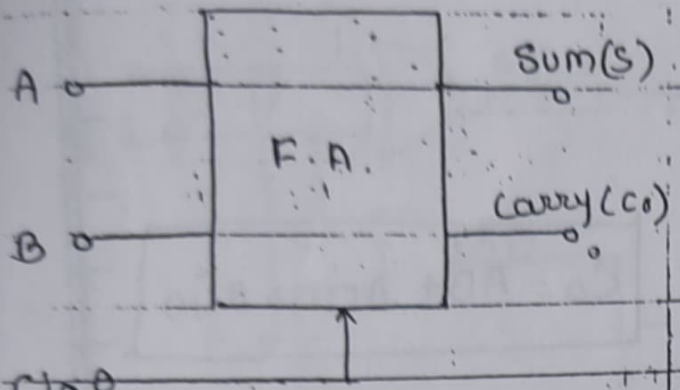
To overcome the drawback of half adder ckt. a 3 single bit adder ckt called Full adder is developed.

It can add two one bit nos. A & B & carry cin. The full adder is a three input & two output combinational ckt.

The block dig. & truth table of full adder is show below.

PAGE NO:
DATE:

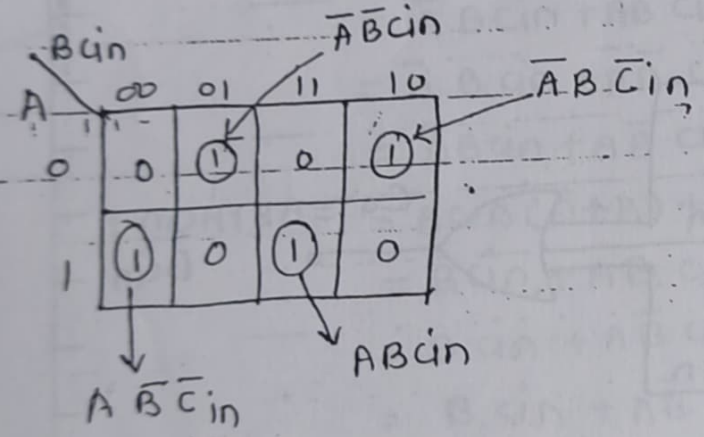
Block diagram



Truth table

A	B	Cin	S	Co
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



$$S = \bar{A}\bar{B}Cin + \bar{A}B\bar{Cin} + AB\bar{Cin} + A\bar{B}\bar{Cin}$$

$$= Cin(\underbrace{\bar{A}\bar{B} + AB}_{\text{EX-NOR}}) + \bar{Cin}(\underbrace{\bar{A}B + A\bar{B}}_{\text{EX-OR}})$$

$$S = Cin(\overline{A \oplus B}) + \bar{Cin}(A \oplus B)$$

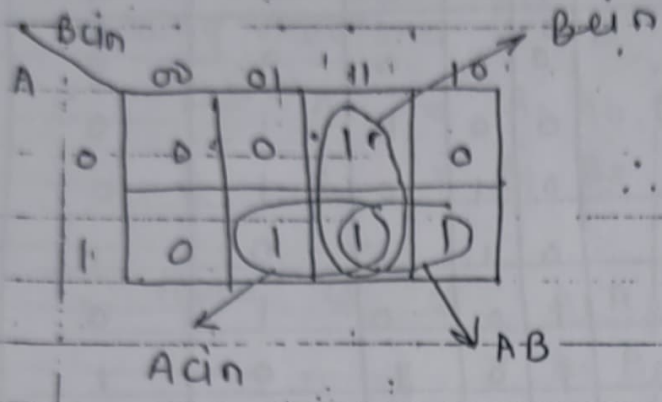
$$= Cin\,\bar{x} + \bar{Cin}\,x = Cin \oplus x$$

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

$$\therefore S = Cin \oplus A \oplus B$$

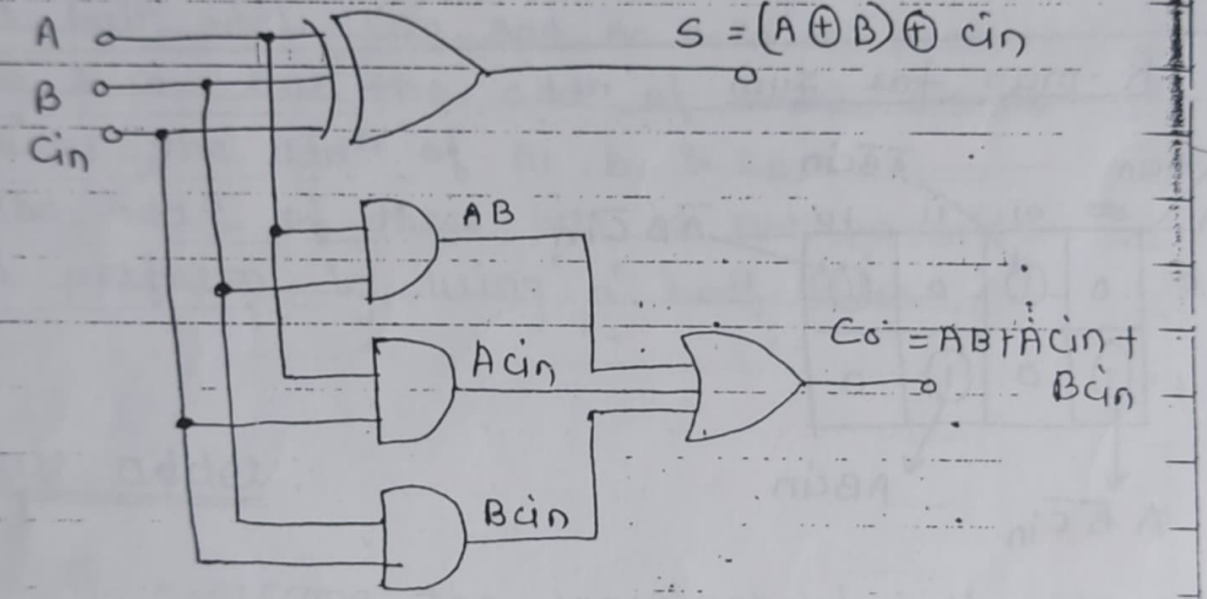
Let
$X = \bar{A}B + A\bar{B}$
$\therefore \bar{A}B + A\bar{B} = A \oplus B$

K-map for Co output:

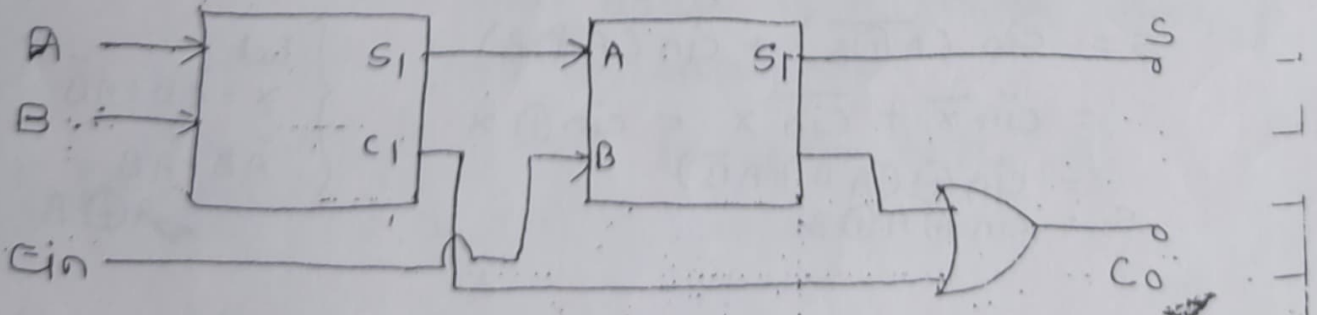


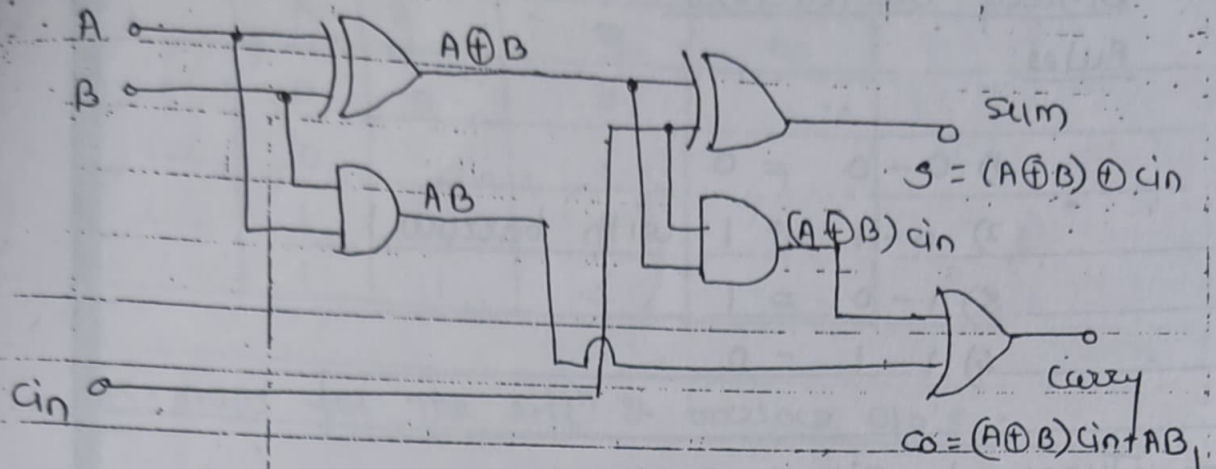
$\therefore$ $C_o = AB + A\bar{C}in + B\bar{C}in$

Logic dig:



✱ Full adder using two half adders:





Proof:

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

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

$$= \bar{A}BC_{in} + A\bar{B}C_{in} + AB$$

$$= \bar{A}BC_{in} + A\bar{B}C_{in} + AB(1 + C_{in})$$

$$= \bar{A}BC_{in} + A\bar{B}C_{in} + AB + ABC_{in}$$

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

$$= BC_{in} + A\bar{B}C_{in} + AB$$

$$= BC_{in} + A\bar{B}C_{in} + AB(1 + C_{in})$$

$$= BC_{in} + A\bar{B}C_{in} + AB + ABC_{in}$$

$$= BC_{in} + AC_{in}(\bar{B} + B) + AB$$

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

$$\boxed{C_o = AB + AC_{in} + BC_{in}}$$

—x—

* Binary Subtractors:

Rules:

1) 0 - 0 = 0
2) 0 - 1 = 1 with borrow 1
3) 1 - 0 = 1
4) 1 - 1 = 0

* Types of Binary Subtractors:

1) Half Subtractor
2) Full Subtractor

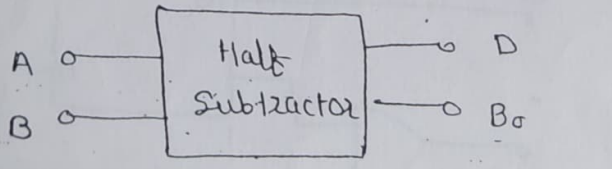
① Half Subtractor

Half subtractor is a combinational ckt with two i/p's & two o/p's (difference & borrow)

It produces the difference betn the two binary bits at the i/p & also produces an o/p (Borrow) to indicate if a 1 has been borrowed.

In the subtraction (A-B), A is called as minued & B is called as subtrahend bit.

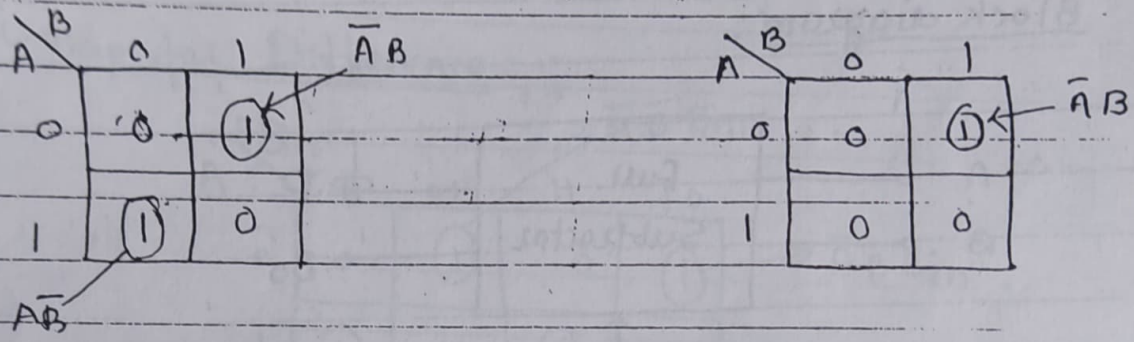
Block diagram



Truth table :

A	B	D	Bo
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K-maps for the diff^{n} & borrow o/p's :

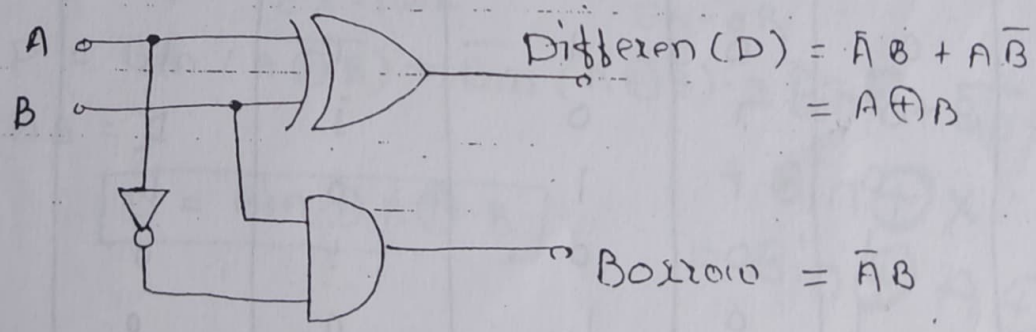


$\therefore D = \bar{A}B + A\bar{B}$

$= A \oplus B$

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

Logic diagram :

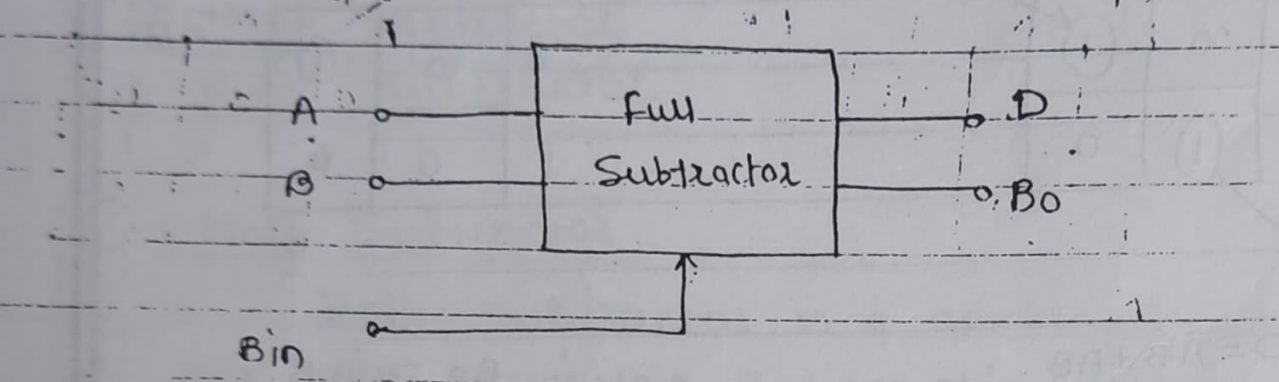


Full Subtractor:

The full subtractor is a combinational CKt. with three i/p's A, B & Bin & two o/p's D & Bo

A is the minuend, B is subtrahed, Bin is the borrow produced by the previous stage. D is the difference o/p & Bo is the borrow o/p.

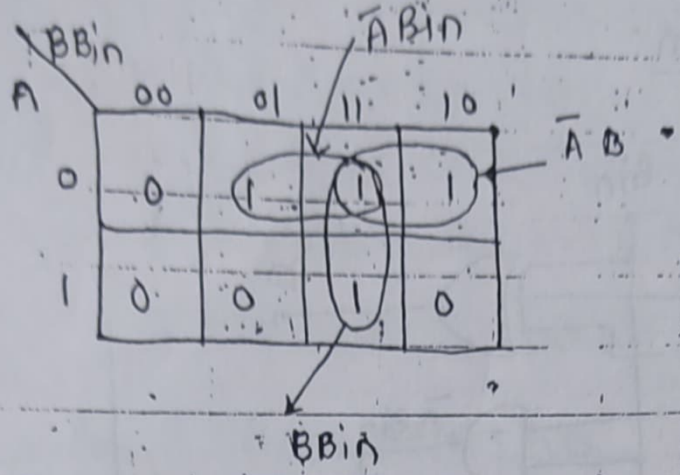
Block diagram:



Truth table:

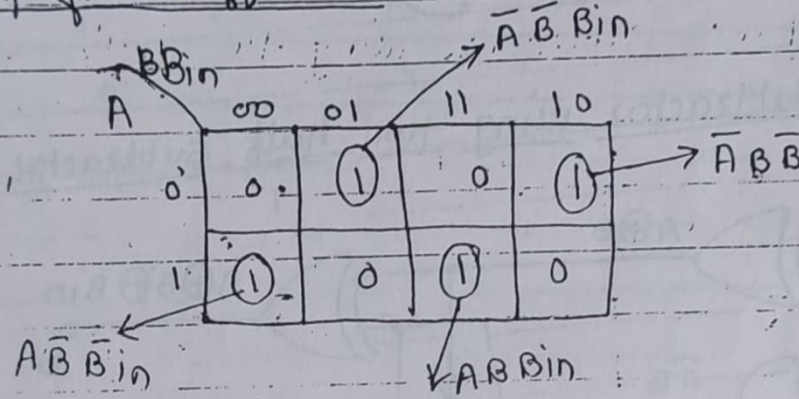
A	B	Bin	D	Bo
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-map for borrow:



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

K-map for Difference:



$$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$$

$$= B_{in}(\underbrace{\bar{A}\bar{B} + AB}_{\text{EX-NOR}}) + \bar{B}_{in}(\underbrace{\bar{A}B + A\bar{B}}_{\text{EX-OR}})$$

$$D = B_{in}(\overline{A \oplus B}) + \bar{B}_{in}(A \oplus B) = B_{in}\bar{X} + \bar{B}_{in}X$$

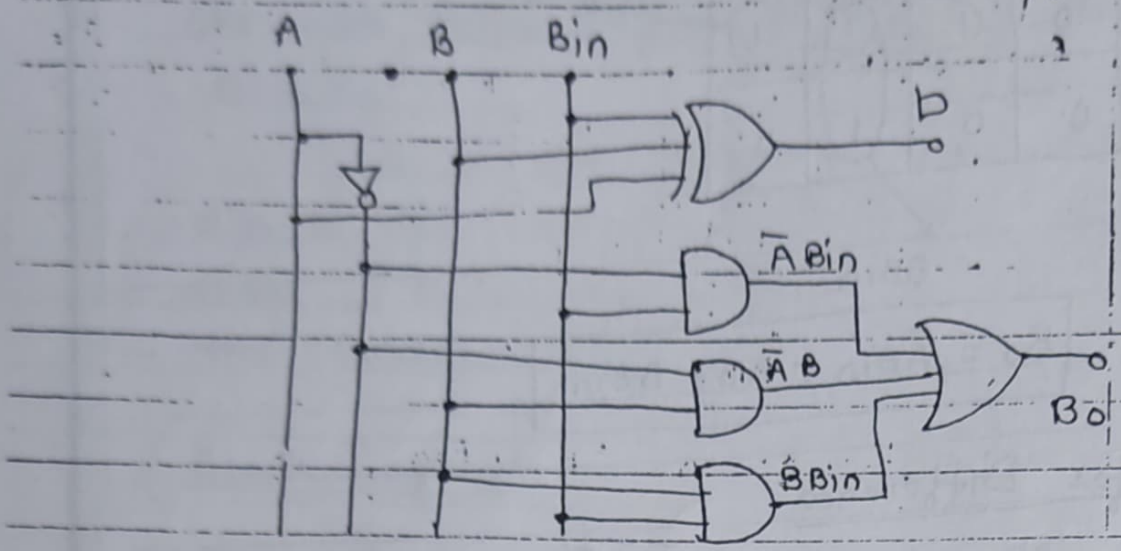
Let $A \oplus B = X$

$$= B_{in} \oplus X$$

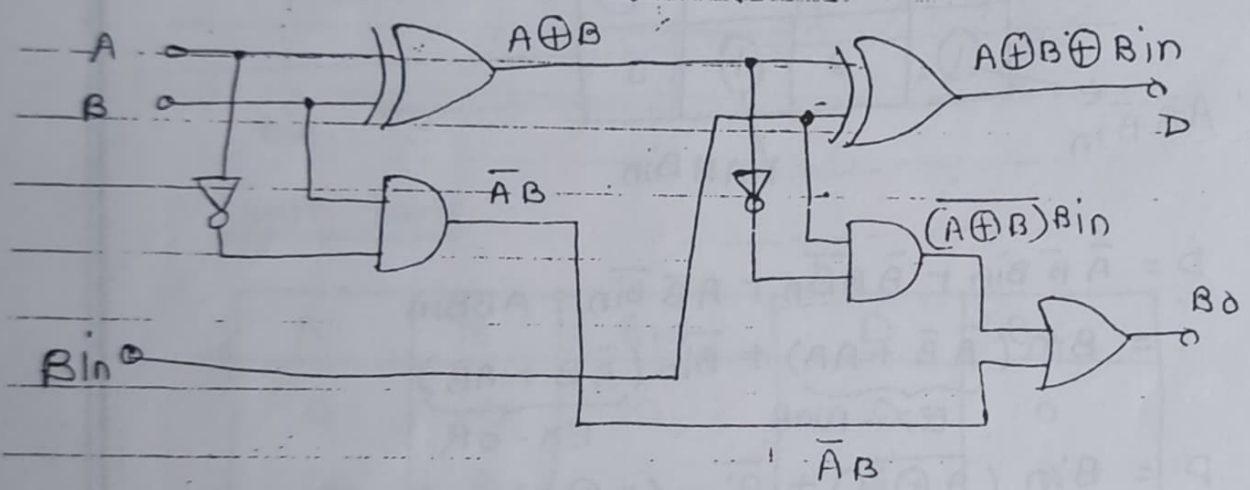
$$= B_{in} \oplus A \oplus B$$

$$\therefore \boxed{D = B_{in} \oplus A \oplus B}$$

Logic diagram:



Full Subtractor using two half subtractor



$$Bo = (\overline{A \oplus B})\, Bin + \bar{A}B$$

$$= (\overline{\bar{A}B + A\bar{B}})\, Bin + \bar{A}B$$

$$= (\bar{A}\bar{B} + AB)\, Bin + \bar{A}B$$

$$= \bar{A}\bar{B}\,Bin + AB\,Bin + \bar{A}B$$

$$= \bar{A}\bar{B}\,Bin + AB\,Bin + \bar{A}B(1 + Bin)$$

$$= \bar{A}\bar{B}\,Bin + AB\,Bin + \bar{A}B + \bar{A}B\,Bin$$

$$B_0 = \bar{A}\bar{B}B_{in} + BB_{in}(A+\bar{A}) + \bar{A}B$$

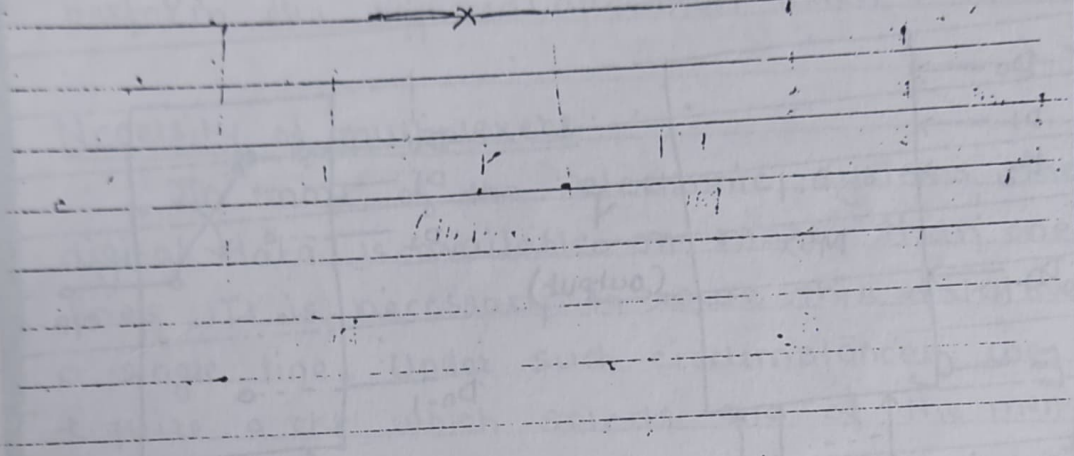
$$= \bar{A}\bar{B}B_{in} + BB_{in} + \bar{A}B \quad \ldots (1)$$

$$= \bar{A}\bar{B}B_{in} + BB_{in} + \bar{A}B(1+B_{in})$$

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

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

$$B_0 = \bar{A}B_{in} + BB_{in} + \bar{A}B$$



* Multiplexer (Data Selector) :

Multiplexer is a special type of combinational ckt. The block dig of multiplexer & its equivalent ckt is shown in fig (a) & (b) resp

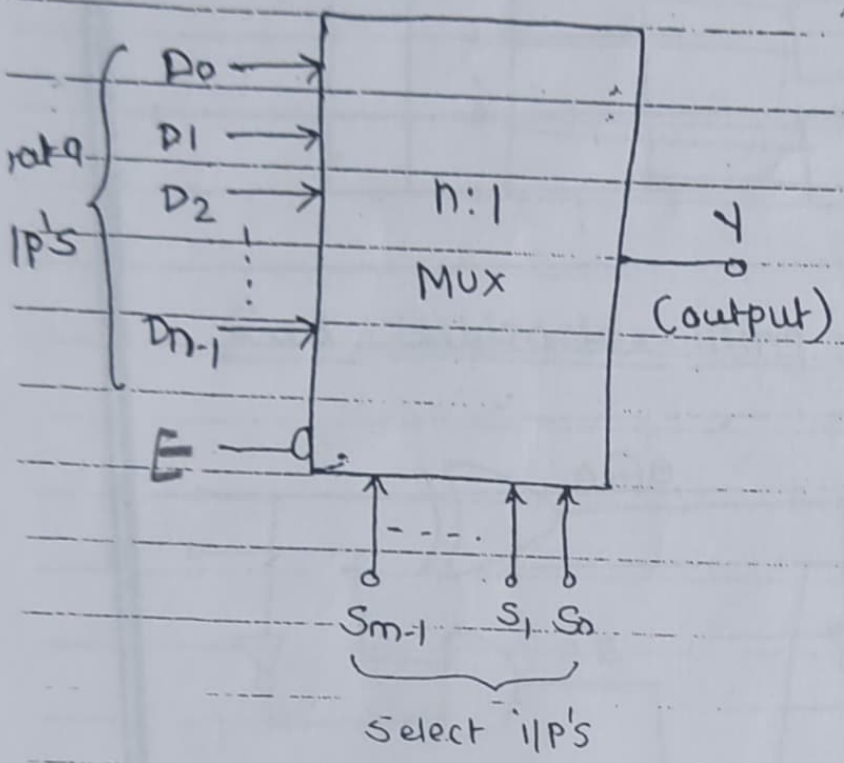


fig (a)

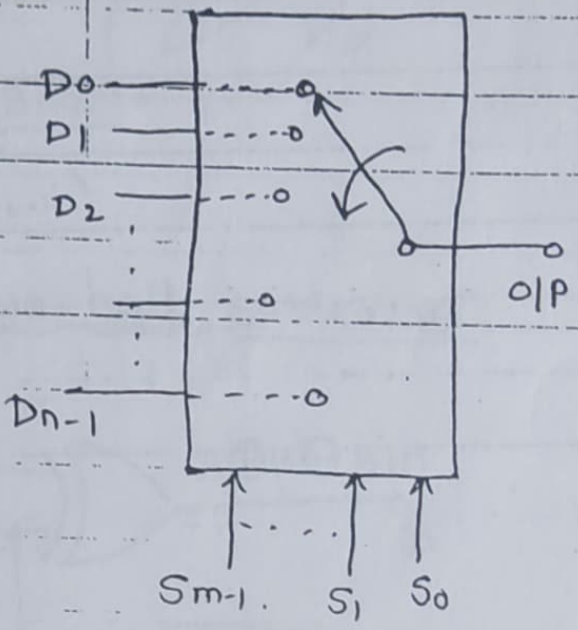


fig (b)

There are n data I/P's, one O/P & a m select I/P's, with $2^m = n$. A multiplexer is a digital ckt. which selects one of the n data I/P's & routes it to the O/P. The selection of one of the n I/P's is done by the select I/P's. To select n I/P's we need m select lines such that $2^m = n$. Depending on the digital

code applied at the select i/p's, one out of n data sources is selected & transmitted to the single o/p Y.

E is called as a strobe or enable i/p which is useful for cascading. It is generally an active low terminal that means it will perform the required operation when it is low.

* Necessity of multiplexers:

In most of the electronic systems, the digital data is available on more than one lines. It is necessary to route this data over a single line. Under such circumstances we require a ckt which selects one of the many i/p's at a time. This ckt is nothing else but a multiplexer. which has many i/p's, one o/p & some select i/p's. Multiplexer improves the reliability of the digital system because it reduces the no. of external wired connections.

* Advantages of multiplexers:

1) It reduces the no of wires.
2) So it reduces the ckt complexity & cost.
3) We can implement many combinational ckt. using MUX
4) It simplifies the logic design.
5) It does not need the K-map & simplification.

Types of multiplexers:

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

1) 2:1 multiplexer:

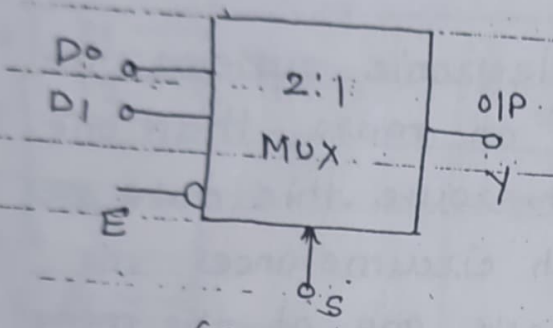


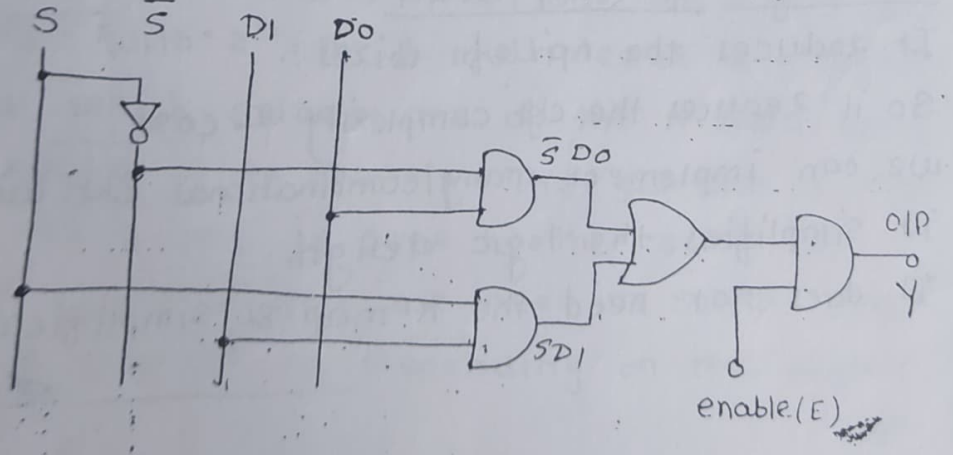
fig (a) - Block dig

Select i/p	output
S	Y
0	D0
1	D1

(b) Truth table

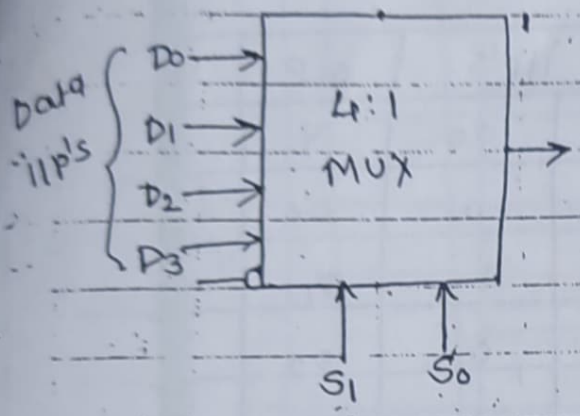
$$\therefore \boxed{Y = \bar{S} D_0 + S D_1} \leftarrow \text{logic exp}^{z}$$

Logic diagram:



2) 4:1 multiplexer :

a) Block diagram :

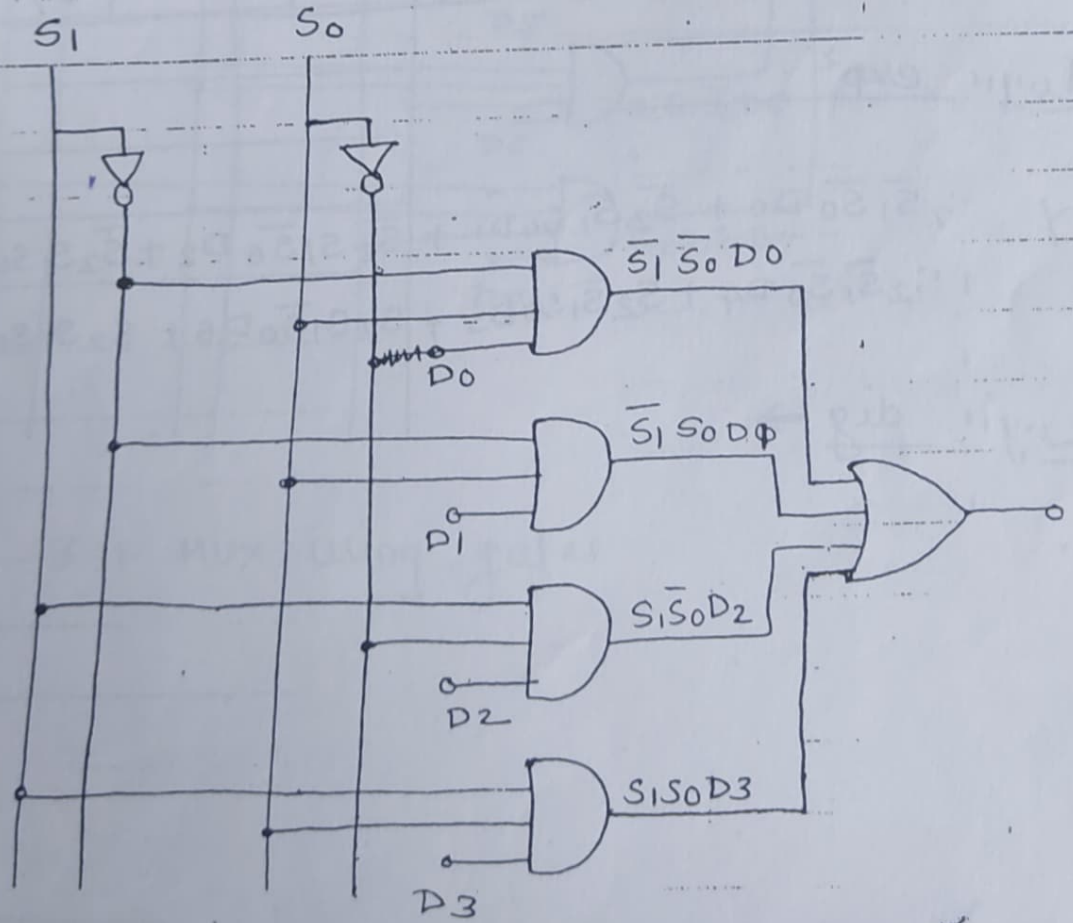


b) Truth table

Select i/p's		O/P
S_1	S_0	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

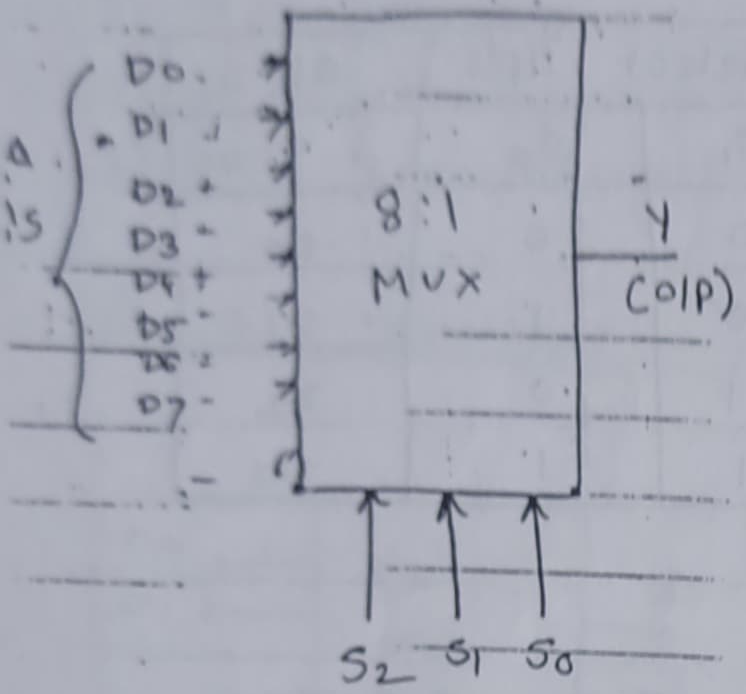
Logic exp^2

$$Y = \bar{S_1}\bar{S_0}D0 + \bar{S_1}S_0D1 + S_1\bar{S_0}D_2 + S_1S_0D3$$



* 8:1 Multiplexer :

(a) Block dig :



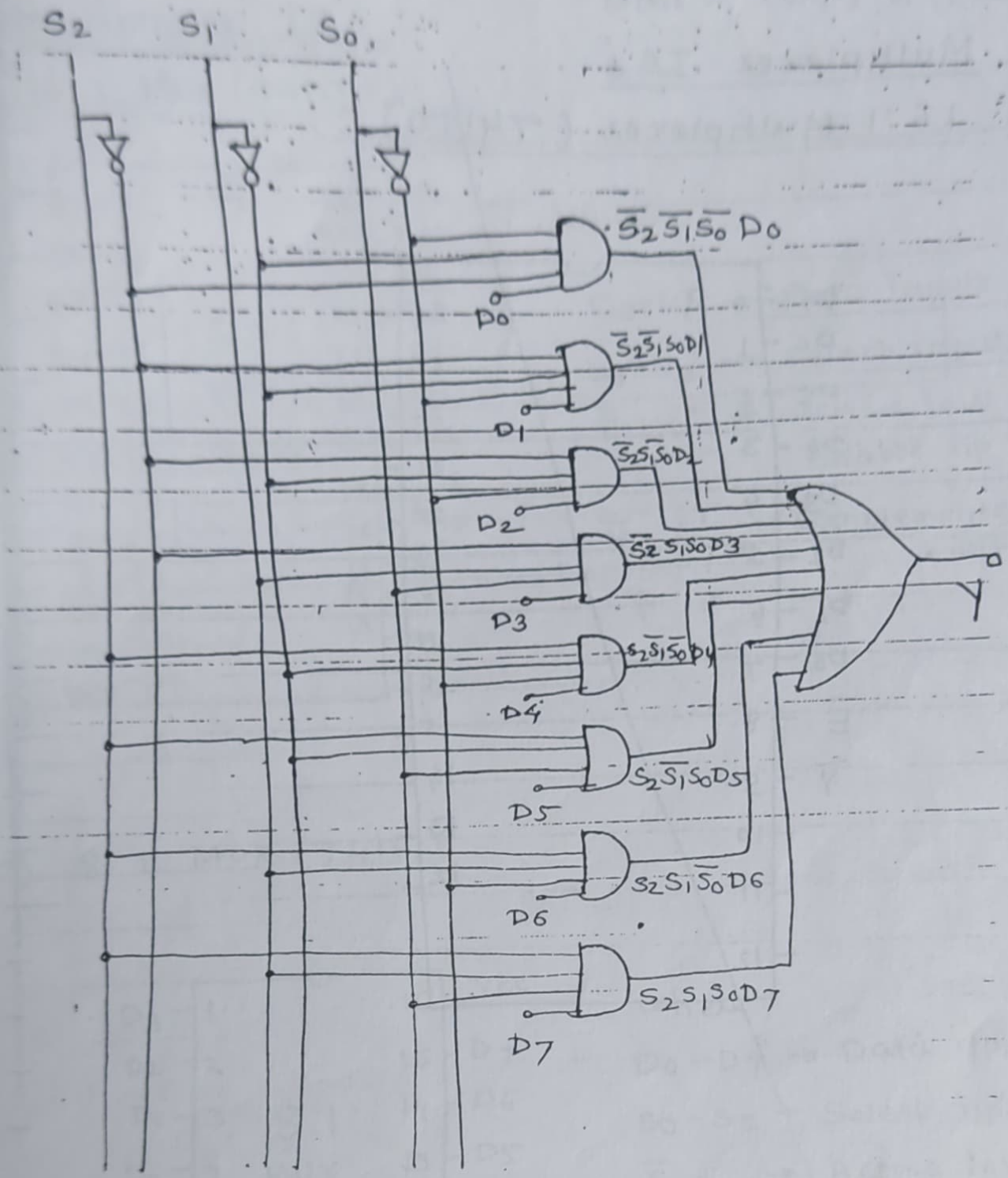
b) Truth table :

Select i/p's			O/P
S_2	S_1	S_0	Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

Logic exp^2 :

$Y = \bar{S_2}\bar{S_1}\bar{S_0}D_0 + \bar{S_2}\bar{S_1}S_0D_1 + \bar{S_2}S_1\bar{S_0}D_2 + \bar{S_2}S_1S_0D_3$
$+ S_2\bar{S_1}\bar{S_0}D_4 + S_2\bar{S_1}S_0D_5 + S_2S_1\bar{S_0}D_6 + S_2S_1S_0D_7$

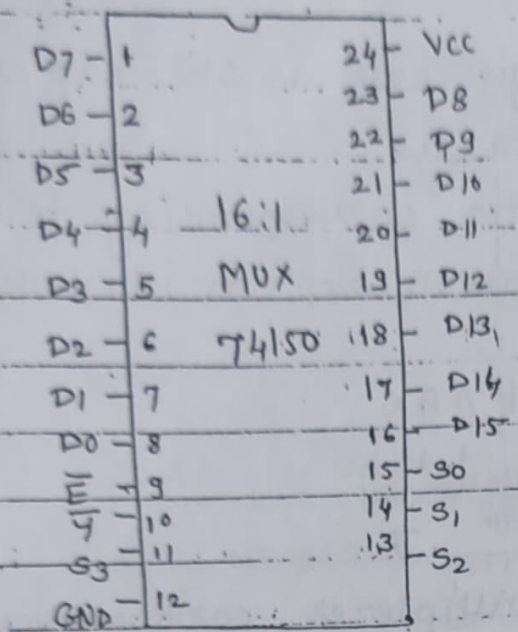
Logic dig →



8:1 MUX using gates

Multiplexer IC's :

1) 16:1 MUX (74150) :



Where,

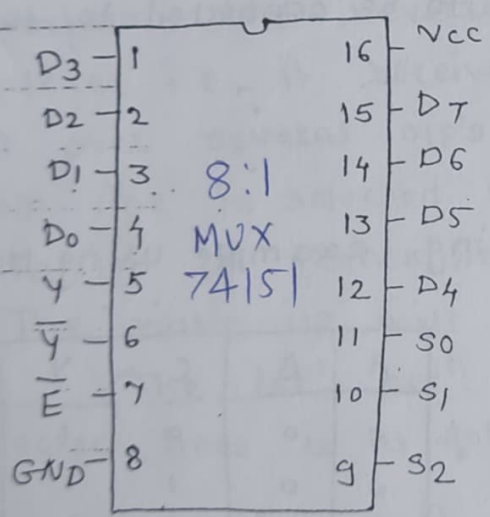
$D_0 - D_{15}$ → Data inputs

$S_0 - S_3$ → Select inputs

$\overline{E}$ → Active low enable i/p

$\overline{Y}$ → Complemented o/p.

2) 8:1 MUX (74151)



where,

$D_0 - D_7$ → Data i/p's

$S_0 - S_2$ → Select i/p's

$\overline{E}$ → Active low enable i/p

Y → o/p

$\overline{Y}$ → Complemented o/p.

* Use of Mux for combinational ckt design:

procedure:

(a) A truth table or logic exp^2 is std. sop or pos form is given to us.

(1) Identify the decimal no. corresponding to each minterm in the given exp^t as illustrated below.

$$Y = \bar{A}\bar{B}\bar{C} + A\bar{B}C + AB\bar{C}$$

000 101 111

0 5 7

2) The i/p lines of a multiplexer, corresponding to these numbers (0, 5, 7) are connected to logic 1 level.

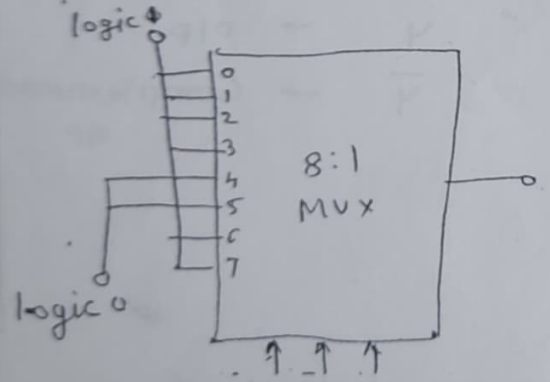
3) All the other i/p lines are connected to the ~~select i/p's~~ logic 0 level.

4) The i/p's (A, B, C) are to be connected to the Select inputs.

Example:

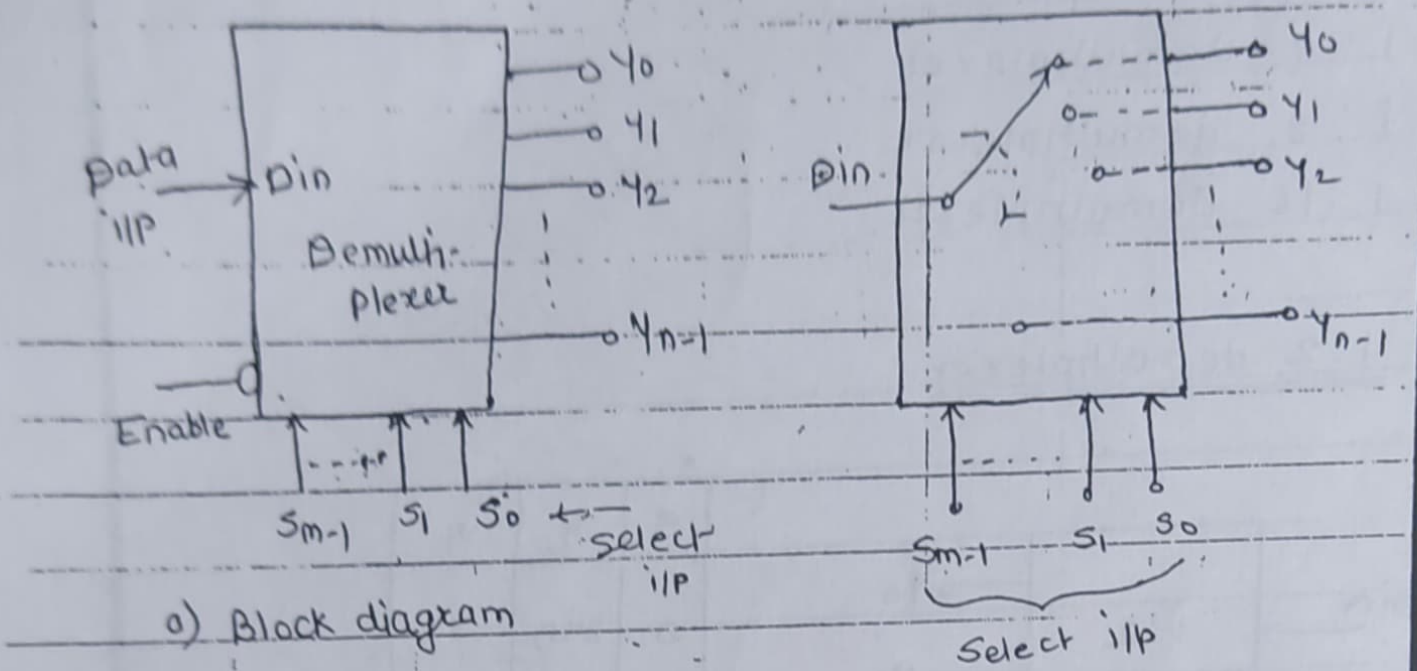
1) Implement the following example using MUX

$Y = \Sigma m(0, 1, 2, 3, 6, 7)$



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

* Demultiplexers:



a) Block diagram

b) equiv. ckt.

The block diagram of a demultiplexer or decoder is shown in fig. It has only one i/p, 'n' outputs & 'm' select inputs. A demux- performs the reverse operation of a multi- plexer i.e. it receives one i/p & distributes it over several o/p's. At a time only one o/p line is selected by the select lines & the i/p is transmitted to the selected o/p line. The enable i/p will enable the demultiplexer.

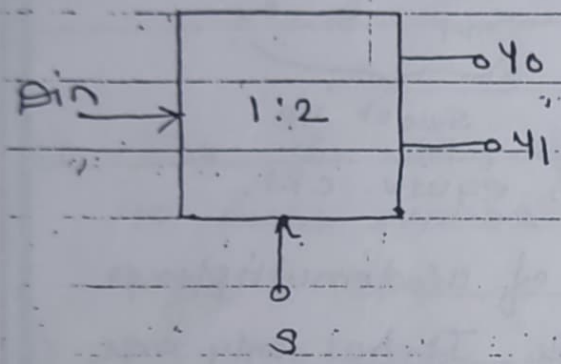
The relⁿ betⁿ the n o/p lines & m select lines is as follows:

$$n = 2^m$$

Types of demultiplexers:

1. 1:2 demultiplexer
2. 1:4 demultiplexer
3. 1:8 demultiplexer
4. 1:16 demultiplexer.

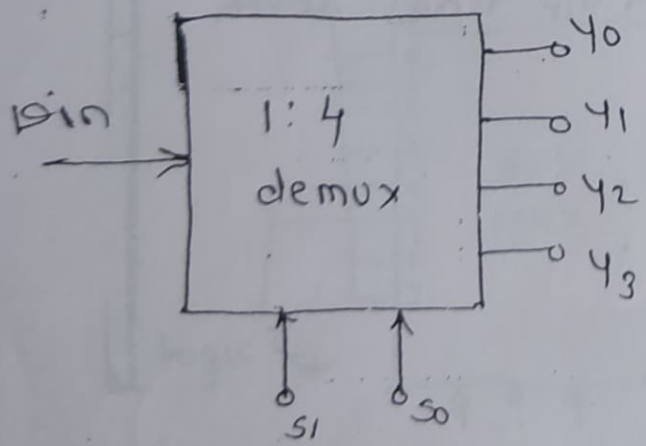
1) 1:2 demultiplexer:



S	Y_0	Y_1
0	Din	0
1	0	Din

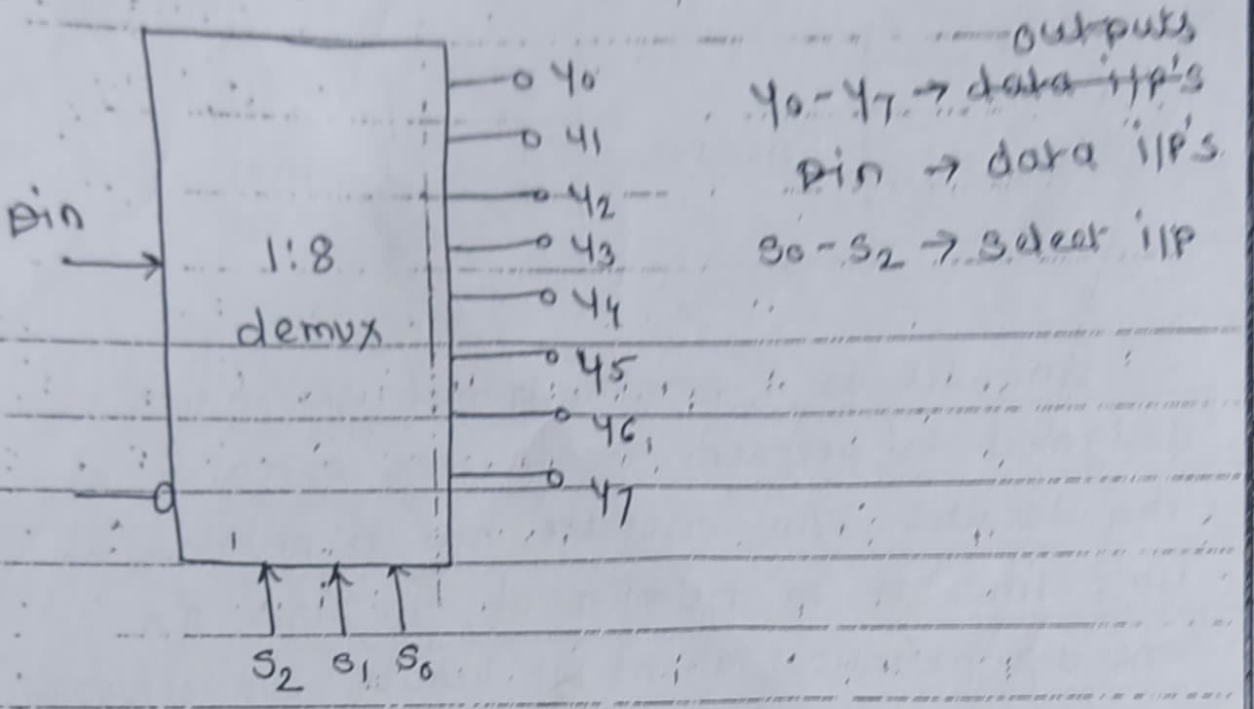
- Din → data i/p
- S → seled i/p
- Y_0 & Y_1 → outputs.

2) 1:4 demultiplexer:



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

3) 1:8 demultiplexer:



Truth table:

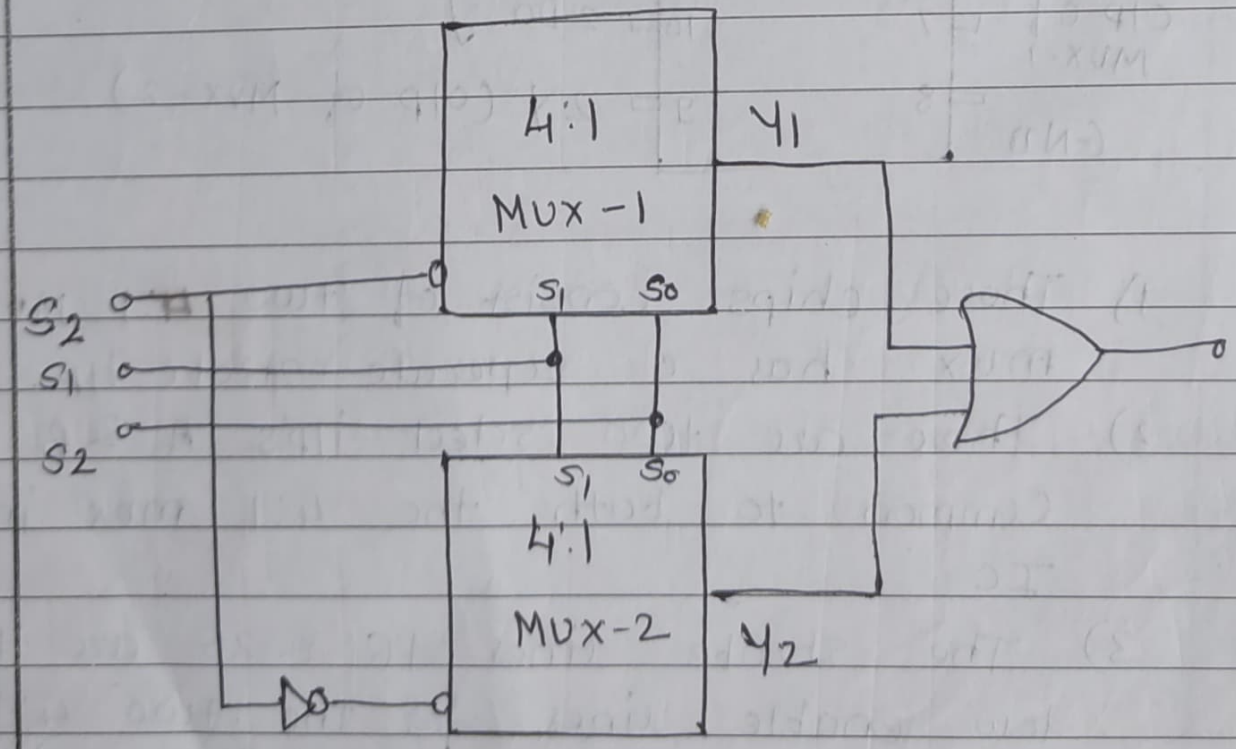
S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	Din	0	0	0	0	0	0	0
0	0	1	0	Din	0	0	0	0	0	0
0	1	0	0	0	Din	0	0	0	0	0
0	1	1	0	0	0	Din	0	0	0	0
1	0	0	0	0	0	0	Din	0	0	0
1	0	1	0	0	0	0	0	Din	0	0
1	1	0	0	0	0	0	0	0	Din	0
1	1	1	0	0	0	0	0	0	0	Din

* Multiplexer Tree:

The multiplexers having more number of i/p's can be obtd by cascading two or more multiplexers with less no. of i/p's. This is called as a multiplexer tree.

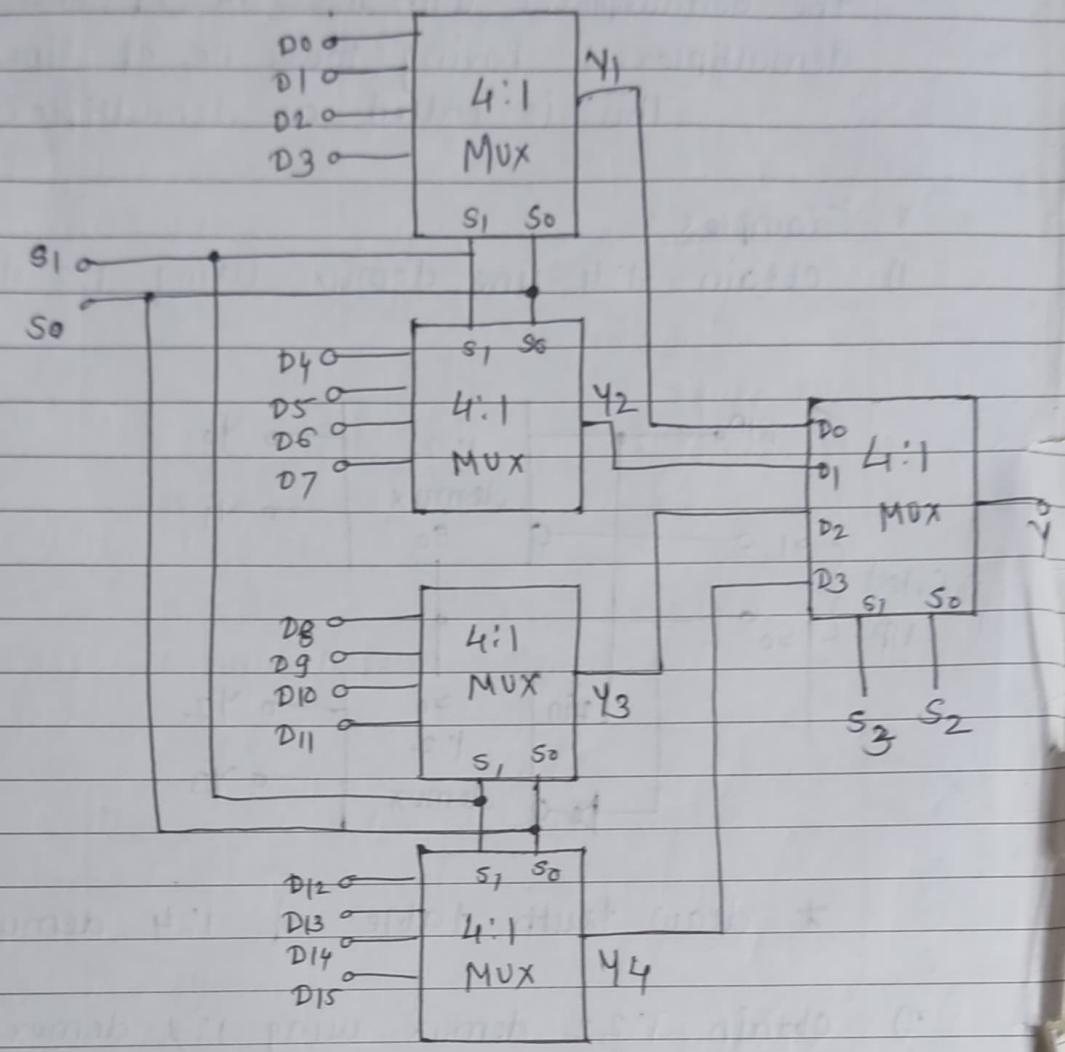
* Example:

1) Obtain 8:1 Mux using 4:1 Mux.



Note: Students should draw the truth table.

2) Implement 16:1 Mux using 4:1 Mux.



* Draw truth table.

HW - Multiplexer tree

1) Implement 16:1 Mux using 8:1 Mux
2) Implement 4:1 Mux using 2:1 Mux
3) Implement 32:1 Mux using 16:1 Mux
4) Implement 32:1 Mux. using 8:1 Mux.

—x—

HW - demultiplexer tree

1) obtain 1:8 line demux using 1:4 line demux
2) Draw 1:16 using 1:4 demux
3) obtain 1:6 demux using 1:8 demux

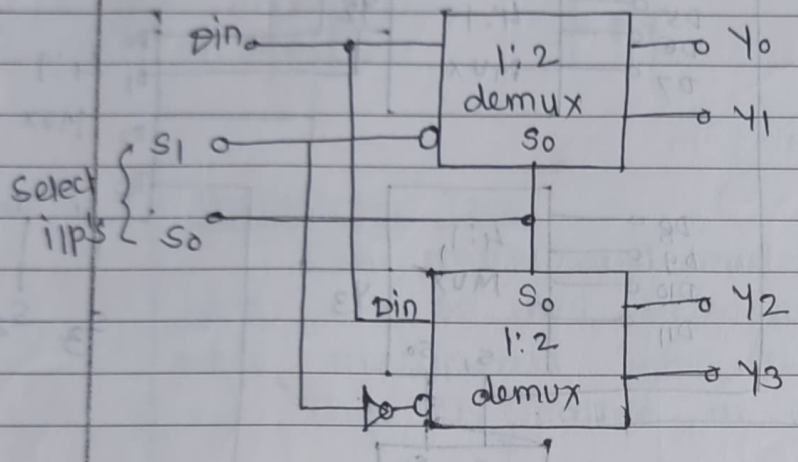
★ Demultiplexer Tree:

Similar to multiplexer we can construct the demultiplexer with more no. of lines using demultiplexers having lower no. of lines.

This is called as demultiplexer tree.

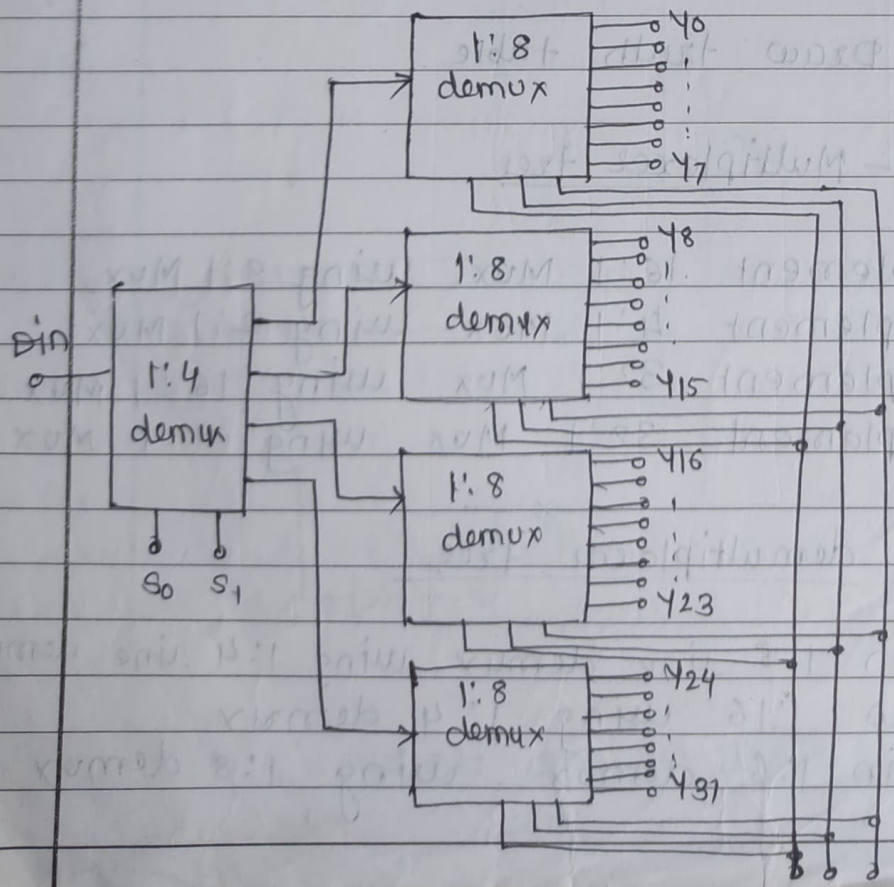
★ Examples:

1) obtain 1:4 line demux using 1:2 demux.



★ draw truth table of 1:4 demux.

2) obtain 1:32 demux using 1:8 demux.



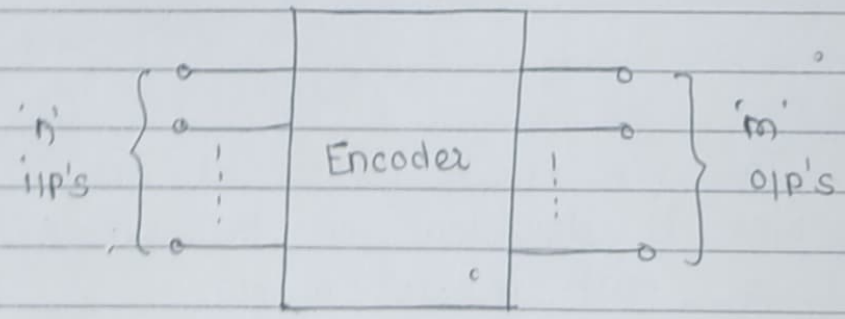
★ draw truth table of 1:32 demux.

2)	Memory is not necessary	2) Memory is necessary
3)	Clock i/p is not necessary	3) Clock i/p is necessary
4)	eg: Adders, subtractors, code converters etc.	4) Flip flop, shift registers, Counters etc.

* **Encoder:**

Encoder is a combinational ckt which is designed to perform the inverse operation of the decoder.

An encoder has 'n' no. of i/p's & 'm' no. of o/p lines. An encoder produces an m bit binary code corresponding to the digital i/p number. Block diagram of encoder is shown in fig.



The encoder accepts an n i/p digital word & converts it into an m bit another digital word.

* **Types of Encoders:**

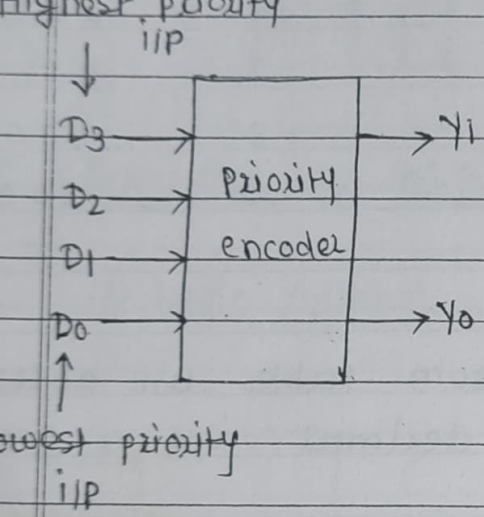
The types of encoders which are going to discuss are as follows.

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

1) Priority Encoder:

This is a special type of encoder. Priorities are given to the i/p lines. If two or more i/p lines are '1' at the same time, then the i/p line with highest priority will be considered.

The block diagram & truth table of a priority encoder is shown in fig.



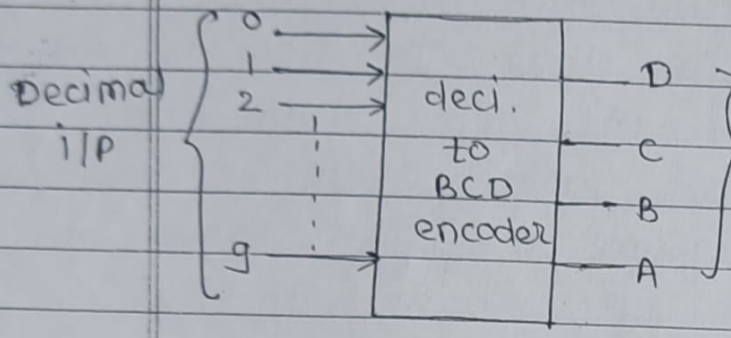
Highest	I/P's		Lowest	O/P	
D3	D2	D1	D0	Y1	Y0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

(a) Block diagram (b) Truth table.

There are four i/p's D0 to D3 & two o/p's Y1 & Y0. Out of four i/p's D3 has the highest priority & D0 has the lowest priority. That means if D3 = 1 then Y1Y0 = 11. Similarly if D3 = 0 & D2 = 1 then Y1Y0 = 10 irrespective of the other i/p's.

2) Decimal to BCD Encoder:

The block diagram of decimal to BCD encoder is shown in fig (a) & truth table in fig (b)



(a) Block diagram

I/P	output D C B A
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

Design of encoder:

Consider o/p D. From table we observe that D=1 only when decimal i/p is 8 or 9

$\therefore \quad D = 8+9$

Similarly the expressions for the o/p's are,

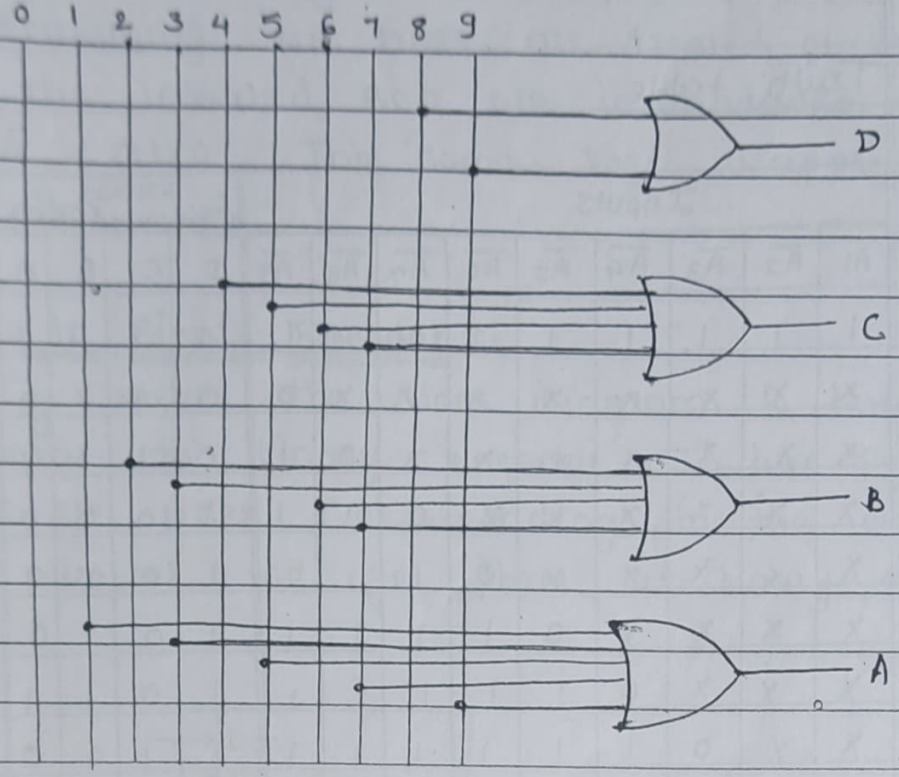
$C = 4+5+6+7$

$B = 2+3+6+7$

$A = 1+3+5+7+9.$

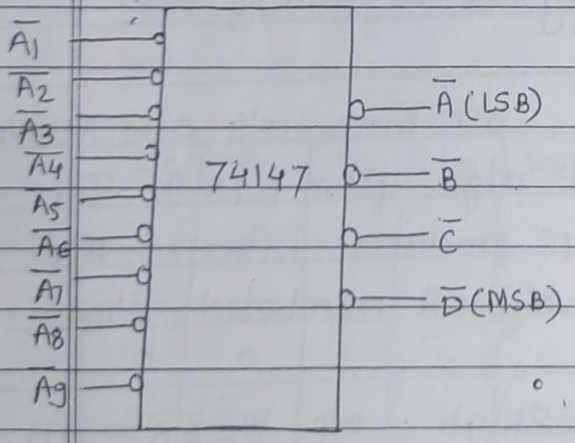
PAGE No.
DATE

Logic diagram using basic gates:



☆ Decimal to BCD Encoder IC 74147

a) Logic symbol



b) Pin configuration

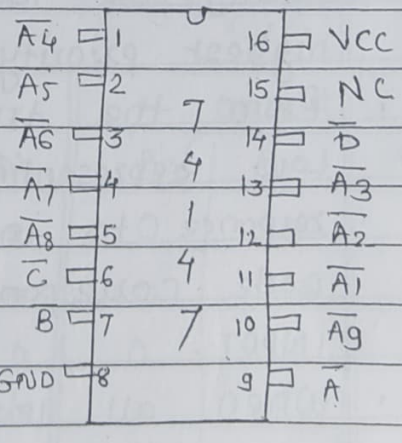


Fig (a) shows the logic symbol & fig (b) shows the pin configuration of IC 74147

A_1 to A_9 are the active low i/p's & A, B, C, D are the active low o/p's.

Truth table

Inputs									Outputs (Inverted BCD)				Normal BCD			
$\overline{A_1}$	$\overline{A_2}$	$\overline{A_3}$	$\overline{A_4}$	$\overline{A_5}$	$\overline{A_6}$	$\overline{A_7}$	$\overline{A_8}$	$\overline{A_9}$	D	C	B	A	D	C	B	A
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
x	x	x	x	x	x	x	x	0	0	1	1	0	1	0	0	1
x	x	x	x	x	x	x	0	1	0	1	1	1	1	0	0	0
x	x	x	x	x	x	0	1	1	1	0	0	0	0	1	1	1
x	x	x	x	x	0	1	1	1	1	0	0	1	0	1	1	0
x	x	x	x	0	1	1	1	1	1	0	1	0	0	1	0	1
x	x	x	0	1	1	1	1	1	1	0	1	1	0	1	0	0
x	x	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1
x	0	1	1	1	1	1	1	1	1	1	0	1	0	0	1	0
0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1

- A_1 to A_9 are the i/p's ~~having~~ with A_1 having the lowest priority & A_9 having the highest priority.
- From the truth table, all nine i/p's are ACTIVE Low representing decimal digit from 1 to 9. In responce to input, chip produces inverted BCD code corresponding to highest numbered ACTIVE INPUT.
- When all i/p's are held high, o/p $\overline{D}\,\overline{C}\,\overline{B}\,\overline{A} = 1111$ i.e. $DCBA = (0000)_2 = (0)_{10}$. Thus a decimal 0 is represented.
- The truth table also shows the normal BCD o/p

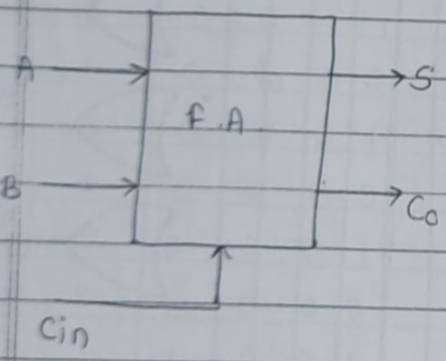
which is actually the inversion of the o/p of IC. As A9 is the highest priority i/p, if $\bar{A}_9 = 0$ then the remaining i/p lines are treated as don't care and the inverted BCD o/p is produced as $\bar{D}\,\bar{C}\,\bar{B}\,\bar{A} = 0110$. The same logic is applicable for other i/p's.

★ Adder: (Binary Adder)

Full adder:

To overcome the drawback of Half Adder circuit, a 3-single bit adder ckt called Full Adder is developed.

It can add two one-bit numbers A and B and carry cin. The full adder is a three i/p & two output combinational ckt. The block dig. & truth table is given below.

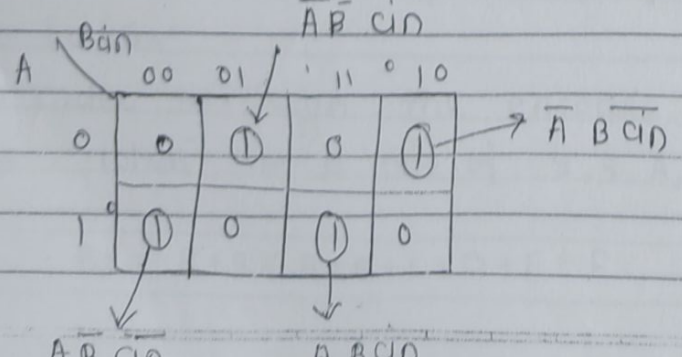


a) Block diagram

Inputs			outputs	
A	B	Cin	S	Co
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

b) Truth table

K-map for sum o/p:



$\therefore S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C_{in}} + AB C_{in} + A\bar{B}\bar{C_{in}}$

$S = C_{in}(\underbrace{\bar{A}\bar{B} + AB}_{\text{Ex-NOR}}) + \bar{C_{in}}(\underbrace{\bar{A}B + A\bar{B}}_{\text{Ex-OR}})$

$\therefore S = C_{in}(\overline{\bar{A}B + A\bar{B}}) + \bar{C_{in}}(\bar{A}B + A\bar{B})$

Let $X = \bar{A}B + A\bar{B}$.

$\therefore S = C_{in}\bar{X} + \bar{C_{in}}X$

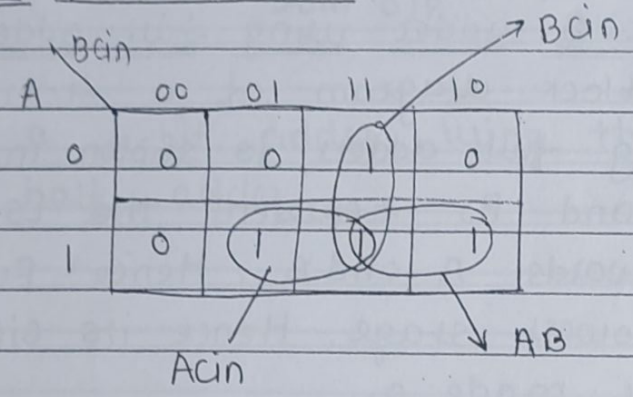
$= C_{in} \oplus X$

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

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

$\therefore \boxed{S = C_{in} \oplus A \oplus B}$

K-map for Co output :



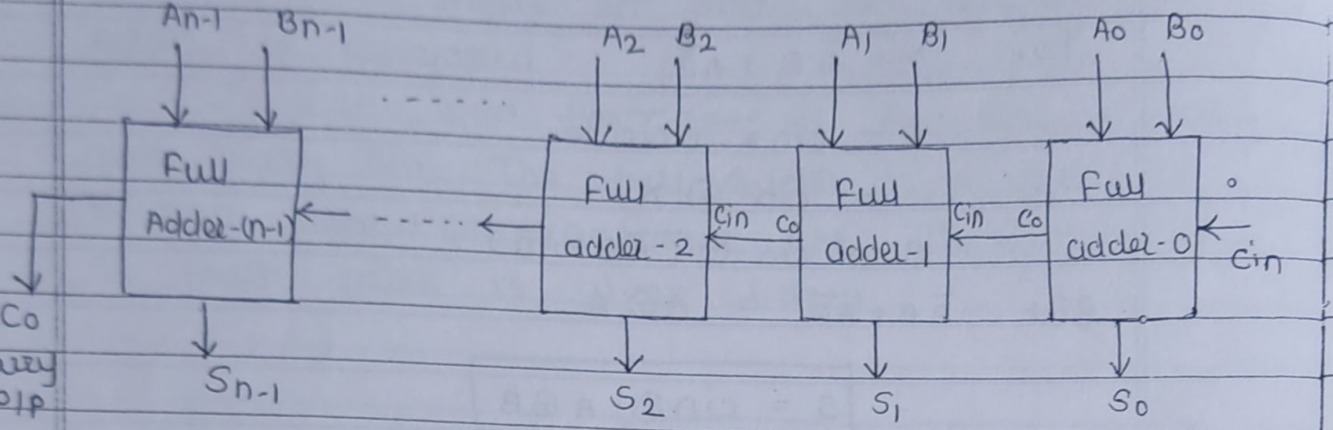
$\therefore \boxed{C_o = AB + AC_{in} + BC_{in}}$

★ n-bit parallel adder :

The full adder is capable of adding only two single digit binary numbers along with a carry i/p. But in practice we need to add binary nos. which are much longer than just one bit.

Two add two n-bit binary numbers we need

to use the n-bit parallel adder shown in fig. It uses a no. of full adders in cascade. The carry output of the previous full adder is connected to the carry i/p of the next full adder.

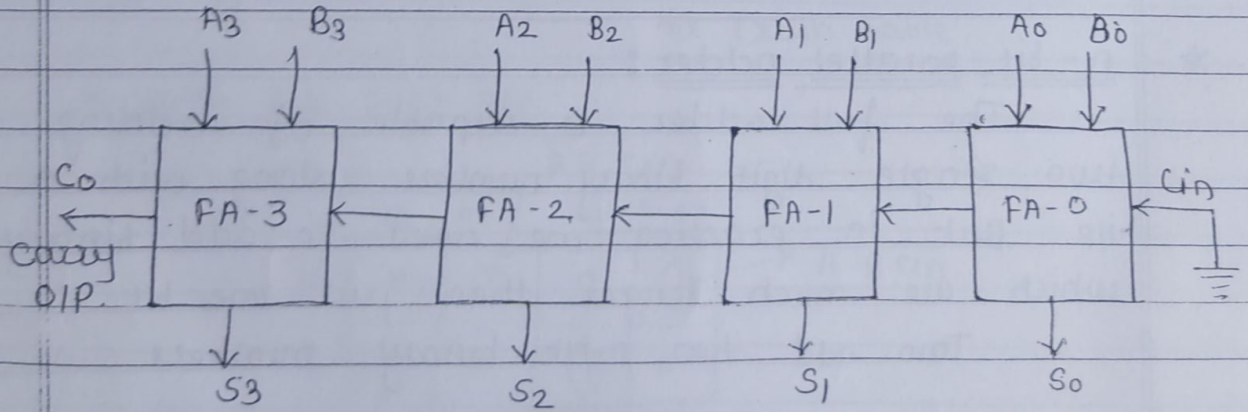


4-bit parallel adder using full adders:

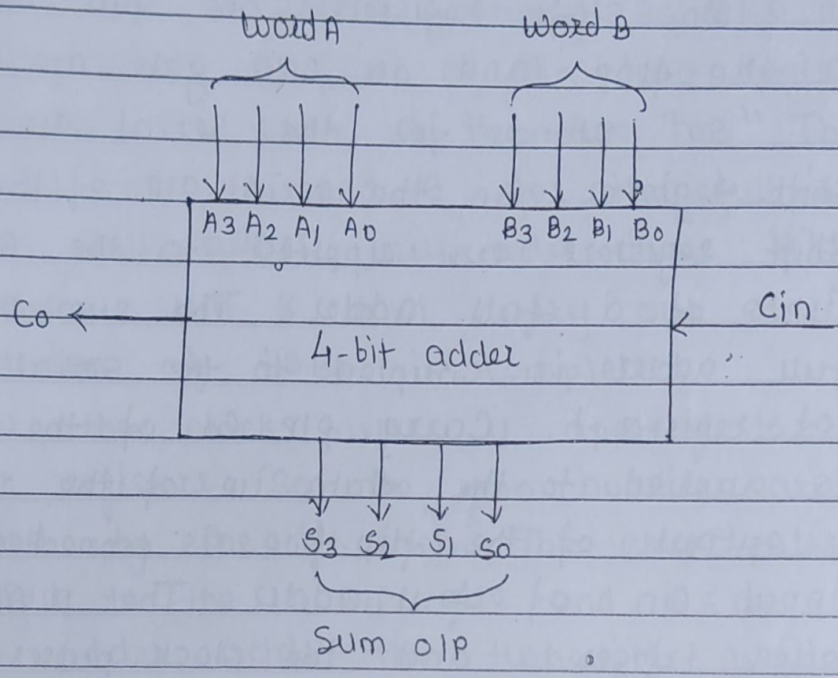
The block diagram of a 4-bit parallel adder using full adders is shown in fig.

A0 and B0 represent the LSBs of the four bit words A and B. Hence Full adder-0 is the lowest stage. Hence its Cin has been permanently made 0.

The rest of connections are exactly same as those of n-bit parallel adder.



The four bit parallel adder is a very common logic ckt. It is normally shown by a block diagram as shown below.



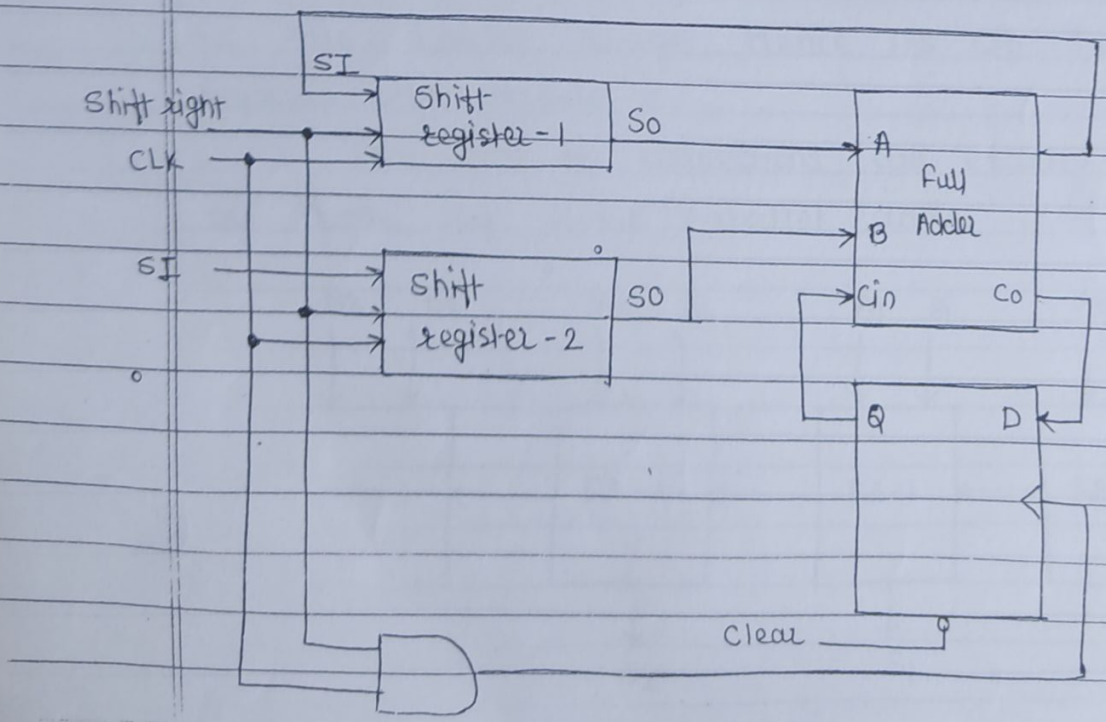
* Design a 4-bit adder using three full adders & one half adder.

* n-bit serial adder:

The ckt diagram of an n-bit serial adder is shown in fig.

Two shift registers, one full adder, a D flip flop and an AND gate are used.

"SO" stands for the serial O/P of the shift register, so the serial O/P of the two shift registers are applied to the A and B I/P's of a full adder. The sum O/P of the full adder is coupled to the serial input SI of register-1. Carry O/P Co of the full adder is applied to the data I/P of the D F/F. The Q output of the flip-flop is connected to carry input Cin of full adder. The D F/F is +ve edge triggered and the clock pulses are obtd. from the output of an AND gate. The two I/P's to the AND gate are clock (CLK) and shift right I/P of the registers.



PAGE No.
DATE

Operation:

1) Let register-1 hold the first number & register-2 hold the other number.
2) The D FlF is cleared initially so Q = 0 and $C_{in} = 0$.
3) The serial o/p (SO) of the two registers will provide the LSBs of the two nos. They will act as bits A and B for the full adder.
4) The full adder will add these bits and produce sum S and carry out C_o. Then the addition of LSBs is complete.
5) Now the clock pulse is applied to both the shift registers. So the two numbers are right shifted by one bit each. The clock pulse gets applied to the D FlF also and $Q = C_{in} = C_{out} = 0$.
6) The adder adds the two bits available at the SO outputs of the two registers and
7) The same operation will continue & we will get the addition of two numbers in a bit by bit manner.
8) Note that the sum of $(1010)_2$ i.e. $(10)_{10}$ and $(0101)_2$ i.e. $(5)_{10}$ is $(1111)_2$ i.e. $(15)_{10}$.

* Features of serial adder:

1) The addition of pair of bits only takes place at any instant of time.
2) (n-1) number of clock cycles are reqd. to complete the addition.
3) The process of addition continues until the shift right control is disabled.
4) Sum is available in the serial form.

PAGE No.
DATE

* Disadvantages of Serial Addition:

1) More time is reqd. to complete the addition.
2) A complexated ckt is reqd.
3) The ckt contains more no. of components.
4) The sum & carry are available in the serial form so result of addition can not be seen at once.

* Comparison of Serial and parallel Addition:

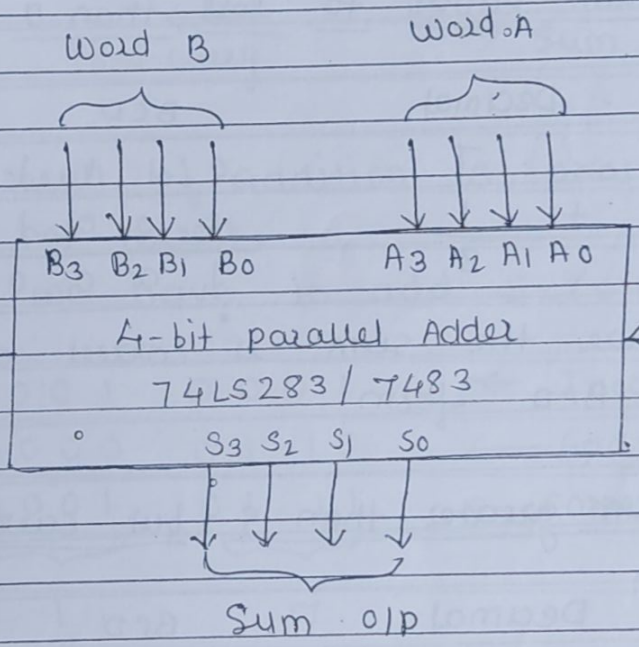
Parallel Adder	Serial Adder
1) Clock pulse is not necessary	1) Clock pulse is necessary
2) All the bits are added simultaneously	2) One pair of bits is added at a time
3) Result is in final form	3) Result is in serial form
4) Time taken to complete the addition is very short.	4) Time taken to complete the addition is very long.
5) Less no. of components are required	5) More number of components are reqd.
6) Less cost	5) More cost
7) Less ckt complexity	7) More ckt complexity.

* 4-Bit binary parallel Adder (IC 7483):

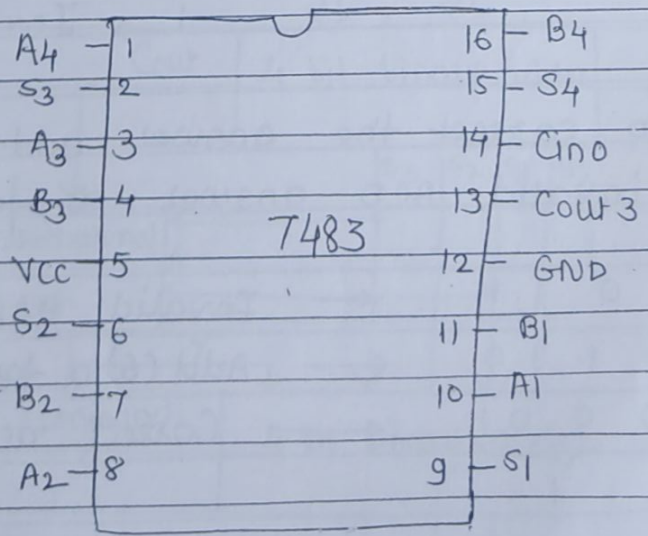
It is the most common binary final adder in the integrated ckt form (IC 74LS83/74LS283)

It is a 4-bit parallel adder which consists of four interconnected full adders alongwith the

look ahead carry ckt. Fig shows the functional symbol of IC 74LS283/7483. $A_3A_2A_1A_0$ is a four bit word A and $B_3B_2B_1B_0$ is another word B. Both these words are applied at the inputs of the IC. Cino is the i/p carry and Cout represents the o/p carry, $S_3S_2S_1S_0$ represent the sum o/ps with S_3 MSB.



* Pin Configuration of 7483:



* Single digit BCD adder using IC7483:

A BCD adder adds two BCD digits and produces a BCD digit. A BCD can not be greater than 9.

The BCD addition can be explained with the help of the following three cases.

Case 1:

Sum equal to less than 9 but carry = 0

Decimal	BCD
7	0 1 1 1
+ 1	0 0 0 1
8	1 0 0 0 ← sum

Here the sum is correct and in the true BCD form.

Case 2:

Sum greater than 9 but carry = 0

Decimal	BCD
7	0 1 1 1
+ 4	0 1 0 0
11	1 0 1 1
	↳ Invalid BCD no.

So to correct the answer add six (0110) to the invalid BCD answer as follows.

1 0 1 1	← Invalid BCD answer
+ 0 1 1 0	← Add (6)10 for correction
0 0 0 1 0 0 0 1	← Correct answer.
1 1	

Case 3 :

Sum less than or equal to 9 but carry = 1

Decimal		BCD	
9		1001	
+ 8	+	1000	
17		1 0001	← wrong result
		↓	↓
		Final carry	Sum is valid BCD no.

The result of addition is $00010001 = (11)_{10}$ which is not correct.

Hence we have to add six (0110) as follows:

0001	0001	← Incorrect answer
0000	0110	← Add 6 for correction
0001	0111	← Correct BCD answer
1	7	

* Block diagram of BCD adder :

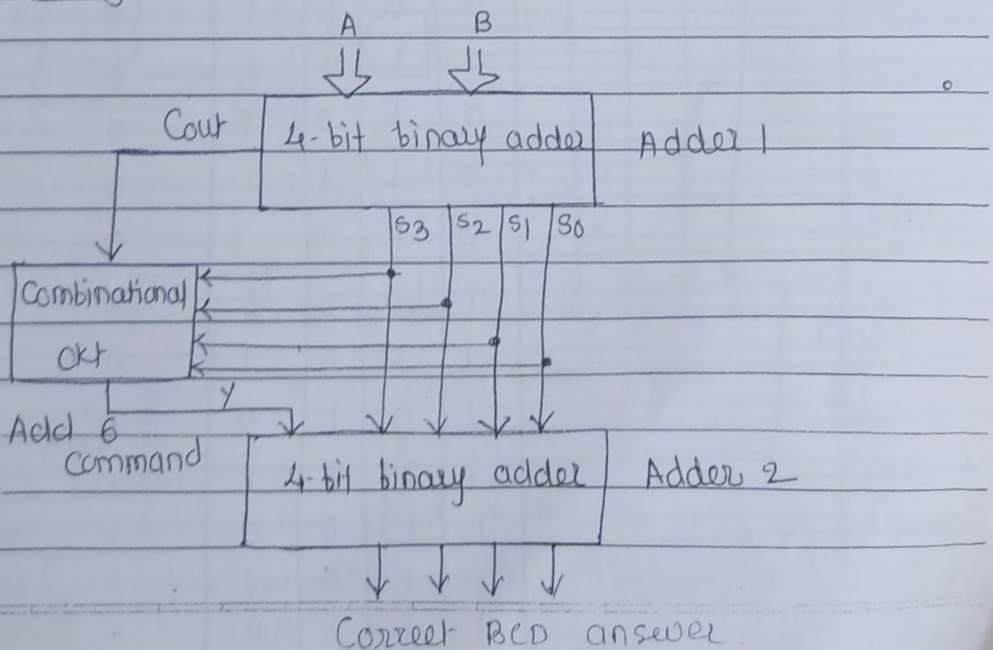


Fig. shows the block diagram of the BCD adder. It consist of a 4-bit binary adder to add the given nos. A & B.

A combinational ckt to check if sum is greater than 9 or carry = 1. One more 4-bit binary adder is used to add six (0110) to the incorrect sum if sum > 9 or carry = 1.

So we have to design the combinational circuit that senses if sum is greater than 9 or carry = 1.

* A single digit BCD adder using 7483:

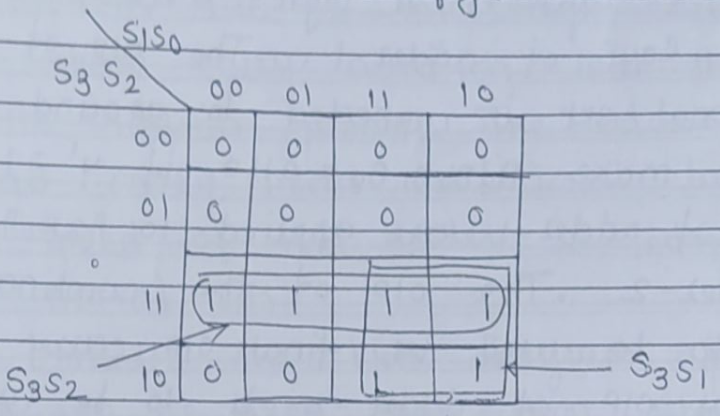
The output of combinational ckt (Y) should be 1 if the sum produced by adder I is greater than 9 i.e. 1001. The truth table is as follows.

Sum bits of adder I →	Inputs S_3	S_2	S_1	S_0	output Y	
	0	0	0	0	0	Sum is valid BCD no. ∴ Y = 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	Sum is invalid BCD no. ∴ Y = 1
	1	0	1	1	1	
	1	1	0	0	1	
	1	1	0	1	1	
	1	1	1	0	1	

PAGE No.
DATE

K-map :

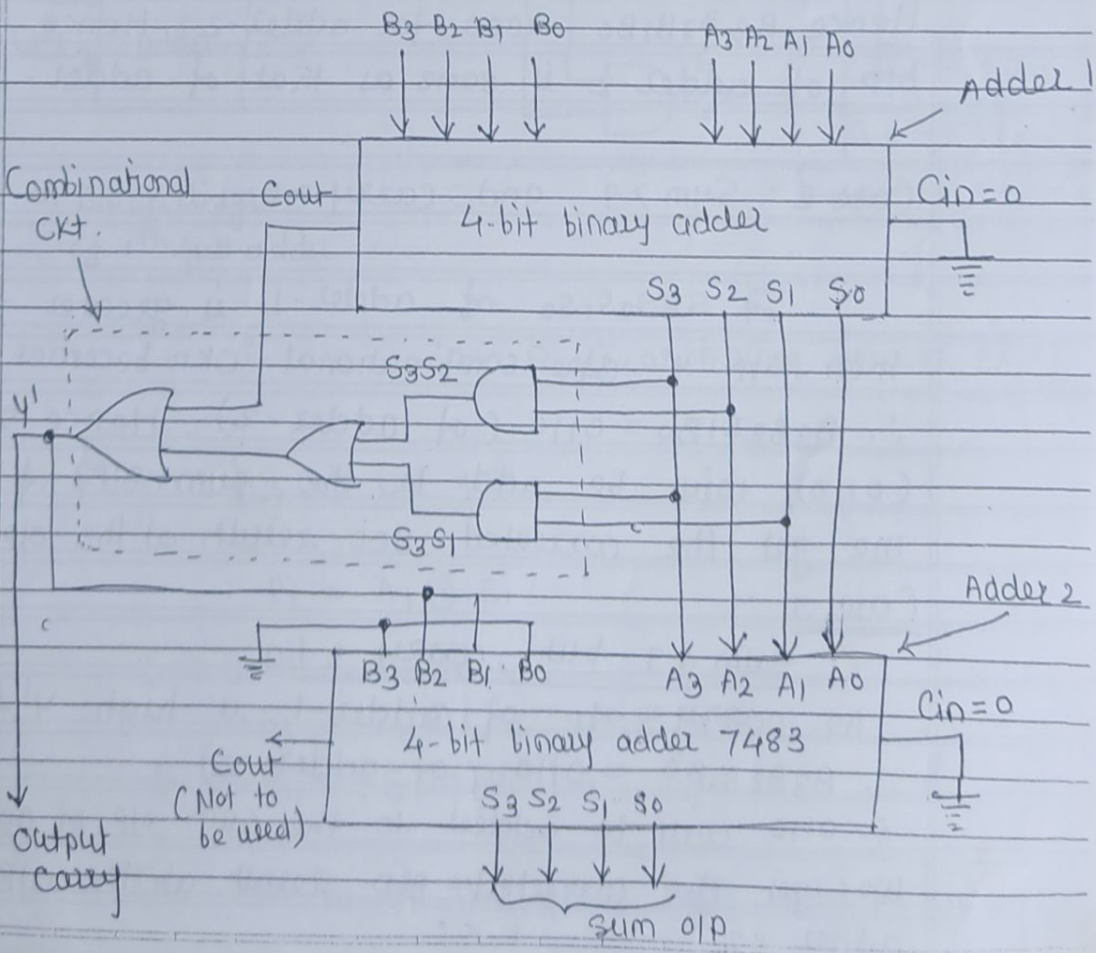
The K-map for y output of the combinational circuit is shown in fig



∴ Simplified Boolean exp^{z} is,

$$Y = S_3S_2 + S_3S_1$$

Block diagram :



The BCD adder is shown in fig. The o/p of the combinational ckt should be 1 if cout of adder-1 is high. Therefor Y is ORed with Cout of adder 1. The o/p of the combinational ckt is connected to ground permanently. This make $B_3B_2B_1B_0 = 0110$ if $y' = 1$. The sum o/p of adder-1 are applied to $A_3A_2A_1A_0$ of adder-2. The o/p of the combinational ckt is to be used as final o/p carry & the carry o/p of adder-2 is to be ignored.

* Operation :

Case 1 : Sum ≤ 9 and carry = 0

The o/p of the combinational ckt $y' = 0$ Hence $B_3B_2B_1B_0 = 0000$ for adder-2. Hence the o/p of adder-2 is same as that of adder-1.

Case 2 : Sum > 9 and carry = 0

If $S_3S_2S_1S_0$ of adder-1 is greater than 9, then o/p y' of combinational ckt becomes 1. $\therefore B_3B_2B_1B_0 = 0110$ (of adder-2). Hence 6 i.e (0110) will be add to the sum o/p of adder-1. We get the corrected BCD result at the o/p of adder-2

Case 3 :

Sum ≤ 9 but carry = 1

As carry o/p of adder-1 is high $y' = 1$.
$\therefore B_3B_2B_1B_0 = 0110$ (of adder-2).
$\therefore$ 0110 will be added to the sum o/p of adder-1. We get the corrected BCD result at the o/p of adder-2.