CREATED BY :-
HARSH PORWAL

Email :- porwalharsh70@gmail.com

Insta :- @hrporwal_007

1 Fundamentals of Digital Systems and Logic Families

Contents

Chapter : 1

Fundamental of Digital System and Logic Families

☆ Decimal Number System

→ $N = 5678.9$

$= (5 \times 10^3) + (5 \times 10^2) + (5 \times 10^1) + (5 \times 10^0) \cdot (9 \times 10^{-1})$

☆ Binary Number System

→ $N = 1101.101$

$= (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \cdot (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$

☆ Octal Number System

→ $N = 5632.471$

$= (5 \times 8^3) + (6 \times 8^2) + (3 \times 8^1) + (2 \times 8^0) \cdot (4 \times 8^{-1}) + (7 \times 8^{-2}) + (1 \times 8^{-3})$

☆ Hexadecimal Number System [8421]

Decimal	Binary	octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

→ $N = 3FD \cdot 84$

$= (3 \times 16^2) + (F \times 16^1) + (D \times 16^0) \cdot (8 \times 16^{-1}) + (4 \times 16^{-2})$

$= (3 \times 16^2) + (15 \times 16^1) + (13 \times 16^0) \cdot (8 \times 16^{-1}) + (4 \times 16^{-2})$

$= (1021.515625)_{16}$

Ex.1 Find the decimal equivalent of $(237.23)_4$

→ $N = (237.23)$

$= (2 \times 4^2) + (3 \times 4^1) + (7 \times 4^0) \cdot (2 \times 4^{-1}) + (3 \times 4^{-2})$

$= (32 + 12 + 7 + 0.5 + 0.1875)$

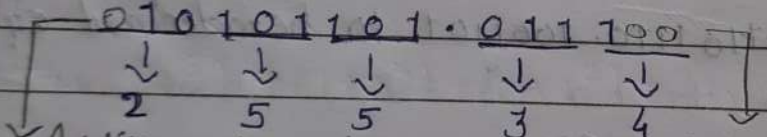
$= (45.6875)_{10}$

☆ Binary to Octal Conversion

Ex.2 Convert 10101101.0111 to octal equivalent.

→ Make group of 3-bit

421



Adding 0 to make group of 3-bit

Adding 0 to make group of 3-bit

$(10101101.0111)_2 = (255.34)_8$

☆ Octal to Binary Conversion

Ex.3 Convert $(125.62)_8$ to Binary.

-> Make group of 3-bit (421)

$$\underset{\downarrow}{1}\ \underset{\downarrow}{2}\ \underset{\downarrow}{5}\ .\ \underset{\downarrow}{6}\ \underset{\downarrow}{2}$$

001 010 101 . 110 010

$\therefore (125.62)_8 = (001010101.110010)_2$

☆ Binary to Hexadecimal Conversion

Ex.4 Convert 110110 1110.100110 1 to hexadecimal equivalent.

-> Make group of 4-Bit

$$\underset{\downarrow}{0011}\ \underset{\downarrow}{0110}\ \underset{\downarrow}{1110}\ .\ \underset{\downarrow}{1001}\ \underset{\downarrow}{1010}$$

3 6 E 9 A

Adding 0 to make group of 4-Bit

Adding 0 to make group of 4-Bit

$\therefore (1101101110.1001101)_2 = (36E.9A)_{16}$

☆ Hexadecimal to Binary Conversion

Ex.5 Convert $(8A9 \cdot B4)_{16}$ to Binary.

→ Make group of 4-Bit (8421)

8	A	9	.	B	4
↓	↓	↓		↓	↓
1000	1010	1001		1011	0100

$\therefore (8A9.B4)_{16} = (100010101001.10110100)_2$

☆ Octal to Hexadecimal Conversion

Ex.6 Convert $(615.25)_8$ to hexadecimal

→ Make group of 3-bit for convert octal to Binary (421)

6	1	5	.	2	5
↓	↓	↓		↓	↓
110	001	101		010	101

→ Make group of 4-bit for convert Binary to hexadecimal. (8421)

0001	1000	1101	.	01010100	
↓	↓	↓		↓	↓
1	8	D		5	4

$\therefore (615.25)_8 = (18D.54)_{16}$

DATE: PAGE:

☆ Hexadecimal to Octal conversion

Ex.7 Convert $(BC66.AF)_{16}$ to octal

(8421)

→ Make group of 4-Bit for convert Hex to Binary

B	C	6	6	.	A	F
↓	↓	↓	↓		↓	↓
1011	1100	0110	0110	.	1010	1111

→ Make group of 3-bit for convert Binary to octal

001	011	110	001	100	110	.	101	011	110
↓	↓	↓	↓	↓	↓		↓	↓	↓
1	3	6	1	4	6		5	3	6

$\therefore (BC66.AF)_{16} = (136146.536)_8$

☆ Convert Decimal into Binary

Ex.8 Convert (12.125) decimal # into Binary

→ Integer Part

2 | 12 → 0
2 | 6 → 0
2 | 3 → 1
2 | 1 → 1
0

$(12)_{10} = (1100)_2$

→ Fractional Part

0.125 x 2 = 0.25 [0]
0.25 x 2 = 0.50 [0]
0.50 x 2 = 1.0 [1]

$(0.125)_{10} = (001)_2$

$\therefore (12.125)_{10} = (1100.001)_2$

☆ Convert Decimal to octal

Ex.9 convert 658.825 decimal into octal

→ Integer Part

8	658	→ 2
8	82	→ 2
8	10	→ 2
	1	→ 1

$(658)_{10} = (1222)_8$

→ Fractional Part

0.825	x	8	= 6.6	[6]
0.6	x	8	= 4.8	[4]
0.8	x	8	= 6.4	[6]

$(0.825)_{10} = (646)_8$

$\therefore (658.825)_{10} = (1222.646)_8$

☆ Convert Decimal to Hexadecimal

Ex.10 Convert 5386.345 decimal into hexadecimal

→

16	5386	→ 10 (A)
16	336	→ 0
16	21	→ 5
16	1	→ 1
	0	

$(5386)_{10} = (150A)_{16}$

→ Fractional Part

0.345	x 16	= 5.52	[5]
0.52	x 16	= 8.32	[8]
0.32	x 16	= 5.12	[5]

$(0.345)_{10} = (585)_{16}$

$\therefore (5386.345)_{10} = (150A.585)_{16}$

☆ Convert decimal into Any Radix Number

Ex 11 Convert the base-5 number $(4433214)_5$ directly to base-12.

→ $(4433214)_5 = (4\times5^6)+(4\times5^5)+(3\times5^4)+(3\times5^3)+(2\times5^2)+(1\times5^1)+(4\times5^0)$

$= (4\times15625)+(4\times3125)+(3\times625)+(3\times125)+(2\times25)+(1\times5)+(4\times1)$

$= 62500+12500+1875+375+50+5+4$

$= 77309$

12	77309	→ 5
12	6442	→ 10 (A)
12	536	→ 8
12	44	→ 8
	3	→ 3

$\therefore (4433214)_5 = (388A5)_{12}$

Ex-12 Convert the decimal number 250.5 to base 3.

→ Integer Part

3	250	→ 1
3	83	→ 2
3	27	→ 0
3	9	→ 0
3	3	→ 0
	1	→ 1

$(250)_{10} = (100021)_3$

→ Fractional Part

$0.5 \times 3 = 1.5 \quad [1]$
$0.5 \times 3 = 1.5 \quad [1]$

$(0.5)_{10} = (11)_3$

$\therefore (250.5)_{10} = (100021.11)_3$

DATE: PAGE:

Ex.13 Convert the decimal number 250.5 to base 7.

→

7	250	→ 5
7	35	→ 0
7	5	→ 5
	0	

$(250)_{10} = (505)_{7}$

→ Fractional Part

$0.5 \times 7 = 3.5$ [3]

$0.5 \times 7 = 3.5$ [3]

$\therefore (250.5)_{10} = (505.33)_{7}$

☆ 1's Complement Representation

Ex.14 Find 1's complement of $(11010100)_2$

→

1	1	0	1	0	1	0	0	Number
↓	↓	↓	↓	↓	↓	↓	↓	Not operation
0	0	1	0	1	0	1	1	1's complement number

☆ 2's Complement Representation

→ The 2's complement is binary number ~~when~~ that results when we add 1 to the 1's complement.

2's complement = 1's complement + 1

Ex-15 Find 2's complement of $(11000100)_2$

→

	1	1	0	0	0	1	0	0	Number
+	0	0	1	1	1	0	1	1	1's complement

			1	1				
	0	0	1	1	1	0	1	1	1's Complement
+							1	1	
	0	0	1	1	1	1	0	0	2's complement

☆ Binary Addition

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$ [carry 1]

Ex. 16 Perform addition of $(1100\ 1100)_2$ and $(1101\ 1010)_2$

→

```
  1 1  1 1        Carry
  1 1 0 0 1 1 0 0   Number : 1
+ 1 1 0 1 1 0 1 0   Number : 2
-----------------
1 1 0 1 0 0 1 1 0   Result
```

Ex. 17 Add $(28)_{10}$ and $(15)_{10}$ by converting them into binary.

→ $(28)_{10} = (011100)_2$
$(15)_{10} = (01111)_2$

$$\begin{array}{r} (28)_{10} \\ +\ (15)_{10} \\ \hline (43)_{10} \end{array}$$

```
  1 1 1          Carry
  0 1 1 1 0 0    Binary of (28)10
+ 0 0 1 1 1 1    Binary of (15)10
---------------
1 0 1 0 1 1      Result: Binary of (43)10
```

Binary Substraction

$0 - 0 = 0$
$0 - 1 = -1$ [Borrow 1]
$1 - 0 = 1$
$1 - 1 = 0$

Ex. 18 Perform $(11101100)_2 - (00110010)_2$

→

	1 1 1	Carry
	1 1 1 0 1 1 0 0	Number : 1
−	0 0 1 1 0 0 1 0	Number : 2
	1 0 1 1 1 0 1 0	Result

$\therefore (11101100)_2 - (00110010)_2 = (10111010)_2$

☆ Binary Substraction using 1's complement method

→ Method : For operation A - B

(i) Take 1's complement of B.
(ii) ~~Convert 1's complement of B.~~ Result: A + 1's com. B
(iii) If carry is genrated then the result is positive and in the true form. Add carry to the result to get the final result.
(iv) If carry is not genrated then the result is negative and in the 1's complement form.

Ex.19 Perform $(28)_{10} - (15)_{10}$ using 6-Bit 1's complement representation.

→ $(28)_{10} = (011100)_2$

$(15)_{10} = (001111)_2$

→ 1's complement of $(15)_{10}$

$= (110000)_2$

```
  [1] 1        Carry
    0 1 1 1 0 0   Binary of (28)10
+   1 1 0 0 0 0   Binary of (15)10
  ---------------
    0 0 1 1 0 0   Result
```

Add extra carry to the result

```
    0 0 1 1 0 0   Result
+             1   carry
  ---------------
    0 0 1 1 0 1   Final Result
```

Ex.20 Perform $(15)_{10} - (28)_{10}$ using 6-Bit 1's complement representation

→ $(15)_{10} = (001111)_2$

$(28)_{10} = (011100)_2$

→ 1's complement of $(28)_{10}$

$= (100011)_2$

```
  1 1 1 1      (carry)
  0 0 1 1 1 1  Binary of (15)10
+ 1 0 0 0 1 1  1's complement of (28)10
-------------
  1 1 0 0 1 0  (Result)
```

→ Carry is not genrated then the result is negative that's why the we have to do 1's complement of Resut

→ 1's complement of resut :

0 0 1 1 0 1 (Final Result)

☆ Binary Subtraction using 2's Complement

→ **Method:** For operation A - B

(i) Take 1's complement of B.
(ii) Take 2's complement of B.
(iii) Result: A + 2's complement of B
(iv) If carry is genrated then the result is positive. In this case carry is ignored.
(v) If carry is not genrated the the result is negative. In this case convert the 2's complement form of Result.

Ex.21 Perform $(28)_{10} - (15)_{10}$ using 6-Bit 2's complement representation.

→ $(28)_{10} = (011100)_2$
$(15)_{10} = (001111)_2$

$$\begin{array}{r} (28)_{10} \\ -(15)_{10} \\ \hline (13)_{10} \end{array}$$

→ 2'n complement of $(15)_{10}$

$$\begin{array}{rl} 110000 & (\text{1's complement of } (15)_{10}) \\ + \quad 1 & \\ \hline 110001 & (\text{2's complement of } (15)_{10}) \end{array}$$

→

$$\begin{array}{rl} & \boxed{1}\,1 \\ \text{Ignore the carry} & \\ & 011100 \quad \text{Binary of } (28)_{10} \\ + & 110001 \quad \text{2'n complement of } (15)_{10} \\ \hline & 000101 \quad \text{Positive Result} \end{array}$$

→ Note: Carry is ignored because result is positive.

Ex.22 Perform $(15)_{10} - (28)_{10}$ using 6-Bit 2's complement representation

→ $(15)_{10} = (001111)_2$

$(28)_{10} = (011100)_2$

$(15)_{10}$

$-(28)_{10}$

$(-13)_{10}$

→ 2's complement of $(28)_{10}$

```
  1 1        Carry
  1 0 0 0 1 1   1's complement of (28)10
+         1     Add 1
-------------
  1 0 0 1 0 0   2's complement of (28)10
```

```
    1 1
  0 0 1 1 1 1   Binary of (15)10
+ 1 0 0 1 0 0   2's complement of (28)10
-------------
  1 1 0 0 1 1   Negative Result
```

→ The result is negative that's why we have to convert 2's complement of Result.

```
  0 0 1 1 0 0   1's complement of Resut
+           1
-------------
  0 0 1 1 0 1   2's complement of Result =
                Final Resut
```

☆ Octal Addition

→ The sum of two octal digit is the same as their decimal sum, provide the decimal sum is less than 8. If the decimal sum is 8 or greater, Substract 8 to obtain the octal digit.

→ A carry of 1 is produced when the decimal sum is correct this way, as illustrade in the following example.

Example:

(a) $4_8 + 2_8 = 6_8$

(b) $6_8 + 7_8 = (13-8)_8 = 5_8$ [carry 1]

(c) $1_8 + 7_8 = (8-8)_8 = 0_8$ [carry 1]

Ex.23 Add $(167)_8$ and $(325)_8$

→

	1	1	Carry
	1	6	7 Number: 1
	3	2	5 Number: 2
5	(9-8)	(12-8)	Sum
5	1	4	Result

$\therefore (167)_8 + (325)_8 = (514)_8$

Ex.24 Add the octal numbers $(341)_8$, $(125)_8$, $(472)_8$ and $(577)_8$.

→

1	2	1	Carry
3	4	1	Number: 1
1	2	5	Number: 2
4	7	2	Number: 3
5	7	7	Number: 4
1	(15-8)	(21-16)	(15-8)
1	7	5	7 Result

$\therefore (341)_8 + (125)_8 + (472)_8 + (577)_8 = (1757)_8$

✡ Octal Substraction using 8's Complement

⇒ How to find 8's complement-

→ (i) Substract each digit of a number from 7 to get the 7's complement of number.
(ii) Add 1 to get the 8's complement of number.

$$\begin{array}{r} 777 \\ -346 \\ \hline 431 \\ +\quad 1 \\ \hline 432 \end{array}$$

⇒ Octal substraction using 8's complement.

(i) Find 8's complement
(ii) Add first number and second number's 8's complement.
(iii) If carry is produced in the addition it is ignored, otherwise find the 8's complement of Result with negative sign.

Ex.25 Use 8's complement method of subtraction to compute $(516)_8 - (413)_8$

→ Find 8's complement of $(413)_8$

$$\begin{array}{r} 7\ 7\ 7 \\ -\ 4\ 1\ 3 \\ \hline 3\ 6\ 4 \\ +\ \quad\quad 1 \\ \hline 3\ 6\ 5 \end{array}$$

$(413)_8$

7's complement of $(413)_8$

8's complement of $(413)_8$

→ $(516)_8 + (365)_8 =$

$$\begin{array}{r} 1\ 1\ \ 1 \\ 5\ 1\ 6 \\ +\ 3\ 6\ 5 \\ \hline 1\ (9-8)\ (8-8)\ (11-8) \\ \boxed{1}\ 1\ 0\ 3 \end{array}$$

8's complement

Result

→ Carry is ignored,

$= (103)_8$

$\therefore (516)_8 - (413)_8 = (103)_8$

DATE: PAGE:

Ex.26 Substract using 8's complement $(376)_8 - (451)_8$

-> Find 8's complement of $(451)_8$

$$\begin{array}{r} 7\ 7\ 7 \\ -\ 4\ 5\ 1 \\ \hline 3\ 2\ 6 \\ +\ \quad\quad 1 \\ \hline 3\ 2\ 7 \end{array}$$

$(451)_8$

7's complement of $(451)_8$

8's complement of $(451)_8$

-> $(376)_8 + (327)_8$

$$\begin{array}{r} \quad 1 \\ 3\ 7\ 6 \\ +\ 3\ 2\ 7 \\ \hline 6\ 4\ (13-8) \\ 6\ 4\ 5 \end{array}$$

Carry

$(376)_8$

8's complement of $(451)_8$

Result

-> No carry, hence take 8's complement of Result

$$\begin{array}{r} 7\ 7\ 7 \\ -\ 6\ 4\ 5 \\ \hline 1\ 3\ 2 \\ +\ \quad\quad 1 \\ \hline 1\ 3\ 3 \end{array}$$

8's complement of Result

$\therefore (376)_8 - (451)_8 = (-133)_8$

Ex. 27 Substract $(45)_8$ from $(66)_8$

→ 8's complement of 45

$= 77 - 45 + 1$

$= 33$

1	1	Carry
	6	6
+	3	3
1	(10-8)	(9-8)
1	2	1

8's complement

Result

→ Carry is ignored,

$= (21)_8$

☆ Hexadecimal Addition

Ex.28 Add $(6E)_{16}$ and $(C5)_{16}$

→

1	1		Carry
	6	E	
+	C	5	
1	(19-16)	(14-16)	
1	3	3	Result

$\therefore (6E)_{16} + (C5)_{16} = (133)_{16}$

Ex.29 Add $(892)_{16}$, $(58F)_{16}$, $(178)_{16}$ and $(48E)_{16}$

→

1	2	2	Carry
	8	9	2
+	5	8	F
+	1	7	8
+	4	8	E
1	(20-16)	(34-32)	(39-32)
1	4	2	7

Result

$\therefore (892)_{16} + (58F)_{16} + (178)_{16} + (48E)_{16} = (1427)_{16}$

DATE: PAGE:

☆ Hexadecimal Subtraction Using 16's Complement.

⇒ How to Find 16's complement of $(ABC)_{16}$

(i) Substract each digit with number of 15' to get 15's complement

(ii) Add 1 to it to get the 16's complement

```
  15  15  15
 - A   B   C
 -----------
  5   7   3
+             1
 -----------
  5   7   4
```

⇒ Substraction Using 16's complement

(i) Find 16's complement.

(ii) Add first Number and 16's complement of second Number

(iii) If carry is produced in the addition it is ignored otherwise find 16's complement and result with negative sign.

Ex.30 Substract $(CB2)_{16} - (972)_{16}$ using 16's complement method.

→ Find 16's complement of $(972)_{16}$.

	15	15	15	
−	9	7	2	$(972)_{16}$
	6	8	D	15's complement
+			1	Add 1
	6	8	E	16's complement

→ $(CB2)_{16} + (68E)_{16} =$

	1	1		carry
	C	B	2	$(CB2)_{16}$
+	6	8	E	16's complement
	(20-16)	(18-16)		
1	(14-16)	(20-16)	(16-16)	
1	3	4	0	

	1	1	1	carry
	C	B	2	$(CB2)_{16}$
+	6	8	E	16's complement
1	(14-16)	(20-16)	(16-16)	
[1]	3	4	0	Result

→ Carry is ignored,

∴ $(CB2)_{16} - (972)_{16} = (340)_{16}$

Ex.3) Substract $(3B7)_{16} - (854)_{16}$ using 16's complement of method.

→ Find 16's complement of $(854)_{16}$

15	15	15	
− 8	5	4	$(854)_{16}$
7	A	B	15's complement
		1	Add 1
7	A	C	16's Complement

→ $(3B7)_{16} + (7AC)_{16} =$

1	1		carry
3	B	7	$(3B7)_{16}$
+ 7	A	C	16's complement
B	(22-16)	(19-16)	
B	6	3	Result

→ No carry, hence take 16's complement of Result,

15	15	15	
− B	6	3	
4	9	C	15's complement of Result
+		1	Add 1
4	9	D	16's complement of Result

→

∴ $(3B7)_{16} - (854)_{16} = (-49D)_{16}$

☆ BCD

Ex.32 Convert 1000 0110 (BCD) to decimal, binary and octal.

→ BCD to Decimal,

1000 0110
↓ ↓
8 6

$= (86)_{10}$

$\therefore (1000\ 0110)_{BCD} = (86)_{10}$

→ BCD to Octal

0100 0011 0
↓ ↓ ↓
1 2 6

$\therefore (10000110)_{BCD} = (126)_8$

→ Decimal to Binary

2	86	→ 0
2	43	→ 1
2	21	→ 1
2	10	→ 0
2	5	→ 1
2	2	→ 0
	1	→ 1

$\therefore (86)_{10} = (1010110)_2$

→ Binary to octal,

001 010 110
↓ ↓ ↓
1 2 6

$\therefore (1010110)_2 = (126)_8$

DATE: PAGE:

☆ BCD Addition

Case:1 Sum equals 9 or less with carry 0

→ Lets consider addition of 6 and 3 in BCD.

```
      1 1
6     0 1 1 0    BCD of 6
+3    0 0 1 1    BCD of 3
-----------------
9     1 0 0 1    Valid BCD Number
```

→ Addition of Normal Digit and addition of BCD code are same that's why it is Valid BCD Numbers.

Case:2 Sum greater than 9 with carry 0

→ Let's consider addition of 6 and 8 in BCD

```
6     0 1 1 0    BCD of 6
+8  + 1 0 0 0    BCD of 8
-----------------
14    1 1 1 0    Invalid BCD (01110) > 9
```

→ The sum of Normal digit and BCD Number are not same and grater than 9 that's why we have to correct it by addition of six (0110) in invalid BCD Number.

Case: 3 Sum equals 9 or less with carry 1

→ Let's consider addition of 8 and 9

```
    1
 8      1000   BCD of 8
+9    + 1001   BCD of 9
17     10001   Invalid BCD Result

      0001      0001   Invalid BCD Result
       ↓         ↓
       1         1
   (carry)
```

→ In this case the Result is incorrect. To get the correct Result we have to do addition of 6(0110)

```
   0001 0001   Invalid BCD Number
+  0000 0110   Add 6
   0001 0111   Valid BCD Number
    ↓    ↓
    1    7
```

Ex.35 Add 96 and 56 BCD Number

→

```
                1   1 1
96        1 0 0 1   0 1 1 0   BCD of 96
+56     + 0 1 0 1   0 1 1 0   BCD of 56
152       1 1 1 0   1 1 0 0   Invalid BCD
        + 0 1 1 0   0 1 1 0
        ---------------------

          1 1 1   1 1
          1 1 1 0   1 1 0 0   Invalid BCD
        + 0 1 1 0   0 1 1 0   Add 66
        ---------------------
        1 0 1 0 1   0 0 1 0   BCD of 152

0001 0101 0010
 ↓    ↓    ↓
 1    5    2
```

∴ 96 + 56 = 152

Ex.36 Add 954 and 463 BCD Number

→

```
                1                    carry
954    1001 0101 0100               BCD of 954
+463 + 0100 0110 0011               BCD of 463
1417   1101 1011 0111               Invalid BCD
```

→ Add 660 to get valid answer,

```
       1 1 1 1 1 1 1                 carry
       1101 1011 0111                Invalid BCD
     + 0110 0110 0000                Add 660
       1010 0001 0111                Valid BCD
                                     of (1417)
```

0001 0100 0001 0111
↓ ↓ ↓ ↓
1 4 1 7

Ex.37 Add 79 and 16 BCD Number

```
 79          1 1 1
+16           0111 1001              BCD of 79
 95         + 0001 0110              BCD of 16
              1000 1111              Invalid BCD
            + 0000 0110              Add 6
              1001 0101              Valide BCD of (95)
```

DATE: PAGE:

☆ BCD Substraction using 9's complement

⇒ **Method : A - B**

(i) Substract each digit of B with 9 to get 9's complement.

(ii) Add two number using BCD addition (A + 9's complement of B)

(iii) If carry is not genrated then find 9's complement and answer with negative sign. otherwise If carry is genrated the result is positive and add carry to Result.

Ex.38 Perform (46) - (22) in BCD using 9's complement

→ Find 9's complement of 22,

~~0100 0110~~
~~0010 0010~~

= 99 - 22
= 77

⇒ $(46)_{10}$ + 9's complement of B

```
  1     1 1
  0100 0110    (46) BCD
+ 0111 0111    9's complement of B
-------------
  1011 1101    Invalid Number
```

→ Add 6 in each digit.

```
  1 1 1 1 1 1
    1 0 1 1 1 1 0 1   Invalid Number
+   0 1 1 0 0 1 1 0   Add 66
  ---------------
  [1] 0 0 1 0 0 0 1 1   Valid Number
```

→ Add carry to the Result.

```
                1 1
    0 0 1 0 0 0 1 1     Valid Number
+               1       Add 1
  ---------------
    0 0 1 0 0 1 0 0     Final Result
      ↓     ↓
      2     4
```

$\therefore (46)_{BCD} - (22)_{BCD} = (24)_{BCD}$

Ex.39 Perform (24) − (56) in BCD using 9's complement.

→ Find 9's complement of 56,
= 99 − 56 = 43

→ Add 24 and 9's complement of 56,

```
    0 0 1 0 0 1 0 0
+   0 1 0 0 0 0 1 1
  ---------------
    0 1 1 0 0 1 1 1
```

→ There is no carry genrate take 9's complement
= 99 − 67 = 32

$\therefore (24)_{10} - (56)_{10} = (-32)_{10}$

Ex.40 Perform 893 - 647 in BCD using 9's complement.

→

893 — 1000 1001 0011
-647 — 0110 0100 0111
246

→ Find 9's complement of 647

= 999 - 647
= 352

→ Add 893 and 9's complement of 647

```
          1   1
  1000 1001 0011   BCD of 893
+ 0011 0101 0010   9's complement 647
-------------------
  1011 1110 0101   Invalid Number
```

→ Add 660 to the Invalid Number,

```
  1 1 1 1 1 1
  1011 1110 0101   Invalid Number
+ 0110 0110 0000   Add 660
-------------------
 [1]0010 0100 0101   Result
```

→ Add carry to the Result,

```
          1
  0010 0100 0101   Result
+              1   Add 1
-------------------
  0010 0100 0110   BCD of 246
                   Final Result
```

☆ BCD Substraction using 10's complement method

-> Substraction of BCD number can also be perfome by representing negative numbers in the 10's complement form.

-> The 10's complement of a decimal number is equal to the 9's complement plus 1

Step:1 Find the 10's complement of negative number.

Step:2 Add two numbers using BCD Addition (Minuend + 10's complement)

Step:3 If carry is not genrated result is negative and find the 10's complement of the result, otherwise result is positive and discard carry.

Ex. $(935)_{11}$

$= 9 \times 11^2 + 3 \times 11^1 + 5 \times 11^0$

$= 1089 + 33 + 5 = (1127)_{10}$

-> $9999 - 1127 + 1 = 8873$

Ex. $(6106)_{10}$

$= 9999 - 6106 + 1$

$= 3894$

Ex. 41 Perform $(46)_{10} - (22)_{10}$ in BCD using 10's complement.

→ Step 1: Find 10's complement of 22

$$\text{10's of } 22 = (99 - 22) + 1 = 78$$

→ Step 2: Add 46 and 10's complement of 22.

```
      1
  0 1 0 0 0 1 1 0      (BCD of 46)
+ 0 1 1 1 1 0 0 0      10's complement of 22
  1 0 1 1 1 1 1 0

  46
+(-22)
  24
```

```
  1 1 1 1 1 1          Carry
  1 0 1 1 1 1 1 0      Invalid number
+ 0 1 1 0 0 1 1 0      Add 6 in each digit
1 0 0 1 0 0 1 0 0 0    Result
    2       4
```

→ Since there is a carry the result is positive and true.

$$(46)_{BCD} - (22)_{BCD} = (24)_{BCD}$$

Ex.42 Perform $(24)_{10} - (56)_{10}$ in BCD using 10's complement

→ Find 10's complement of 56

$= (99 - 56) + 1 = 44$

→ Add $(24)_{10}$ and 10's complement of 56

	1	Carry
24	0 0 1 0 0 1 0 0	BCD of 24
-56	+ 0 1 0 0 0 1 0 0	10's complement of 56
-32	0 1 1 0 1 0 0 0	Result

Since carry is 0 the answere is negative.

→ Take 10's complement of answer

10's complement of 68 = $(99 - 68) + 1$

$= (-32)_{10}$

☆ Other 4-Bit BCD code

→ 2-4-2-1

Decimal digit	2-4-2-1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 1 1 0
9	1 1 1 1

☆ Other 4-Bit BCD code

→ 3 3 2 1
→ 4 2 2 1
→ 5 2 1 1
→ 5 3 1 1
→ 5 4 2 1
→ 6 3 1 1
→ 7 4 2 1

Ex. Represent $(7)_{10}$ using all 4-bit BCD code

-> 3321

$(7)_{10} = 1101$

-> 4221

$(7)_{10} = 1101$

-> 5211

$(7)_{10} = 1100$

-> 5311

$(7)_{10} = 1011$

-> 5421

$(7)_{10} = 1010$

-> 6311

$(7)_{10} = 1010$

-> 7421

$(7)_{10} = 1000$

☆ Excess - 3 code

Decimal Digit	Excess-3 code
0	0 0 1 1
1	0 1 0 0
2	0 1 0 1
3	0 1 1 0
4	0 1 1 1
5	1 0 0 0
6	1 0 0 1
7	1 0 1 0
8	1 0 1 1
9	1 1 0 0

Ex. Find xs-3 code for 37

→

3	7	
0 0 1 1	0 1 1 1	BCD
0 1 1 0	1 0 1 0	BCD + 3 = xs-3 code

→ $(37)_{10} = (0110\ 1010)$ xs-3

☆ Excess -3 Addition

→ Add two excess number using Binary Addition

→ If carry = 1 → add 3 $(0011)_2$ to the sum of two digit

= 0 → Substract 3 $(0011)_2$ from the sum

Ex.4) Perfom the excess-3 addition of 8 and 6

→

```
  1  1  1
     1  0  1  1
  +  1  0  0  1
  ---------------
  1  0  1  0  0
```

→ There is carry 1 add 3

```
        1  1
     0  0  0  1     0  1  0  0
  +  0  0  1  1     0  0  1  1
  ---------------------------
     0  1  0  0     0  1  1  1
```

Excess-3 Subtraction

→ Complement the subtrahend (B).
→ Add xs-3 code of A and complement of B
→ If carry = 0 → Result Negative. Substract $3_{(0011)_2}$
1 → Result is positive. Add $3_{(0011)_2}$. and end around carry.

Ex. Perform the xs-3 subtraction of 8 and 5

→

```
  1 1 1 1
    1 0 1 1    xs-3 of 8
+   0 1 1 1    Complement of 5 in xs-3
  ---------
  [1] 0 0 1 0
   |
 carry
```

→ Add 3

```
      1
  0 0 1 0
+ 0 0 1 1    Add 3
  ---------
  0 1 0 1
```

→ Add end around carry

```
  0 1 0 1
+       1
  ---------
  0 1 0 0    Exces-3 for 3
```

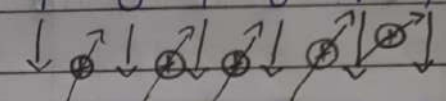
☆ Gray to Binary Conversion

→ The most significant Bit (MSB) of the Binary number is the same as the most significate Bit (MSB) of the grey code number. So write down MSB as it is.

→ To obtain the next Binary Digit perform an exclusive between the bit just written down and the next grey code bit. write down the result.

Ex. Convert gray code 101011 into its binary equivalent.

→ Gray Code : 1 0 1 0 1 1



Binary code : 1 1 0 0 1 0

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

☆ Binary to Gray code conversion

→ The MSB of the Gray code is the same as the MSB of the Binary number. So write down MSB as it is.

→ To obtain the next gray digit perform exclusive or operation between previous and current Binary Bit.

Ex. Convert 10111011 in binary into gray code.

→ Binary code: 1 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 1 ⊕ 1

Gray code: 1 1 1 0 0 1 1 0

Ex. Convert $(46)_{10}$ to gray code.

→ $(46)_{10} = (101110)_2$

→ Binary code: 1 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 0

Gray code: 1 1 1 0 0 1

2 Logic Gates and Boolean Algebra

Contents

Chapter: 2

Logic Gates and Boolean Algebra

☆ Logic Gates

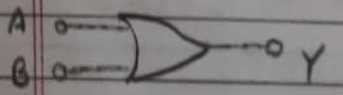
→ AND Gate

Boolean Expression:

$Y = A \cdot B$

Input		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

→ OR Gate

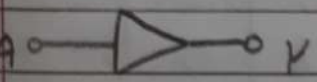


Boolean Expression:

$Y = A + B$

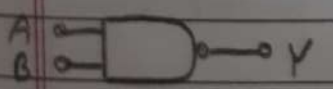
Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

→ NOT Gate



Input	Output
A	Y
0	0
1	1

→ NAND Gate



Boolean Expression:

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

Input		Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

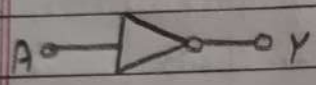
→ NOR Gate

Boolean Expression:

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

Input		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

→ NOT Gate

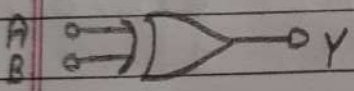


Boolean Expression:

$$Y = \overline{A}$$

Input	Output
0	1
1	0

→ Ex-OR Gate

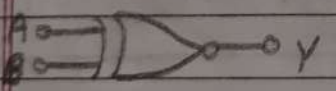


Boolean Expression:

$$Y = A \oplus B$$

Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

→ Ex-NOR Gate



Boolean Expression:

$$Y = \overline{A \oplus B}$$

Input		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

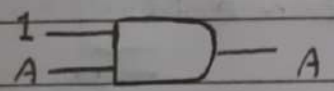
☆ Boolean Algebra

→ Boolean algebra is a mathamatical system that defines a series of logical operation (AND, OR, NOT) performed on sets of variable (a, b, c, ...) When stated in this form, the expression is called Boolean equation or switching equation.

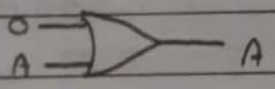
* Closure property

→ Identity Property

$A \cdot 1 = 1 \cdot A = A$



[AND]



[OR]

→ Commutative Property

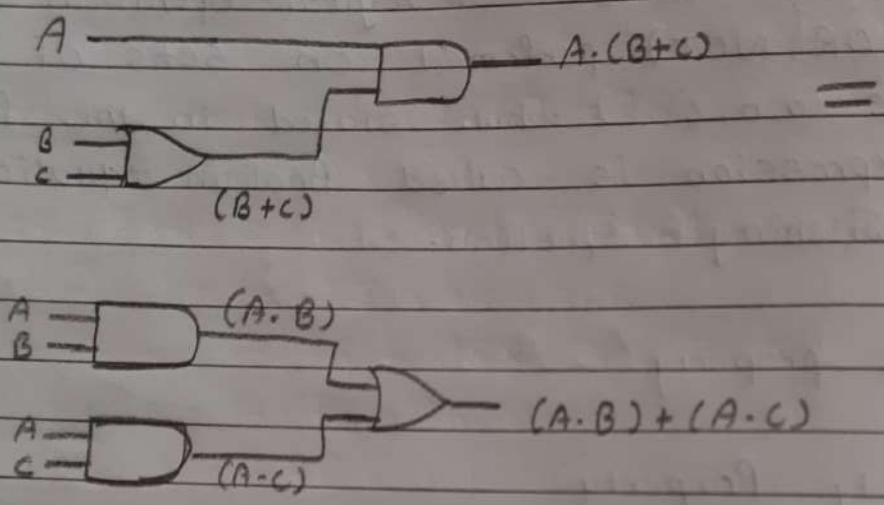
With respect to + : $A + B = B + A$
With respect to · : $A \cdot B = B \cdot A$

→ Associative Property

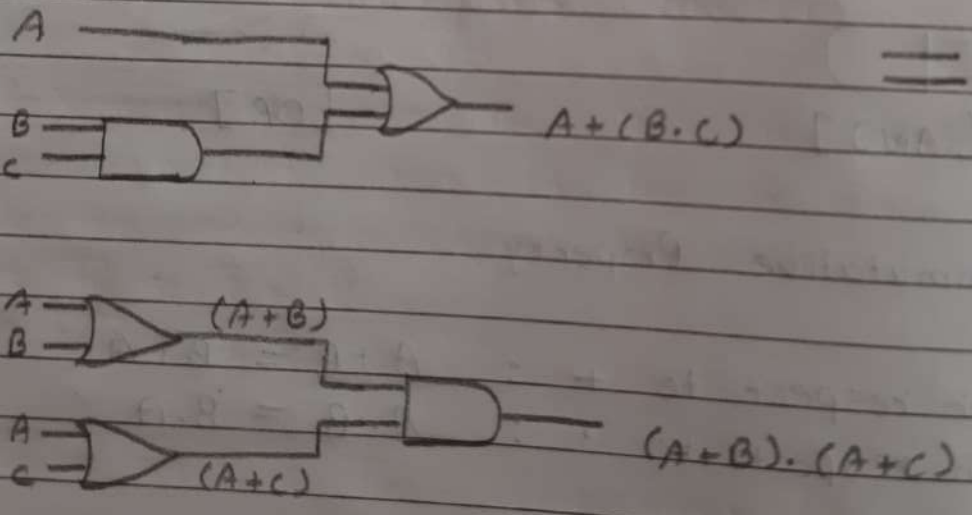
Associative Property of + : $A + (B + C) = (A + B) + C$
Associative Property of · : $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

-> Distributive Property

Property of · over + : $A \cdot (B+C) = (A \cdot B) + (A \cdot C)$
Property of + over · : $A + (B \cdot C) = (A+B) \cdot (A+C)$



(Distributive Property of ·over +)



(Distributive Property of + over ·)

→ Complement Property

- For every Binary element
  $A = 1, \quad \overline{A} = 0$
- Complement Property of $\cdot$ is $A \cdot \overline{A} = 0$
  Complement Property of $+$ is $A + \overline{A} = 1$

→ Absorption Property

(i) $A + AB = A(1+B)$
$= A$

(ii) $A(A+B) = AA + AB$
$= A + AB$
$= A(1+B)$
$= A$

* Properties of Boolean algebra

→ $A + 0 = A$	$A \cdot 1 = A$
→ $A + B = B + A$	$AB = BA$
→ $A(B+C) = AB + AC$	$A + BC = (A+B)(A+C)$
→ $A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
→ $A + A = A$	$A \cdot A = A$
→ $A + 1 = 1$	$A \cdot 0 = 0$
→ $\overline{\overline{A}} = A$	
→ $A + AB = A$	$A(A+B) = A$
→ $A + \overline{A}B = A + B$	$A \cdot (\overline{A} + B) = AB$
→ $A + (B+C) = (A+B) + C$	$A(BC) = (AB)C$

Ex. 1 Prove that $A + \bar{A}B = A + B$

→ $A + \bar{A}B = A + AB + \bar{A}B$

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

$= A + B \cdot 1$

$= A + B$

Ex. 2 Prove that $A \cdot (\bar{A} + B) = AB$

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

$= A\bar{A} + AB + ABB$

$= AB + ABB$

$= AB + AB$

$= AB$

☆ ~~Boolean~~ DeMorgan's Theorems

→ DeMorgan suggest two theorems that from an important part of Boolean algebra. In the equation from they are

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

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

(i) $\overline{AB} = \bar{A} + \bar{B}$: The complement of the product is equal to the sum of the complements.

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

(ii) $\overline{A+B} = \overline{A} \cdot \overline{B}$: The complement of sum is equal to the product of the complements.

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

* Principle of Duality

→ Starting with a boolean relation you can derive another boolean relation by,
(i) Changing each OR sign to an AND sign
(ii) Changing each AND sign to an OR sign.
(iii) Complementing any 0 or 1 appearing in the expression.

Ex.3 Simplify $AB + \overline{AC} + A\overline{B}C(AB+C)$

→
$= AB + \overline{AC} + AA\overline{B}BC + A\overline{B}CC$
$= AB + \overline{AC} + A\overline{B}CC$
$= AB + \overline{AC} + A\overline{B}C$
$= AB + \overline{A} + \overline{C} + A\overline{B}C$
$= \overline{A} + B + \overline{C} + A\overline{B}C$
$= \overline{A} + A\overline{B}C + B + \overline{C}$
$= \overline{A} + \overline{B}C + B + \overline{C}$
$= \overline{A} + B + \overline{C} + \overline{B}C$
$= \overline{A} + B + \overline{C} + \overline{B}$
$= \overline{A} + \overline{C} + 1$
$= 1$

Ex.4 (i) $\bar{A}B + \bar{A}B\bar{C} + \bar{A}BCD + \bar{A}B\bar{C}\bar{D}E$

(ii) $(P+Q+R)(\bar{P}+\bar{Q}+\bar{R})P$

→ (i) $\bar{A}B + \bar{A}B\bar{C} + \bar{A}BCD + \bar{A}B\bar{C}\bar{D}E$

$= \bar{A}B(1 + \bar{C} + CD + \bar{C}\bar{D}E)$

$= \bar{A}B$

→ (ii) $(P+Q+R)(\bar{P}+\bar{Q}+\bar{R})P$

$= (P.\bar{P} + P.\bar{Q} + P.\bar{R} + \bar{P}.Q + Q.\bar{Q} + Q.\bar{R} + \bar{P}.R + \bar{Q}.R + R.\bar{R})P$

$= (P\bar{Q} + P\bar{R} + \bar{P}Q + Q\bar{R} + \bar{P}R + \bar{Q}R)P$

$= P.P\bar{Q} + P.P\bar{R} + P.\bar{P}Q + P.Q\bar{R} + P.\bar{P}R + P.\bar{Q}R$

$= P\bar{Q} + P\bar{R} + PQ\bar{R} + P\bar{Q}R$

$= P\bar{Q}(1+R) + P\bar{R}(1+Q)$

$= P\bar{Q} + P\bar{R}$

Ex.5 (a) $xyz + \bar{x}y + xy\bar{z}$

(b) $(\overline{A+B})(\overline{\bar{A}+B})$

→ (a) $xyz + \bar{x}y + xy\bar{z}$

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

$= xy + \bar{x}y$

$= y(x+\bar{x})$

$= y$

(b) $(\overline{A+B})(\overline{\bar{A}+B})$

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

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

$= \bar{B}$

Ex. 6 Find the complement of the following Boolean function and reduce to a minimum number of literals. $\bar{B}D + \bar{A}B\bar{C} + ACD + \bar{A}BC$

→ $Y = \bar{B}D + \bar{A}B\bar{C} + ACD + \bar{A}BC$

$Y = \bar{B}D + \bar{A}B + ACD$

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

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

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

$= AB\bar{C} + A\bar{D}(B + \bar{C} + 1) + \bar{B}\bar{D}(\bar{A} + \bar{C} + 1)$

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

Ex. 7 Prove that: (i) $\overline{(\overline{(A\bar{B} + ABC)} + A(B + A\bar{B}))} = 0$

(ii) $A\bar{B}C + \bar{A}BC + ABC = AB + AC$

→ (i) $\overline{(\overline{(A\bar{B} + ABC)} + A(B + A\bar{B}))} = 0$

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

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

$= (A\bar{B} + ABC)(\bar{A} + \bar{A} \cdot \bar{B})$

$= A(\bar{B} + BC)(\bar{A}(1 + \bar{B}))$

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

$= 0$

→ (ii) L.H.S $= A\bar{B}C + \bar{A}BC + ABC$

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

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

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

$= C(A + B)$

$= AC + BC$

$= R.H.S.$

writangel
DATE: PAGE:

Ex.9 Show that the dual of the EX-OR is equal to its complement.

→ $A \oplus B = A\bar{B} + \bar{A}B$

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

According to duality property,

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

According to complement property

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

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

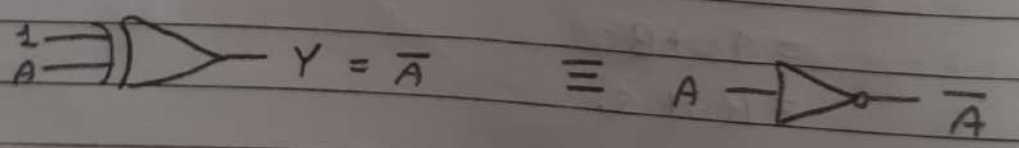
∴ For equation (1) and (2) prove that duality of Ex-OR is equal to its complement.

Ex.10 Design a NOT gate using two input Ex-OR Gate.

→ Ex-OR Gate : $A\bar{B} + \bar{A}B$ Consider $B = 1$

$Y = A.(\bar{1}) + \bar{A}(1) = A \cdot 0 + \bar{A} = \bar{A}$

→ Thus by connecting any one input to logic 1 two input Ex-OR Gate can be used as NOT gate.



Ex.11 $AB + BC + A'C = AB + A'C$

→ $AB + BC + \bar{A}C = AB + BC(A + \bar{A}) + \bar{A}C(B + \bar{B})$
$= AB + ABC + \bar{A}BC + A\bar{B}C + \bar{A}\bar{B}C$
$= AB(1 + C) + \bar{A}C(B + \bar{B} + \bar{B})$
$= AB + \bar{A}C$

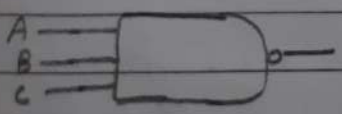
Ex.12 $xyz + \bar{x}z + yz$ Simplify to minimum number

→ (i) $\bar{x}\bar{y}z + \bar{x}yz + x\bar{y}$
$= \bar{x}z(\bar{y} + y) + x\bar{y}$
$= \bar{x}z + x\bar{y}$

(ii) $xy + \bar{x}z + yz$
$= xy + \bar{x}z + yz(x + \bar{x})$
$= xy + \bar{x}z + xyz + \bar{x}yz$
$= xy(1 + z) + \bar{x}z(1 + y)$
$= xy + \bar{x}z$

Ex.13 Simplify expression and draw a diagram.
$F = \overline{ABC} + \overline{A\bar{B}C} + \bar{A}B\bar{C} + A\overline{BC} + A\bar{B}C$

→ $F = \overline{ABC} + \overline{A\bar{B}C} + \bar{A}B\bar{C} + A\overline{BC} + A\overline{\bar{B}C}$
$= \bar{A} + \bar{B} + \bar{C} + \bar{A}C + \bar{B}C + \bar{A}B\bar{C} + A\bar{B} + A\bar{C} + A\bar{B}C$
$= \bar{A}(1 + C + B\bar{C}) + \bar{B}(1 + C + A + AC) + C(1 + A)$
$= \bar{A} + \bar{B} + \bar{C} = \overline{ABC}$



Ex.14 Show that $A \odot B = AB + \overline{A}\overline{B} = \overline{(A \oplus B)} = \overline{(A\overline{B} + \overline{A}B)}$
Also construct the corresponding diagram.

→ $\overline{A \oplus B} = \overline{A\overline{B} + \overline{A}B} = \overline{(A \cdot \overline{B})} \cdot \overline{(\overline{A}\,B)}$

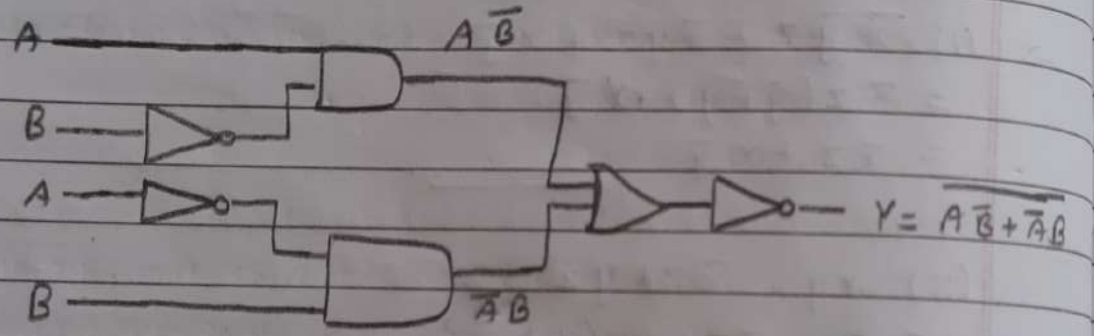
$= (\overline{A} + \overline{\overline{B}})\,(\overline{\overline{A}} + \overline{B})$

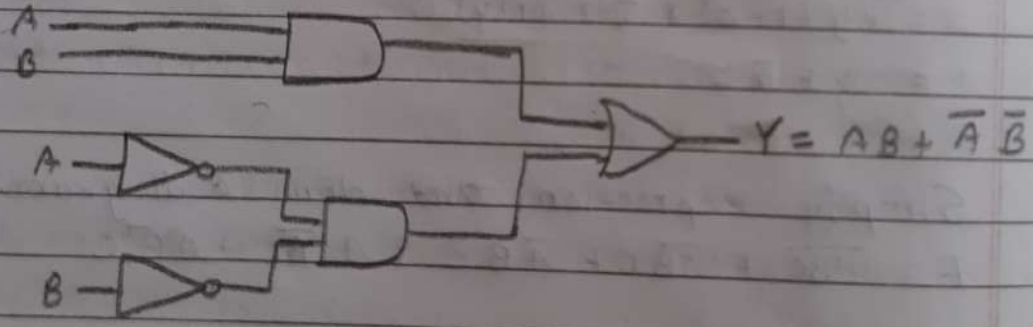
$= (\overline{A} + B)(A + \overline{B})$

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

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

$= A \odot B$





3 Digital Logic Families

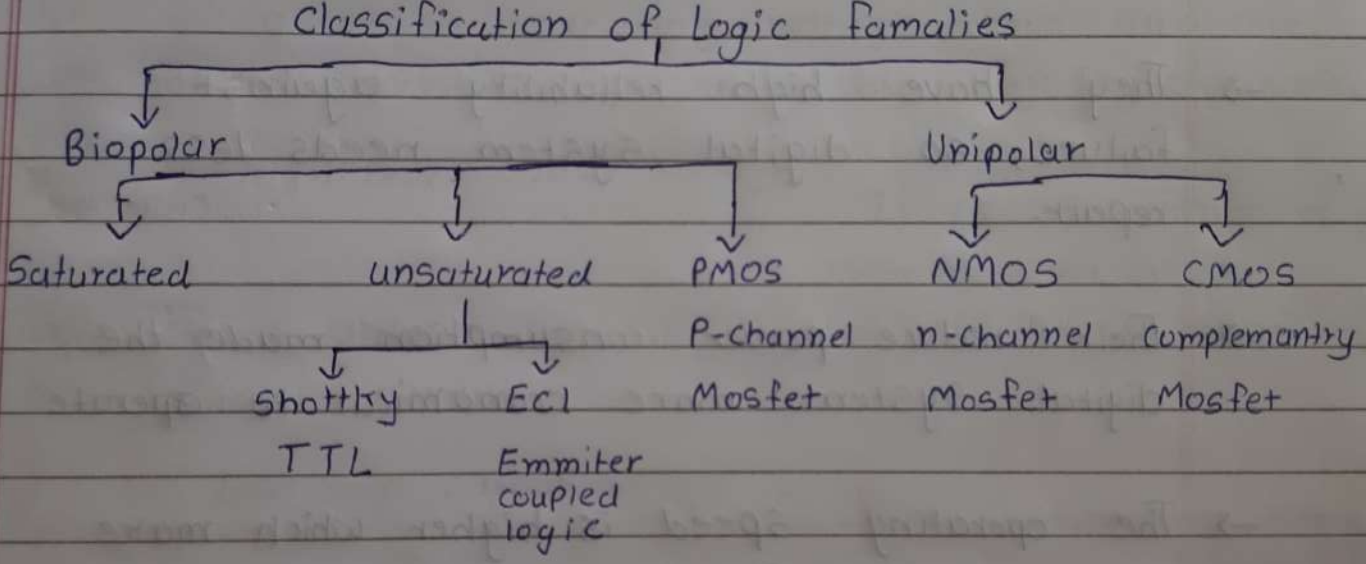
Contents

Chapter : 3

Digital Fundamentals

☆ Introduction

→ A digital logic circuit famaily is a group of compatible device with the same logic levels. and supply voltage.



RTL : Registor Transitor logic
DTL : Diode Transistor logic
DCTL : Direct coupled transistor logic
I^2L : Integrated injection logic
HTL : High Threshold logic
TTL : Transistor transistor logic

* Advantage of digital Integrated Circuits

→ Ics pack more circuitry in small package so that the overall size almost any digital system is reduced.

→ The cost of ICs is very low, which makes them economical to operate.

→ They have high reliability against failure, so digital system needs less repair.

→ Their reduce power consumption made the digital system more economical to operate.

→ The operating speed is higher which make them suitable for high-speed operation.

→ The use of ICs reduce the number of external wiring connection because many of them are internal to the package.

☆ Characteristics of Digital ICs

→ $V_{IH(min)}$ - High-Level Input voltage : It is minimum voltage level required for a logical 1 at an Input. Any voltage below this level will not be acceptable as a High by logic circuit.

→ $V_{IL(max)}$ - Low-level Input voltage : It is the maximum voltage level required for logic 0 at an input. Any voltage above this level will not be acceptable as a low by the logic circuit.

→ $V_{OH(min)}$ - High Level output voltage : It is the minimum voltage level at a logic circuit output in the logical 1 state under defined load condition.

→ $V_{OL(max)}$ Low Level output voltage : It is maximum voltage level at a logic circuit output in the logical 0 state under defined load condition

→ I_{IH} High Level Input current : It is the current that flows into an input when a specified high level voltage is applied to that input.

→ I_{IL} Low Level Input current : It is the current that flows into an input when a specified low level voltage is applied to that input

→ I_{OH} High Level output current : It is the current that flows from an output in the logical 1 state under specified load condition.

→ I_{OL} Low-Level output current : It is the current that flows from an output in the logical 0 state under specified load condition

☆ Transistor - Transistor Logic (TTL)

→ The TTL is so named because of its dependence on transistors alone to perform basic logic operation.

→ It is the most popular logic family. It is also the most widely used bipolar digital IC family.

→ The TTL using transistors operating in saturated mode It is the fastest of the saturated logic families.

→ The basic TTL logic circuit is the NAND gate. Good speed, Low manufacturing cost, wide range of circuits and availability in SSI and MSI are its merits.

→ TTL logic family consists of serval subfamalies or series such as: Standard TTL, High speed TTL, Low power TTL, Schottky TTL, Advanced Shottky TTL, Advance low power schottky TTL and Fast TTL.

→ The difference between various TTL are their in electrical characteristics such as delay time, Power dissipation, switching speed, fan-in, fan-out, noise margi

→ For standart TTL, 0V to 0.8V is treated as a logic 0 and 2V to 5V is treated as logic 0.

→ If a terminal left open in TTL, It is equivalent to connecting it to High.

☆ Schottky TTL

→ The standard TTL, Low power TTL and high speed TTL series operate using saturated switching.

→ When a transistor is saturated excess charge carries will be stored in the base region and they must be removed before the transistor can be turned off.

→ So owing to storage time delay the speed is reduced.

→ The Shottky TTL 74S series reduce this storage time delay by not allowing the transistor to go into full saturation.

→ This is acomplished by using a schottky barrier diode (SBD) between the base and the collector of each transistor.

→ Virtually, all modern TTL devices incorporate this so-called Shottky clamp.

→ The SBD has a forward voltage of 0.25 V.
→ The speed of 74S series is twice of the 74H series.
→ Schottky TTL has more than three times the switching speed of standard TTL at expense of approximantly doubling the power consuption.

☆ Tri-State TTL

→ The third TTL configuration is the tri-state configuration.

→ It utilizes the advantage of high speed of operation of the totem-pole configuration and wire Anding of open-collector configuration.

→ It is called tri-state TTL because it allows three possible output states : High, Low and High Impedence. (Hi-z)

→ In the Hi-z state, both are the transistor in the totem-pole arrangment are turned off, so that the output terminal is a High impedence to ground or Vcc.

→ In fact the output is open or floating terminal that is neither a Low nor a High.

→ In practice output terminal is not an exact open circuit but has a resistance of serval MΩ or more relative to ground and Vcc.

4 Combinational Digital Circuits

Contents

Chapter : 4

Combinational Digital Circuits

☆ Standard Representation for logic Function

→ Boolean expression are constructed by connecting the boolean constant and variables with Boolean operation. These boolean expression are also known as Boolean formulae.

→ We use boolean expressions to describe switching function or Boolean function. For example, if the boolean expression $(A + \overline{B})C$ is used to describe the function f, then boolean function written as,

$f(A, B, C) = (A + \overline{B})C$ or $f = (A + \overline{B})C$

→ Based on the structure of boolean expression it can be categorized in different formula. One such categorization are the normal formula.

→ Consider the four-variable boolean Function :

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

Product terms

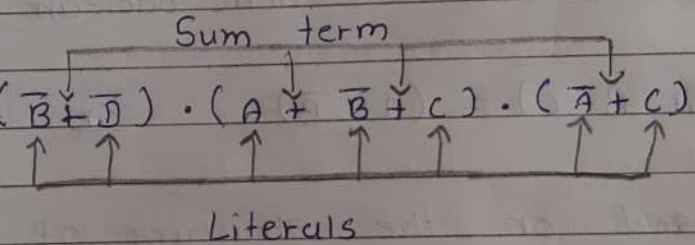
Literals

→ In this boolean function the variables are appeared either in complement or an uncomplement form.

→ Each variable in complement and uncomplement form is called literal. Thus above Boolean function consists of six literals.
They appear in product terms.

→ A product term is defined as either as a literal or a product of literals.
Function consist three product term
$A\bar{B}C$, $AC\bar{D}$.

→ Consider another four-variable Boolean function



→ The above Boolean function consists of seven literals. Here they appeare in the sum term.

→ A sum term is defined as either a literal or a sum litreal. Above function contains three sum term. These literals and terms are arranged in one of the two form:
- Sum OF Product (SOP)
- Product OF Sum (POS)

☆ Sum of Product form

→ The word sum and product are derived from the symbolic representation of the OR and AND function by (+) and (·). But we realize that these are not arithmetic operators in the usual sense. A product term in any group of literals that are ANDed together.

→ A sum term is any group of literal that are ORed together such as A + B + C.
A SOP is a group of product terms ORed together.

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

Sum

Product term

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

→ Each product term consists of one or more literals appearing in either complement or uncomplement form.

→ For Example in the sum of product expression $ABC + A\bar{B}\bar{C}$, the first product term contains literals A, B and C in their uncomplement form The second term B, C contains complement form.

→ The SOP is also known as disjunctive normal form (DNF).

✡ Product of Sum form

→ A product of sum is any groups of sum terms ANDed together.

Ex. $f(A, B, C) = (A + B) \cdot (\overline{B} + C)$

Sum

Product

$f(P, Q, R, S) = (P + Q) \cdot (R + \overline{S}) \cdot (P + \overline{S})$

Sum

→ Each of these product of sums expressions consists two or more sum term (OR) that are ANDed together.

→ Each sum term consits of one or more literals appearing in either complement and uncomplement form.

→ The POS is also known as conjuctive normal form. (CNF)

☆ Standard SOP and POS forms

→ We can realize that in the SOP form all the individual term do not involve all litreals.

→ For example in expression $AB + AB\bar{C}$ the first product term do not contain litreal C.

→ If each term in SOP form contains all the literals then SOP form known as standard or canonical SOP form.

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

→ Each individual term in the standard SOP form is called minterm.

→ One standard SOP expression is as shown above.

→ If each term in POS form contains all the literals then the POS form is known as standard or canonical POS form.

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

→ Each individual term in the standard POS form is called muxterm.

→ One standar POS expression is as shown above.

☆ Convert SOP to Standard SOP

→ Step 1 : Find the missing literal in each product term if any.

Step 2 : AND each product term having missing literals with terms from by ORing the literals and its complement

Step 3 : Expand the term by applying distributive law and recorder the litreals in the product terms.

Step 4 : Reduce the expression by ommiting repeated product term if any.
Because $A + A = A$

Ex. Convert given expression into SSOP form.
$f(A, B, C) = A + ABC$

→ Step 1 : Find missing literals

$f(A, B, C) = A + ABC$
(B) and (C) missing

→ Step 2 : AND Product term with (missing literal + it's complement)

$f(A, B, C) = A \cdot (B + \overline{B})(C + \overline{C}) + ABC$

→ Step 3: Expand the term and recorder literals

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

→ Step 4: Ommit repeated product term

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

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

☆ Convert POS to standard POS

→ Step 1: Find Missing literals in each sum term if any.

Step 2: OR each sum term having missing literal with term having form by ANDing the literals and it's complement

Step 3: Expand the term by applying distributive law and recorder literals in the sum terms

Step 4: Reduce the expression by ommiting repeated sum term if any
Because $A \cdot A = A$

Ex. Convert the given expression in SPOS form.
$f(A,B,C) = (A+B)(B+C)(A+C)$

→ Step 1: Find the missing literals in each sum term

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

(B+C) → A is missing
(A+C) → B is missing
(A+B) → C is missing

→ Step 2: OR sum term (missing literal · complement)

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

→ Step 3: Expand the term and recorder literals.

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

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

→ Step 4: Ommit repeated sum term

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

☆ Karnaugh-Maps Representation (K-Map)

→ During the process of simplification of boolean expression we have to predict each successive step.

→ We can never be absoulatly certain that an expression simplified by bodean algebra alone.

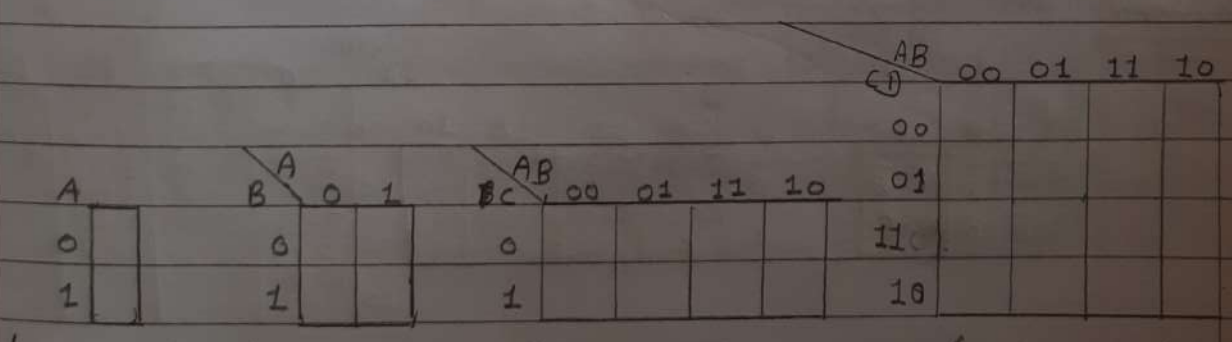
→ On the other hand the map gives us systematic approach for simplifying a boolean expression.

→ The map method was proposed by Veitch and modified by karnaugh, hence it is known as Veitch diagram or the karnaugh map.

→ It contains boxes called cells.

→ Each of the sell represent one of the 2^n possible product that can be formed from n variables.

2-Variable = 2^2 = 4 cell
3-Variable = 2^3 = 8 cell
4-Variable = 2^4 = 16 cell



(1-Variable) (2-Variable) (3-Variable) (4-Variable)

→ Product term are assign to the cell of a k-Map by labeling each row and each column of the map with a variable with it's complement and with a combination of variable and complements.

→ The Product term corresponding to given cell is then the product of all variable in the row and column where the cell is located.

→ When we move from one cell to the next along any row or from one cell to the next along any column one and ~~one~~ only one variable in the product term change to a complemented or to an uncomplemeted form.

A \ B	$\bar{B}$ 0	B 1
$\bar{A}$ 0	0	2
A 1	1	3

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

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

Maxtern

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

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

CD \ AB	00	01	11	10
00	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$	$A+\bar{B}+\bar{C}+\bar{D}$	$A+B+\bar{C}+\bar{D}$	$A+\bar{B}+\bar{C}+\bar{D}$
01	$\bar{A}+\bar{B}+\bar{C}+D$	$A+\bar{B}+\bar{C}+D$	$A+B+\bar{C}+D$	$A+\bar{B}+\bar{C}+D$
11	$\bar{A}+\bar{B}+C+D$	$A+\bar{B}+C+D$	$A+B+C+D$	$A+\bar{B}+C+D$
10	$\bar{A}+\bar{B}+C+\bar{D}$	$A+\bar{B}+C+\bar{D}$	$A+B+C+\bar{D}$	$A+\bar{B}+C+\bar{D}$

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

AB / CD	00	01	11	10
0	$AB\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\overline{C}$
1	$\overline{A}B\overline{C}$	$\overline{A}BC$	ABC	$A\overline{B}C$

AB / CD	00	01	11	10
00	$\overline{A}\overline{B}C\overline{D}$	$\overline{A}\overline{B}C\overline{D}$	$AB\overline{C}\overline{D}$	$A\overline{B}C\overline{D}$
01	$\overline{A}\overline{B}C\overline{D}$	$\overline{A}\overline{B}\overline{C}D$	$AB\overline{C}D$	$A\overline{B}\overline{C}D$
11	$\overline{A}\overline{B}CD$	$\overline{A}\overline{B}CD$	$ABCD$	$\overline{B}ACD$
10	$\overline{A}\overline{B}C\overline{D}$	$\overline{A}\overline{B}C\overline{D}$	$ABC\overline{D}$	$A\overline{B}C\overline{D}$

→ Insted of writing actual product term corresponding shorthand minterm notation are written in the cell rows and colums are maked with gray code insted of variables

A / B	0	1
0	m_0	m_2
1	m_1	m_3

AB / C	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

AB / CD	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

→ Insted of writing actual product term corresponding shorthand maxtern notation are written in the cell rows and colums are marked with gray cod insted of variables

A / B	0	1
0	M_0	M_2
1	M_1	M_3

AB / C	00	01	11	10
0	M_0	M_2	M_6	M_4
1	M_1	M_3	M_7	M_5

AB / CD	00	01	11	10
00	M_0	M_4	M_{12}	M_8
01	M_1	M_5	M_{13}	M_9
11	M_3	M_7	M_{15}	M_{11}
10	M_2	M_6	M_{14}	M_{10}

* Representing SSOP on K-Map

→ A Boolean expression in the sum of products from can be plotted on the K-Map by placing a 1 in each cell corresponding to a term (minterm) in the sum of products expression. Remaining cells are filled with zero.

Ex. $Y = AB\bar{C} + ABC + \bar{A}\bar{B}C$ Plot boolean expression on K-Map.

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

C \ AB	00	01	11	10
0	0	2	1 6	4
1	1 1	3	1 7	5

Plot boolean expression on the K-Map
$Y = \bar{A}B\bar{C}\bar{D} + A\bar{B}C\bar{D} + A\bar{B}CD + AB\bar{C}D$

CD \ AB	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}B\bar{C}\bar{D}$	$AB\bar{C}\bar{D}$	$A\bar{B}\bar{C}\bar{D}$
01	$\bar{A}\bar{B}C\bar{D}$	$\bar{A}BC\bar{D}$	$ABC\bar{D}$	$A\bar{B}C\bar{D}$
11	$\bar{A}\bar{B}CD$	$\bar{A}BCD$	$ABCD$	$A\bar{B}CD$
10	$\bar{A}\bar{B}C\bar{D}$	$\bar{A}BC\bar{D}$	$ABC\bar{D}$	$A\bar{B}C\bar{D}$

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

☆ Representing SPOS on K-Map

→ A boolean expression in the product of sums can be plotted on the K-Map by placing 0 in each cell corresponding to a term (maxterm) in the expression. Remaining cells are filled with 1.

Ex. Plot boolean expression on K-Map

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

→

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

C \ AB	00	01	11	10
0	1 $_0$	1 $_2$	0 $_6$	0 $_4$
1	0 $_1$	1 $_3$	1 $_7$	0 $_5$

Ex. Plot boolean expression on K-Map

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

→

CD \ AB	00	01	11	10
00	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$	$\bar{A}+B+\bar{C}+\bar{D}$	$A+B+\bar{C}+\bar{D}$	$A+\bar{B}+\bar{C}+\bar{D}$
01	$\bar{A}+\bar{B}+\bar{C}+D$	$\bar{A}+B+\bar{C}+D$	$A+B+\bar{C}+D$	$A+\bar{B}+\bar{C}+D$
11	$\bar{A}+\bar{B}+C+D$	$\bar{A}+B+C+D$	$A+B+C+D$	$A+\bar{B}+C+D$
10	$\bar{A}+\bar{B}+C+\bar{D}$	$\bar{A}+B+C+\bar{D}$	$A+B+C+\bar{D}$	$A+\bar{B}+C+\bar{D}$

CD \ AB	00	01	11	10
00	1 $_0$	1 $_4$	0 $_{12}$	1 $_8$
01	1 $_1$	1 $_5$	1 $_{13}$	0 $_9$
11	1 $_3$	1 $_7$	1 $_{15}$	1 $_{11}$
10	0 $_2$	1 $_6$	0 $_{14}$	1 $_{10}$

☆ K-Map Rules for grouping cells.

→ Group is not contain zero and cell contains '1' must be grouped

→ We can group 1, 2, 4, 8, ..., 2^n cells.

→ Each group should be as large as possible.

→ Group may overlap.

→ Opposite grouping and corner grouping is allowed.

→ There should be as few group as possible.

Ex. Using K-Map find boolean function and it's complement for: $f(A,B,C,D) = \Sigma(1,2,3,4,6,8,9,10,11,12,14)$

→

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

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

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

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

Ex. Obtain the simplified expression in sum of product for the following boolean function.

(a) $f = \Sigma(0, 1, 4, 5, 10, 11, 12, 14)$

(b) $f = \Sigma(11, 12, 13, 14, 15)$

→ (a) $F = \Sigma(0, 1, 4, 5, 10, 11, 12, 14)$

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

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

→ (b) $f = \Sigma(11, 12, 13, 14, 15)$

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

$Y = AB + AC\bar{D}$

Ex. Simplify the boolean function with K-Map

$F = \bar{A}BC + \bar{B}C\bar{D} + \bar{A}BC\bar{D} + A\bar{B}\bar{C}$

→

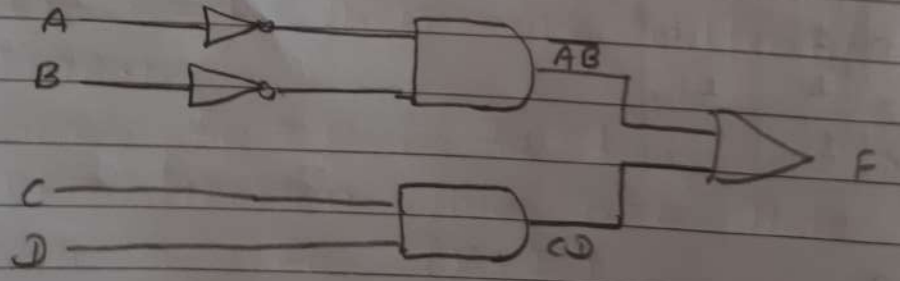
CD \ AB	00	01	11	10
00	1 (0)	1 (4)	12	1 (8)
01	1	5	13	1 (9)
11	3	7	15	11
10	1 (2)	1 (6)	14	1 (10)

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

Ex. Implement the Function $F = \Sigma(1,3,7,11,15)$ with don't care condition $d = \Sigma(0,2,5)$

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

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



Ex. Obtain the simplified expression in sum of product using K-Map.

$F = \bar{x}z + \bar{w}x\bar{y} + w(\bar{x}y + x\bar{y})$

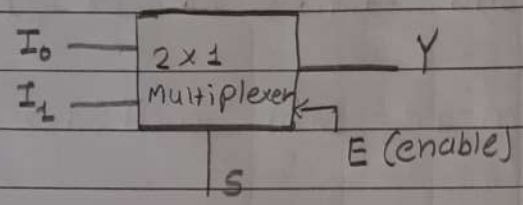
→ $F = \bar{x}z + \bar{w}x\bar{y} + w\bar{x}y + x\bar{y}w$

yz \ wx	00	01	11	10
00		1	1	
01	1	1	1	1
11	1			1
10				1

Multiplexer

* 2x1 multiplexer.

-> Block Diagram:-



-> $[2^n \times 1]$ mux

$n = 1$, 1 Selection line

2 Input , 1 Output

-> In above figure of [2x1 mux] there are two input I_0 and I_0 and one output Y.

-> There is a one selection line.

-> If $E = 0$ there is no matter what is input output will be 0. $(Y = 0)$

-> If $E = 1$ there are different Input and Output.

Like, $I_0 = 0$, $I_1 = 1$

-> If Selection line $S = 0$ there is a output will be I_0. $(Y = I_0)$

-> If selection line $S = 1$ there will be output I_1 $(Y = I_1)$

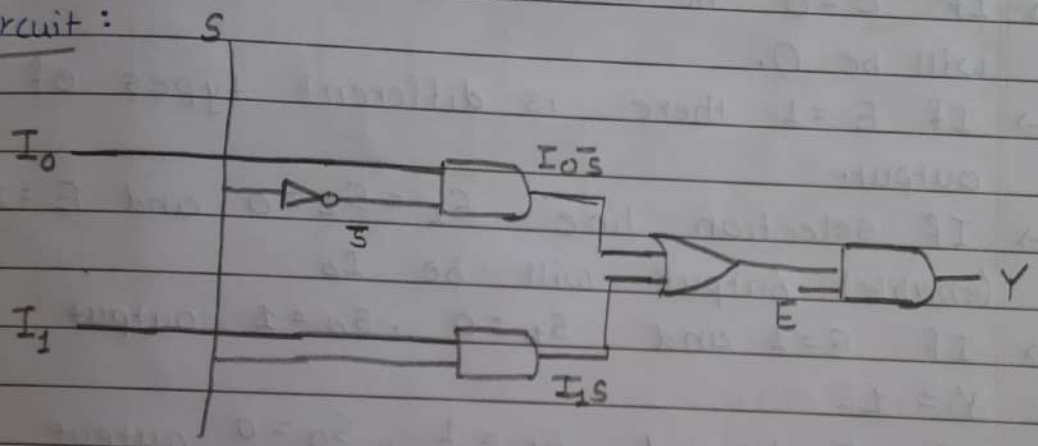
-> Truth table :

E	S	Y
0	x	x
1	0	I_0
1	1	I_1

-> Expression = $I_0\bar{S}E + I_1SE$

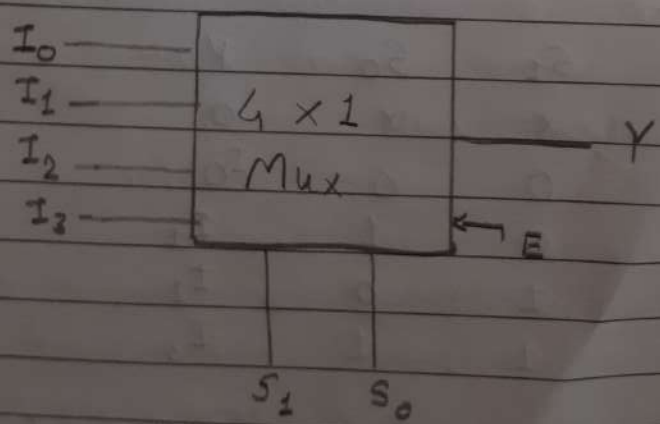
$= E(I_0\bar{S} + I_1S)$

-> Circuit :



* 4 x 1 Multiplexer

-> Block Diagram :



→ $[2^n \times 1]$ mux.

→ $[4 \times 1]$ mux

$n = 2$, Selection line = 2

4 Input , 1 Output

→ In above figure $[4 \times 1]$ mux there are 4 inputs I_0, I_1, I_2, I_3 and one output Y.

→ There is a 2 selection line S_1 and S_0

→ There is a E.

→ If $E = 0$ no matter what inputs output will be 0.

→ If $E = 1$ there is different types of output.

→ If selection line $S_1 = S_2 = 0$ and $E = 1$ (enable) output will be I_0

→ If $E = 1$ and $S_1 = 0$, $S_0 = 1$ output $Y = 1$.

→ If $E = 1$ and $S_1 = 1$, $S_0 = 0$ output $Y = 2$.

→ If $E = 1$ and $S_1 = 1$, $S_2 = 1$ output $Y = 3$.

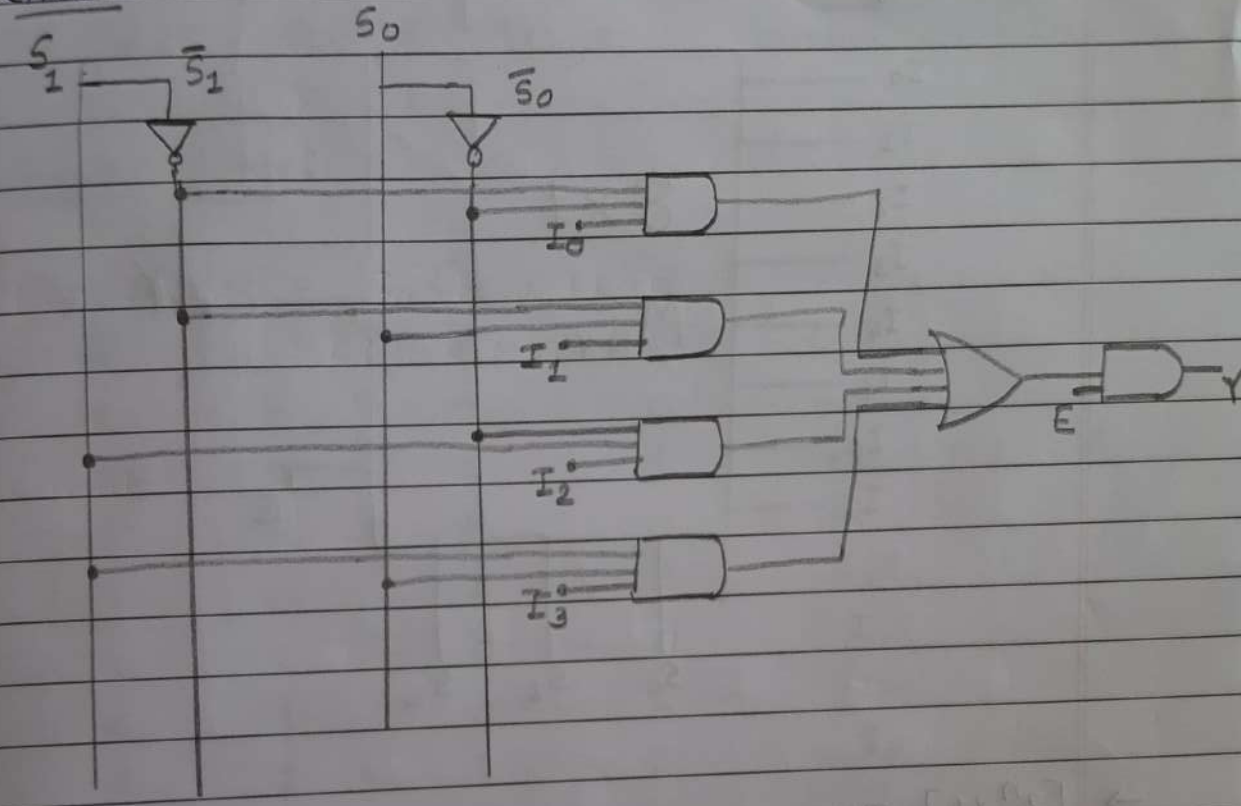
→ Truth table :

E	S_1	S_0	Y
0	x	x	0
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3

→ Boolean Expression = $I_0 \bar{S_1} \bar{S_0} E + I_1 S_0 \bar{S_1} E + I_2 S_1 \bar{S_0} E + I_3 S_1 S_0 E$

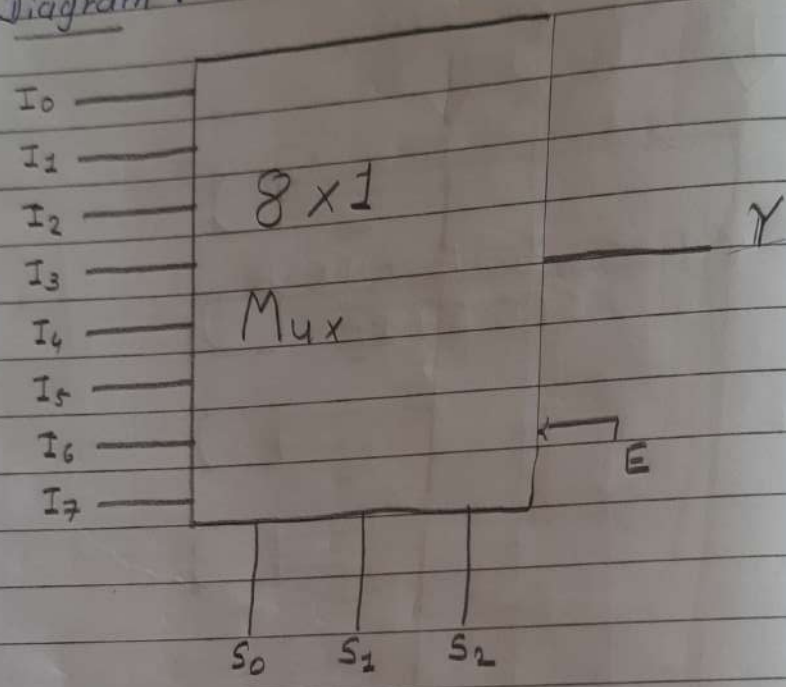
Expression = $E[I_0 \bar{S}_1 \bar{S}_0 + I_1 \bar{S}_1 S_0 + I_2 S_1 \bar{S}_0 + I_3 S_1 S_0]$

→ Circuit :



* 8 x 1 Multiplexer

→ Block Diagram :



→ $[2^n \times 1]$ mux
$[8 \times 1]$ mux
$n = 3$, Selection line = 3
8 Input, 1 Output

→ In above $[8 \times 1]$ mux figure there is a 8 input and 1 output Y.
→ There are 3 selection line.
→ If E = 0 there is no matter what is Input Output will be 0.
→ If E = 1 there is a different types of output depends on Input.
→ If E = 1, $S_0 = S_1 = S_2 = 0$ output $Y = I_0$.
→ If E = 1, $S_0 = S_1 = 0$, $S_2 = 1$ output $Y = I_1$.
→ If E = 1, $S_0 = 0$, $S_1 = 1$, $S_2 = 0$ output $Y = I_2$.

→ If, $E = 1$, $S_0 = 0$, $S_1 = 1$, $S_2 = 1$ output $Y = I_3$.

→ If $E = 1$, $S_0 = 1$, $S_1 = S_2 = 0$ output $Y = I_4$.

→ If $E = 1$, $S_0 = 1$, $S_1 = 0$, $S_2 = 1$ output $Y = I_5$.

→ If $E = 1$, $S_0 = 1$, $S_1 = 1$, $S_2 = 0$ output $Y = I_6$.

→ If $E = 1$, $S_0 = S_1 = S_2 = 1$ output $Y = I_7$.

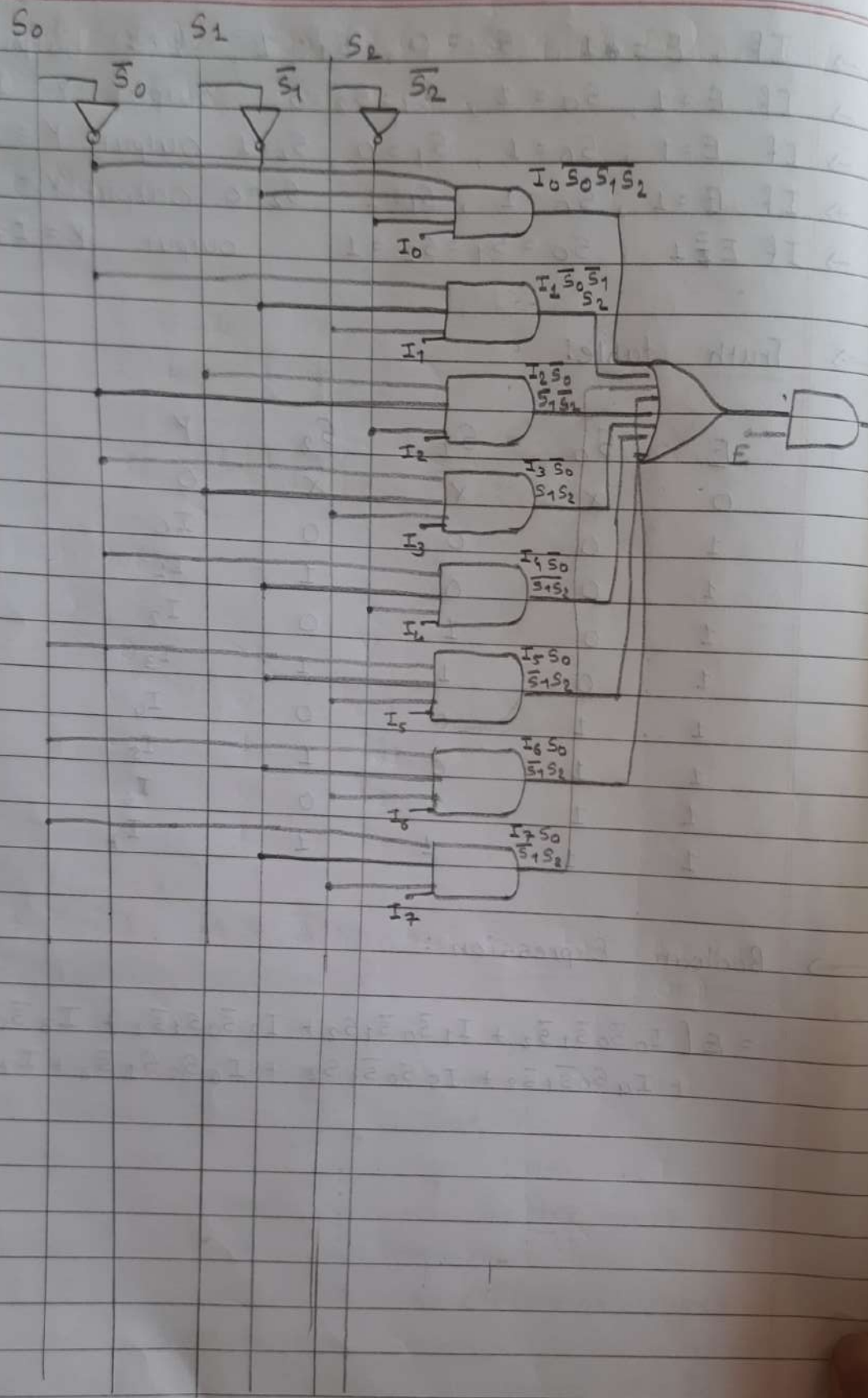
→ Truth table:

E	S_0	S_1	S_2	Y
0	x	x	x	0
1	0	0	0	I_0
1	0	0	1	I_1
1	0	1	0	I_2
1	0	1	1	I_3
1	1	0	0	I_4
1	1	0	1	I_5
1	1	1	0	I_6
1	1	1	1	I_7

→ Boolean Expression :

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

→ Circuit :


S0
S1
S2
S̄0
S̄1
S̄2
I0 S̄0 S̄1 S̄2
I0
I1 S̄0 S̄1 S2
I1
I2 S̄0 S1 S̄2
I2
I3 S0 S̄1 S̄2
I3
I4 S0 S̄1 S2
I4
I5 S0 S1 S̄2
I5
I6 S0 S1 S̄2
I6
I7 S0 S1 S2
I7
F

Ex. $F(A, B, C, D) = \Sigma m(0, 2, 6, 9, 11, 13)$ make it by 4×1 mux.

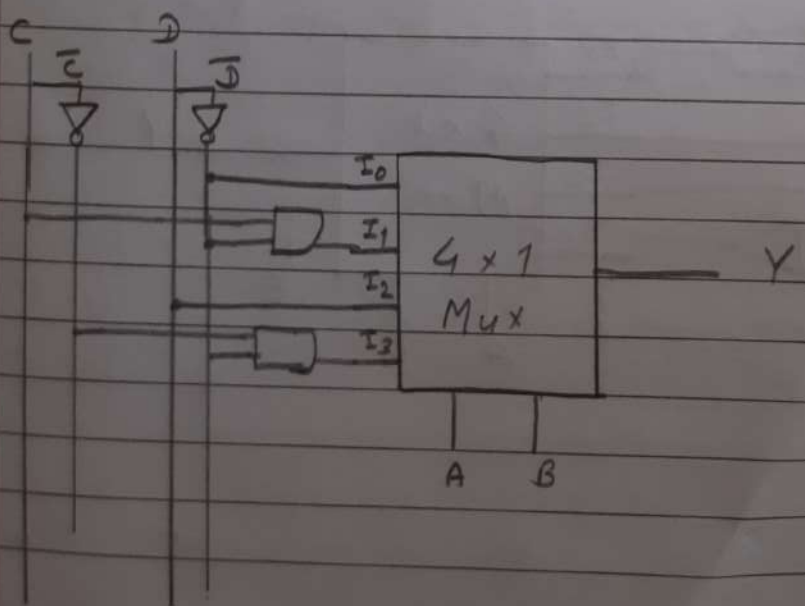
→ K-Map :

CD \ AB	00	01	11	10
00	1 0	4	12	8
01	1	5	1_{13}	1_9
11	3	7	15	1_{11}
10	1_2	1_6	14	10

→ Truth table :

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

→ $I_0 = \overline{D}$, $I_1 = C\overline{D}$, $I_2 = D$, $I_3 = \overline{C}D$



⊕ = Ex-OR

Ex. $F(A,B,C,D) = \Sigma m\ (1,4,6,8,9,13,14)$
make it by 4x1 mux

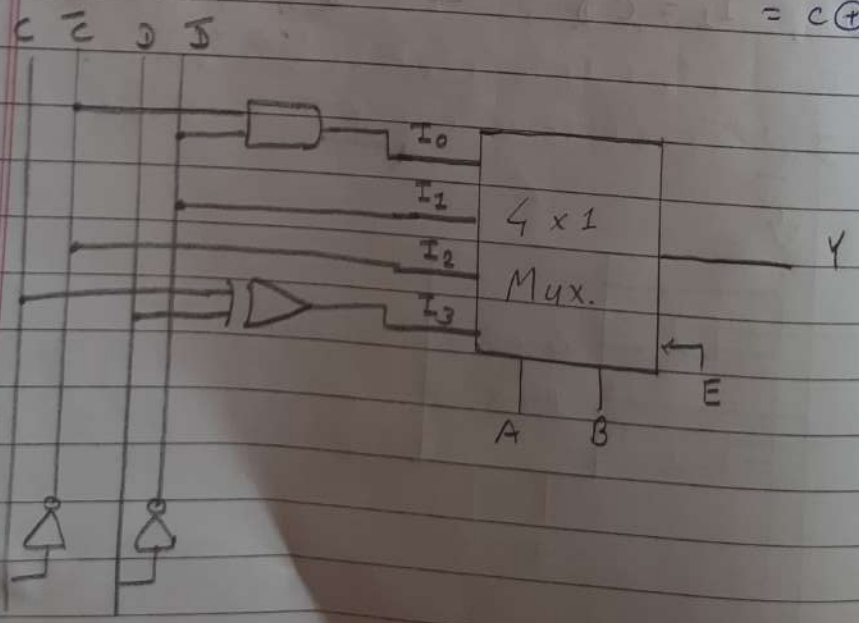
→ K-Map :

CD \ AB	00	01	11	10
00	0	1_4	12	1_8
01	1_1	5	1_{13}	1_9
11	3	7	15	11
10	2	1_6	1_{14}	10

→ Truth-table :

S_0	S_1	Y
0	0	I_0
0	1	I_{01}
1	0	I_2
1	1	I_3

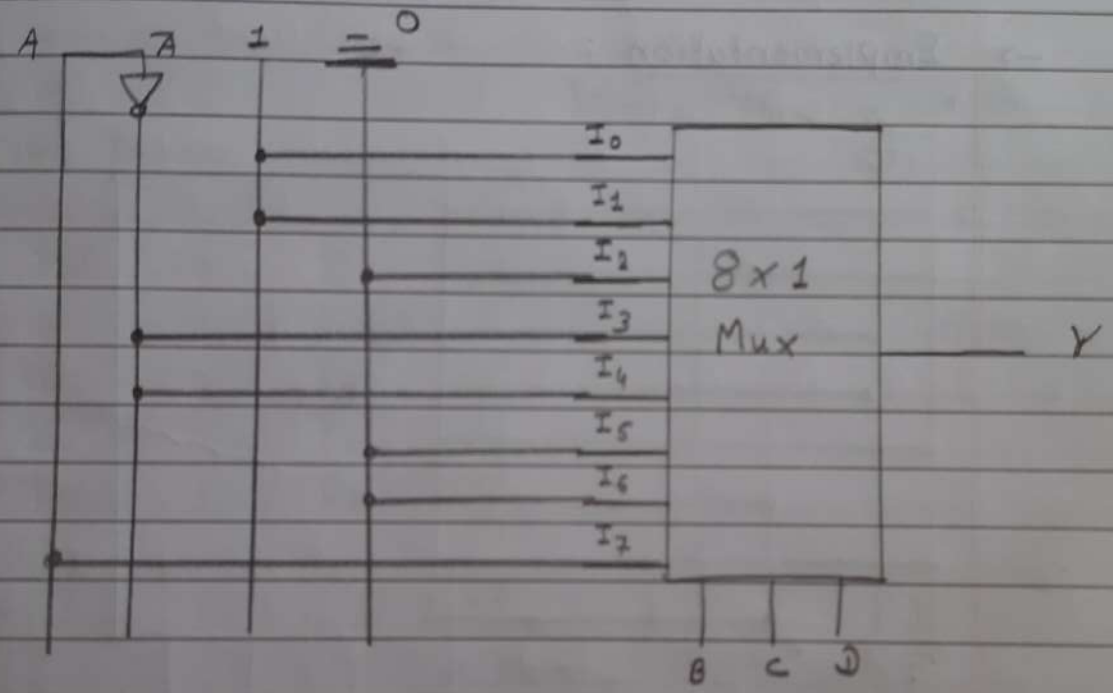
→ $I_0 = \bar{C}D$, $I_1 = \bar{D}$, $I_2 = \bar{C}$, $I_3 = \bar{C}D + D\bar{C}$
$= C \oplus D$



Ex. Design 8:1 multiplexer using four variable
$F(A, B, C, D) = (0, 1, 2, 3, 4, 8, 9, 15)$

→ Implementation Table :

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A

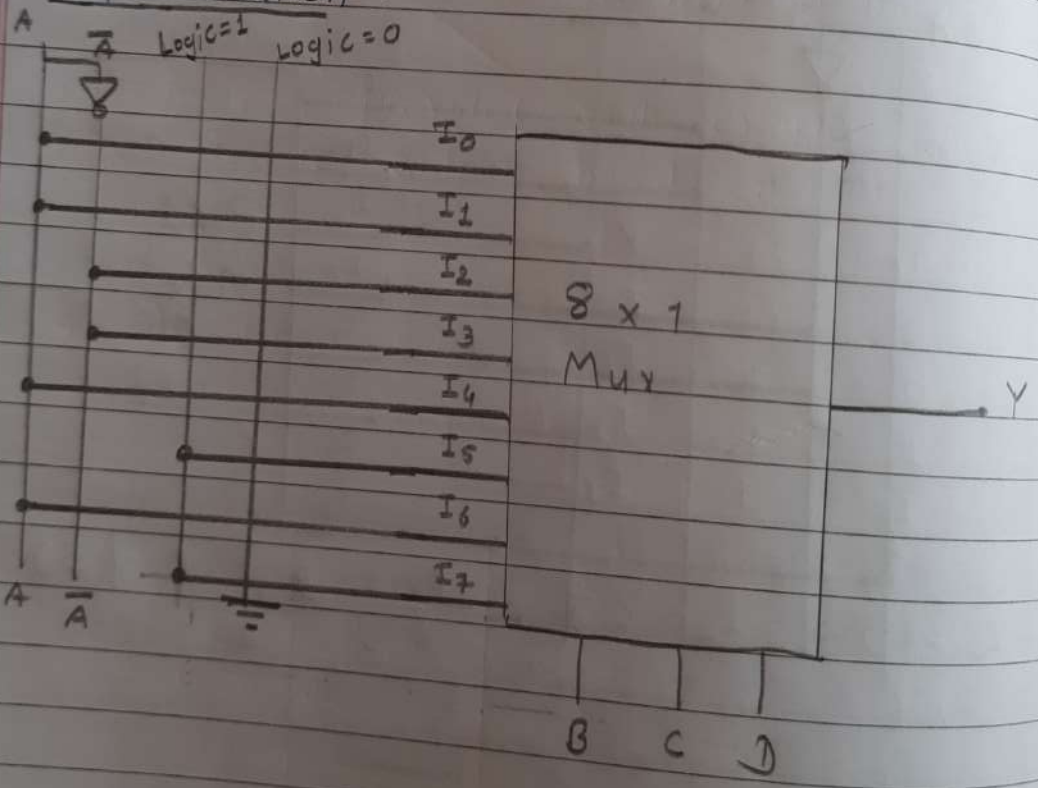


Ex. $F(A, B, C, D) = \Sigma_m (2, 3, 5, 7, 8, 9, 12, 13, 14, 15)$
make it by 8:1 mux.

→ Implementation Table :

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	A	A	$\bar{A}$	$\bar{A}$	A	1	A	1

→ Implementation :



* Application of Multiplexer

-> They are used as a data selector to select one out of many data Input.

-> They can be used to Implement combinational logic circuit.

-> They are used in time multiplexing system.

-> They are used frequency multiplexing system.

-> They are used in A/D and D/A converter.

-> They are used in data acquistion system.

* Multiplexer ICs

IC number	Function
74150	16:1 mux
74151	8:1 mux
74153	Dual 4:1 mux
74157	Quad 2-Input mux.

☆ Decoder

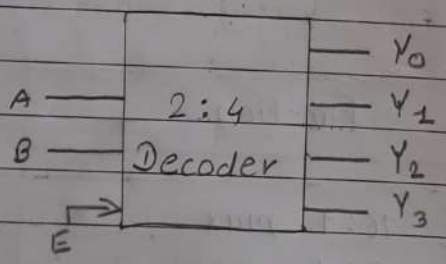
→ A decoder is a multiple-input, multiple-output logic circuit which converts coded inputs into coded output, where the input and output codes are different.

→ The encoded information is presented as n input producing 2^n possible output. The 2^n input values are from 0 through $2^n - 1$

→ A decoder is provided with enable input to activate decode output based on data Inputs.

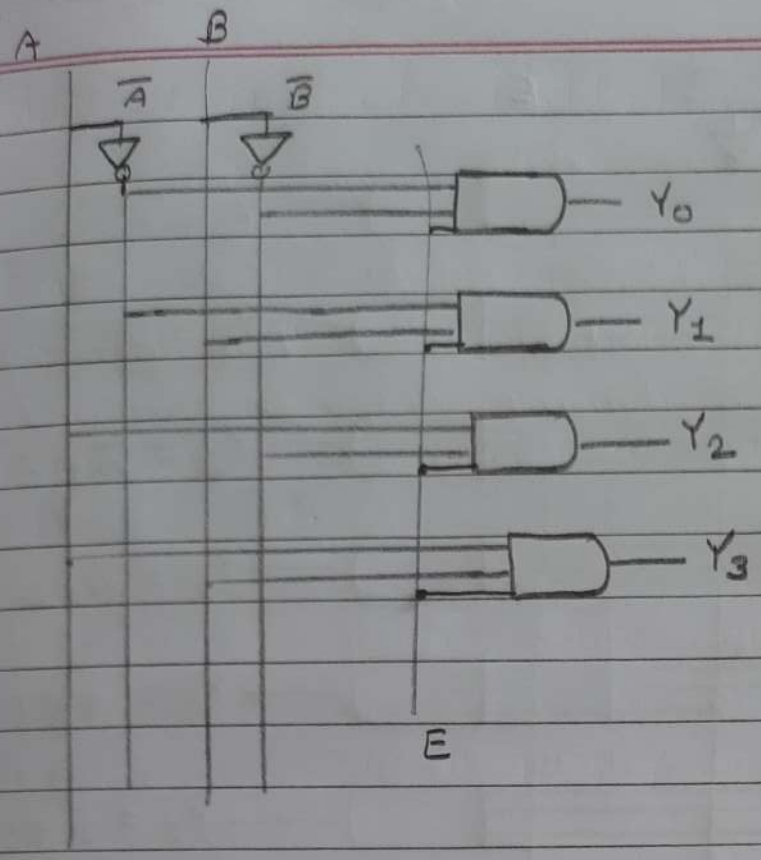
* 2:4 Decoder

→ Diagram:



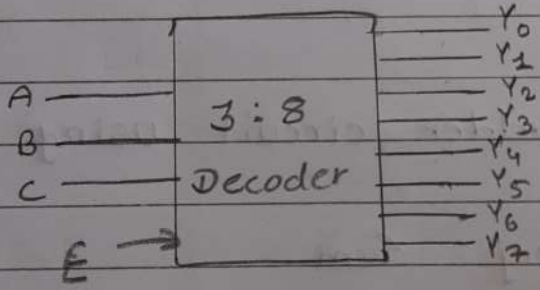
→ Truth Table:

E	A	B	Y_0	Y_1	Y_2	Y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

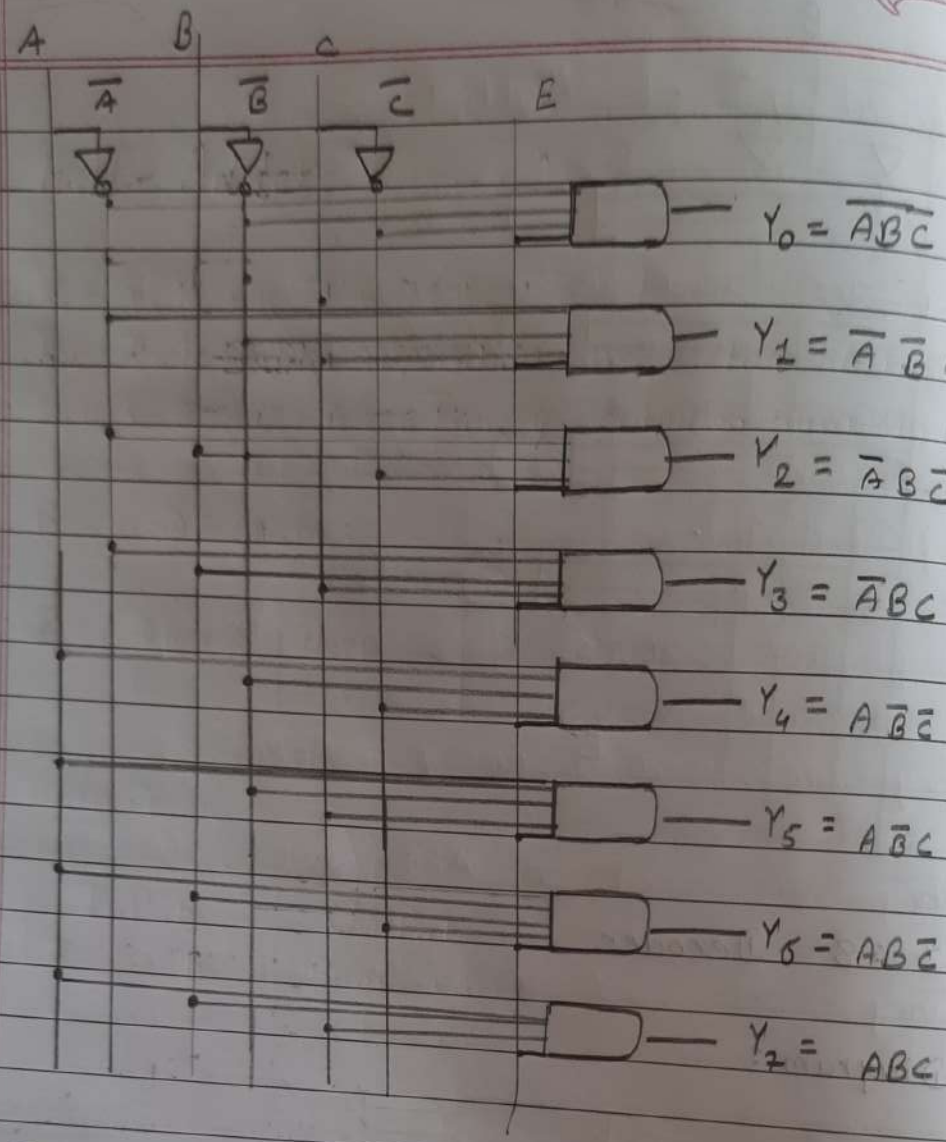


* 3 : 8 Decoder

→ Diagram :



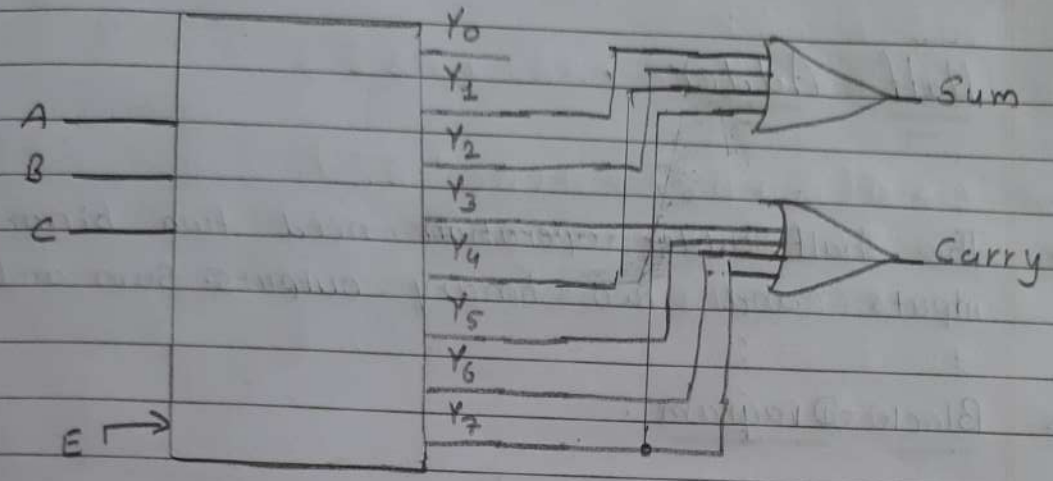
E	A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1



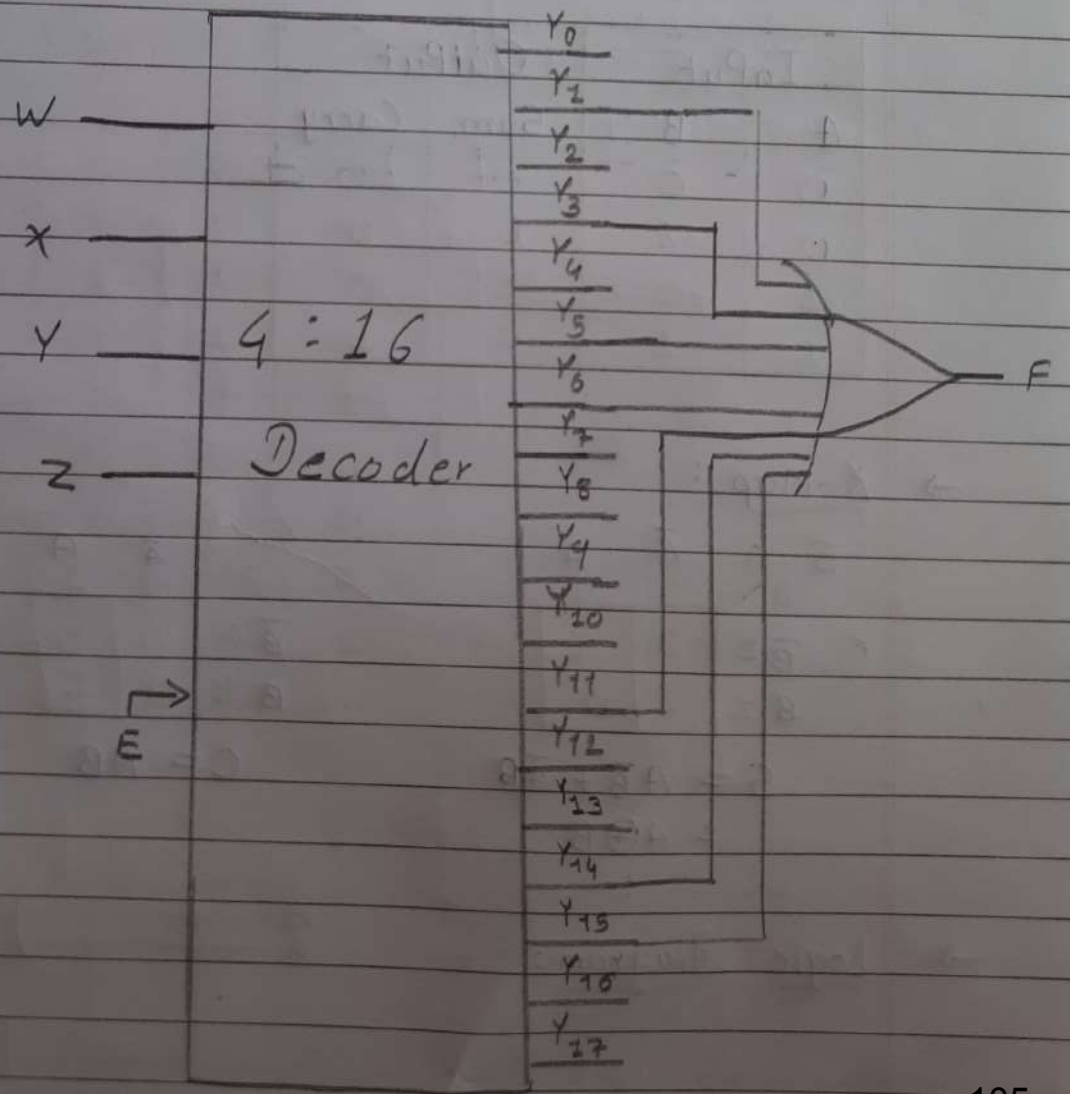
Ex. Design a Full adder circuit using 3:8 Decoder.

→

A	B	C	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



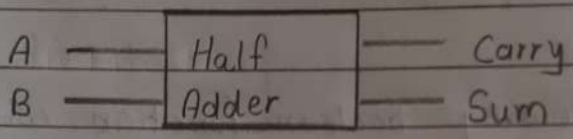
Ex. Implement following boolean function with a decoder
$f(w, x, y, z) = \Sigma(1, 3, 5, 6, 11, 14, 15)$



☆ Half-Adder

→ The half-adder operation need two binary inputs and two binary outputs: Sum and carry.

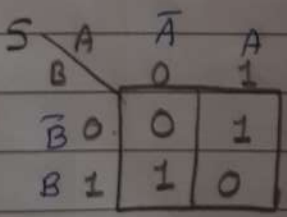
→ Block-Diagram:



→ Truth-Table:

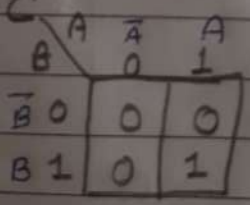
InPut		OutPut	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

→ K-Map:



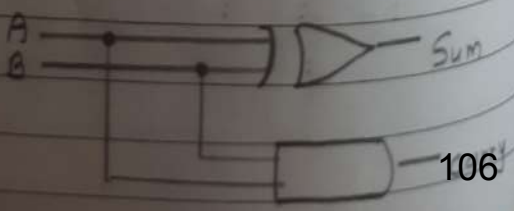
$S = A\bar{B} + \bar{A}B$

$= A \oplus B$



$C = AB$

→ Logic Diagram:

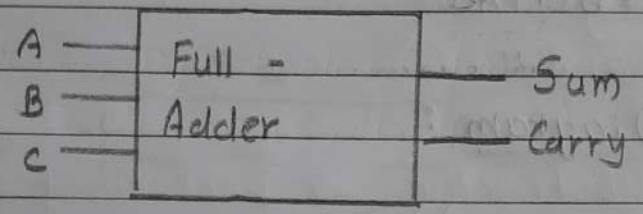


☆ Full adder

→ A full adder is a combinational circuit that forms the arithmetic sum of three input bits.

→ Full adder contains 3 input and 2 output: Sum and carry

→ Block-Diagram:



→ Truth-Table:

A	B	C	S	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

→ K-Map:

S

C \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

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

Carry

C \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

AB, AC, CB

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

$C_o = AB + BC + AC$

→ $S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$

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

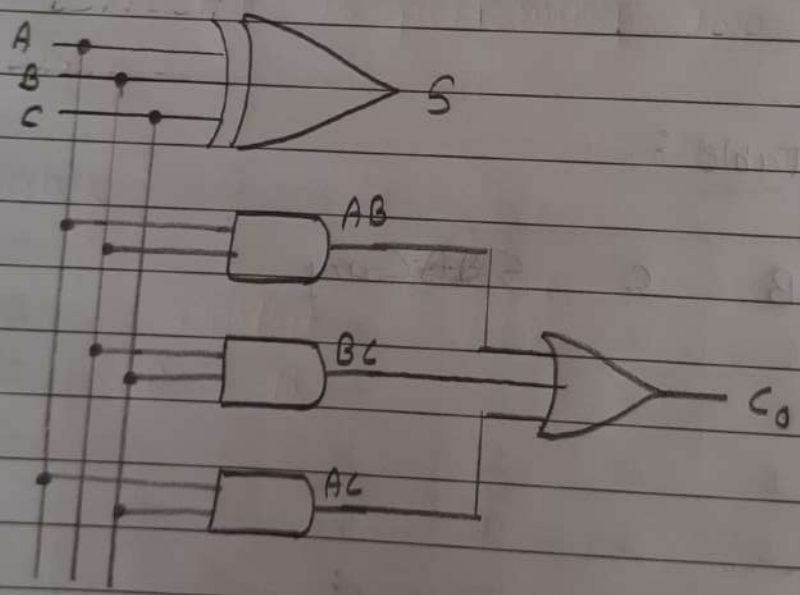
$= \bar{A}(B \oplus C) + A(B \oplus C)$

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

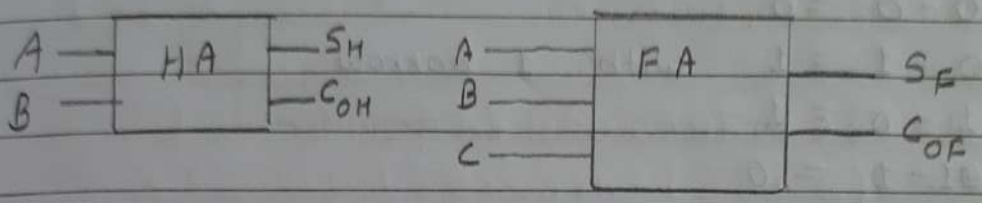
$= A \oplus B \oplus C$

→ $C = AB + BC + AC$

⟶ Logic Diagram :



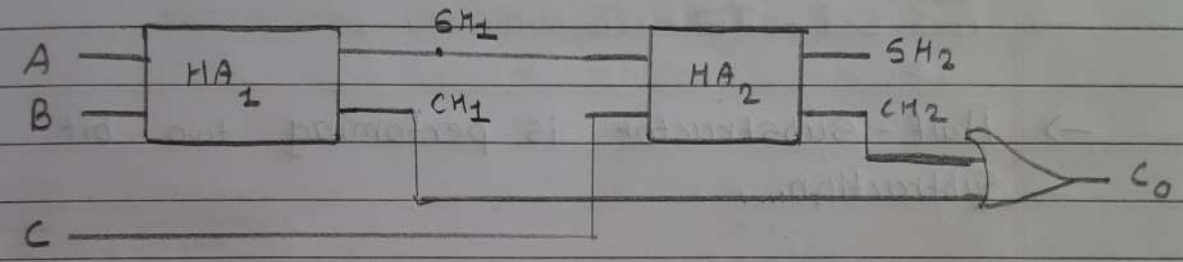
☆ Full adder circuit using half adder



→ $S_H = A \oplus B$
$C_{OH} = A \cdot B$

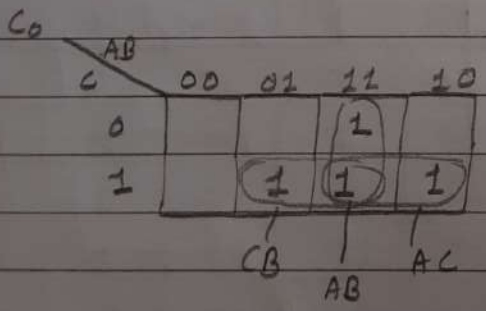
$S_F = A \oplus B \oplus C$
$C_{OF} = AB + BC + CA$

→



→ $SH_1 = A \oplus B = A\bar{B} + \bar{A}B$
$CH_1 = AB$

→ $SH_2 = S_{H1} \oplus C = A \oplus B \oplus C$
$CH_2 = S_{H1} \cdot C$
$= (A\bar{B} + \bar{A}B)C$
$= A\bar{B}C + \bar{A}BC$



→ $C_0 = CH_2 + CH_1$
$= AB + A\bar{B}C + \bar{A}BC$
$= AB + BC + AC$

☆ Substractor

→ 0 - 0 = 0
0 - 1 = 1 with 1 borrow
1 - 0 = 1
1 - 1 = 0

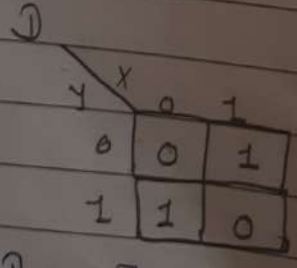
☆ Half - Substractor

→ Half - substractor is perfoming two bit subtraction.

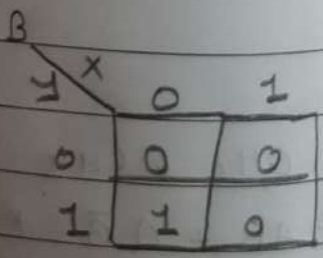
→ Truth - table :

Input		OutPut	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

→ K - Map :

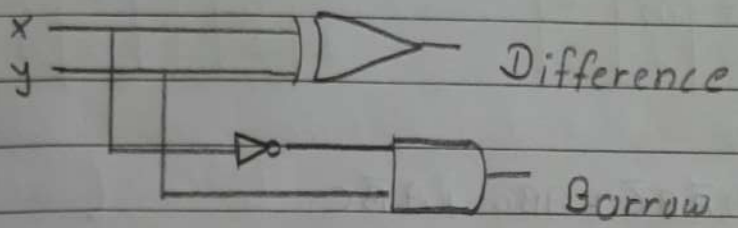


$D = x\bar{y} + \bar{x}y$
$= x \oplus y$



$B = \bar{x}y$

→ Logic - Diagram :



→ Notice : This is x-y Performance above.
If we want to perform y - x then Borrower will be $B = x\bar{y}$

☆ Full - Substractor

→ It perfoms 3-bit substraction.
(A - B - C) or A - B (B + C)

→ Truth-table

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

→ K-Map :

D

C \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

B_0

	00	01	11	10
0	0	1	0	0
1	1	1	1	0

→ $D = \bar{A}B\bar{C} + \bar{A}B\bar{C} + ABC + A\bar{B}C$

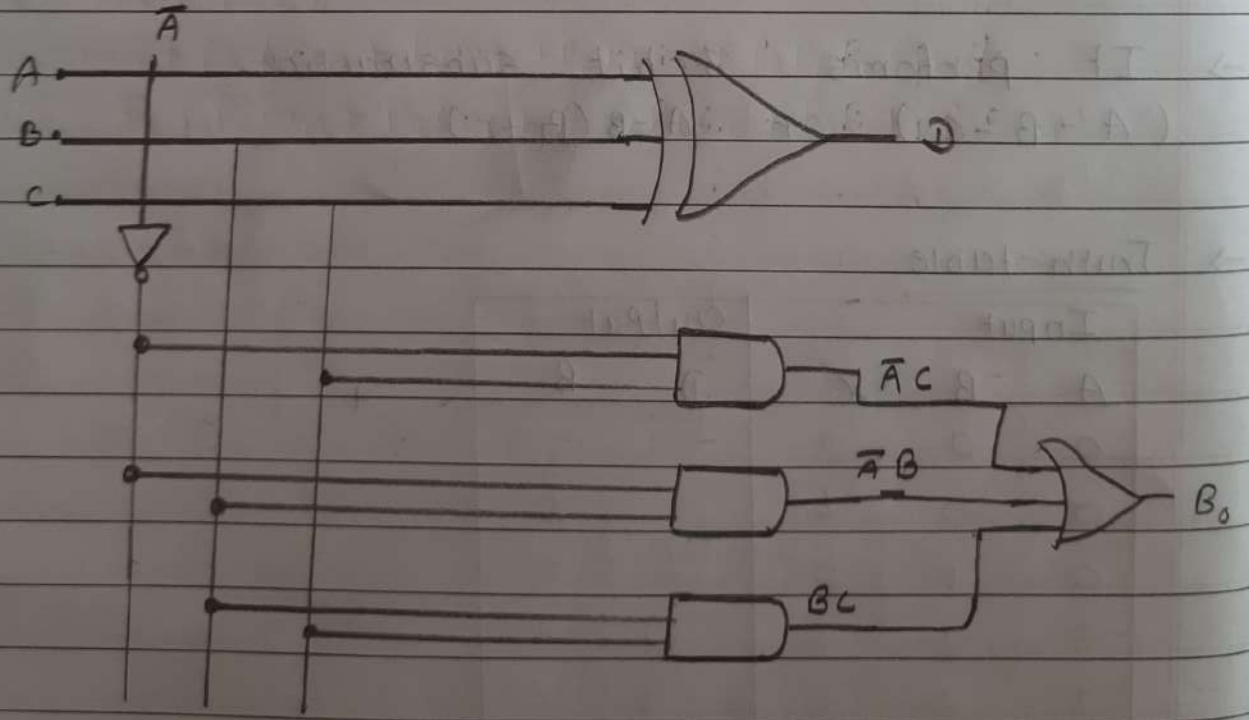
$= \bar{A}(\bar{B}C + B\bar{C}) + A(BC + \bar{B}\bar{C})$

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

$= A \oplus B \oplus C$

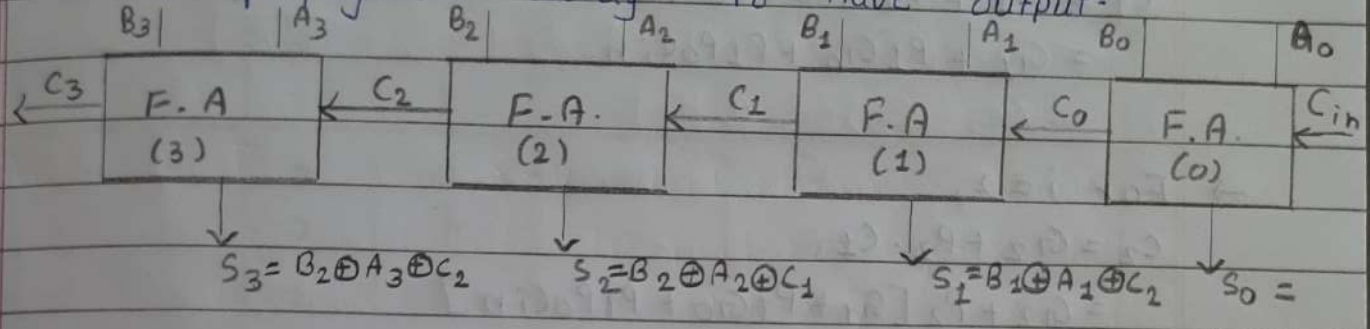
→ $B = C\bar{A} + \bar{A}B + BC$

→ Logic-Circuit :



☆ Carry Look Ahead Adder

→ CLA Adder is faster in terms of operation speed.
→ By full adder when we have parallel adder it takes propogation delay to have output.



→ Truth-table :

B_1	A_1	C_0	C_1
0	0	0	0
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	1

Sum > 1 = Carry

→ $C_1 = B_1A_1C_0 + B_1A_1\bar{C}_0 + B_1\bar{A}_1C_0 + \bar{B}_1A_1C_0$

$= B_1A_1(C_0 + \bar{C}_0) + C_0(B_1\bar{A}_1 + \bar{B}_1A_1)$

$= B_1A_1 + (B_1 \oplus A_1).C_0$

→ Take C_i,

$C_i = \underbrace{B_iA_i}_{\text{Carry genrator } (G_i)} + \underbrace{(B_i \oplus A_i)}_{\text{carry Prpogator } (P_i)} . C_{i-1}$

$\therefore C_i = G_i + P_i . C_{i-1}$

→ For $i = 0$,

$C_0 = G_0 + P_0 C_{in}$

→ For $i = 1$,

$C_1 = G_1 + P_1 [G_0 + P_0 C_{in}]$ $\quad [\because C_0 = [G_0 + P_0 C_{in}]]$

$= G_1 + P_1 G_0 + P_1 P_0 C_{in}$

→ For $i = 2$,

$C_2 = G_2 + P_2 \cdot C_1$

$= G_2 + P_2 [G_1 + P_1 G_0 + P_1 P_0 C_{in}]$

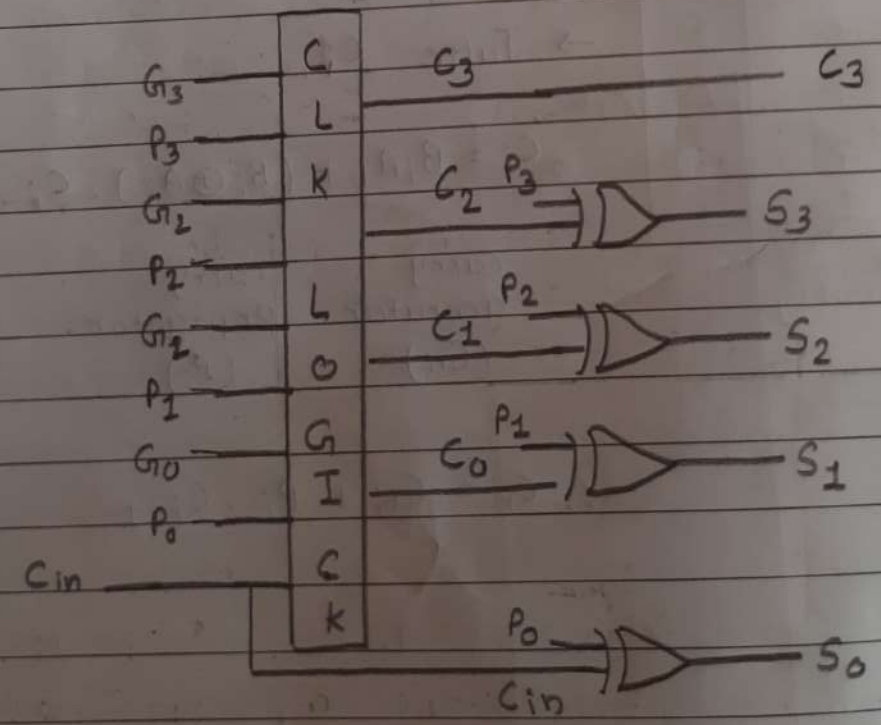
$= G_n + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$

→ For $i = 3$,

$C_3 = G_3 + P_3 C_2$

$= G_3 + P_3 [G_n + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}]$

$= G_3 + P_3 G_n + P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in}$



Digital Comprator

Ex. Design 2-bit comprator using gates

→ Truth-Table

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	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

A>B

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

A=B

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

A<B

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

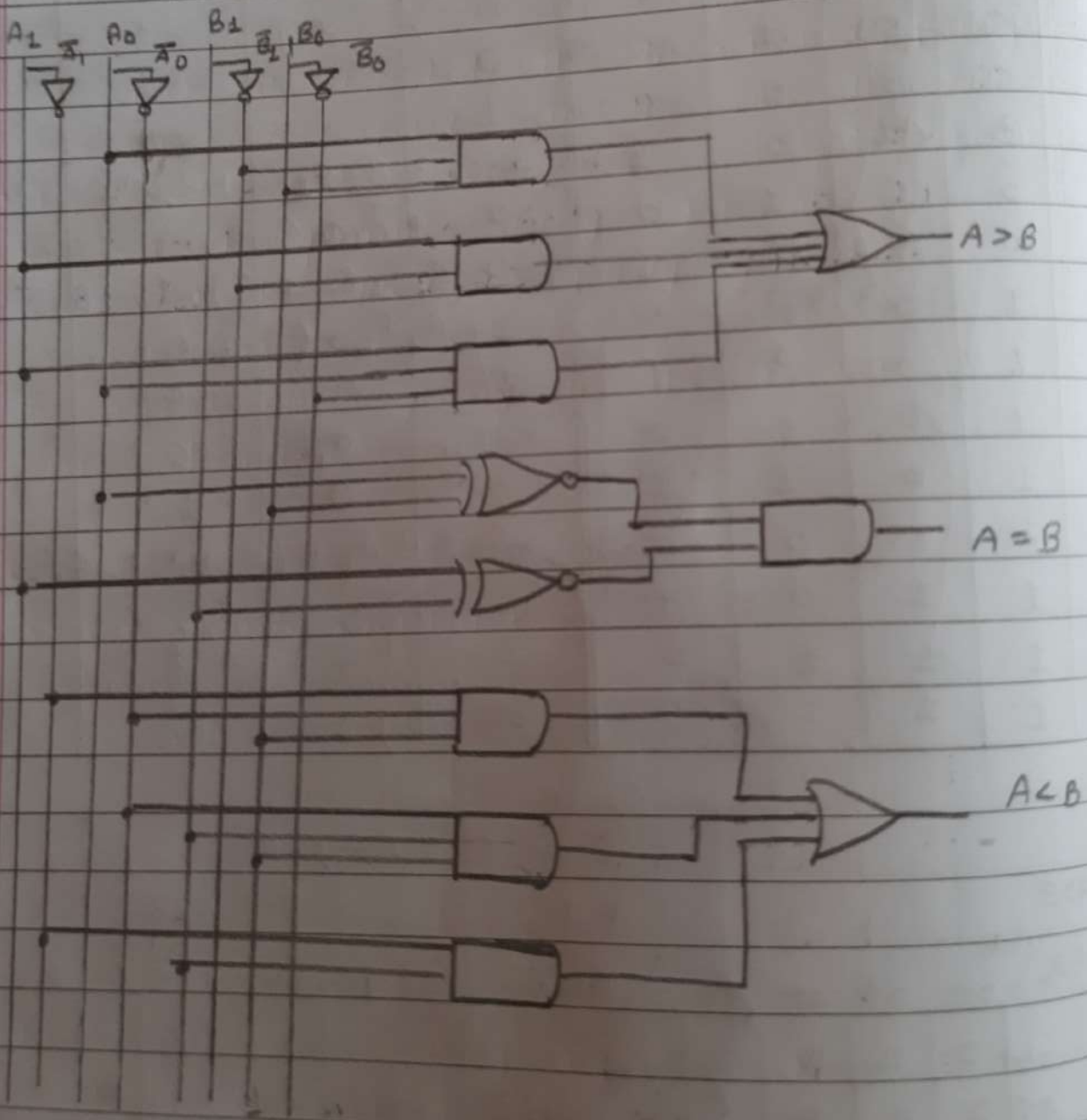
→ $A > B = A_0 \overline{B_1} B_0 + A_1 \overline{B_1} + A_1 A_0 \overline{B_0}$

→ $A < B = \overline{A_1} \overline{A_0} B_0 + \overline{A_0} B_1 B_0 + \overline{A_1} B_1$

→ $A = B = \overline{A_1} A_0 B_1 \overline{B_0} + \overline{A_1} A_0 \overline{B_1} B_0 + A_1 A_0 B_1 B_0 + A_1 \overline{A_0} B_1 \overline{B_0}$

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

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



Ex Design 1-Bit comprator using basic gates

→ Truth-table:

Input		Output		
A	B	A > B	A = B	A < B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

A>B

B \ A	0	1
1	0	0
0	0	1

A=B

B \ A	0	1
1	0	1
0	1	0

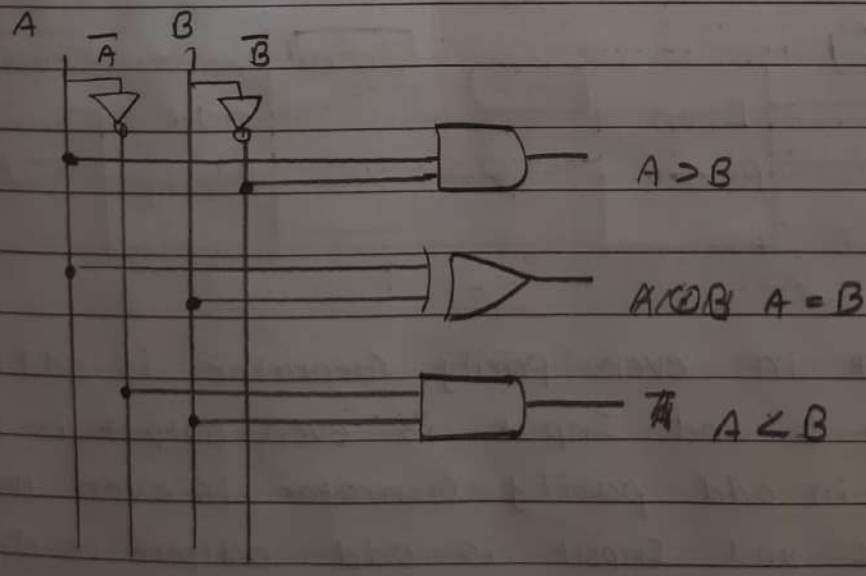
A<B

B \ A	0	1
1	1	0
0	0	0

→ $A > B = A\overline{B}$

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

$A = B = \overline{A}\,\overline{B} + AB = \overline{A \oplus B} = A \odot B$



☆ Parity Genrator

→ A parity bit is used for the purpose of detecting error during transmission of binary information.

→ A parity bit is an extra bit included with a binary massage to make the number either odd or even.

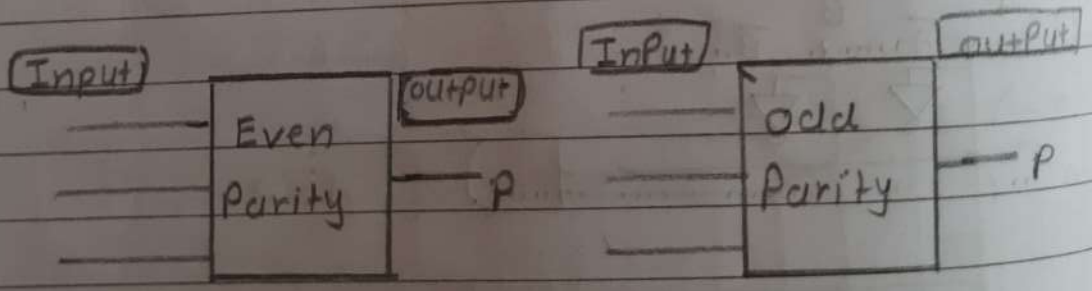
→ An error is detected if the checked parity does not correspond with the one transmitted.

→ The circuit genrates parity bit in the transmitor is called as parity genrator.

→ The circuit check parity in the reciver is called parity checker.

* 3- bit Parity Genrator

→



→ If Input in even parity Genrator is odd then output is 1 and Input is even output is 0.

→ If input in odd parity Genrator is even then output is 1 and Input is odd output is 0.

→

A	B	C	Even Parity	Odd Parity
0	0	0	0	1
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	0

→ Even

C \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

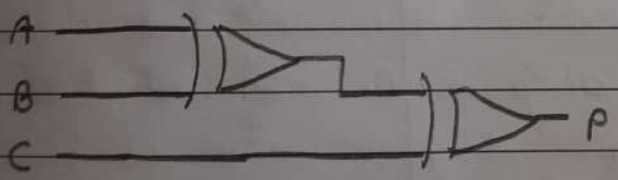
Even = $A \oplus B \oplus C$

Odd

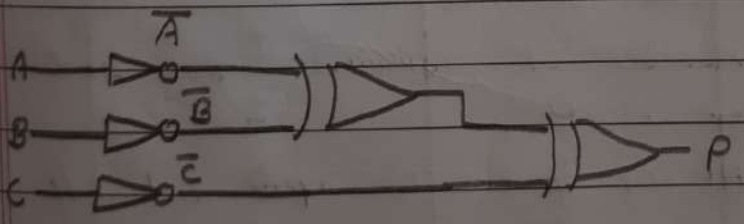
C \ AB	00	01	11	10
0	1	0	1	0
1	0	1	0	1

Odd = $\overline{A \oplus B \oplus C}$

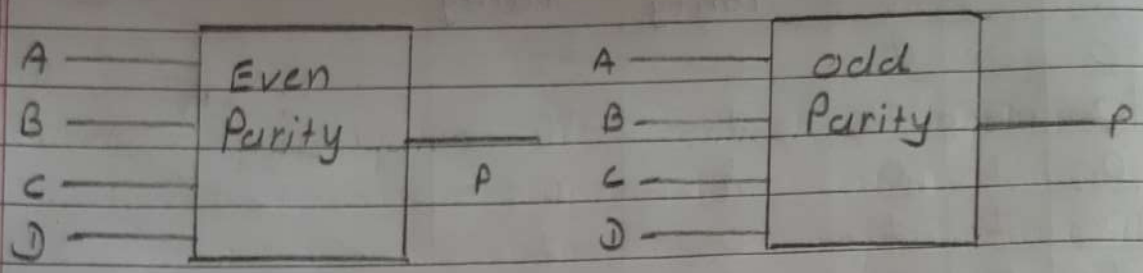
(Even)
→



(Odd)
→



* 4-Bit Parity genrator



→ In even parity genrator if input is odd then output is 1 and Input is even output is 0.

→ In odd parity Genrator if input is even then output is 1 and input is odd output is 0.

→

A	B	C	D	Even Parity	Odd Parity
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	1

Even

CD \ AB	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

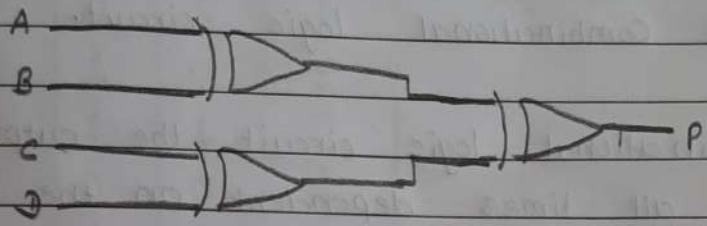
Odd

	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

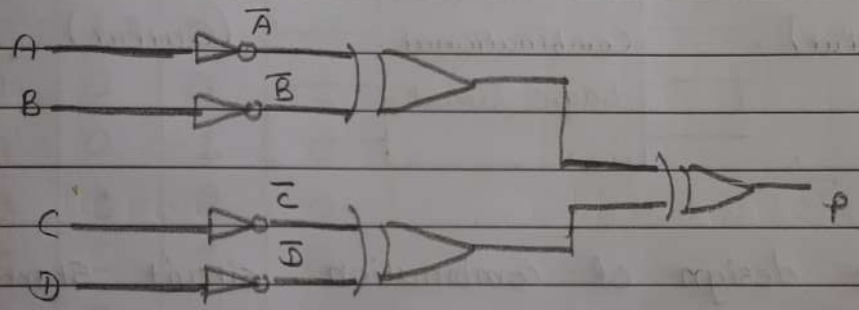
Even = $A \oplus B \oplus C \oplus D$

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

-> Even:



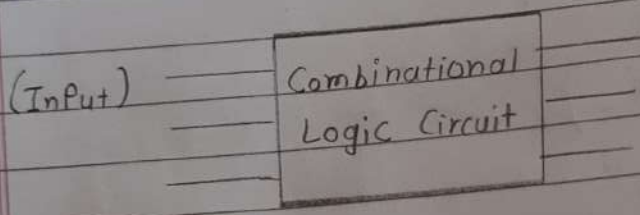
-> Odd:



☆ Combinational Design Example

→ When logic gates are connected together to produce specific output for certain combination of input variable, with no storage involved that is called combinational logic circuit.

→ In combinational logic circuit, the output variables are at all times dependent on the combination of input variables.



→ The design of combination circuit starts from outline of the problem statment and ends in a logic circuit diagram or set of boolean function which the logic diagram can be easily obtained.

→ Steps: (1) The Problem definition.
(2) Detemination number of avaluible input and output variables.
(3) Assigning a letter or symbol to input and o/p variable
(4) Derive the truth table that shows relationship between input and output variables.
(5) Obtain simplified boolean expression for each output.
(6) Obtain the logic diagram.

Ex: Design a combinational logic circuit output is high only when Majority of input (A, B, C, D) are low.

→ Step 1: Derive truth table.

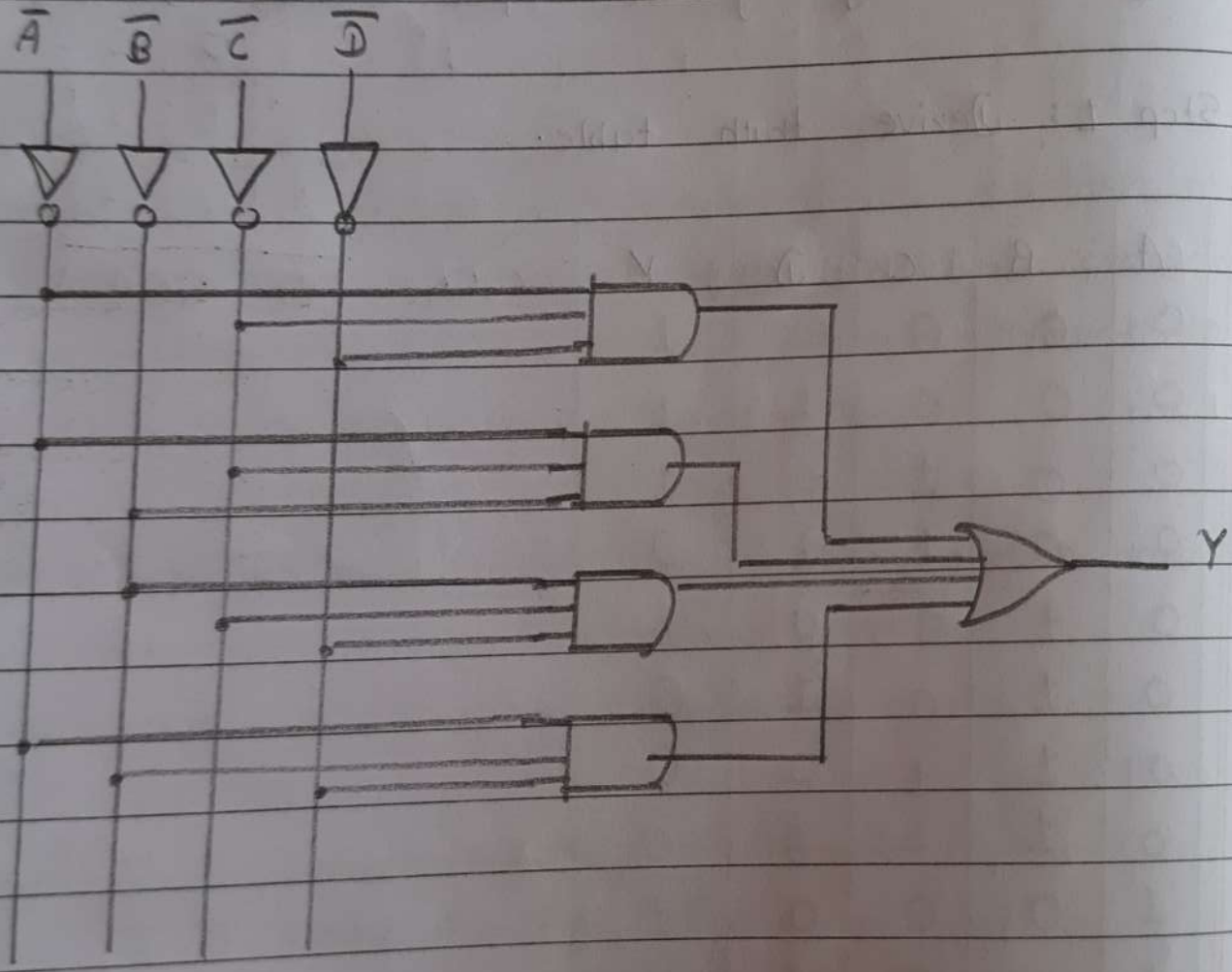
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

→ Step 2: K-Map

CD \ AB	00	01	11	10
00	1_0	1_4	0_{12}	1_8
01	1_1	0_5	0_{13}	0_9
11	0_3	0_7	0_{15}	0_{11}
10	1_2	0_6	0_{14}	0_{10}

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

→ Step 3 : Logic Diagram.



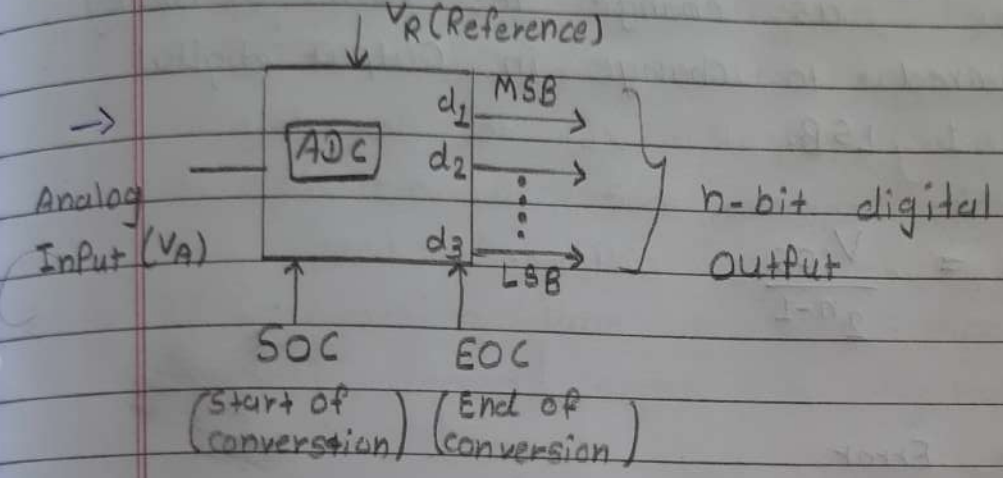
6 A / D and D / A Converters

Contents

Chapter : 6

A/D and D/A Convertors

☆ Performance parameter of ADC (Specification)



→ An electronic integrated circuit which transforms a signal from Analog to digital form.

→ Analog is a continuos signal.
(changing)

→ Digital is a discrete form signal.
(stable)

→ Analog signal are directly measurable quantities

→ Digital signal only have two states. For digital computer. we refer to binary state 0 and 1.

* Resolution

-> It is define as maximum number of digital output word.

-> It is also as change the value of analog signal required to change in output digital world by 1 LSB.

$$\text{Resolution} = \frac{V_{IFS}}{2^{n-1}}$$

* Quantization Error

-> Basicly, when Analog signal converted into digital signal that conversion may not be accurate or properly because the signal get approximate.

-> For example if signal is present 1/8 or between 1/8 and 2/8 the digital fomo signal becomes 001 this ~~digital signal~~ approximate is called as quantization and error occurs during this process is called quantization error.

-> The quantization error can be reduced by increasing number of bits when we increasing number of bits resolution of the ADC is also get increase.

$$Q_E = \frac{V_{IFS}}{(2^n - 1)\,2}$$

* Conversion Time

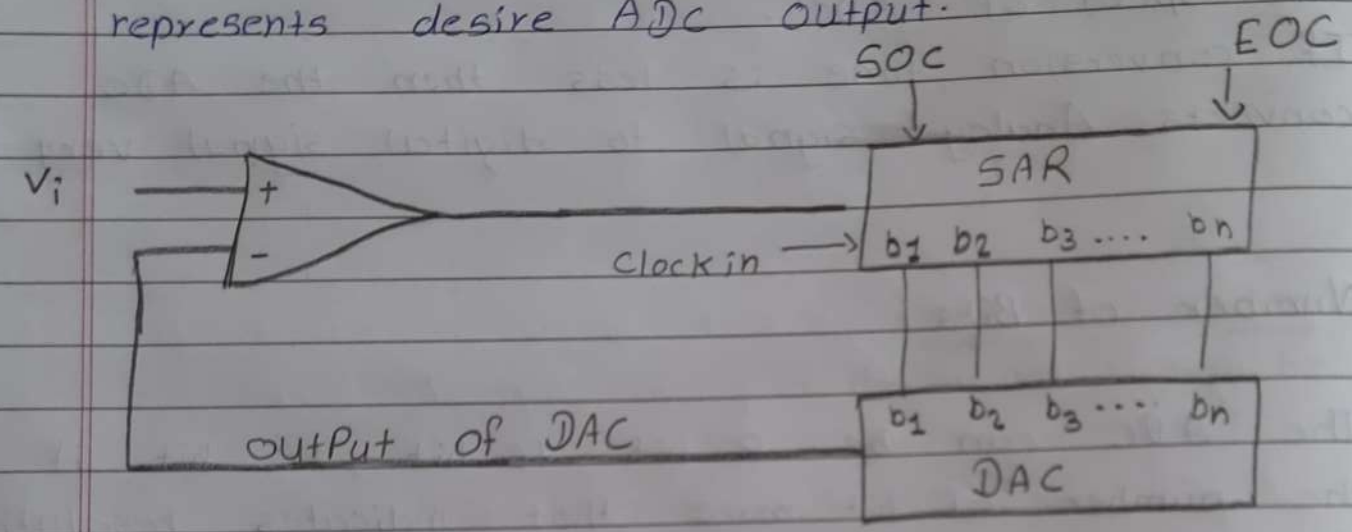
-> The block diagram of ADC consist two signal that is SOC and EOC.
-> Conversion time is a time required convert Analog signal into digital signal.
-> SOC and EOC called as conversion time. ideally it is 0 and practically as small as possible because it also increase decide the speed of ADC.
-> If conversion time is less then the ADC converts Analog signal to digital signal very fast.

* Number of Bits

-> The ADC can be 8 bit, 16 bit, 12 bit if the number of bit more that indicates resolution of ADC is more. If the resolution is high there is a less chances of quantization error.

Successive Approximate ADC

→ In this technique, the basic idea is to adjust the DAC's input code such as that it's output when is within $\pm \frac{1}{2}$ LSB of the analog input V_1 to be A/D converted. The code that achives this represents desire ADC output.



→ The succesive approximation method uses very ~~difficult~~ efficient code searching strategy is called binary search. It completes searching process for n-bit in just n clock periods.

→ It consits of a DAC, a comparator and a successive approximation register (SAR).

→ The external clock input sets the internal timing parameters. The SOC indicates A/D conversion process is activated and EOC indicates the completion of A/D conversion process.

☆ Comparison between ADC

Parameter	Flash (Parrallel)	Successive approximation	Dual Slope
(i) Speed	Fastest	Fast	Slow
(ii) Accuracy	Less	Medium	More
(iii) Resolution	upto 2^8	upto 2^{16}	upto 2^{16} or more
(iv) I/p hold time	Very less	Depends on number of bits.	It is max, hence Sample and hold circuit is required.
(v) Cost	Very costly	Medium	less
(vi) Applications	High speed fiber optics, Digital storage, Imaging and many more where high speed A/D conversion required	Advantge of high speed with excellent resolution. Hence these are most popular and use in data acquisition systems.	These are used when high accuracy and resolution is required and Speed is not a important criteria.

7 Semiconductor Memories and Programmable Logic Devices

Contents

Chapter : 7

Semiconductor memories And Programmable Logic Device.

☆ Memory

→ Memorys are made up for register. Each register in memory is one storage location also called memory location. Each memory location identified by an adress.

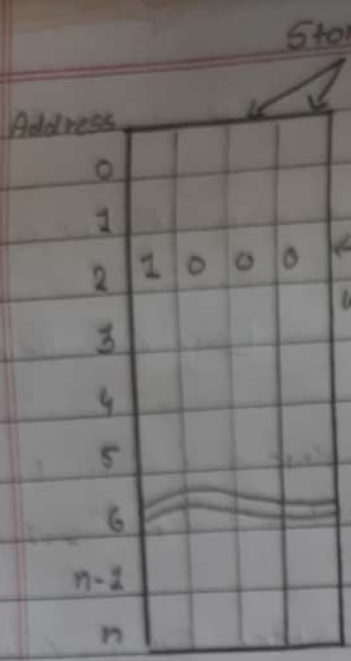
→ The number of storage location can vary from a few in some memories to hundred of thousand in other

→ Each location consits one or more bits. Genrally, the total number of bits that a memory can store is its capacity. Most of the capacity is specified in terms of bytes. (group of 8 bits)

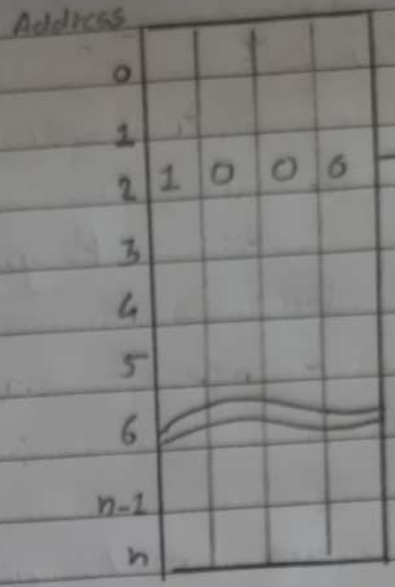
→ Each register consits of storage elements which stores one bit of data. A storage element is called cell.

→ A data stored in memory by a process is called writing. and are retrieved from the memory process called reding.

→ Memory unit stores binary information in groups of bit called words. A word in memory is an entity of bits that moves in and out of storage as a unit. A word having group of 8-bits is called a byte.



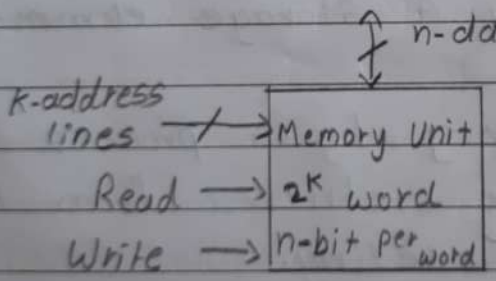
[write operation]



[Read operation]

→ The communication between memory and it's enviroment is achived through data lines, address selection line, and control lines that specify the direction of transfer.

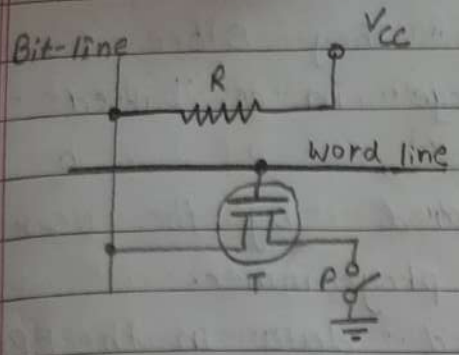
→ The data lines ~~stored~~ provide the information to be stored in memory and the K adress lines specify the particular word choosen among the many avalaible.



→ When there are k address lines we can access 2^k memory words.

→ Ex. k = 10 we can access 2^{10} = 1024 memory wor

☆ Read Only Memory (ROM)



open: Data stored (logic 1)
close: Data stored (logic 0)

→ We can not write data in read only memories. It is non-volatile memory. it can hold data even if power is turned off.

→ Genrally ROM is used to store the binary codes for the sequence of instruction you want the computer data such as look up tables.

→ This type of information does not change.

→ It is important to note that we give the name RAM to static and dynamic read/write memory device, that does not mean the ROMs that we are using are also not random access devices. Most of ROMs are accessed randomly with unique adresses.

→ It consits of Transistor (T) switch (P).

→ Transistor (T) is driven by the word line.

→ Content of cell can be read from the cell when word line logic is 1. A logic value 0 reads the transistor is connected to ground through switch (P).

→ If switch (P) is open, a logic value 1 is read, The bit line is connected through a resistor to the power supply. A sense circuit at the end of bit line genrates the proper output value. Data is stored into a ROM when it's manufactured.

→ Types: Masked ROM, PROM, EPROM, EEPROM or E^2PROM

☆ EPROM (Erasable Programmable ROM)

→ EPROMs use MOS circuitry. They store 1's and 0's as a packet of charge in a buried layer of the IC chip.
→ EPROMs can be programmed by the user with a special EPROM programmer.
→ We can erase the stored data in the EPROM by exposing the chip to ultra-violate light through its quaterz window for 15 to 20 minutes.
→ It is not possible to erase selective information when erase the entire information is lost.
→ The chip can be reprogrammed.
→ This memory is ideally suitable for product devlopment, experimental project.

☆ EAPROM

→ It is an aeronym for electrically Alterable programmable ROM. It is non-volatile ROM. It is form of PROM in which the contents of selected memory locations can be changed by applying suitable electrical signals without removing the EAPROM from the printed circuit board.

→ Advantage: We can selectively program or selectively erase EAPROM.
• Erasing time 5 to 10 ms and programing time 250 µs to 1 ms for particular location. This time is quite low.

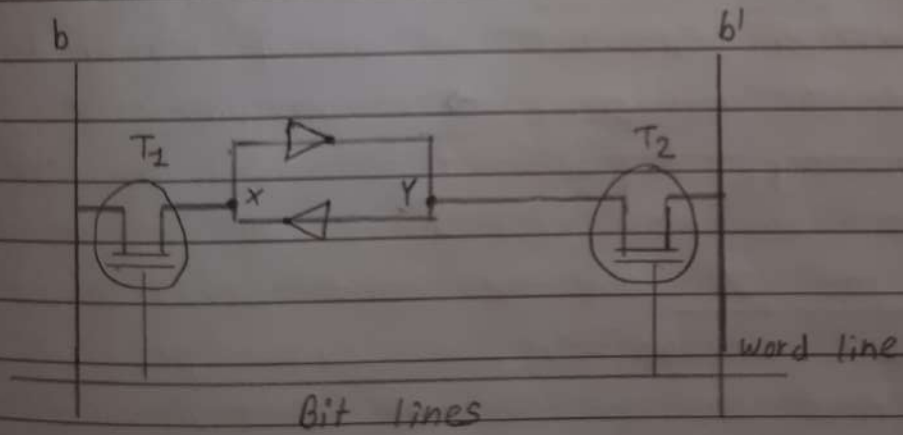
→ Disadvantage: It is expensive.
- It is not available from a variety of source.
- It is not popular as this technology is still not matured.

☆ Read and Write Momory (RAM)

→ There are two types of Ram.
(i) Static RAM (ii) Dynamic RAM

* Static RAM (SRAM)

→ Memory that consits of circuit capable of retaining their state as long as power is applied are known as static memorics. This are Random acess memory (RAM) and hence combinely called static RAM memories.

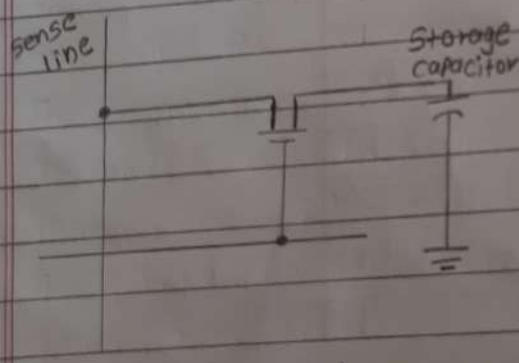


→ The latch is connected two bit lines by transistor T_1 and T_2. The word line controls the opening and closing of transistor T_1 and T_2. When word line is logico level the transistor are off and the latch retains its state.

⇒ Read Operation : Word line is made logic 1 (high) so that both transistors are ON. Now if the cell is in state 1, the signal on bit line b is high and the signal on bit line b' is low. The operation is true if the cell is in state 0. The b and b' are complement of each other. The sense circuit connected to bit lines that monitor the state of b and b' set the output accordingly.

⇒ Write : The state to be set is placed on the line b and its complement is placed on line b' and the word line is activated.
This action forces the cell into the corresponding state and write operation is completed.

* Dynamic RAM (DRAM)



→ DRAM stores data as a charged on the capacitor.
→ A DRAM contains thousand of such memory cell.
→ When COLUMN (sense) and ROW (control) lines go high the MOSFET conducts and charges the capacitor.

→ When COLUMN and ROW lines go low, the MOSFET opens and the capacitor retains its charge. In this way it stores 1 bit. Since only single MOSFET and capacitor needed.

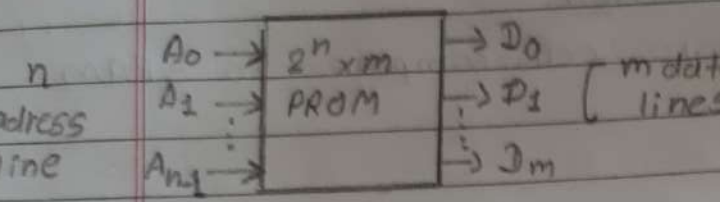
→ The DRAM contains more cells compare to SRAM per unit area.

→ The disadvantage of DRAM is that it needs refreshing ~~after~~ of charge on the capacitor after every few ms.
This complicate the system design. Since it requires extra hardware to control refreshing of DRAM.

☆ Comparision between RAM and ROM

Parameter	RAM	ROM
Definition	It is read/write memory.	It is Read only memory.
Volatility	It is volatile memory. Contenst are lost if Power turned off.	It is non-volatile memory Contents retain if power turned off.
Types	Static RAM (SRAM), Dynamic RAM (DRAM)	PROM, EPROM and EEPROM or E^2PROM
Speed	Read data faster than ROM.	Read speed is slower than RAM.
Use	RAM allows to read data quickly to run application.	It is used to store Program.

ROM / PROM as a PLD



→ ROM stands for read only memory and PROM stands for programmable read only memory.
→ The only difference in ROM and PROM is that ROM is programmed by manufacture while PROM is user programmable.
→ It consits of (n) input lines and (m) output lines
→ Each bit combination of the input variable is called an address.
→ Each bit combination of the output variables is called a word.
→ The number of bits per word is equal to the number of output lines, m.
→ The address specified in binary number denotes one of the minterms of n variables.
→ The number of distinct addresses possible with n-input variables is 2^n.
→ There are 2^n addresses and words in the PROM.
→ The word avalaible on the output lines at any given time depends on the address value applied to the input lines.
→ Let us consider 64 x 4 PROM
The PROM consists of 64 words of 4-bits each. This means there are four outPut lines and Particular word from 64 words presently avalaible on the output lines determined from the six input lines.

→ There are only six inputs in a 64 x 4 PROM because $2^6 = 64$ and with six variables, we can specify 64 addresses. For each address input there are unique selected word.

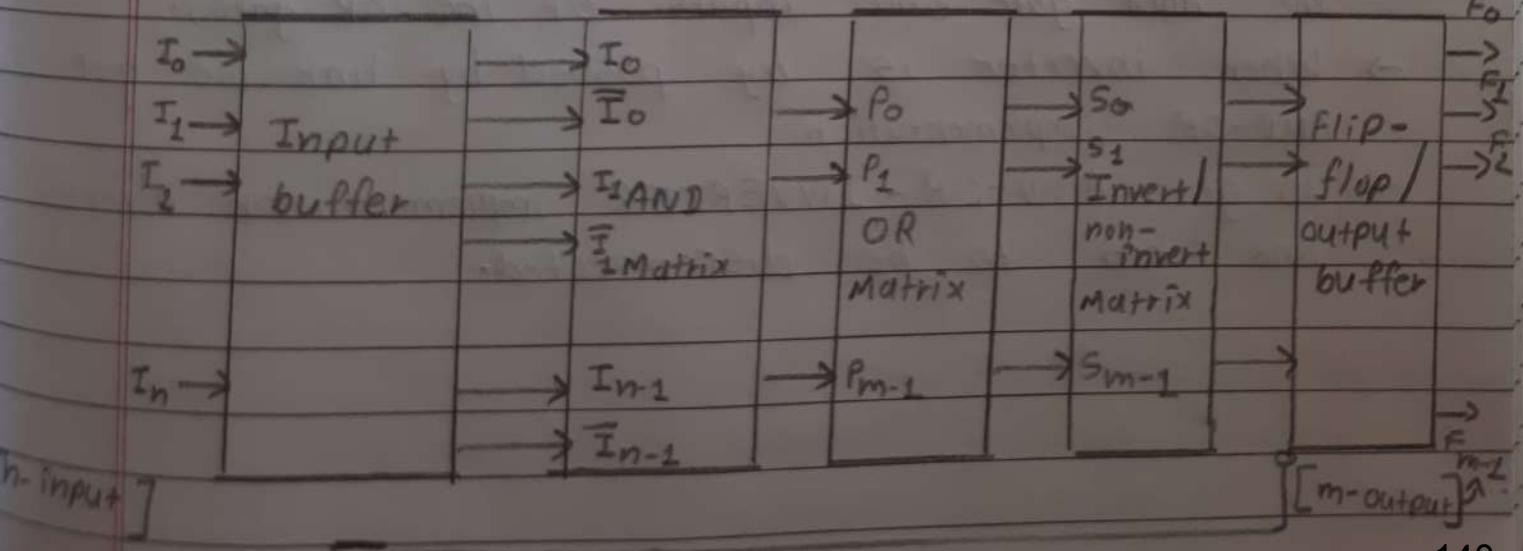
→ If ~~output~~ input address is 000000, word number ~~is~~ 0 is selected and applied to the output line.

→ If the input address is 111111, word number 63 is selected and applied to the output lines.

☆ Programmable Logic Array (PLA)

→ The combinational circuit do not use all the mintems every time. ~~Oeis~~ Occasionally, they have don't care conditions.

→ When implemented with a PROM becomes an address input that will never occur. The result is that not all the bit patteren available in the PROM are used, which may be considered a waste of available equipment.



→ For cases where the number of don't care conditions is excessive, it is more economical to use a second type of LSI component called Programmable Logic Array.

→ A PLA is similar to a PROM in concept however it does not provide full decoding of the variables and does not genrate all the minterm as in the PROM.

→ The PLA replace decoder by group of AND gate each of which can be programmed to genrate a product term of the input variables.

→ In PLA both AND and OR gates have fuses at the inputs, therefore in PLA both AND and OR gates are programmable.

→ The product term constitute a group of m AND gates and the sum terms constitute a group of m OR gates, called OR matrix.

→ Fuses are inserted between all n-inputs and their complement values to each of the AND gates.

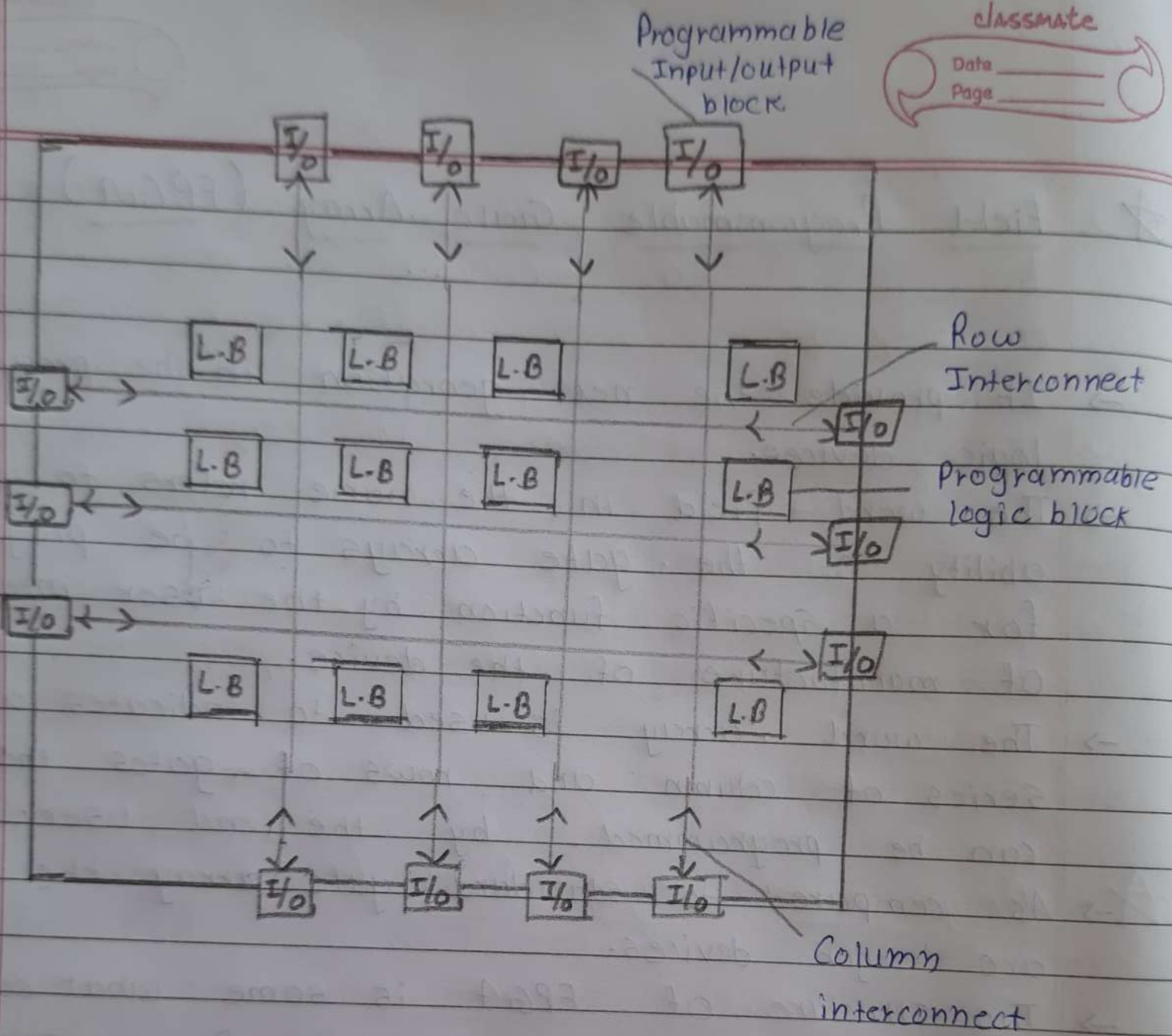
→ Fuses are also provided between the outputs of the AND gate and inputs of the OR gates.

→ When inverter is by passed by link we get AND-OR implementation.

→ To get AND-OR-INVERTER implementation inverter link has to be disconnected.

☆ Field Programmable Gate Array (FPGA)

→ It provide the next genration in the programmable logic devices.
→ The word field in the name refers to the ability of the gate arrays to be programmed for a specific function by the user intead of manufacture of the device.
→ The word array is used to indicates a series of column and rows of gates that can be programmed by the end user.
→ As compared to standard gate array the FPGA are larger devices.
→ The structure of FPGA is some what complicated than the basic cell structure for standard gate array.
→ The logic box of FGPA are called logic blocks or configurable logic blocks (CLBs)
→ The basic structure of FPGA consits of an array of logic blocks with programmable row and column interconecting channel surrounded by programmable I/o blocks.


Programmable Input/output block
I/O
L.B
Row Interconnect
Programmable logic block
Column interconnect