BCD to Excess-3 Converter

Truth table

Binary	BCD Input				Excess-3 output			
	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

$E_3 =$

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$G_2 = B_2B_0$

$G_3 = B_1B_2$

$G_1 = B_3$

$E_3 = B_0 + B_2B_0 + B_2B_1$

$E_3 = B_3 + B_2(B_0 + B_1)$

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

$B_2(B_1+B_0) + B_2\overline{B}_1\overline{B}_0$

$\overline{B}_2(B_1+B_0) + B_2(\overline{B_1+B_0})$

$= B_2 \oplus (B_1+B_0)$

$E_2 =$

B_3B_2 \ B_1B_0	00	01	11	10
$\overline{B}_3\overline{B}_2$	0	1	1	1
$\overline{B}_3B_2$	1	0	0	0
B_3B_2	X	X	X	X

$G_1 = \overline{B}_2B_1$

$E_2 = \overline{B}_2B_1 + \overline{B}_2B_0 + B_2\overline{B}_1\overline{B}_0$

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	×	×	×	×
10	1	0	×	×

$G_1 = \bar{B}_1\bar{B}_0$ $\quad$ $G_2 = B_1B_0$

$$E_1 = \bar{B}_1\bar{B}_0 + B_1B_0$$

$$\boxed{E_1 = B_1 \oplus B_0}$$

$E_0 =$

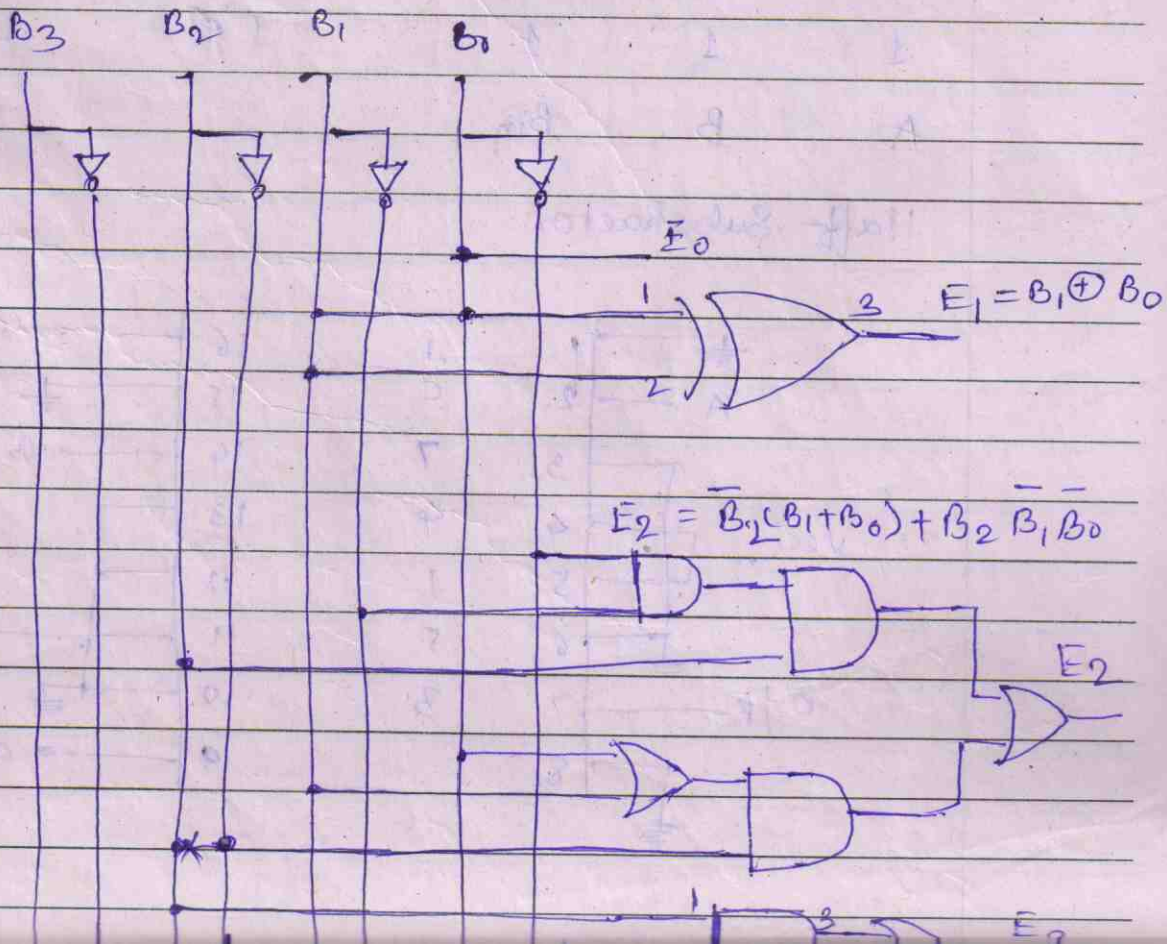
B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	×	×	×	×
10	1	0	×	×

G_1 → $G_2 = B_0$

$E_0 = B_1\bar{B}_0 + \bar{B}_1\bar{B}_0$

$= \bar{B}_0$

$$\boxed{E_0 = \bar{B}_0}$$



Half Substractor:-

Input		Output		
A	B	Diff	Borrow	
0	0	0	0	D_0
0	1	1	1	D_1
1	0	1	0	D_2
1	1	0	0	D_3

Full Substractor:-

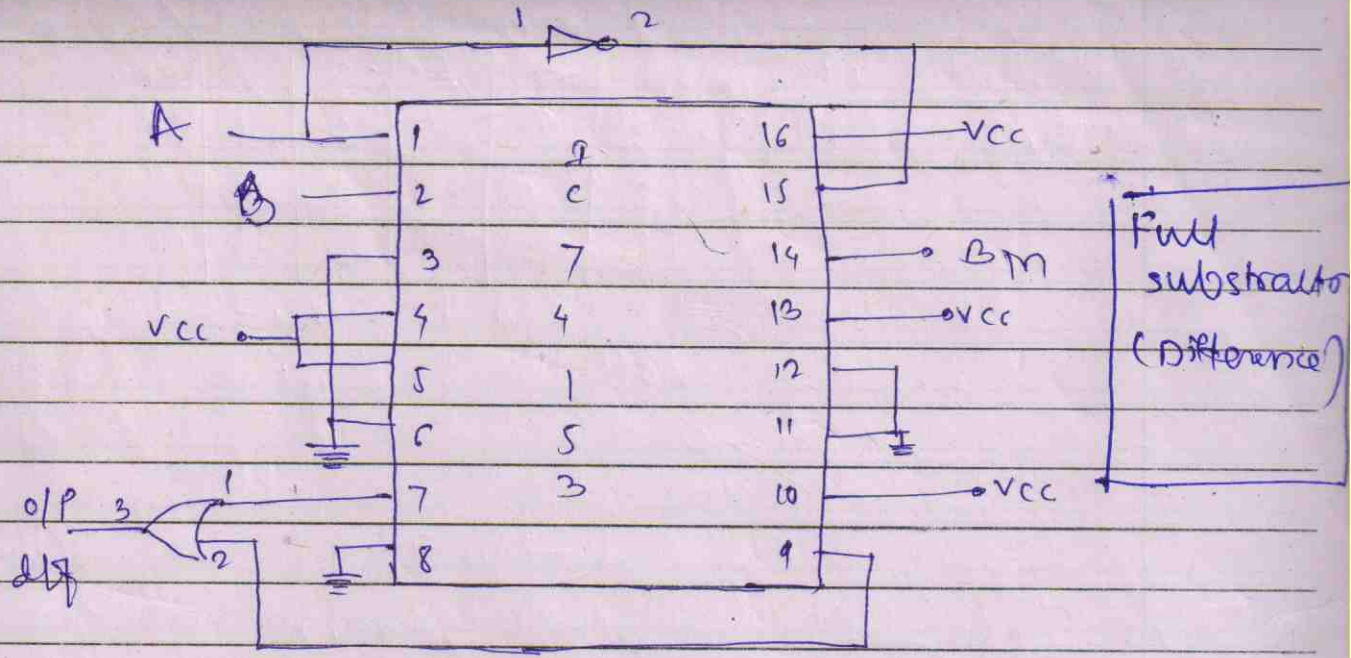
(S2) (S1)

input			output		
Cin	A B	B Cin	Diff	Borrow	
0	0	0	0	0	D_0
0	0	1	1	1	D_1
0	1	0	1	1	D_2
0	1	1	0	1	D_3
1	0	0	1	0	D_4
1	0	1	0	0	D_5
1	1	0	0	0	D_6
1	1	1	1	1	D_7

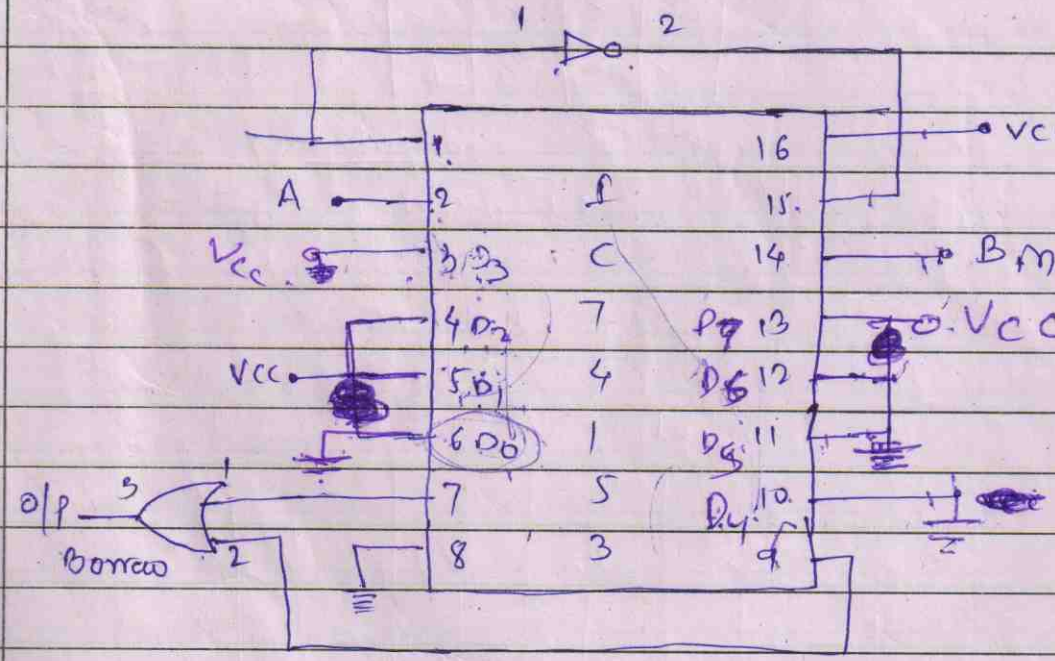
A B Bin

Half Substractor

full Substractor (Difference)



full Substractor (Borrow)



BCD Adder:-

- BCD is less effiient than binary.
- BCD Binary needs more bits than binary to encode the same decimal no.
- BCD anthmetic is more complicated than Binary anthmetic.
- The advantage of a BCD code is that the conversion from decimal to BCD or vice versa is simpler.

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

BCD & Binary are same.

BCD & binary are different

Advantages of BCD codes.

1. It is very similar to decimal system.
2. We need to remember binary equivalent of decimal no. 0 to 9 only.

Disadvantages.

1. The addition & substraction of BCD have different rules
2. BCD arithmetic is more complicated.
3. BCD needs more no. of bits than binary to represent the same decimal no. so BCD is less efficient than binary.

BCD addition:-

- In BCD addition we have to deal with three different situations.
- Assume that two 4 bit BCD nos. A & B are being added. Then three cases to be considered are.

1. Sum is equal to or less than 9 & carry is 0.
2. Sum is greater than 9 & carry is 0.
3. Sum is less than or equal to 9 but carry is 1.

(1) The addition is to be carried out as normal binary addition.

(1) Sum equal to or less than 9 with carry 0.

6		0 1 1 0	
+ 3	+	0 0 1 1	Sum is a
9	[0]	1 0 0 1 →	valid BCD no.

2		0 0 1 0	
+ 6		0 1 1 0	
8	[0]	1 0 0 0	→ Sum is a valid BCD no.

Requires no correction.

(2) Sum grater than 9 but carry = 0.

- In this case the sum of two BCD no. is greater than 9 that means it is an invalid BCD no. but the final carry is 0.
- Whenever this occurs the sum has to be corrected by the addition of six (0110) in the invalid BCD no.

①   + 6   0 1 1 0

8   1 0 0 0

14   [0] 1 1 1 0 → Invalid BCD No. with carry '0'.

```
     1 1 1 0
   + 0 1 1 0     ← add six for correction.
 [1] 0 1 0 0     ← BCD for 14 valid.
  1     4
```

② + 6   0 1 1 0

7   0 1 1 1

13   [0] 1 1 0 1 ← Invalid BCD

```
     1 1 0 1
   + 0 1 1 0     ← add 6 to correct
 [1] 0 0 1 1
  1     3
```

Case 3   Sum less than or equal to 9 but carry = 1

→ Sum of two BCD no. is less than or equal to 9 i.e. it is a valid BCD no. & final carry = 1.

→ Indicates that the answer is wrong & needs correction.

– Add 6 to the sum to correct the answer.

Decimal.

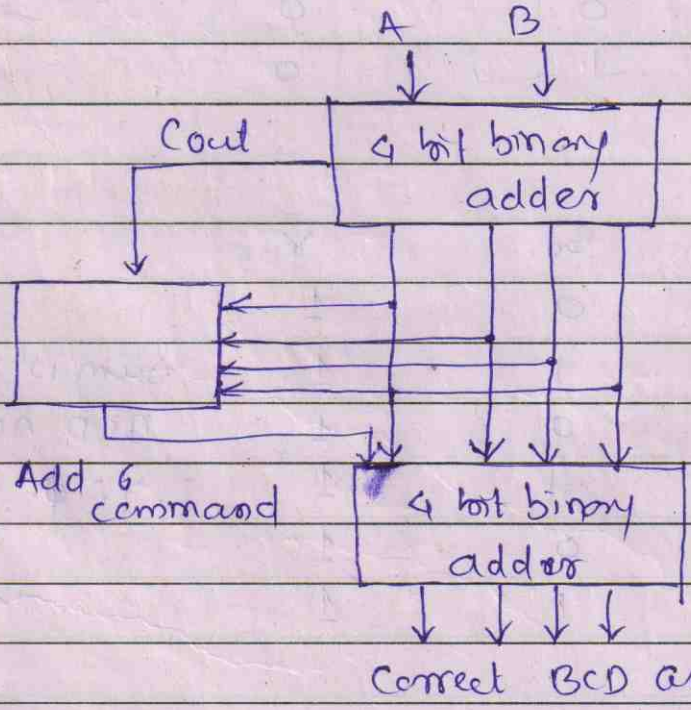
```
   8        1 0 0 0
 + 9      + 1 0 0 1
  17    [1] 0 0 0 1   ← Sum is correct BCD
                        & carry is 1.

        0 0 0 1  0 0 0 1      Incorrect BCD
           1        1
```

	0001	0001	Incorrect BCD result
+	0000	0110	Add 6.
→	0001	0111	
	1	7	Correct result.

Block diagram of BCD adder:

4-bit, BCD adder should consist of the following blocks.

1) A 4-bit binary adder to add the given numbers A & B.
2) A combinational circuit to check if Sum is greater than 9 or carry = 1.
3) One more 4-bit binary adder to add six (0110) to the incorrect sum if Sum > 9 or carry = 1.



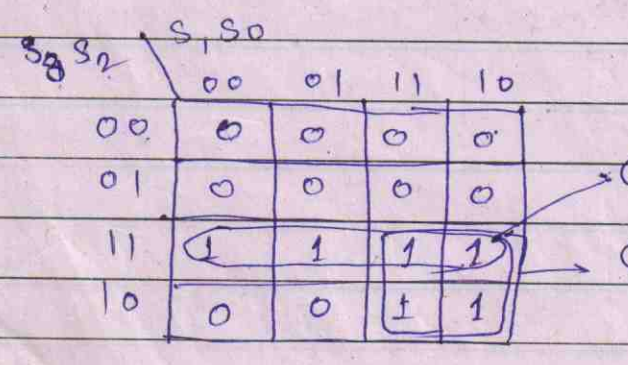
Design of Combinational circuit :-

The o/p of combinational circuit should be 1 if the Sum produced by adder 1 is greater than 9 i.e. 1001. The truth table is

Sum bits of adder 1

Inputs				outputs	
S_3	S_2	S_1	S_0	Y	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	Sum is valid
0	0	1	1	0	BCD no.
0	1	0	0	0	$\therefore Y = 0$.
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	

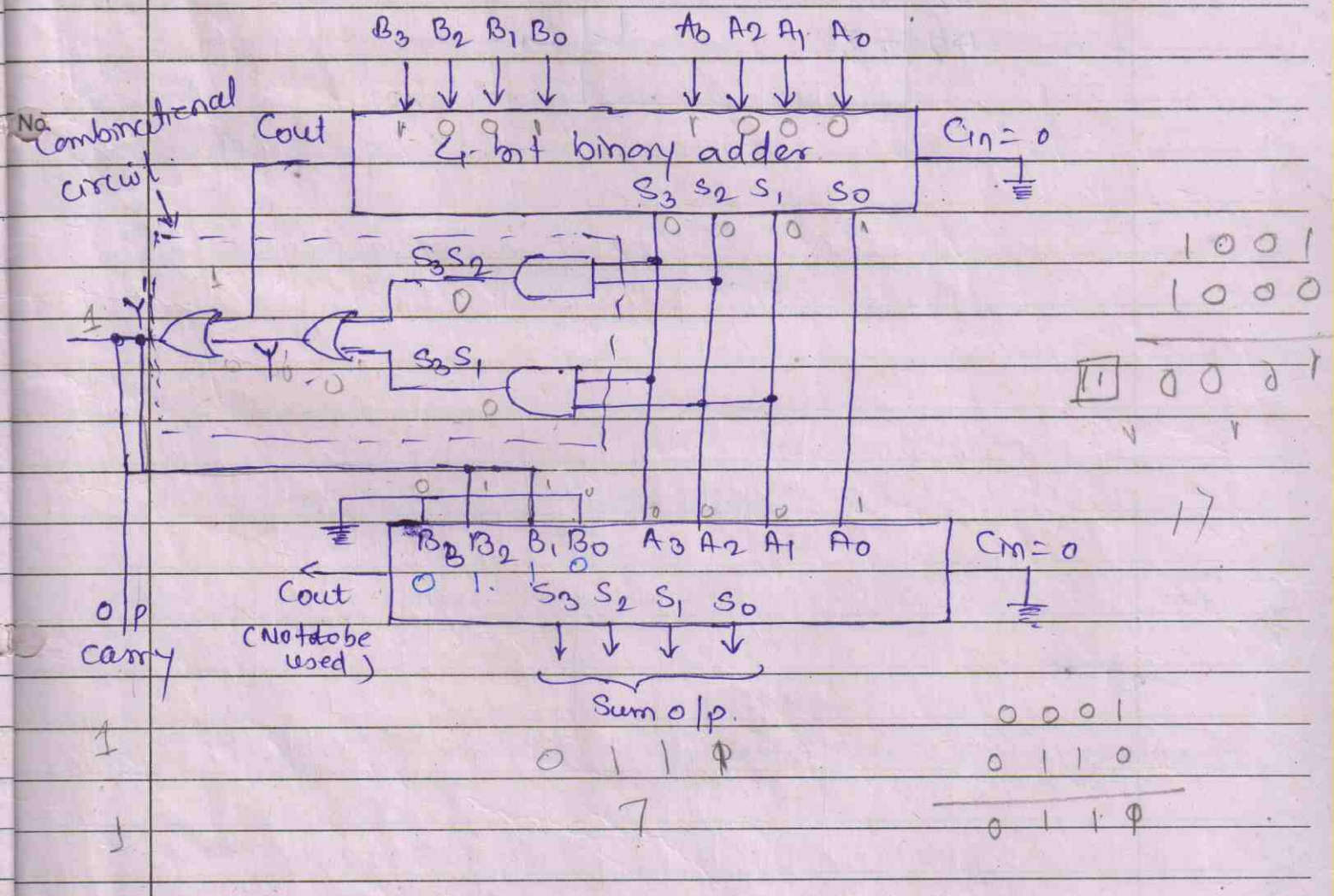
S_3	S_2	S_1	S_0	Y	
1	0	1	0	1	
1	0	1	1	1	Sum is invalid
1	1	0	0	1	BCD no.
1	1	0	1	1	$\therefore Y = \phi$.
1	1	1	0	1	
1	1	1	1	1	



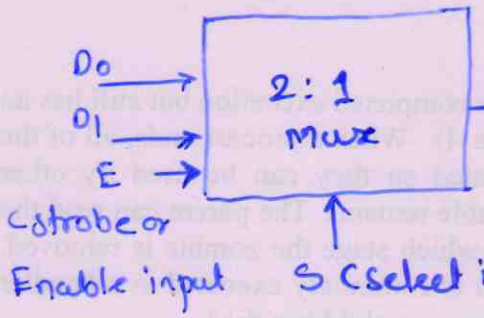
$Y = S_3S_2 + S_3S_1$ — Combinational circuit.

→ The output of the combinational circuit should be 1 if Cout of adder-1 is high. Therefore Y is ORed with Cout of adder 1.

→ The o/p of combinational circuit is connected to $B_1 B_2$ inputs of adder-2 & $B_3 = B_0 = 0$ as they are connected to ground permanently. This makes $B_3 B_2 B_1 B_0 = 0110$ if $Y' = 1$.



2 : 1 Multiplexer :-



Enable	select i/p S	o/p Y
0	X	0
1	0	D_0
1	1	D_1

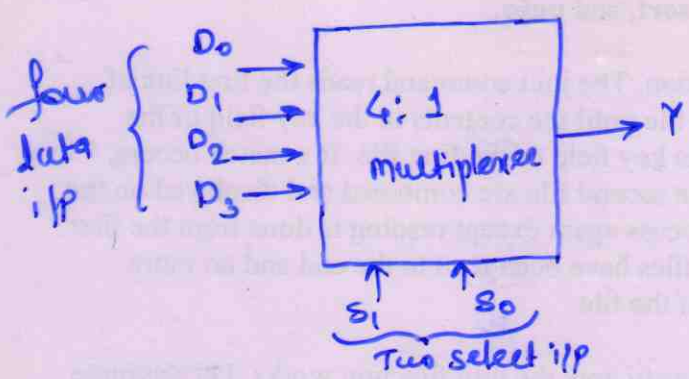
Realization of a 2 : 1 Mux using gates :-

Enable	Select S	D_1	D_0	o/p Y	
0	X	X	X	0	
1	0	X	0	0	
1	0	X	1	1	$\leftarrow Y = E\bar{S}D_0$
1	1	0	X	0	
1	1	1	X	1	$\leftarrow Y = ESD_1$

from ~~the~~ truth table it is clear that $Y = 1$ for only two cond?

$$Y = E\bar{S}D_0 + ESD_1$$
$$Y = E(\bar{S}D_0 + SD_1)$$

A 4 : 1 Multiplexer :-

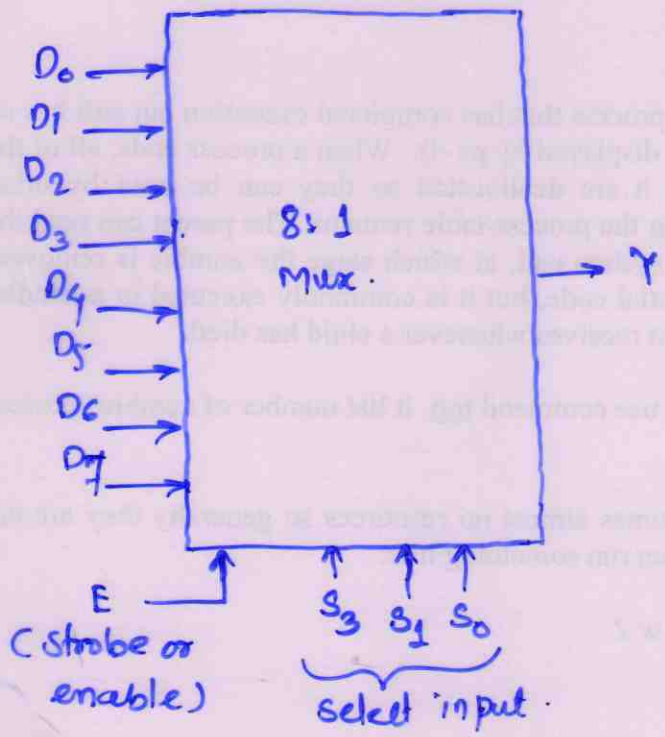


Select i/p S_1	Select i/p S_0	o/p Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

→ Here $n = 4$, hence no. of select lines i.e. $m = 2$ so that $2^m = n$. The strobe terminal has not been included to reduce the complexity.

$$Y = \bar{S}_1\bar{S}_0D_0 + \bar{S}_1S_0D_1 + S_1\bar{S}_0D_2 + S_1S_0D_3$$

8:1 Multiplexer:-



Enable	select i/p			o/p.
E	S_2	S_1	S_0	Y
0	X	X	X	0
1	0	0	0	D_0
1	0	0	1	D_1
1	0	1	0	D_2
1	0	1	1	D_3
1	1	0	0	D_4
1	1	0	1	D_5
1	1	1	0	D_6
1	1	1	0	D_7

→ i. When the Strobe or enable input is 0, the o/p will be 0 irrespective of any i/p.

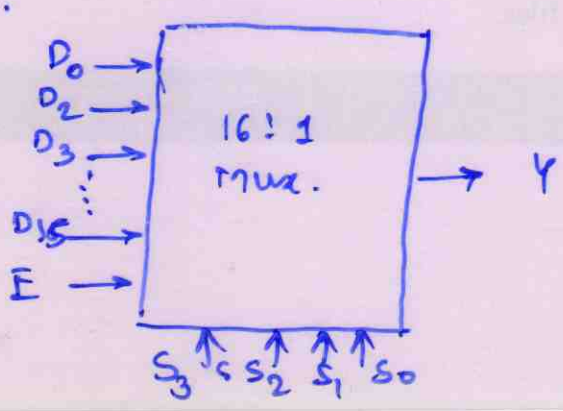
→ With E = 1, we can select any one of the input data i/ps & Connect it to the o/p.

16:1 Mux:.

→ It has 16 data i/p, 4 select lines, one enable or strobe i/p & one o/p.

→ When the Strobe or enable input E=0, the multiplexer o/p Y=0 irrespective of the status of other i/p.

→ With E=1, we can select any one of the 16 inputs with the help of select i/p lines & Connect the selected i/p to the o/p.



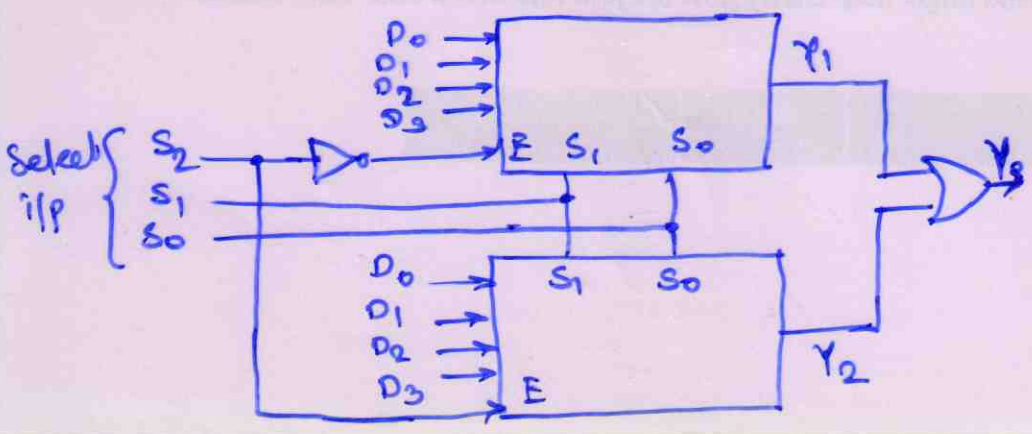
mux 3

Enable	Select i/p				o/p.
E	S_3	S_2	S_1	S_0	Y
0	0	0	0	0	D_0
1	0	0	0	1	D_1
1	0	0	1	0	D_2
1	0	0	1	1	D_3
1	0	1	0	0	D_4
1	0	1	0	1	D_5
1	0	1	1	0	D_6
1	0	1	1	1	D_7
1	1	0	0	0	D_8
1	1	0	0	1	D_9
1	1	0	1	0	D_{10}
1	1	0	1	1	D_{11}
1	1	1	0	0	D_{12}
1	1	1	0	1	D_{13}
1	1	1	1	0	D_{14}
1	1	1	1	1	D_{15}

Multiplexer Tree :→

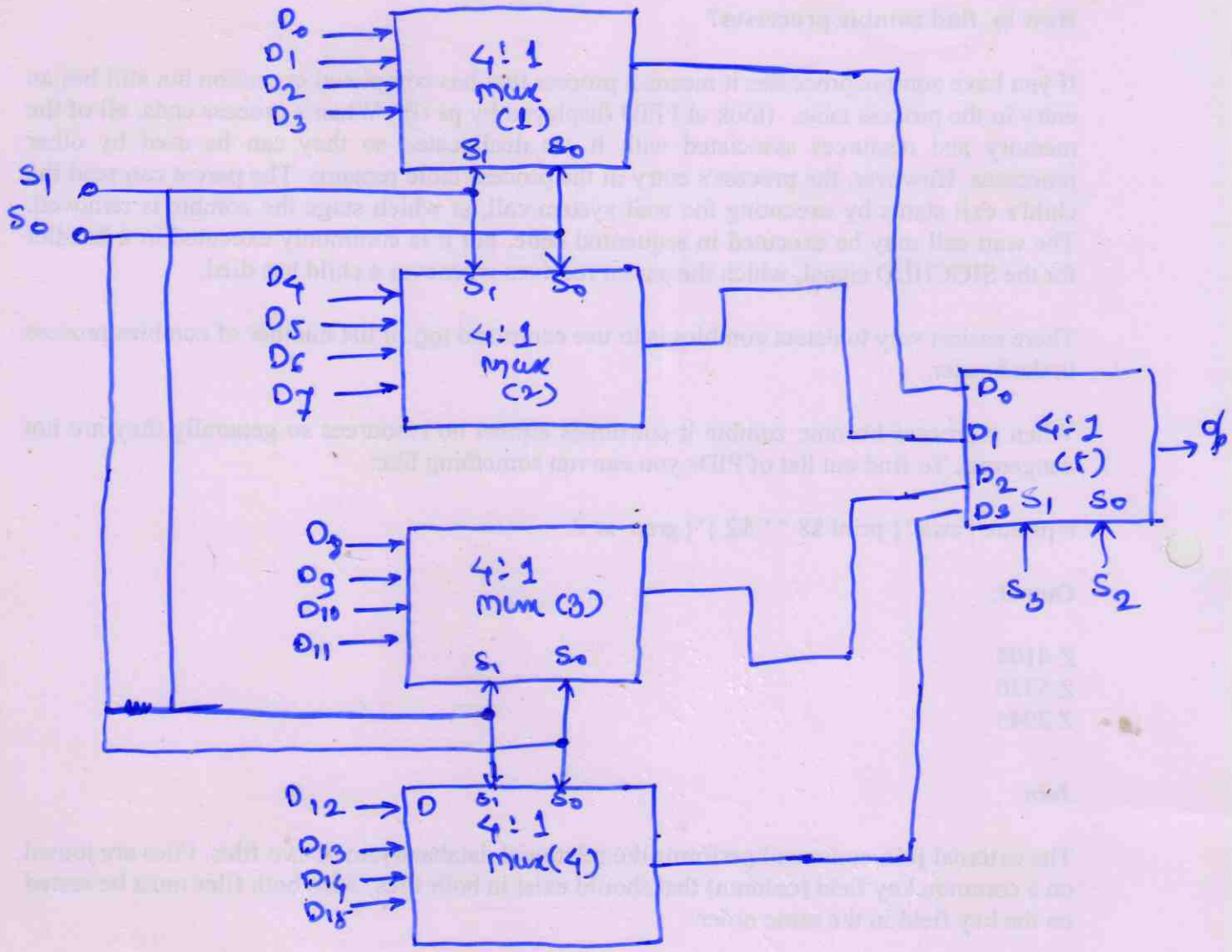
The multiplexers having more no. of i/p can be obtained by cascading two or more multiplexers with less no. of i/p. This is called as multiplexer tree.

Exp: obtain an 8:1 multiplexer using two 4:1 multiplexers.

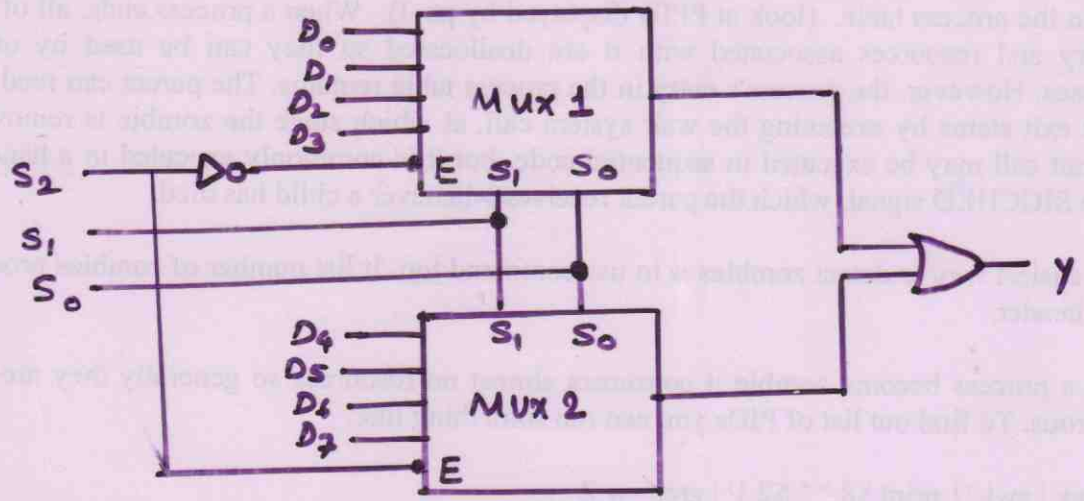


Select i/p			o/p Y	
S_2	S_1	S_0		
0	0	0	D_0	mux 1 is enable
0	0	1	D_1	
0	1	0	D_2	
0	1	1	D_3	
1	0	0	D_4	mux 2 is enable
1	0	1	D_5	
1	1	0	D_6	
1	1	1	D_7	

Exp.:2 Implement a 16:1 multiplexer using 4:1 multiplexer (4)



* Cascading of Multiplexer
  - This is achieved by enable / strobe input



8:1 multiplexer by cascading two 4:1 multiplexer

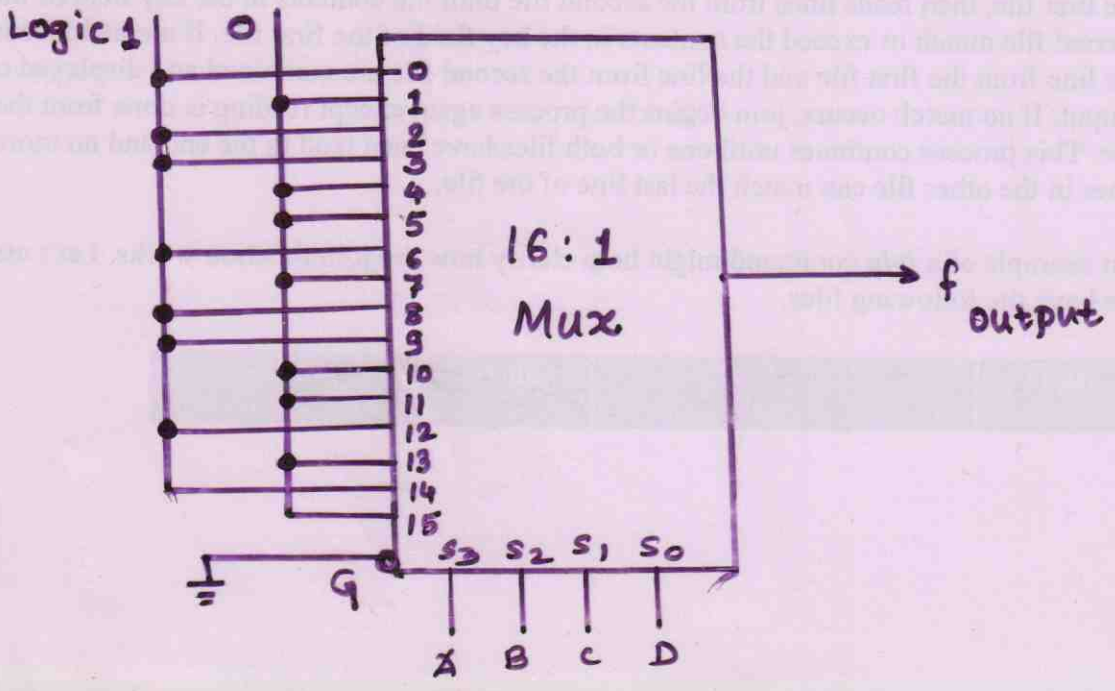
* Assignment : 1. Construct 16:1 multiplexer by using 4:1 Mux.
  2. Design a 2:1 Mux using the basic gates.

- Combinational Logic Design Using Multiplexer

  Advantage
  1. Logic design is simplified
  2. Not necessary to simplify the logic expression
  3. Minimizes the number of IC's required to be used

- Implement the expression using a Multiplexer

  $f(ABCD) = \Sigma m(0, 2, 3, 6, 8, 9, 12, 14)$

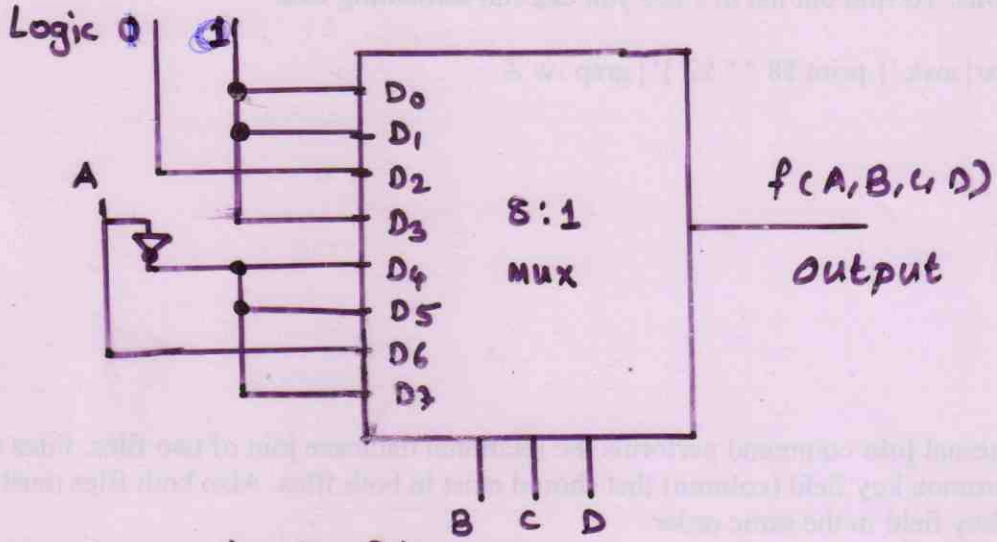


- Implementation of 4 variable function using 8:1 Mux.

- Implement the following function using 8:1 mux. (May-98, 8M)

$f(A,B,C,D) = \Sigma m(2,4,5,7,10,14)$

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	1	(2)	3	(4)	(5)	6	(7)
A	8	9	(10)	11	12	13	(14)	15
	0	0	1	0	$\bar{A}$	$\bar{A}$	A	$\bar{A}$



Logic Diagram

- Assignment: Implement a 16:1 mux using 4:1 Mux (May 08, May 01, May 00, 8 Marks)

* Impliment the following function using 4:1 Mux

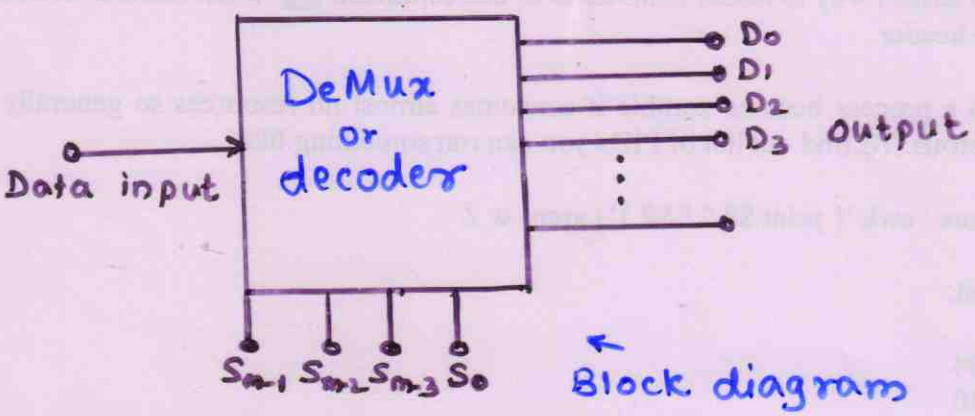
$F(A,B,C,D) = \Sigma m(0,1,5,9,10,15)$

→

	Mux input			
	D_0	D_1	D_2	D_3
$\bar{C}\bar{D}$	(0)	4	8	12
$\bar{C}D$	(1)	(5)	(9)	13
$C\bar{D}$	2	6	(10)	14
CD	3	7	11	(15)
	$\bar{C}\bar{D}+\bar{C}D$ $= \bar{C}$	$\bar{C}D$	$\bar{C}D+C\bar{D}$ $C \oplus D$	CD

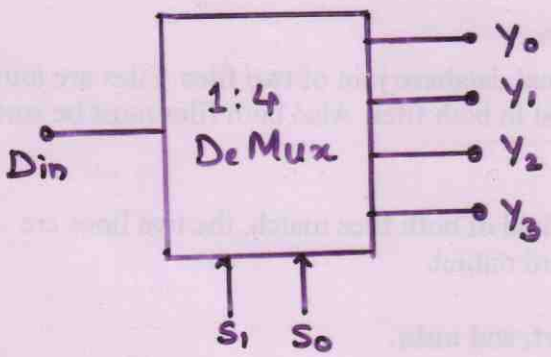
Demultiplexer

- Accepts single input and distributes it over several outputs
- The select input determines which output the data input will be transmitted
- The number of output line is n and the number of select line is m then $n = 2^m$



Block diagram

* Implementation of 1:4 demultiplexer



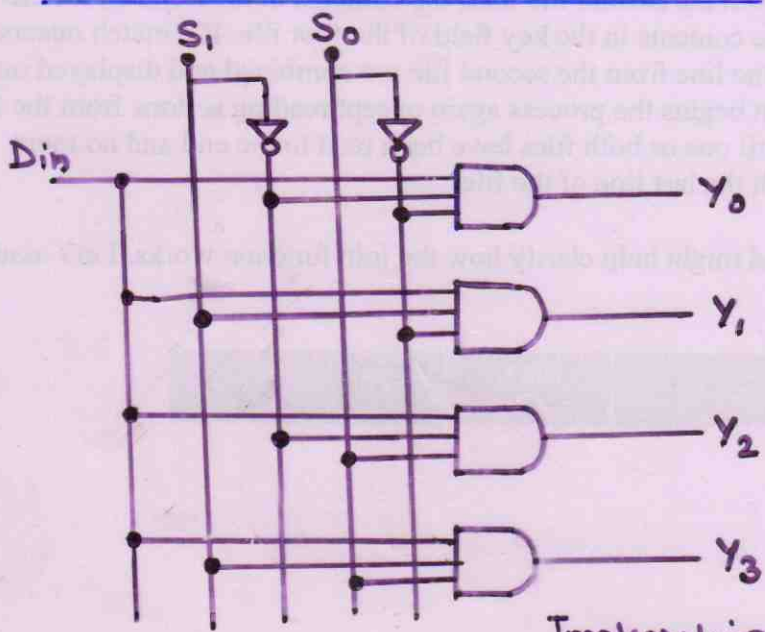
$Y_0 = \bar{S}_1 \bar{S}_0 Din$

$Y_1 = S_1 \bar{S}_0 Din$

$Y_2 = \bar{S}_1 S_0 Din$

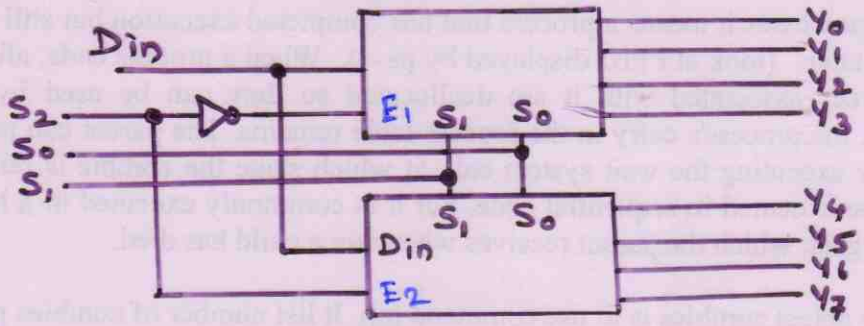
$Y_3 = S_1 S_0 Din$

Implementation using logic gates:



Implemation of 4:1 DeMux

* Cascading De-Mux



8:1 De-Mux using 4:1 De-Mux

- Implementation of full substractor using 1:8 De-Mux

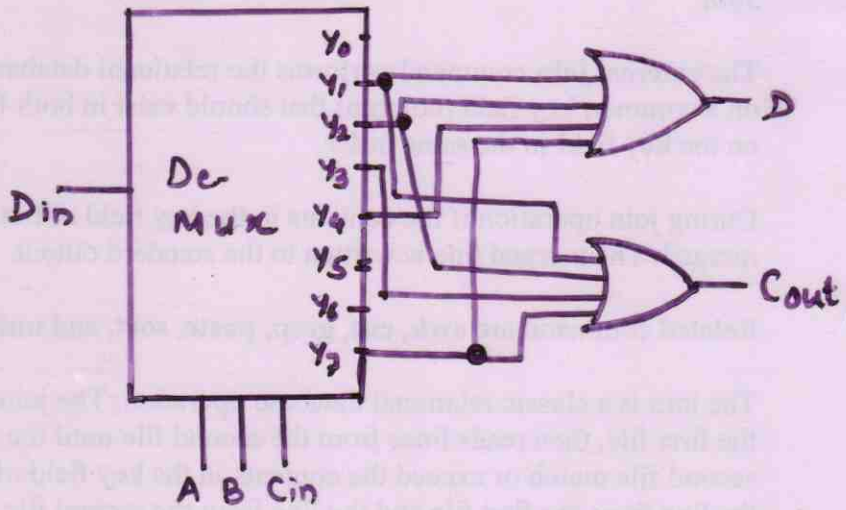
→ Truth table for full substractor

Input			output	
A	B	Cin	D	Cout
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

From truth table

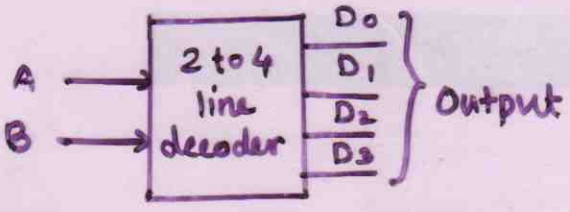
$D = f(ABCin) = \sum m\,(1,2,4,7)$

$Cout = f(A,BCin) = \sum m\,(1,2,3,7)$



- De-Coder

1. Decoder is combinational ckt
2. Like De-Mux it does not have Data input line

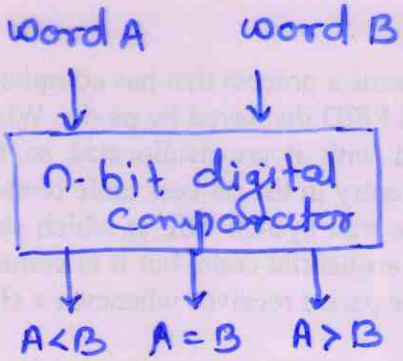


Block diagram

Truth table

Input		outputs			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Magnitude Comparator:-



- fig- shows the digital comparator, is a combinational circuit designed to compare the two n-bit binary words applied as it's inputs.

- The Comparator has ~~to~~ has three outputs A<B, A=B & A>B. Depending on the result of comparision one of these inputs will go high.

1- Bit Magnitude Comparator:-

- 1 bit Comparator Compares the two ~~Sing~~ Single bit no. A & B & produces an o/p that indicates the result of the comparision.

Truth table of 1-bit Comparator.

Inputs		outputs.		
A	B	Y_1: A<B,	Y_2: A=B	Y_3: A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

$$Y_1 = (A<B) = \bar{A}B$$
$$Y_2 = (A=B) = \bar{A}\bar{B} + AB = \overline{A \oplus B}$$
$$Y_3 = A\bar{B}.$$

A - 2 - bit Comparator :-

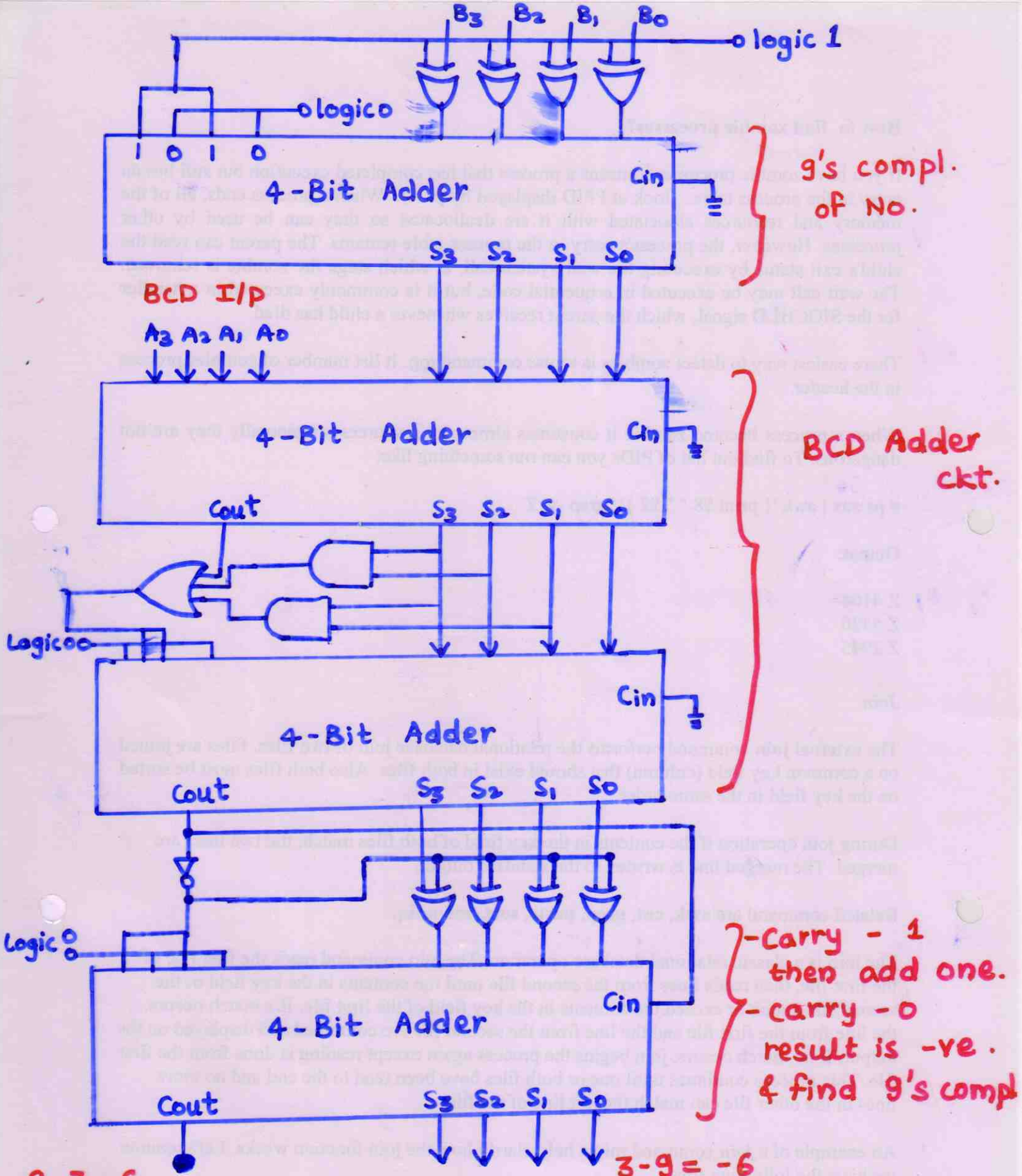
- for a 2-bit Comparator, each input word is 2 bit long.
→ Truth table for 2-bit Comparator.

Input				output		
A_1	A_0	B_1	B_0	$A<B$	$A=B$	$A>B$
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

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

$$A = B = (A_1 \odot B_1)(A_0 \odot B_0) \quad \text{where } \odot = \text{EX-NOR}$$

$$A > B = A_0 \bar{B}_1 \bar{B}_0 + A_1 A_0 \bar{B}_0 + A_1 \bar{B}_1$$



9 - 3 = 6

1001 - BCD for 9
+0110 - 9's comp. of 3
1111 - Invalid BCD
+0110 - Add 6
10101
0110 — BCD for 6

3 - 9 = -6

0011 - BCD for 3
+ 0000 - 9's compl. of 9
0011 - Carry is not generated, result is -ve & in 9's compl. form

Ans. 0110 - 9's compl. of 0011