Monday

Date 05/09/22
Page

STLD → Switching Theory & Logic Design

★ Number Systems:

1) Decimal Number System $(\ )_{10}$ → 0-9
2) Binary Number System $(\ )_2$ → 0-1
3) Octal Number System $(\ )_8$ → 0-7
4) Hexadecimal Number System $(\ )_{16}$ → 0-9, A, B, C, D, E, F

A, B, C, D, E, F
↓ ↓ ↓ ↓ ↓ ↓
10 11 12 13 14 15

Base → RADIX → No. of digits present in the system

★ Conversion:

(1) A) DECIMAL TO BINARY

Ex1: $(92)_{10} \longrightarrow (?)_2$

2	92	0
2	46	0
2	23	1
2	11	1
2	5	1
2	2	0
	1	

$(92)_{10} \longrightarrow (1011100)_2$

Ex2:

2	243	1
2	121	1
2	60	0
2	30	0
2	15	1
2	7	1
2	3	1
	1	

$(243)_{10} \longrightarrow (?)_2$

$(243)_{10} \longrightarrow (11110011)_2$

Ex3: $(0.92)_{10} \longrightarrow (?)_2$

$0.92 \times 2 = 1.84 \longrightarrow 1$

$0.84 \times 2 = 1.68 \longrightarrow 1$

$0.68 \times 2 = 1.34 \longrightarrow 1$

$0.34 \times 2 = 0.68 \longrightarrow 0$

$(0.92)_{10} \longrightarrow (0.1110)$

Ex4: $(11.48)_{10} \longrightarrow (?)_2$ $\qquad (11)_{10} \longrightarrow (1011)_2$

2	11	1
2	5	1
2	2	0
	1	

$0.48 \times 2 = 0.96 \longrightarrow 0$

$0.96 \times 2 = 1.92 \longrightarrow 1$

$0.92 \times 2 = 1.84 \longrightarrow 1$

$0.84 \times 2 = 1.68 \longrightarrow 1$

$(0.48)_{10} \longrightarrow (0111)_2$

$(11.48)_{10} \longrightarrow (1011.0111)_2$

1) B) BINARY TO DECIMAL :

Most Significant Bit (MSB) ↑ 1 0 1 1 1 ↑ (LSB) Least Significant Bit

Ex1:

$2^2\ 2^1\ 2^0\ \ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ 2^{-5}\ 2^{-6}$

111.011011

$(?)_{10} = 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 1 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-6}$

$= 4 * 2 + 1 + 0 + 1/4 + 1/8 + 0 + 1/32 + 1/64$

$= 7 + 0.25 + 0.125 + 0.03125 + 0.015625$

$= (7.421875)_{10}$

7/09/22

2) A) DECIMAL TO OCTAL

Ex: $(79.37)_{10} = (?)_8$

8	79	7 ↑
8	9	1
	1	

$(79)_{10} \longrightarrow (117)_8$

$0.37 \times 8 = 2.96 \longrightarrow 2$

$0.96 \times 8 = 7.68 \longrightarrow 7$

$0.68 \times 8 = 5.44 \longrightarrow 5$

$\Rightarrow (79.37)_{10} \longrightarrow (117.275)_8$

B) OCTAL TO DECIMAL

Ex. $(234.45)_8 \longrightarrow (?)_{10}$

$8^2\ 8^1\ 8^0\ \ 8^{-1}\ 8^{-2}$

$2\ 3\ 4\ .\ 4\ 5$

$= 2 \times 64 + 3 \times 8 + 4 \times 8^0 + 4 \times 8^{-1} + 5 \times 8^{-2}$

$= 128 + 24 + 4 + 0.5 + 0.125 + 0.07$

$= 156.695$

$\Rightarrow (234.45)_8 \longrightarrow (156.57)_{10}$

3) DECIMAL TO HEXADECIMAL

$(234.45)_{10} \rightarrow (?)_{16}$

16	234	10 ↑
	14 ←	

$0.45 \times 16 = 7.2$ → 7

$0.2 \times 16 = 3.2$ → 3

$(EA)_{16}$

$(234.45)_{10} \longrightarrow (EA.733)_{16}$

5) HEXADECIMAL TO DECIMAL

$(D4E.79B)_{16} \longrightarrow (?)_{10}$

$16^2\ 16^1\ 16^0\ \ 16^{-1}\ 16^{-2}\ 16^{-3}$

D 4 E . 7 9 B

$16^1 = 16$

$16^2 = 256$

$16^3 = 4096$

$16^4 = 65536$

$= D \times 16^2 + 4 \times 16^1 + E \times 16^0 + 7 \times 16^{-1} + 9 \times 16^{-2} + B \times 16^{-3}$

$= 13 \times 16^2 + 4 \times 16 + 14 \times 16^0 + 7 \times 16^{-1} + 9 \times 16^{-2} + 11 \times 16^{-3}$

$= 3328 + 64 + 14 + 0.4375 + 0.035 + 0.0026$

$= 3406 + 0.475$

$\Rightarrow (D4E.79B)_{16} \longrightarrow (3406.475)_{10}$

4) BINARY TO OCTAL

$(1101110010.001110)_2$

↓

To convert binary number to octal, we need to make groups of 3 digits for integer part decimal points to MSB side and for fractional part decimal to LSB side.

(001 101 110 010 . 001 110)

Decimal point

⇒ $(1562.16)_{10}$ =

⇒

Binary Table

Decimal No.	Binary No. (8 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 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

5) BINARY TO HEXADECIMAL CONVERSION:

In this we need to make groups of 4 digits for integer part decimal point to MSB and for fractional part decimal point to LSB.

0011 0111 0010 . 0011 1000

3 7 2 3 8

$\Rightarrow (372.38)_{16}$

8/09/22

* Binary Arithmetic

• ADDITION

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10

1 + 1 + 1 = 1 + 10 = 11

1 + 1 + 1 + 1 = 10 + 10 = 100

• Ex

```
  11110.11
  10011.00
+ 00010.11
 ---------
 110100.10
```

```
  111
  11110.11
  10011.00
 ---------
 110001.11
```

```
   111 1
  1.10001.11
  000010.11
 ----------
  110100.10
```

Date
Page

- SUBTRACTION

$0 - 0 = 0$
$0 - 1 = 1$ with borrow 1
$1 - 0 = 1$
$1 - 1 = 0$

$10 = 1 + 1$
$10 - 1 = 1$
$1 + 1 - 1 = 1$

Ex.

$$\begin{array}{r} 1\,1\,\overset{0}{1}\,\overset{10}{1}\,\overset{10}{0}.1\,1 \\ -\,1\,0\,0\,1\,1.0\,0 \\ \hline 0\,1\,0\,1\,1.1\,1 \end{array}$$

- MULTIPLICATION

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

~~1 1 0~~

Ex.

$$\begin{array}{r} 1\,1\,0\,1\,1 \\ \times\ 1\,0\,1 \\ \hline 1\,1\,0\,1\,1 \\ 1\,1\,0\,1\,1\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0 \\ \hline 1\,0\,0\,0\,0\,1\,1\,1 \end{array}$$

~~- DIVISION :~~
~~0 ÷ 0 =~~

~~* - Division~~

- DIVISION

```
          1011
   101 ) 11̇0111 (
       - 101↓↓|
         00111|
           -101↓
           0101
           -101
            000
```

★ Complements

r → radix

(1) r's complement → $r^m - N$

(2) (r-1)'s complement → $r^m - r^{-m} - N$

143.75 — N → Number

n=3, m=2

n → no. of digits in integer part

m → no. of digits in fractional part

Ex:

Q. Take r's complement of $(143.75)_{10}$

Sol.

$= 10^3 - 143.75$

$= 1000 - 143.75$

$= 856.25$

Direct method

Method 2: for 10's complement

(143.75)

Subtract every digit from 9 then +1 at the end.

```
  999.99
- 143.75
  856.24
+      1
  856.25
```

Date
Page

- **(r-1)'s complement:**

$= r^n - r^{-m} - N$

$(143.75)_{10} = 10^3 - 10^{-2} - 143.75$

$= 1000 - 0.01 - 143.75$

$= 1000 - 143.75$

$= 856.24$

Direct method:

9's complement

$$\begin{array}{r} 999.99 \\ 143.75 \\ \hline 856.24 \end{array}$$

- **2's complement**

I^st method: $(11011.01)_2 \xrightarrow{\text{2's complement}} (2^5)_{10} - (11011.01)_2$

Step II: convert to Binary

$= (32)_{10} - (11011.01)_2$

$= 100000 - 11,011.01$

$= 000100.11$

II^nd method

Direct

$$\begin{array}{rl} 11011.01 & \xrightarrow{1\text{'s}} 00100.10 \\ & \quad + \quad 1 \\ & \xrightarrow{2\text{'s}} 00100.11 \end{array}$$

$$\begin{array}{r} 100000.00 \\ 11011.01 \\ \hline 000100.11 \end{array}$$

→ for 2's complement, find the 1's complement then add 1 at the end.

- 1's complement.

1st method:

$(1011.110)_2 \xrightarrow{\text{1's complement}} 2^4 - 2^{-3} - (1011.110)_2$

$= (16)_{10} - (0.125)_{10} - (1011.110)_2$

$= (15.875)_{10} - (1011.110)_2$

2	15	1
2	7	1
2	3	1
	1	—

$= (1111.111)_2 - (1011.110)_2$

$= 0100.001$

$0.875 \times 2 = 1.75 \hookrightarrow 1$

$0.75 \times 2 = 1.50 \hookrightarrow 1$

$0.50 \times 2 = 1.0 \hookrightarrow 1$

$$\begin{array}{r} 1111.111 \\ -1011.110 \\ \hline 0100.001 \end{array}$$

12/09/22

- **Arithmetic Using Complements:-**

Case I: when minuend > subtrand

I. Subtraction → Using 1's comp.

15 → 1 1 1 1 Minuend

− 0 1 0 1 Subtrand

original Ans 1 0 1 0

Ans: $(1010)_2$

1 1 1 1 ① 1's

+ 1 0 1 0 Rule

① 1 0 0 1 if carry

+ → 1 comes

Ans 1 0 1 0 out in MSB add it to LSB

② In 2's case, carry is neglected.

Example:

1 1 1 0 1

− 1 0 0 0 1

0 1 1 0 0

→

1 1

1 1 1 0 1

+ 0 1 1 1 0

① 0 1 0 1 1

+ → 1

0 1 1 0 0

II. Subtraction − Using 2's comp.

1 1 1 0 1

− 1 0 0 0 1

0 1 1 0 0

1 1 1 1

1 1 1 0 1

+ 0 1 1 1 1

① 0 1 1 0 0

↓

neglecting MSB

Ans: $(01100)_2$

Case II: Minuend < subtrand

Using 1's

A 0111
B 1010

A − B = ?

```
  111
  0111
+ 0101
  1100   no carry generated
```

So the answer will be complement of the value

Answer is $(-0011)_2$

```
  0101
+    1
  1010
```

Using 2's

```
A  0111   →   0111
B  1010     + 0110   no carry generated
            ------
            1 1 0 0 1
```

```
  0010
+    1
```

Ans: $(-0011)_2$

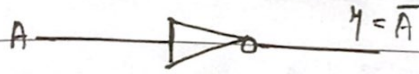
13/09/22 Logic Gates :-

- Basic Gates
  - NOT (−)
  - AND (·)
  - OR (+)
- Universal Gates
  - NAND
  - NOR
- Application of 2 types of gates
  - EX-OR → $A\bar{B} + \bar{A}.B = A \oplus B$
  - EX-NOR → $\bar{A}\bar{B} + AB = A \odot B = \overline{A \oplus B}$

Page

Basic Gates:-

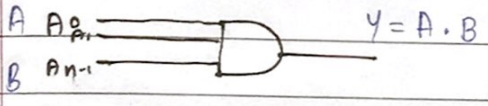
(1) NOT Gate :-



A	Y = Ā
0	1
1	0

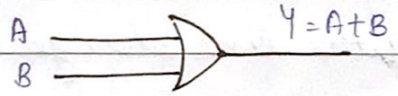
combination = 2^n

(2) AND Gate :-



A	B	Y = A . B
0	0	0
1	0	0
0	1	0
1	1	1

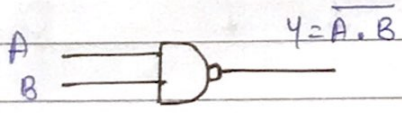
(3) OR Gate :-



A	B	Y = A + B
0	0	0
1	0	1
0	1	1
1	1	1

Universal Gates: These gates can ~~be~~ replace basic gates.

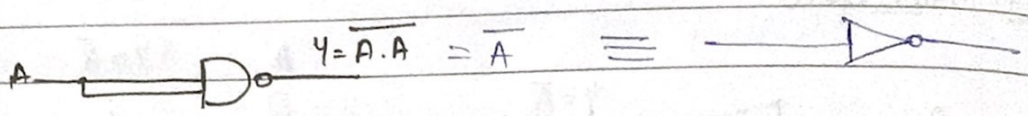
(1) NAND Gate



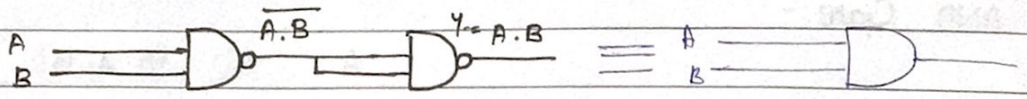
A	B	Y = $\overline{A . B}$
0	0	1
0	1	1
1	0	1
1	1	0

Basic gates using NAND gates :-

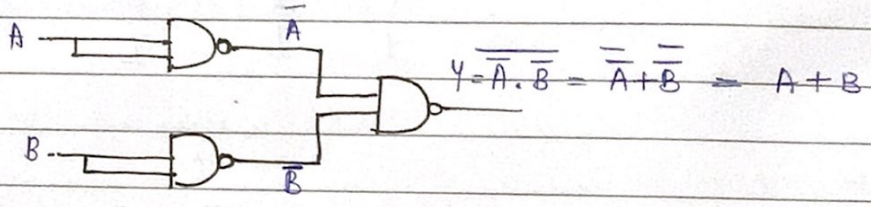
(I) NOT



(II) AND

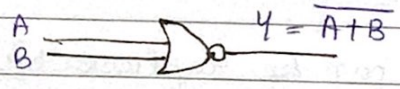


(III) OR



De-Morgan's Theorem: break the line change the sign

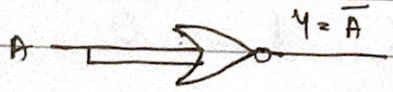
(2) NOR Gate:



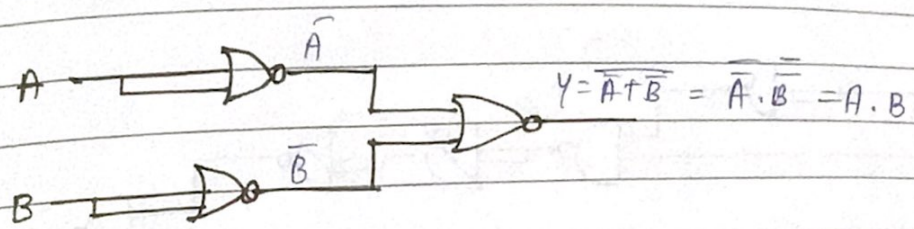
A	B	$Y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Basic gates using NOR Gate

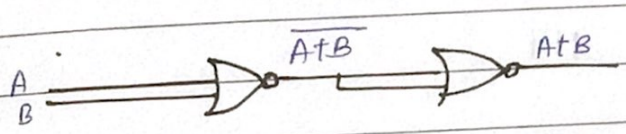
(I) NOT



(II) AND



(III) OR



15/09/22

✱ EX-OR :-

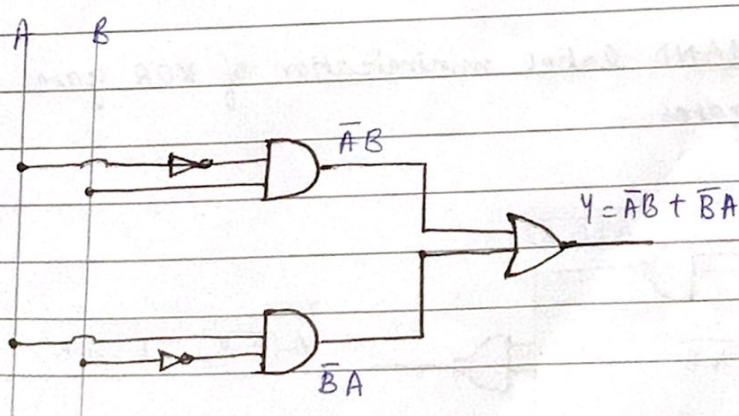
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	0	0

~~Same~~ → when inputs are same XOR is 0

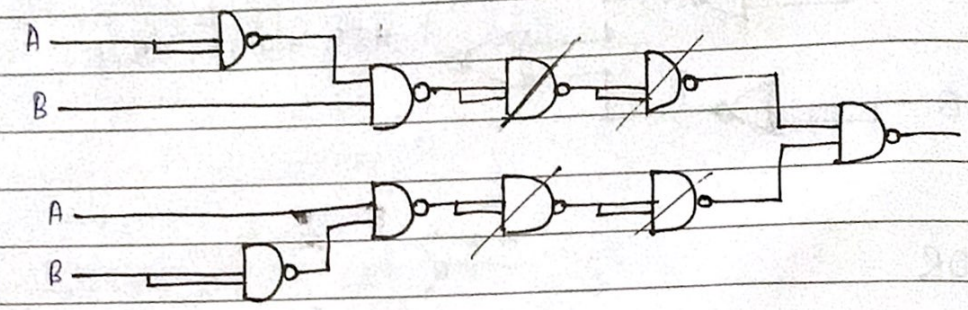
↳ when inputs are opposite XOR is 1

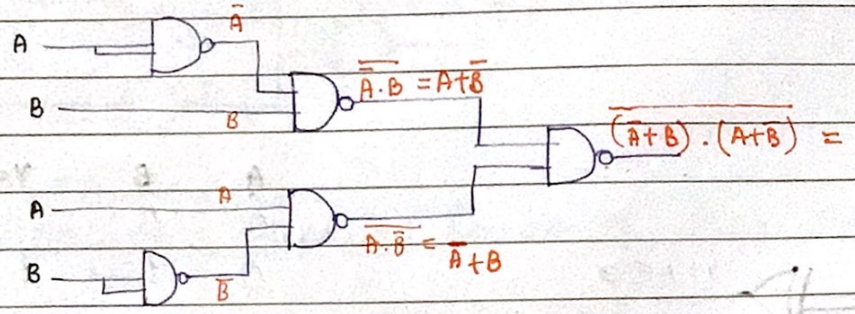
$\bar{A}.B + \bar{B}A$

EX-OR using Basic gate



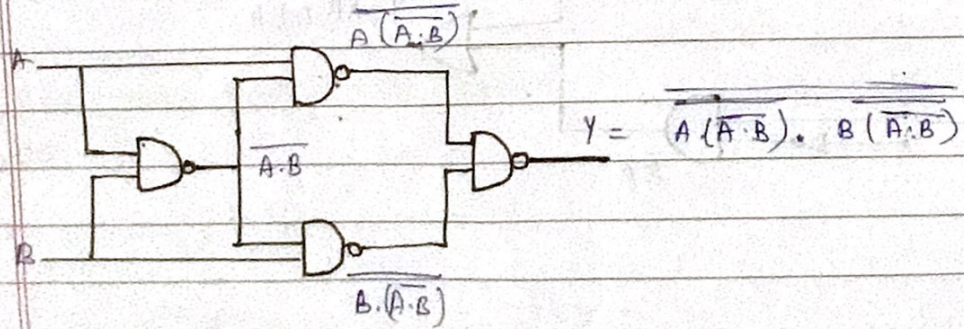
Ex-OR gate using NAND Gate only





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

Q Draw the NAND label minimization of XOR gates using minimum gates.



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

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

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

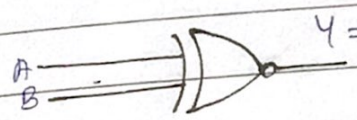
$$= A\overline{A} + A\overline{B} + B\overline{A} + B.\overline{B}$$

$A\overline{A} = 0$

$A\overline{A} \rightarrow 0$, $B.\overline{B} \rightarrow 0$

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

✱ EX-NOR :-



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

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

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

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

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

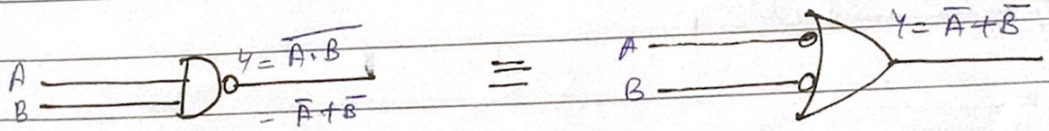
$A\overline{A} \rightarrow 0$, $B\overline{B} \rightarrow 0$

A	B	$Y = A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1

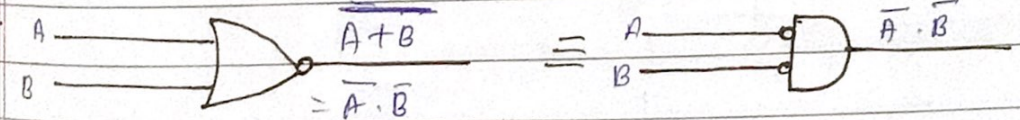
$$A \odot B = A\overline{B} + AB$$

16/09/22. Two level NAND / NOR Implementation :-

NAND



NOR



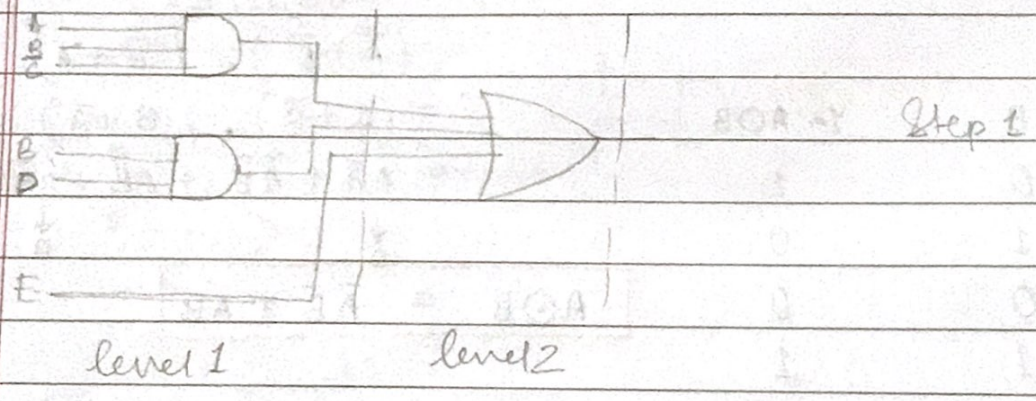
★ AND-OR logic to NAND/NOR logic:

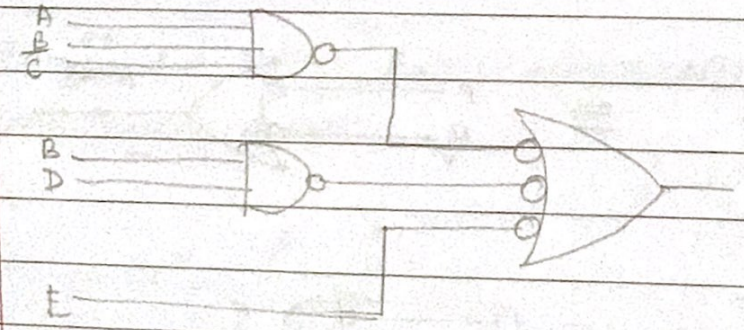
- AND-OR logic to NAND/NAND implementation:

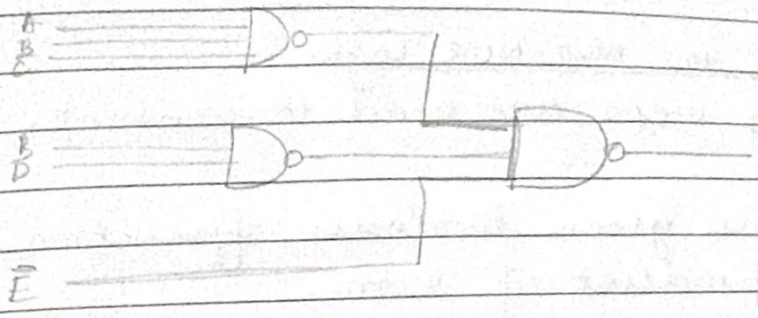
1. Draw and AND-OR logic for the simplified boolean expression.
2. Replace ~~An~~ AND gate by NAND gate and OR gate by bubbled OR gate
3. Replace bubbled OR gate by NAND gate

Ex: Implement NAND NAND logic for given expression:-

$Y = AB\bar{C} + BD + E$

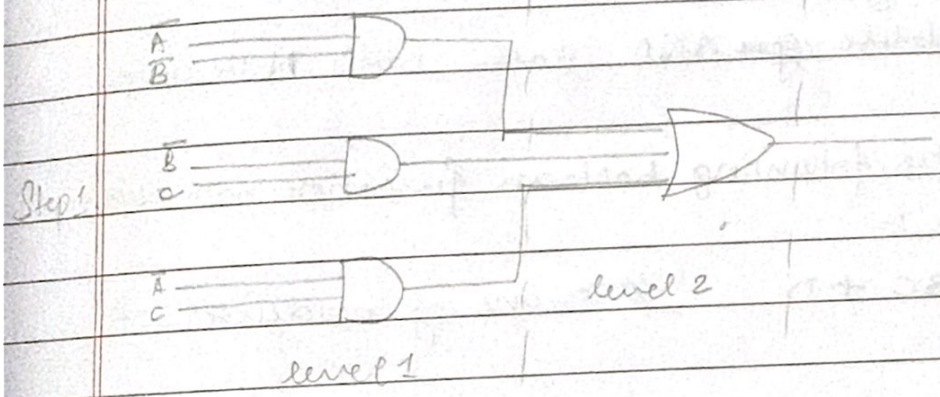


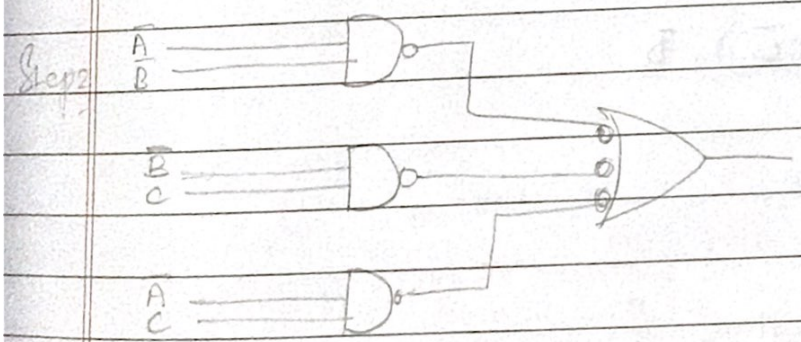


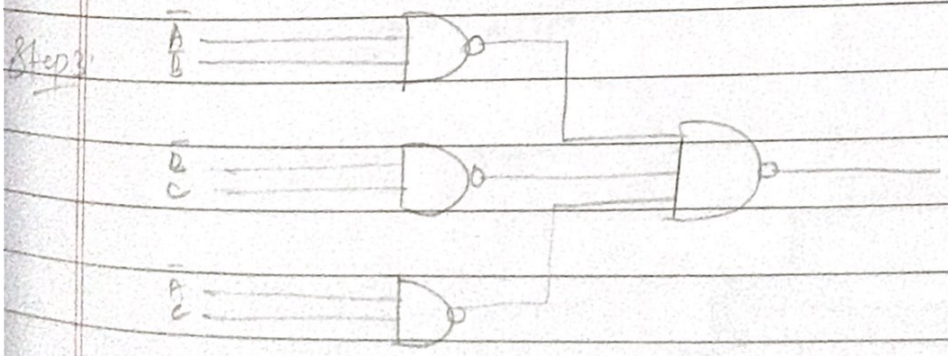


Ex2: Implement the boolean function with NAND NAND logic

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







- **OR-AND Logic to NOR-NOR Logic :-**

The following steps are used to convert,

1. Simplify the given boolean function in the form of product of sum.
2. Draw OR-AND logic circuit for the simplified expression.
3. Replace OR gate by NOR gate and AND gate with bubbled AND gate
4. Replace bubble of AND gate with NOR gate.

Ex1: Implement the following boolean function with NOR-NOR gate logic :-

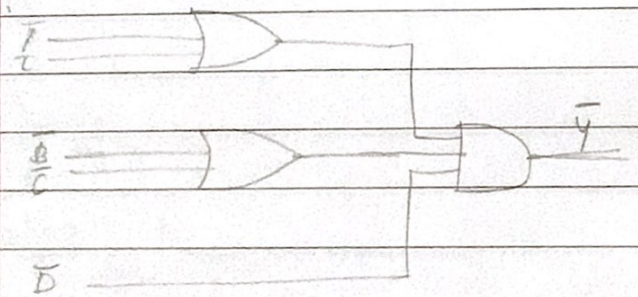
$Y = AC + BC + D$ // sum of products (SOP)

Solution:

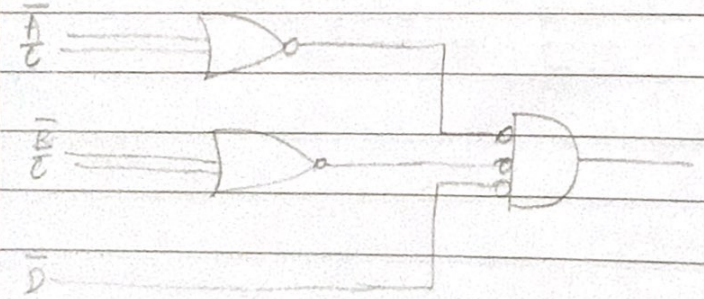
By applying De morgan's theorem

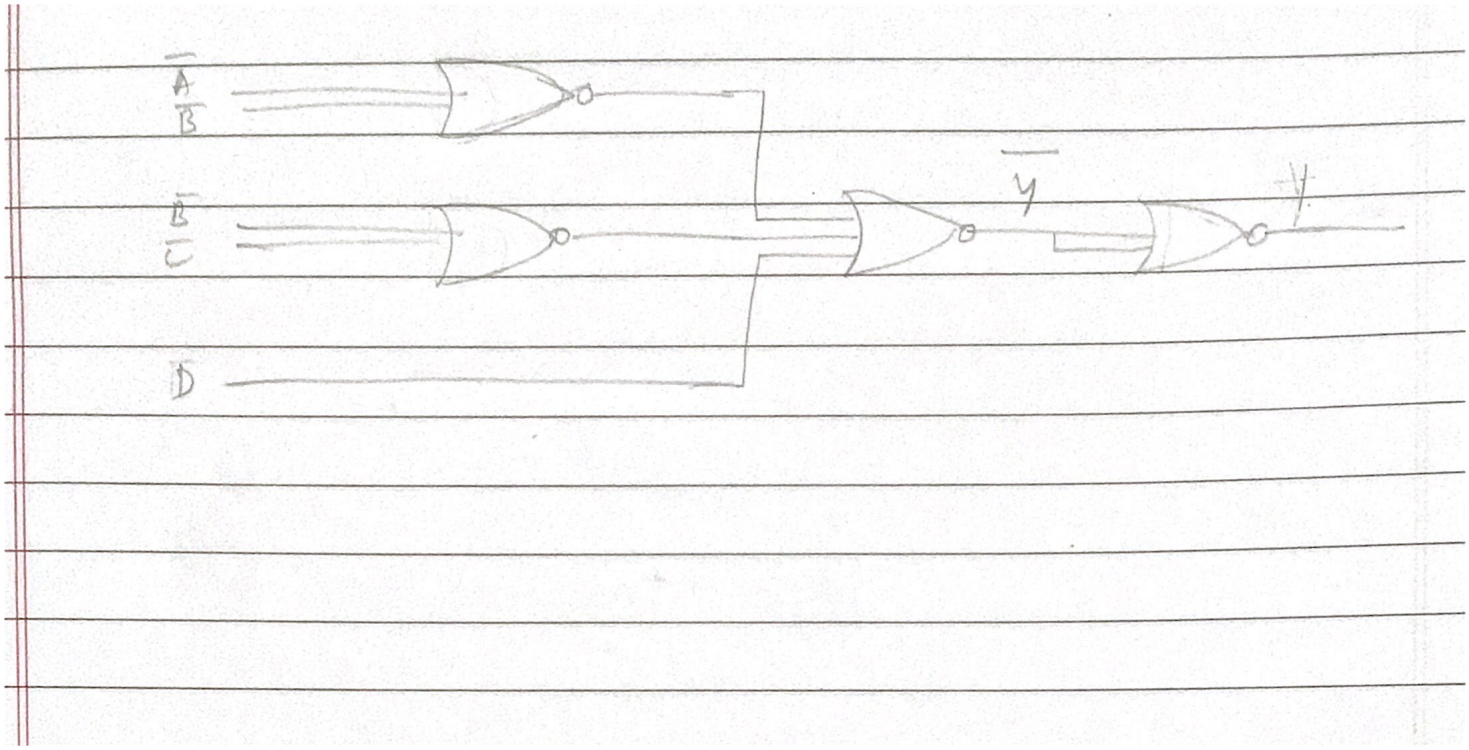
$$\overline{Y} = (\overline{A}+\overline{C}).(\overline{B}+\overline{C}).\overline{D}$$

Step 2:



Step 3:




A
B
E
C
D
y
y

19/09/22

* Multilevel NAND Implementation :-

The following steps are used to implement :-

Step 1: Draw the logic circuit using basic gates for the given boolean expression.

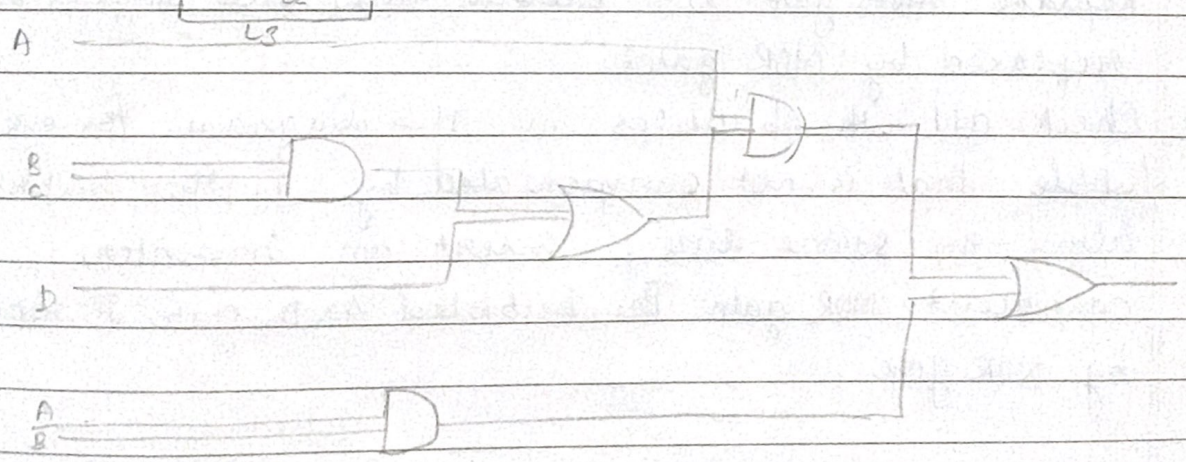
Step 2: Replace all the AND gates using NAND gates and OR gates are replaced by bubbled OR gate.

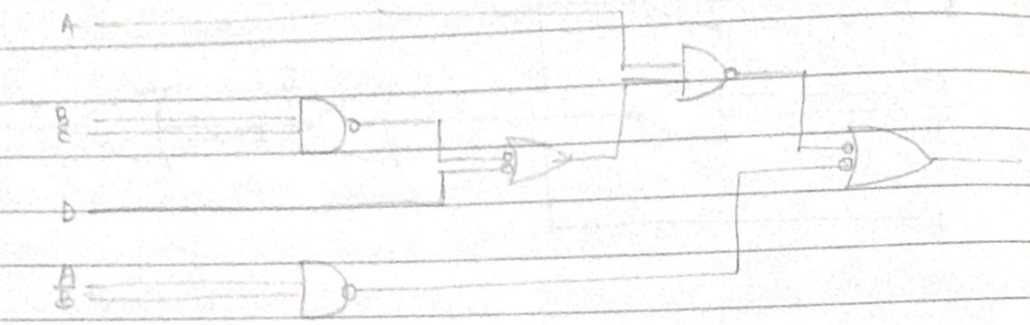
Step 3: Check all the bubbles in the diagram. For every bubble that is not compensated by another bubble along the same line. Insert a NOT gate and replace it with equivalent NAND gate. The bubbled OR is also replaced by NAND gate.

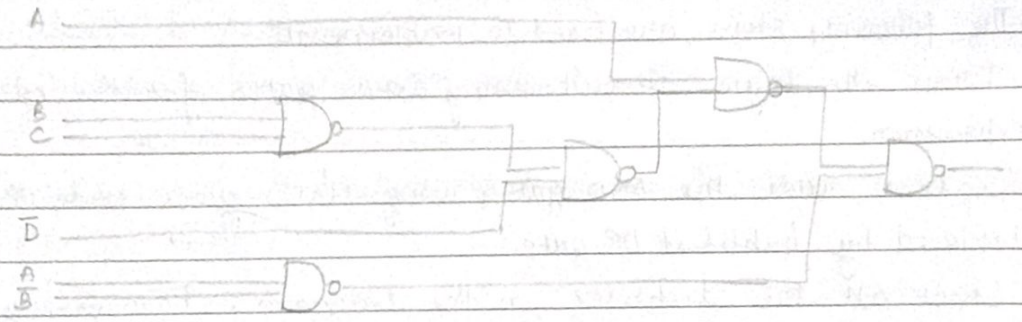
Q.

$$Y = A.(BC+D) + A\overline{B}$$

L1 L2 L3 L1







✱ Multilevel NOR Implementation :-

The following steps are followed to implement:

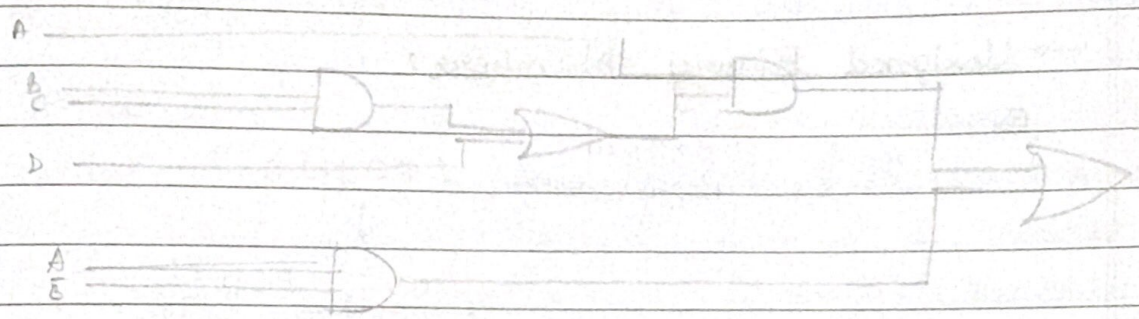
Step 1: Draw the logic diagram using basic gate for the given boolean expression.

Step 2: Replace AND gate with bubbled AND gate, and OR gates are replaced by NOR gates.

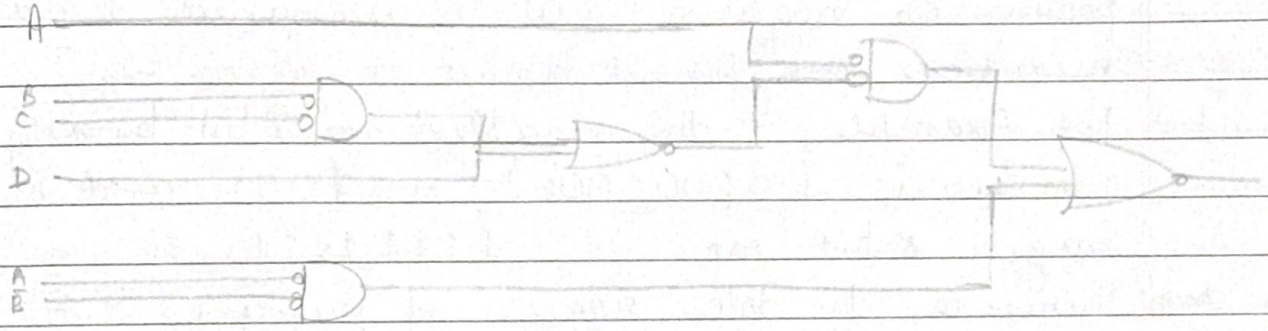
Step 3: Check all the bubbles in the diagram. For every bubble that is not compensated by another bubble along the same line, insert an inverter equivalent NOR gate. The bubbled AND gate is replaced by NOR gate.

Q. $Y = A(BC + D) + A\bar{B}$

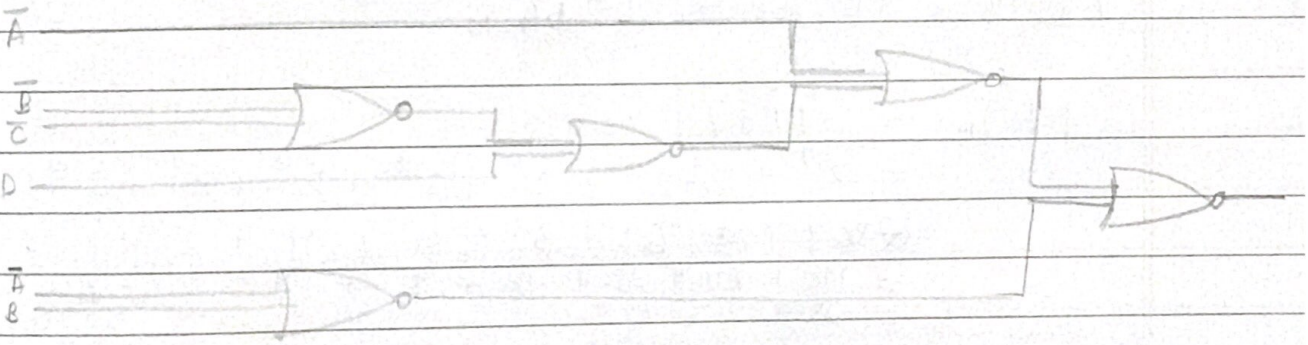
Step1:



Step2:



Step3:



20/09/22

* Sign/Magnitude Representation: & Unsigned

↳ Unsigned Binary Numbers:

Ex:

$(22)_{10}$ $\xrightarrow{\text{unsigned Representation}}$ 10110

$\downarrow$

$(0001\,0110)_2$

↳ In some applications, all the datas are either positive or negative but in unsigned binary number we forget about + or − sign.

↳ For Example, the smallest 8 bit binary number is $(0000\,0000)_2$ is $(00)_2$ and the largest 8 bit no. is $(1111\,1111)_2$.

↳ Therefore, the total range of unsigned 8 bit binary no. is from $(00)_H$ to $(FF)_H$ (H represent 16)

i.e. $(0)_{10}$ to $(255)_{10}$

$(FF)_H$ = $2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$1\ 1\ 1\ 1\ 1\ 1\ 1\ 1$

$= 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$

$= 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1$

$= 255$

↳ Signed Binary Numbers:

$+(89)_{10}$ $\xrightarrow{\text{8 bit Representation}}$ 0 1 0 1 1 0 0 1 (MSB)

Sign Bit

$-(89)_{10}$ 8 bit Representation 1 1 0 1 1 0 0 1

−ve sign

1 represents −ve

0 represents +ve

8 bit largest negative no: | 1 | 1111111 | $-(127)_{10}$

largest positive no.: | 0 | 1111111 | $+(127)_{10}$

→ If the data, has positive as well as negative number then the sign binary numbers must be used. The + sign and − sign are represented by 0 and 1 respectively.

→ The MSB of the binary no. here, is used to represent the sign and the remaining bits represents the no.

Range

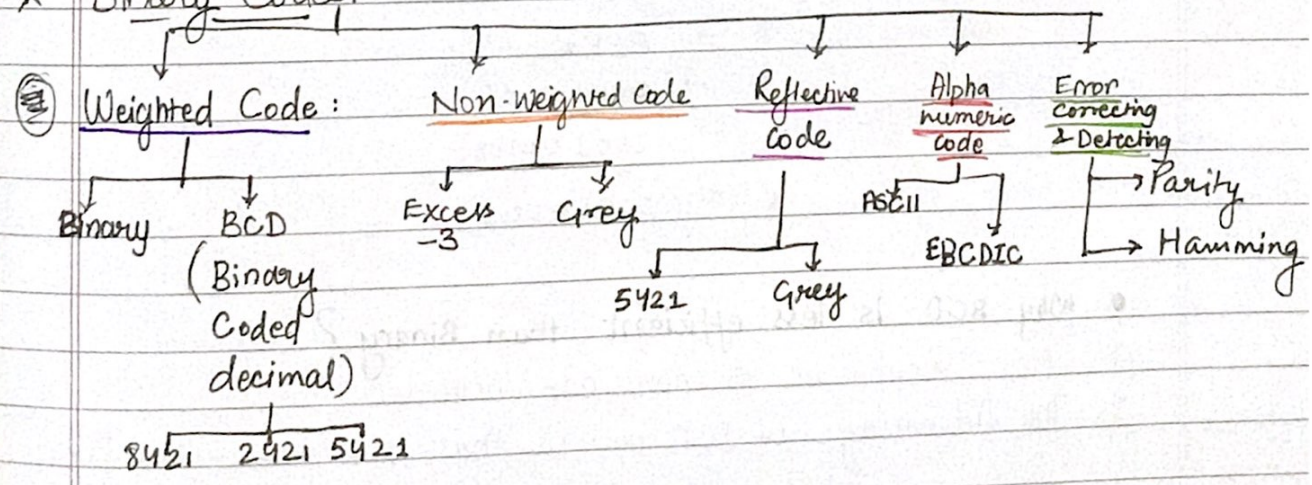
↳ Range: The signed magnitude numbers range from −127 to +127

★ Sign One's Complement :

$(-7)_{10}$

Sign Magnitude:	1 0000111
Signed 1's complement:	1 1111000
Signed 2's complement:	1 1111001

★ Binary Codes :-



(1) Weighted Code :

↳ In weighted codes, each position of the digit represents a specific weight. In 4 bit binary number each digit will have a weight of 8, 4, 2, 1 respectively.

A) Binary :

B) Binary Coded Decimal :

↳ In this code each decimal digit is represented by a 4 bit binary number the valid digits ranges from 0 to 9.

Decimal	BCD code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	X → No 4 bit exists, only 8 bit can
11	X represent this.
12	X → (00010010)
13	X → (00010011)
14	X → (00010100)
15	X → (00010101)

- Why BCD is less efficient than Binary ?

1. BCD Arithmetic is more complicated.
2. The Advantage of BCD no. is that the conversion from

decimal to BCD and vice-versa is quite simple.

3. BCD needs more bits (digit) to represent in comparison to binary.

- **BCD Arithmetic:**

Case I: Sum is equal to or less than 9 with carry 0

Ex:

Decimal		
2	0010	If Sum is
6	0110	
+8	1000 → $(+8)_{BCD}$	

Conclusion: If sum less than or equal to 9 with final carry equal to 0 then sum is in proper BCD form and requires no correction.

Case II: Sum is greater than 9 with carry 0

Ex:

Decimal		
7	0111	
+ 6	0110	
	1101	
	+ 0110	Add 6 for correction
13	0001 0011	
	1 3	
	→ VALID NO.	

22/09/22

Case III: Sum is ~~greater~~ less than 9 but with carry other than 0. i.e. 1

Decimal	BCD
9	1001
+ 8	+ 1000
17	10001
	+ 0110
	0001 0111

Add +6 for correction → 1 7

Q1. Add $(57)_{10}$ and $(26)_{10}$ using BCD arithmetic.

Sol.

Decimal	BCD
57	0101 0111
26	+ 0010 0110
83	0111 1101 → as this no. is considered invalid
	+ 0110
	1000 0011

considering carry as 0 ←

Add +6 for correction → 8 3

Q2. Add $(83)_{10}$ and $(34)_{10}$ using BCD arithmetic.

Sol.

Decimal	BCD
83	1000 0011
34	0011 0100
117	1011 0111
	+ 0 0110
	1011 1101

Adding +6 for correction

```
  1 1
  1011  0111
+ 0110  0000
---------------
  0001  0001  0111
    ↓     ↓     ↓
    1     1     7
```

Q3. Add $(569)_{10}$ and $(687)_{10}$ using BCD arithmetic.

Sol.

Decimal

```
  569
+ 687
------
 1256
```

```
  0101  0110 | 1001      → carry is 1
+ 0110  1000 | 0111
---------------------
  1011  1111   0000
+ 0110  0110   0110   ← Adding +6 for correction
---------------------
0001  0010  0101  0110
  ↓     ↓     ↓     ↓
  1     2     5     6
```

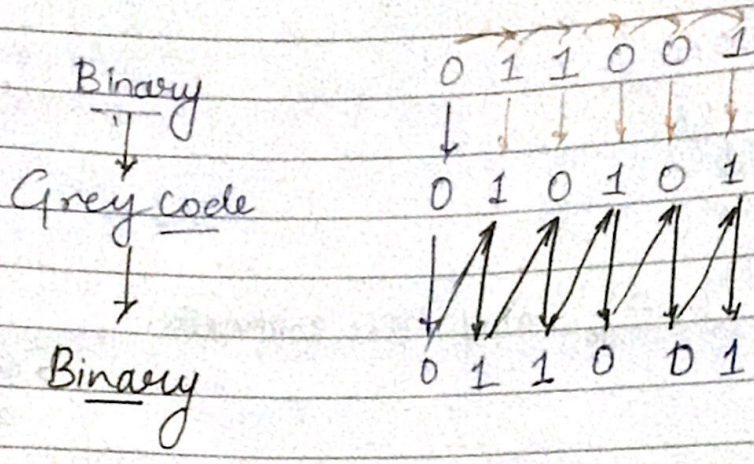
(2) Non-Weighted code: These codes are not assigned with any weight to each digit position.

Ex: Excess-3 and Grey code

A. Excess-3

Decimal	Excess-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

B) Grey code :- (1) Non-Weighted (2) Reflective code (3) X-OR implement



XOR
like → 0
unlike → 1

22/09/22

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

getting repeated as mirror image

→ generally change in one bit i.e. to get stable condition

We prefer grey code in digital system ∵ in quad there is only a single bit change in the grey code for successive binary numbers.

Mirror image:
$(0 \text{ to } 2^n - 1)$ mirroring $(2^n \text{ to } 2^{n+1} - 1)$

Grey Code is called Reflective code or Self-complementary code?

- Self Reacting / Self complementary

Ans. Grey codes are called self-reflecting code because the 2 LSBs from 4 to 7 to are the mirror image for those from 0 to 3 similarly, the 3 significant Bs from 8 to 15 are the mirror images from 0 to 7. In general, 2^n to 2^{n+1} are the mirror images of from 0 to $2^n - 1$

BOOLEAN ALGEBRA

★ Basic properties:

1) Commutative law: • $A + B = B + A$
   • $A \cdot B = B \cdot A$

2) Associative law: • $(A + B) + C = A + (B + C)$
   • $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

3) Distributive law: • $A \cdot (B + C) = A \cdot B + A \cdot C$

4) AND law: • $A \cdot 0 = 0$
   $A \cdot 1 = A$
   $A \cdot A = A$
   $A \cdot \overline{A} = 0$ * → false

5) OR law: • $A + 0 = A$
   $A + 1 = 1$
   $A + A = A$
   $A + \overline{A} = 1$ → true

6) $\overline{\overline{A}} = A$

7) Other laws:
   (1) $A + AB = A$
   (2) $A + \overline{A}B = A + B$
   (3) $(A + B)(A + C) = A + BC$

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

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

Shortcut
$X+YZ = (X+Y)(X+Z)$

(3) $(A+B)(A+C) = A+BC$
$= A.A + A.C + B.A + BC$
$= A + A.B + A.C + BC$
$= A(1+B) + AC + BC$
$= A.1 + AC + BC$
$= A(1+C) + BC$
$= A.1 + BC$
$= A + BC$

★ **Duality Theorem**: Theorem of boolean algebra

↳ **Duality Principle**: It states that in a two-valued boolean algebra, the dual of an algebraic expression can be obtained simply by interchanging AND(.) and OR(+) operators and by replacing ones(1's) by (0's) and zeros(0's) by ones(1's).

Ex: $A + BC \Rightarrow A.(B+C)$
$= A.B + A.C$
$= A$

Q. Find the dual of the following expression

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

Solving $= A.A + A.B$

$= A(1+B)$

Dual $= A$

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

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

$= 0 + AB$

$= AB$

3. $(A+B)(A+C) = A.(B+C)$

Date
Page

* De Morgan's Theorem: combination of two theorems

(1) $\overline{A\,B} = \overline{A} + \overline{B}$ NAND ≡ Bubble OR

(2) $\overline{A + B} = \overline{A} \cdot \overline{B}$ NOR ≡ Bubbled AND

Q1. Simplify: $Y = \overline{(\overline{AB} + \overline{A} + AB)}$ $= \overline{\overline{AB}} \cdot \overline{\overline{A}} \cdot \overline{AB}$

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

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

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

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

$= A \cdot 0 = 0$

30/09/22

* Canonical form: (standard form)

$F(A,B,C) = A\overline{B} + B\overline{C} + ABC$ // in simplified form

↳ SOP : $= A\overline{B}(C + \overline{C}) + (\overline{A} + A)B\overline{C} + ABC$

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

Q. Convert $A\overline{B}$ to POS form

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

$\cdot (A\overline{B} + C)(A\overline{B} + \overline{C})$

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

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

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

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

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

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

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

Minterm & Maxterm

0 → False 1 → True
1 → True 0 → False

Term	A	B	C	SOP Minterm (m)	POS Maxterm (M)
0	0	0	0	m_0 $\bar{A}\bar{B}\bar{C}$	M_0 $A+B+C$
1	0	0	1	m_1 $\bar{A}\bar{B}C$	M_1 $A+B+\bar{C}$
2	0	1	0	m_2 $\bar{A}B\bar{C}$	M_2 $A+\bar{B}+C$
3	0	1	1	m_3 $\bar{A}BC$	M_3 $A+\bar{B}+\bar{C}$
4	1	0	0	m_4 $A\bar{B}\bar{C}$	M_4 $\bar{A}+B+C$
5	1	0	1	m_5 $A\bar{B}C$	M_5 $\bar{A}+B+\bar{C}$
6	1	1	0	m_6 $AB\bar{C}$	M_6 $\bar{A}+\bar{B}+C$
7	1	1	1	m_7 ABC	M_7 $\bar{A}+\bar{B}+\bar{C}$

★ SOP and POS expression from Truth Table :

- Ex: HALF ADDER

A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Expression in SOP form:

  $S = \bar{A}B + \bar{B}A = A \oplus B \Rightarrow m_1 + m_2$

  $C = AB \Rightarrow m_3$

- POS Expression

  $S = (A+B)(\bar{A}+\bar{B}) \rightarrow M_0 \cdot M_3$

  $C = (A+B)(A+\bar{B})(\bar{A}+B) \Rightarrow M_0 \cdot M_1 \cdot M_2$

A	B	C	Y	Minterm	Maxterm
0	0	0	0		M_0
0	0	1	1	m_1	
0	1	0	0		M_2
0	1	1	0		M_3
1	0	0	1	m_4	
1	0	1	0		M_5
1	1	0	0		M_6
1	1	1	1	m_7	

Minterm 0 (1) → 0 → False 1 → True

SOP =

$F(A,B,C) \Rightarrow m_1 + m_4 + m_7$

$\Rightarrow \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$

$\Rightarrow \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC$

POS → Maxterm (0) → 1 → False 0 → True

$F(A,B,C) = M_0 M_2 M_3 M_5 M_6$

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

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

Q. Simplify the expression given below.

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

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

$Y = AB + B + AB$

$\boxed{Y = AB}$

~~$\Rightarrow AB \rightarrow$~~

3/10/22

★ Karnaugh Map (K-Map):

→ As boolean algebra is ~~limited~~ upto 6 variables.

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

Case I: Using 2 variable SOP 1 → True 0 → False. Minterm

$A \backslash B$	$\bar{B}$ (0)	B (1)
$\bar{A}$ (0)	$\bar{A}\bar{B}$ 0	$\bar{A}B$ 1
A (1)	$A\bar{B}$ 2	AB 3

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

	A	B
0 → 0	0	0
1 → 0	1	1
2 → 1	0	0
3 → 1	1	1

Case II: Using 3 Variable SOP

Horizontal Orientation:

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

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

A	B	C	
0	0	0	← 0
0	0	1	← 1
0	1	0	← 2
0	1	1	← 3
1	0	0	← 4
1	0	1	← 5
1	1	0	← 6
1	1	1	← 7

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

Vertical Orientation:

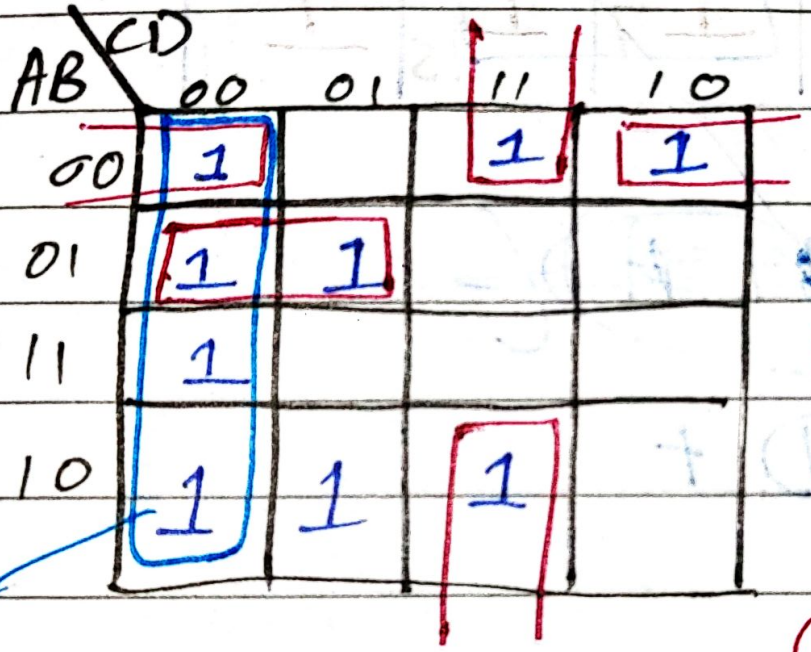
BC \ A	0	1
00	0	4
01	1	5
11	3	7
10	2	6

Case III Using 4 variables:

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

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

4th Oct '22



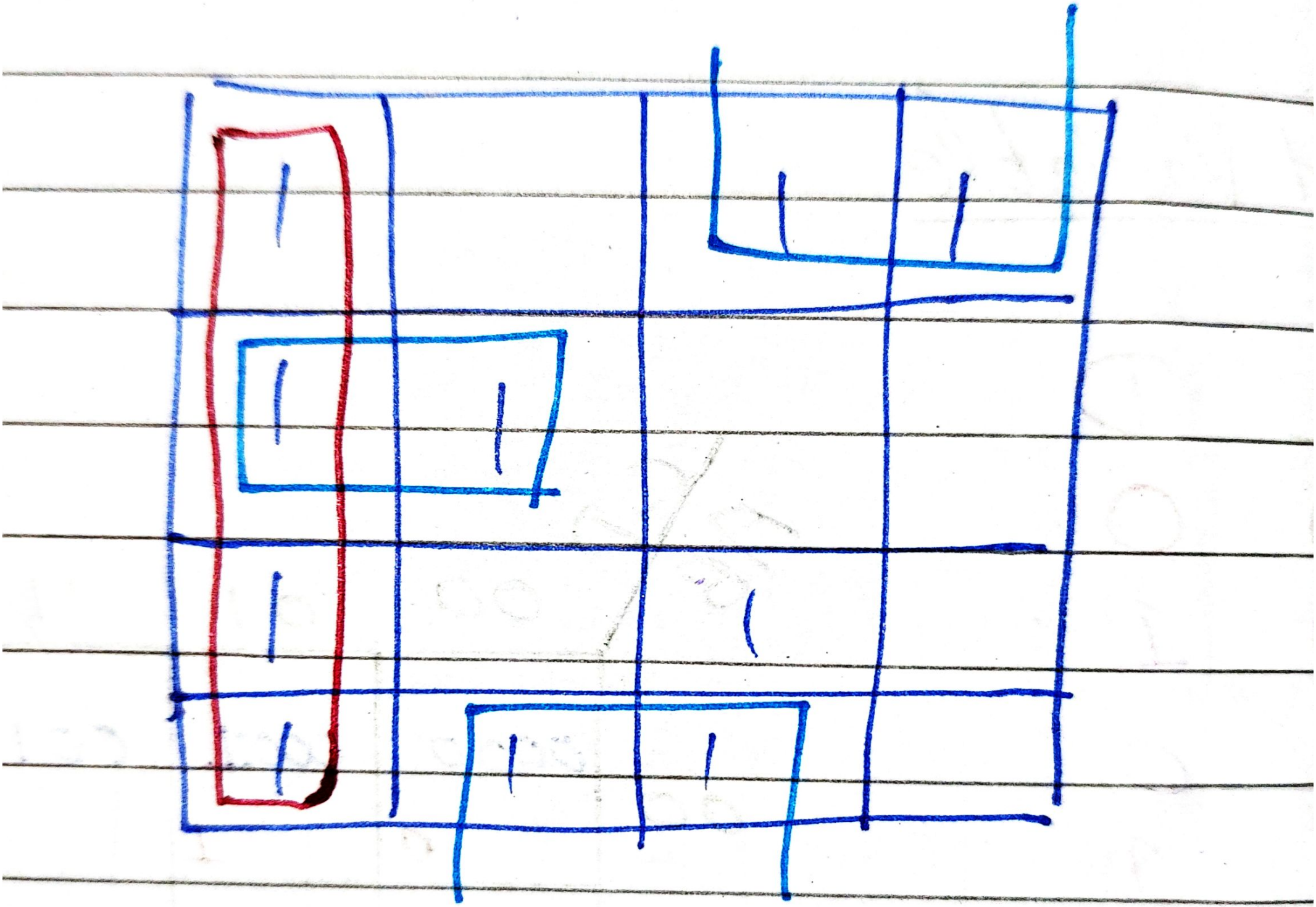
$f(A,B,C,D) = \sum m(0,2,3,4,5,8,7,11,12)$

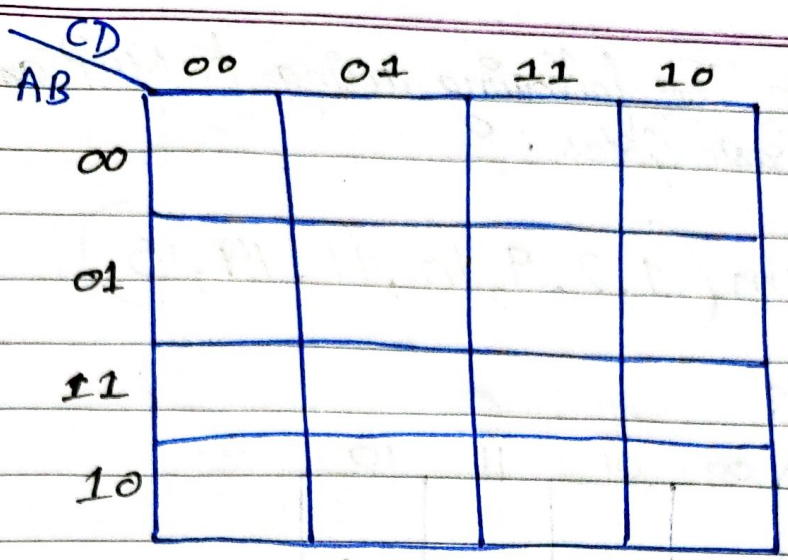
(1) Single - 1(1's)

(2) Pair - 2(1's)

(3) Quad - 4(1's)

(4) octate - 8(1's)



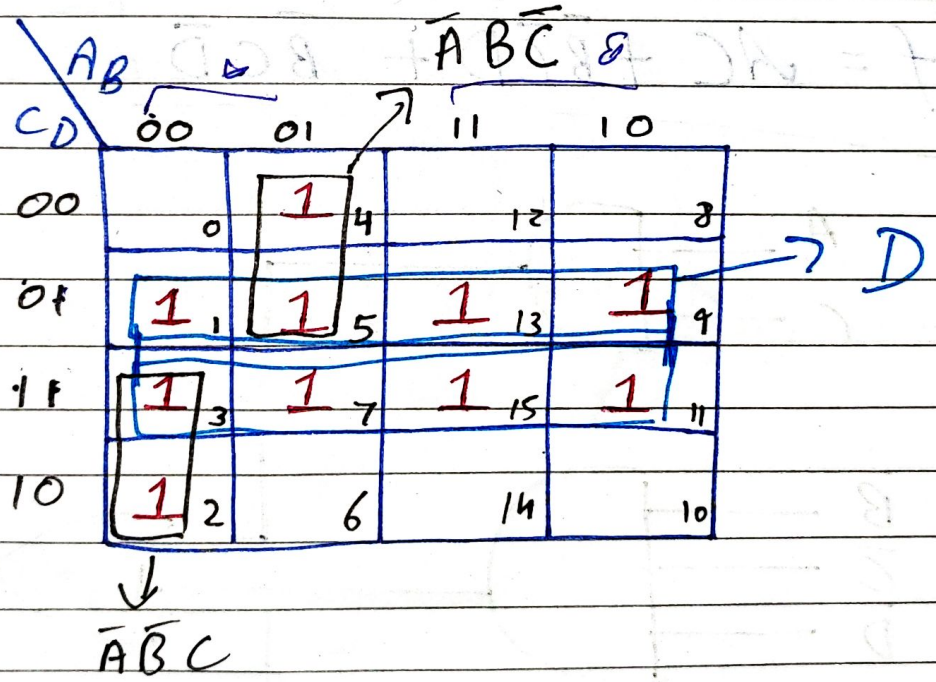


Realise.

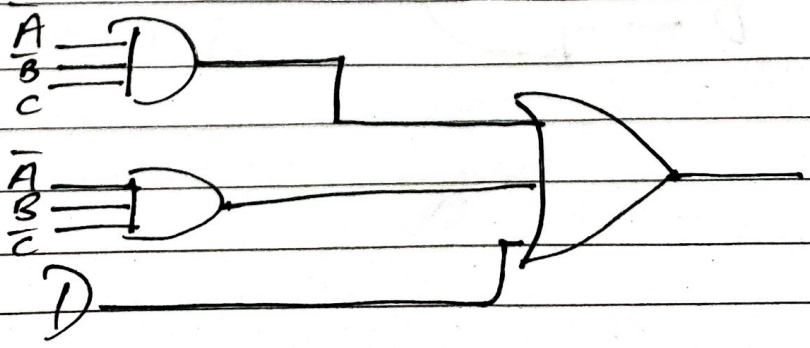
Q→ Using K-Map Reduce the following expression using Minimum no. of gates.

$$Y = \sum m(1, 2, 3, 4, 5, 7, 9, 11, 13, 15)$$

Ans)



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



Page No.
Date

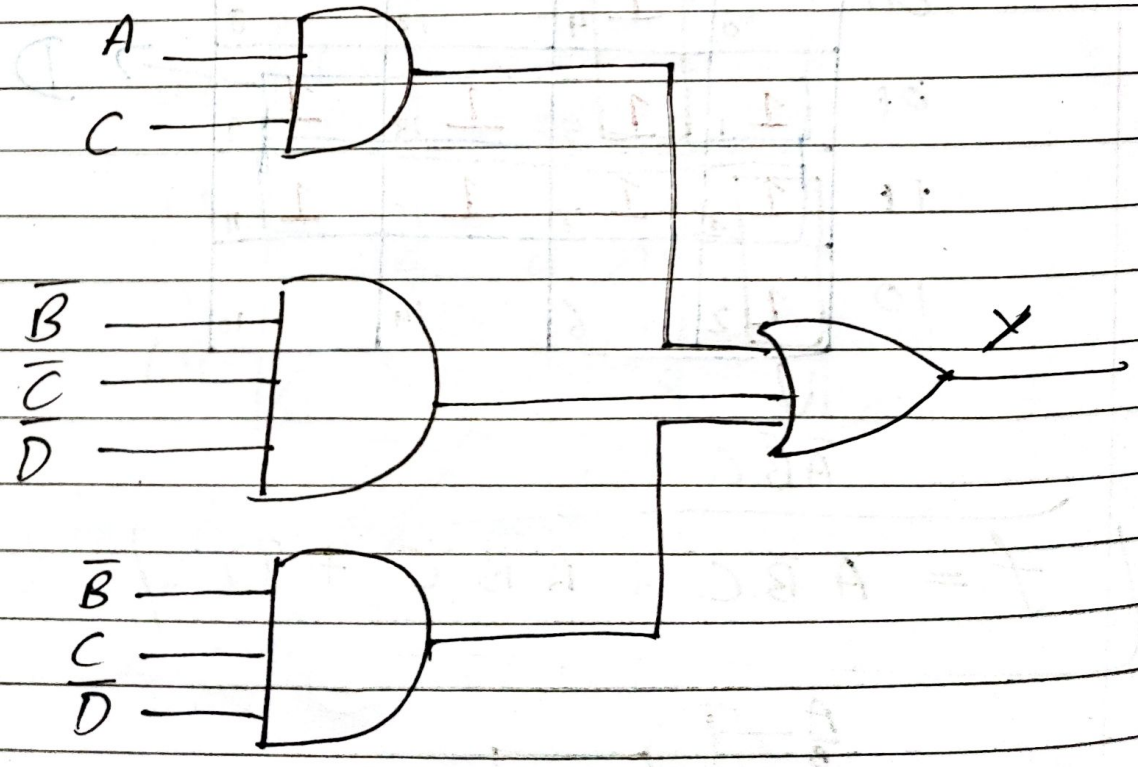
Ques Minimise the following using K-MAP and Realise using Basic Gates.

$$Y = \sum m(1,2,9,10,11,14,15)$$

Ans)

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

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



Page No.
Date

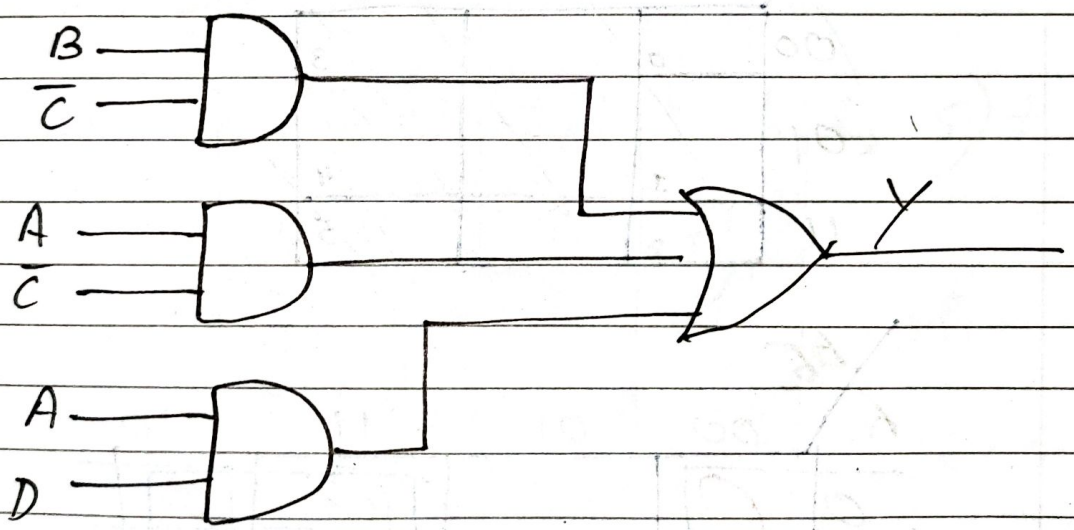
Ques→ Minimise The following exp. using K-MAP and Basic Gates.

$$Y = \sum m(4, 5, 8, 9, 11, 12, 13, 15)$$

Ans)

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

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

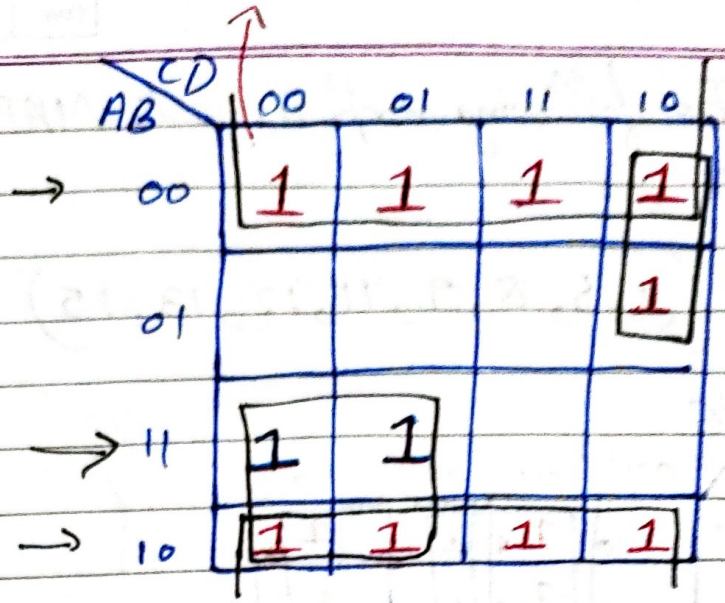


Q→ Minimise the logic expression:

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

$\bar{B}$

Ans)

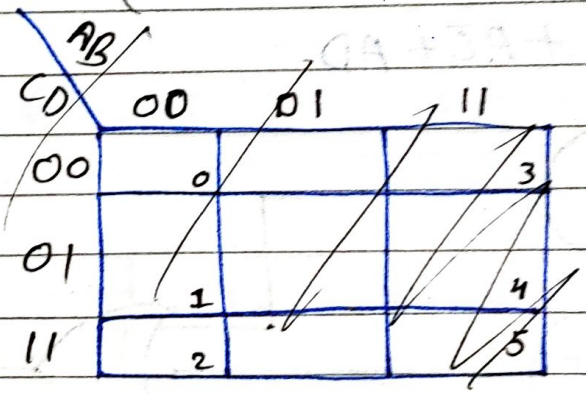


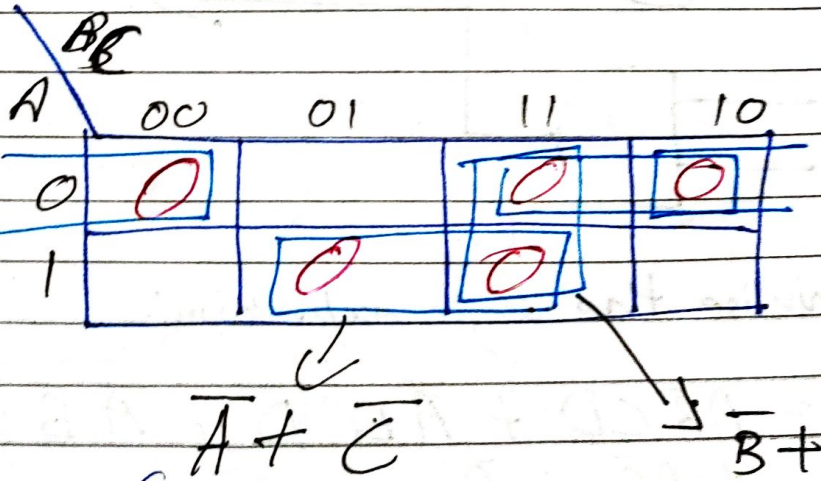
$= \bar{B} + A\bar{C} + \bar{A}B\bar{C}$

Q Simplify the following expression using K-Map

$Y = \Pi M(0, 2, 3, 5, 7)$

Ans)





$(A+C)$

$\bar{A} + \bar{C}$

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

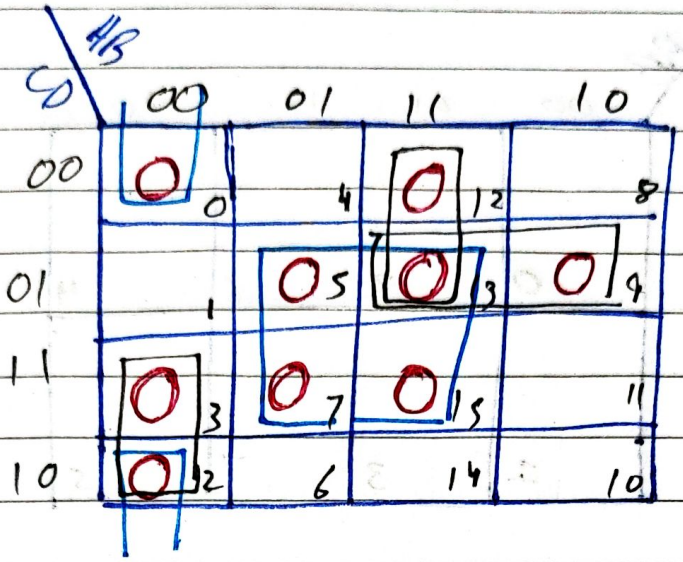
$Y = (A+C)(\bar{A}+\bar{C})(\bar{B}+\bar{C})$

Page No. | Date

Q→ Minimise The following expression :

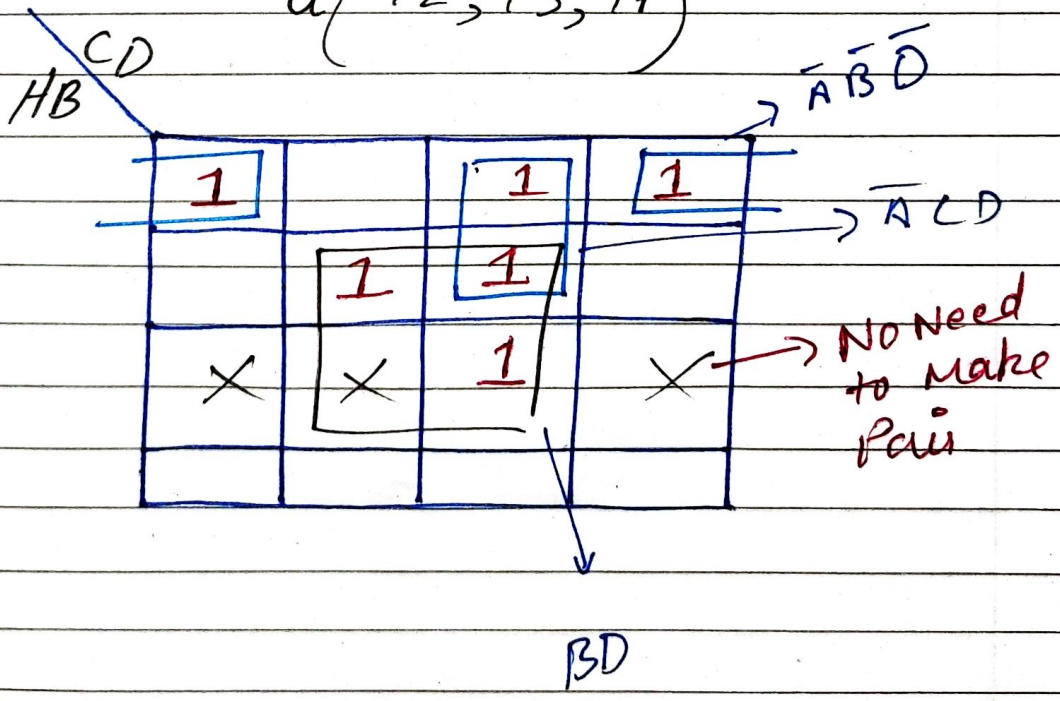
$$Y = \prod M(0,2,3,5,7,9,12,13,15)$$

Ans



Don't care → X or d

$$Y = \sum m(0,2,3,5,7,15) + d(12,13,14)$$

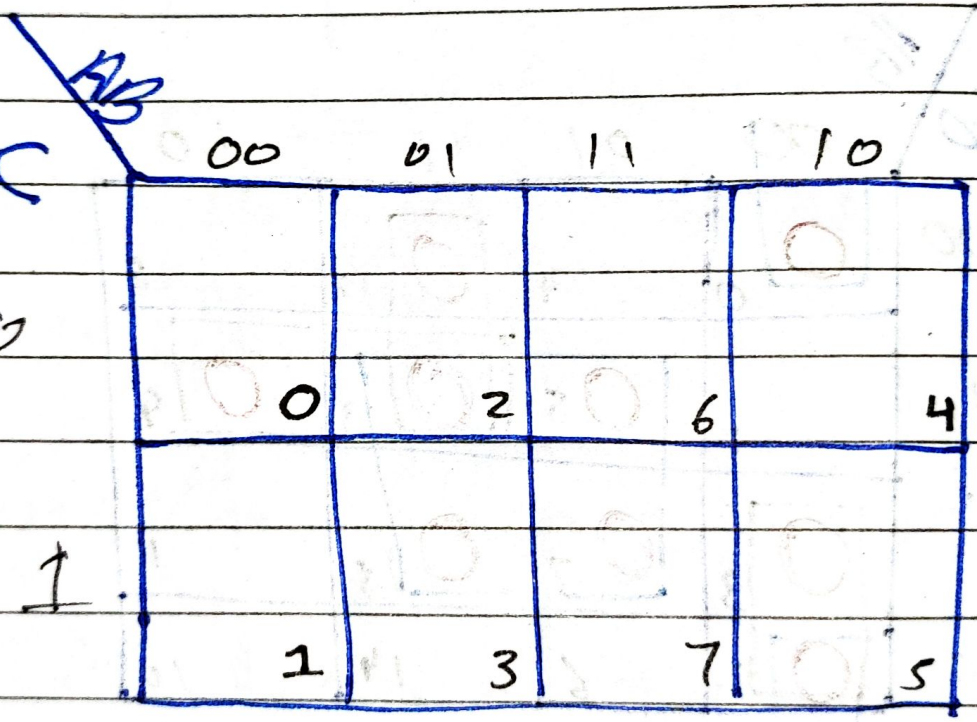


Page No.
Date

Ques) Simplify using Don't Care:

$$Y = \sum m(1, 3, 7, 11, 15) + d(0, 2, 5)$$

Ans)



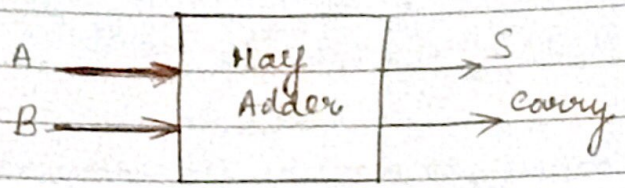
06/10/22 Thursday

Sequential - memory & logic gates
Combinational logic circuit

★ Combinational logic Circuit : 1) Adders
2) Subtractors
3) Code converters
4) Magnitude Comparator

★ Adders → Half adder
→ full adder

① Half Adder :



Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

A \ B	0	1
0		1
1	1	

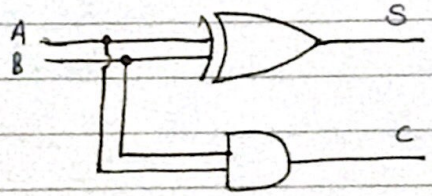
$S = A\bar{B} + \bar{A}B$

$= A \oplus B$

A \ B	0	1
0		
1		1

$C = AB$

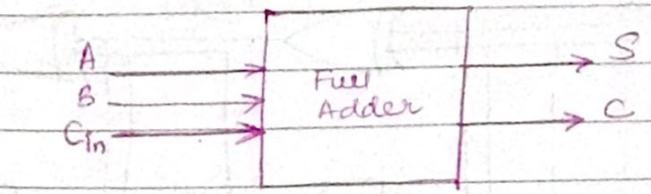
Logic

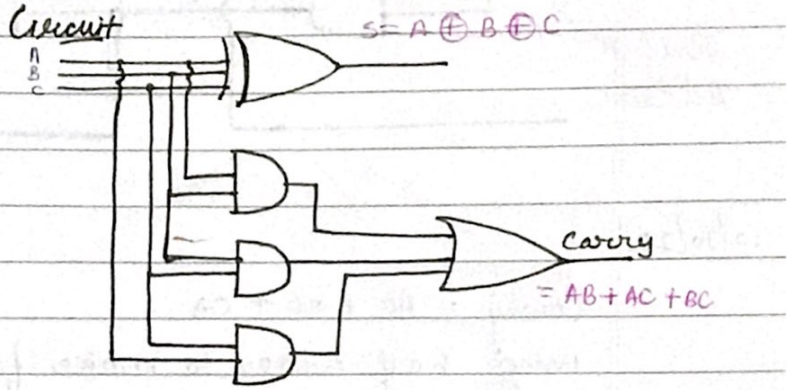


- Drawback of Half Adder : A half adder can add only 2 inputs. ~~However Why we~~ while we add 2 nos. the result may consist Carry. This carry needs to be added into successive bits. This can only be done with the help of full adder.

☆ Full Adder:

Inputs			Outputs	
A	B	C_{in}	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





Sum

A \ BC	00	01	11	10
0	0	1	3	1 (2)
1	1 (4)	5	1 (7)	6

→ Sum

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

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

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

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

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

$S = C \oplus A \oplus B$

[using $\bar{A}\bar{B} + AB = A \odot B$ &

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

using $\overline{A \odot B} = A \oplus B$

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

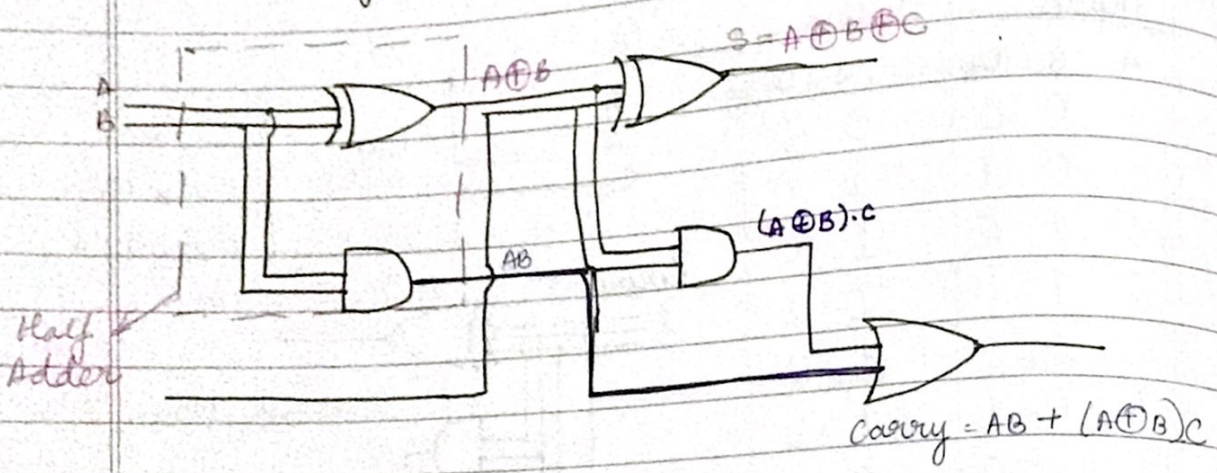
Carry

A \ BC	00	01	11	10
0			1	→ BC
1		1	1	1 → AB

AC

$C = AC + AB + BC$

★ Full adder using Half adder:-



10/10/22

Carry = AB + BC + CA

using half adder to make full adder

we get carry = $AB + (A \oplus B)C$

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

$= A\bar{B}C + \bar{A}BC + AB(1+C)$ [1+C=1]

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

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

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

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

$= AB + AC(B+\bar{B}) + BC$

Carry = AB + AC + BC (Hence Proved)

★ Subtractors:-

0 - 0 = 0

1 - 0 = 1

0 - 1 = 1 with borrow

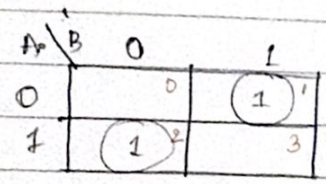
1 - 1 = 0

* Half - Subtractor:

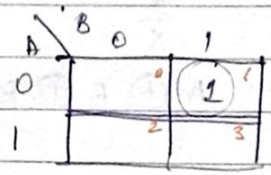
→ Truth Table:

	A	B	D	Borrow
0	0	0	0	0
1	0	1	1	1
2	1	0	1	0
3	1	1	0	0

→ Difference



→ Borrow

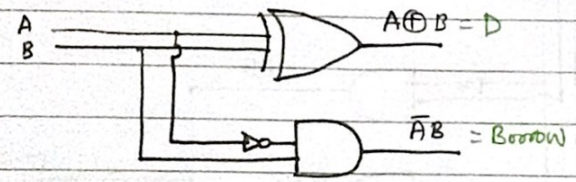


$D = A\bar{B} + \bar{A}B$

$= A \oplus B$

$Borrow = \bar{A}B$

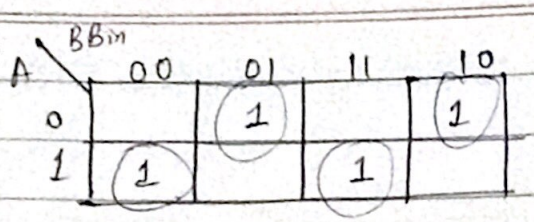
Logic: Diagram

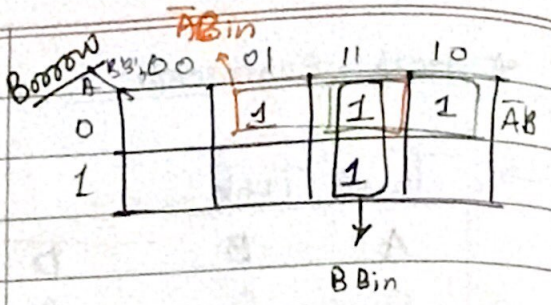


* Full Subtractor:

	A	B	B_{in}	D	Borrow
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

Difference





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

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

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

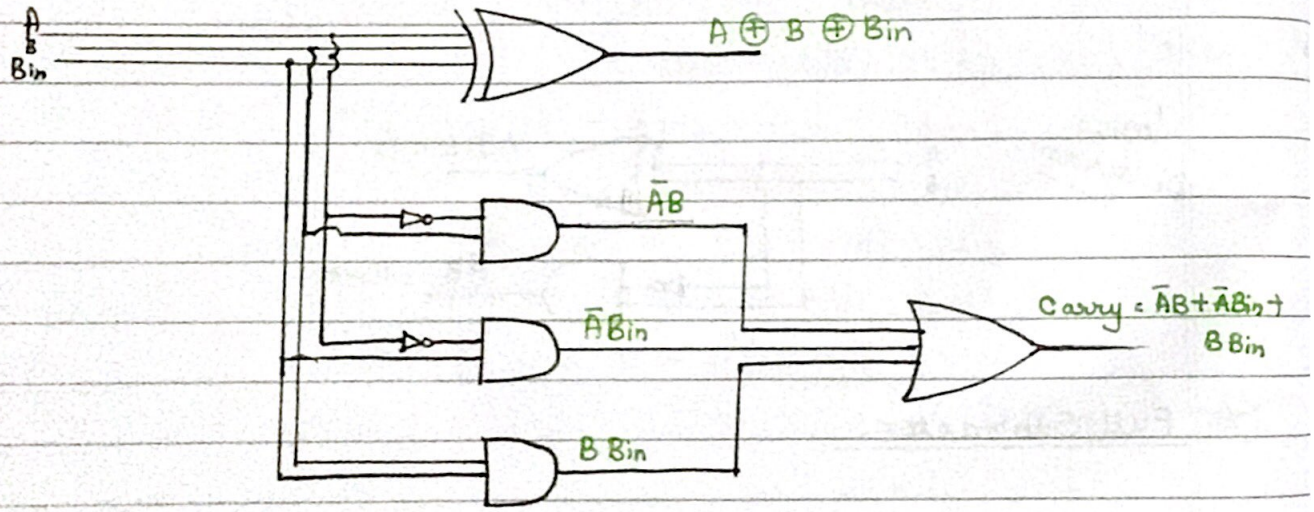
$= \overline{(A \oplus B)}\, B_{in} + (A \oplus B)\bar{B}_{in}$

($\overline{(A \oplus B)}$ → X, $(A \oplus B)$ → X)

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

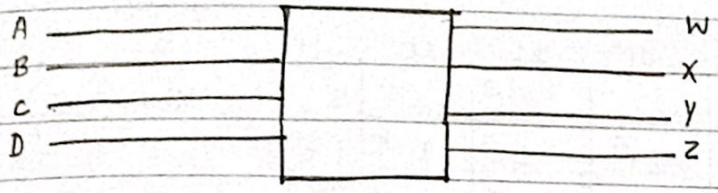
$Borrow = \bar{A}B_{in} + \bar{A}B + BB_{in}$

Logic design:



11/10/22

Code Converter:



A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

K-Map:

- For W

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

W = A

- For X

AB \ CD	00	01	11	10
00	0	1	3	2
01	1 4	1 5	1 7	1 6 → $\bar{A}B$
11	12	13	15	14
10	1 8	1 9	1 11	1 10 → $A\bar{B}$

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

- For Y

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

$\bar{A}BC$ ↑ ; $B\bar{C}$ ↓ ; $A\bar{B}C$ ↓

$Y = B\bar{C} + \bar{B}C$
$= B \oplus C$

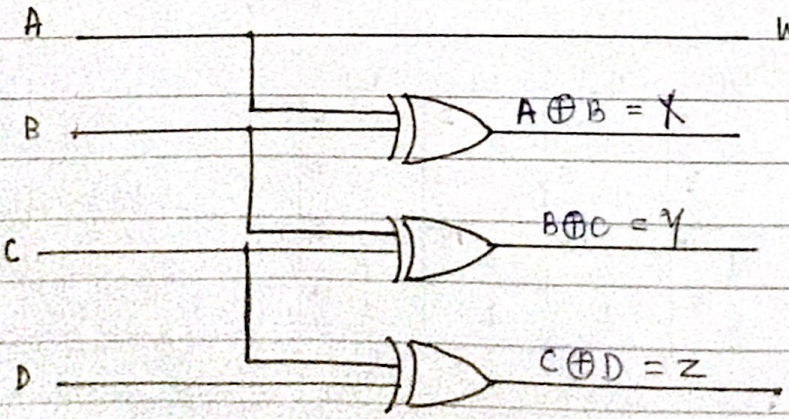
- For Z

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

$\bar{C}D$ ↓ ; CD ↓

$Z = \bar{C}D + C\bar{D}$
$= C \oplus D$

Logic design



★ Magnitude Comparator : (1 - bit)

A	B	X = A < B	Y = A = B	Z = A > B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

For X

A \ B	0	1
0		1
1		

For Y

A \ B	0	1
0	1	
1		1

For Z

A \ B	0	1
0		
1	1	

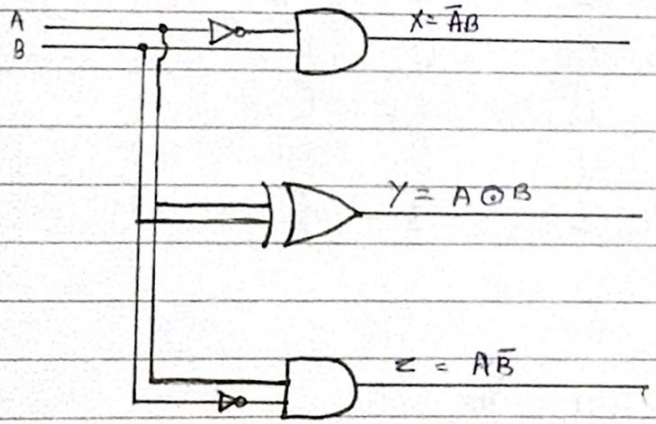
$X = \bar{A}B$

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

$= A \odot B$

$Z = A\bar{B}$

Logic Design



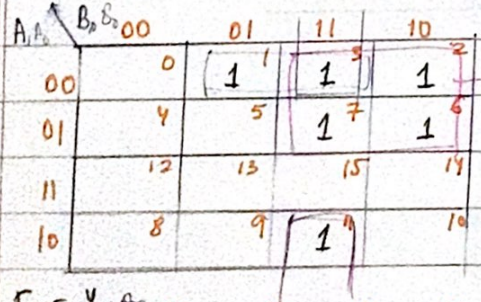
13/10/22

★ 2-bit Magnitude Comparator:-

$A \rightarrow A_1A_0$ $B \rightarrow B_1B_0$

	A		B		X	Y	Z
	A_1	A_0	B_1	B_0	$A < B$	$A = B$	$A > B$
0	0	0	0	0	0	1	0
1	0	0	0	1	1	0	0
2	0	0	1	0	1	0	0
3	0	0	1	1	1	0	0
4	0	1	0	0	0	0	1
5	0	1	0	1	0	1	0
6	0	1	1	0	1	0	0
7	0	1	1	1	1	0	0
8	1	0	0	0	0	0	1
9	1	0	0	1	0	0	1
10	1	0	1	0	0	1	0
11	1	0	1	1	01	0	0
12	1	1	0	0	0	0	1
13	1	1	0	1	0	0	1
14	1	1	1	0	0	0	1
15	1	1	1	1	0	1	0

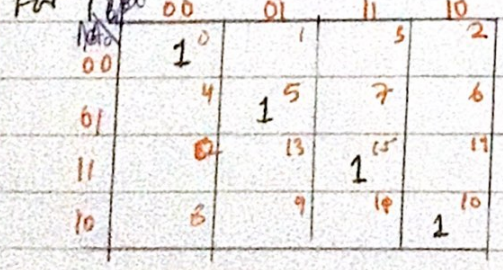
★ For X



$\bar{A}_1B_1 + \bar{A}_1\bar{A}_0B_0 + \bar{A}_0B_1B_0 = X$

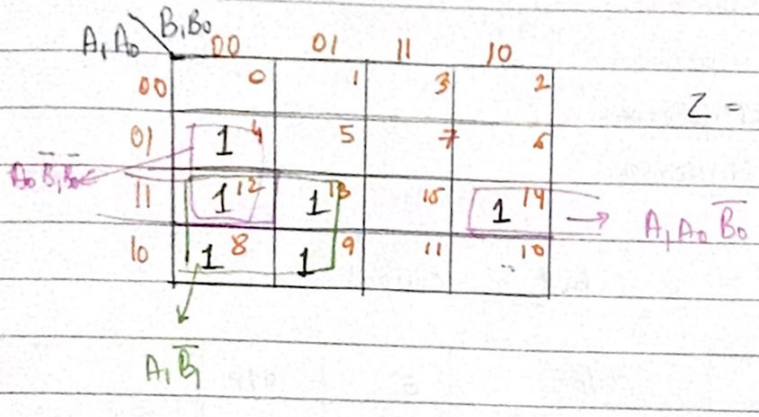
$\bar{A}C + \bar{A}\bar{B}D + \bar{B}CD = X$

★ For Y



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

★ For Z



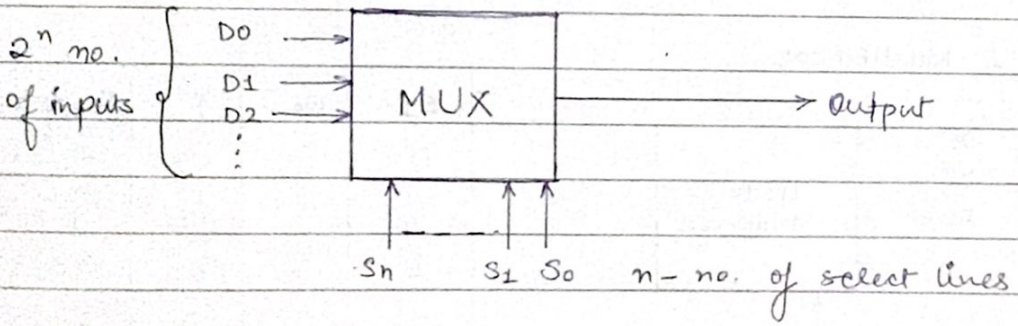
$Z = A_1\bar{B_1} + A_1A_0\bar{B_0} + A_0\bar{B_1}\bar{B_0}$

14/10/22

★ Multiplexer / Data Selector:-

Multiplexer is combination service which select one of many data inputs and give single output. The selection of one of input is done by select lines. If these are small n select lines then 2^n inputs can be selected.

↳ Advantages of Multiplexer:

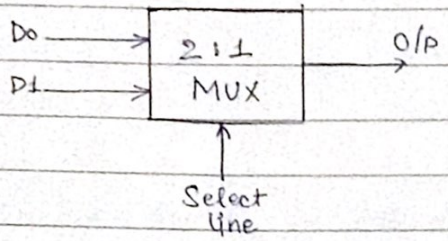


1) It reduces the no. of wires
2) It reduces the circuit complexity and cost
3) We can implement many complex circuit using NAND gate
4) It simplify the logic design
5) It doesn't need K-Maps and simplification

↳ Classification of multiplexer

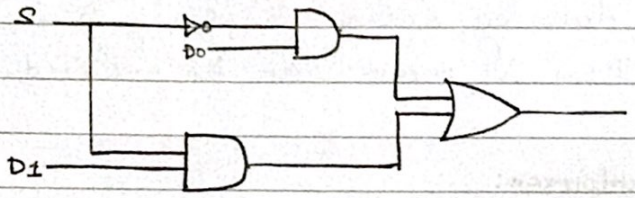
1) 2:1 multiplexer
2) 4:1 multiplexer
3) 8:1 multiplexer
4) 16:1 multiplexer

a) 2:1 multiplexer ⇒ 2 input – 1 output

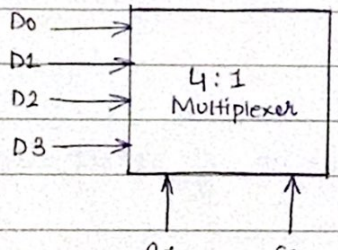


S	O/P	
0	D0	$\bar{S}D_0$
1	D1	SD_1

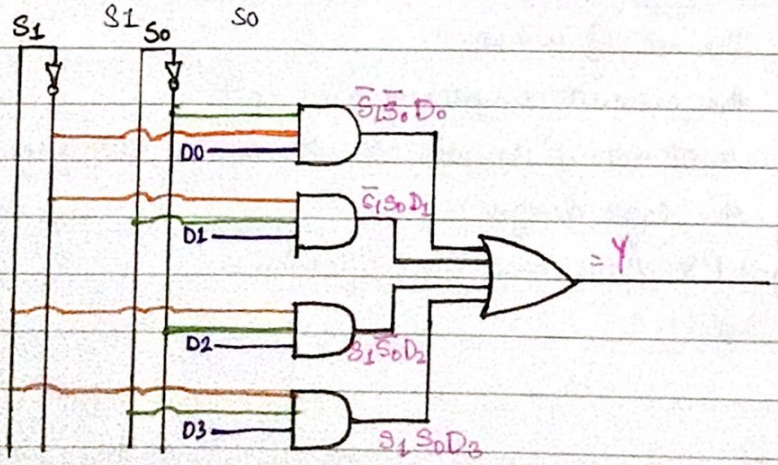
Logic design



b) 4:1 Multiplexer



S_1	S_0	Y	Expression
0	0	D_0	$\bar{S}_1\bar{S}_0D_0$
0	1	D_1	$\bar{S}_1S_0D_1$
1	0	D_2	$S_1\bar{S}_0D_2$
1	1	D_3	$S_1S_0D_3$



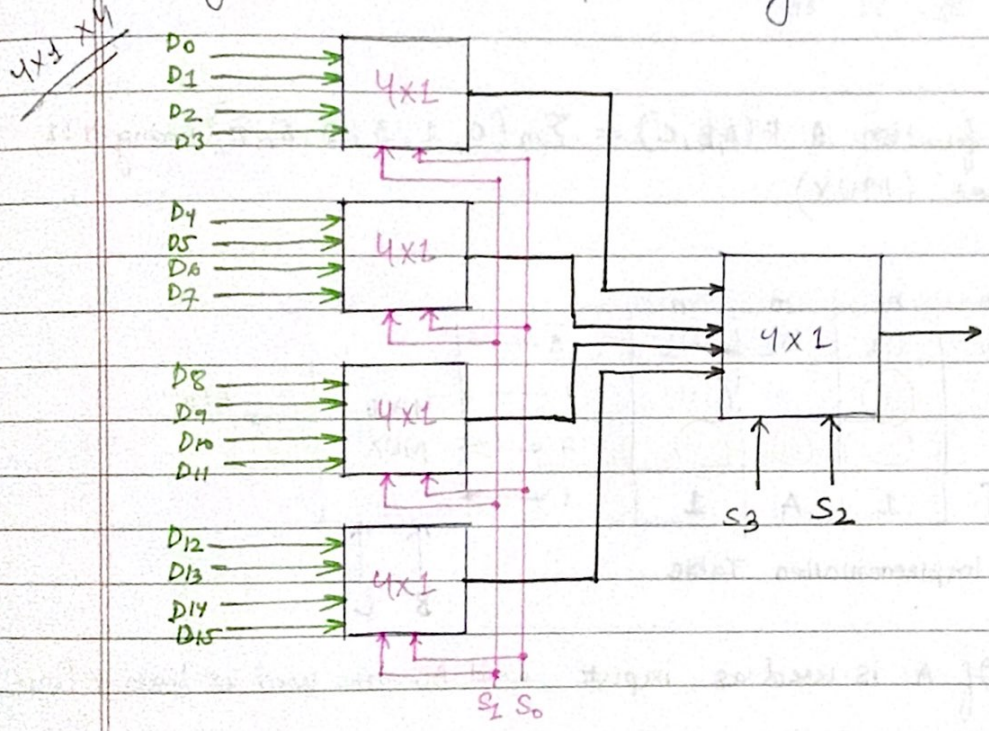
17/10/22

★ Applications of Multiplexer :-

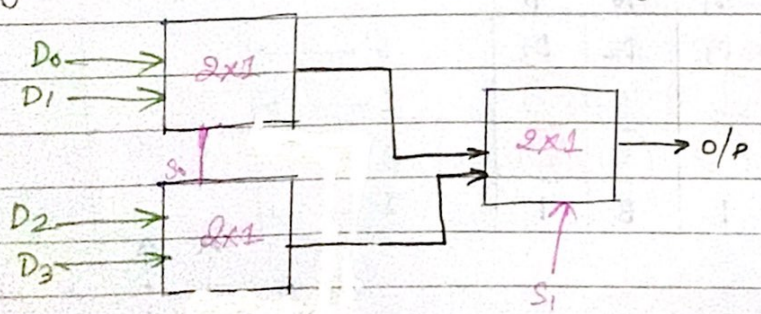
1. D/A (Digit to Analog converter)
2. As a Data Selector
3. In designing combinational circuits
4. Parallel to Serial Converter
5. Data Acquistion System

★ Multiplexer Trees :

Q. Design a 16:1 Multiplexer using 4:1 multiplexer

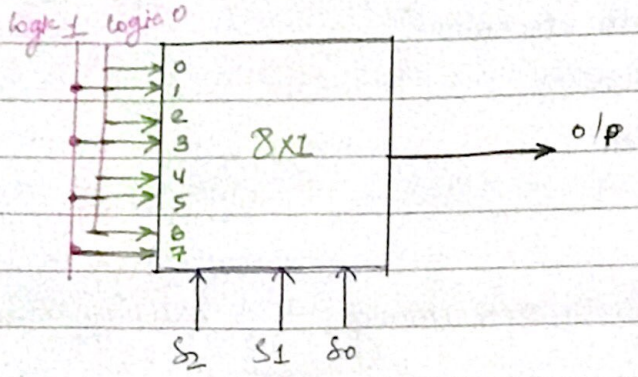


Q. Design a 4:1 Multiplexer using 2:1 multiplexer



★ Boolean function implementation using Multiplexer

Q1. Implement $F(A,B,C) = \sum m(1,3,5,7)$ using 8:1 Multiplexer



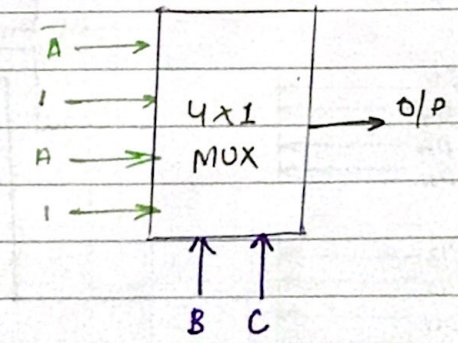
Q2. Implement function $F(A,B,C) = \sum m(0,1,3,5,6,7)$ using 4:1 Multiplexer. (MUX)

Sol.

Case I

	00	01	10	11
	D_0	D_1	D_2	D_3
$\bar{A}$ 0	0	1	2	3
A 1	4	5	6	7
Sol.	$\bar{A}$	1	A	1

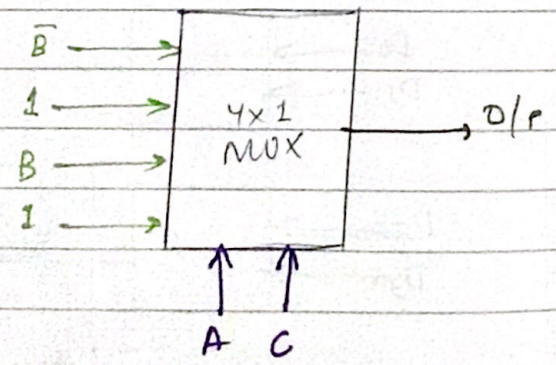
Implementation Table



Case I: If A is used as input and BC are used as Select lines

Case II: If B is used as input and AC are used as Select lines

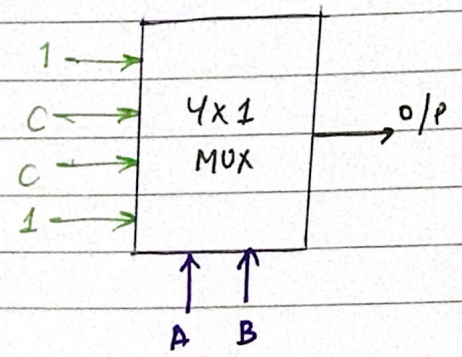
	00	01	10	11
	D_0	D_1	D_2	D_3
$\bar{B}$ 0	0	1	4	5
B 1	2	3	6	7
	$\bar{B}$	1	B	1



Case III:

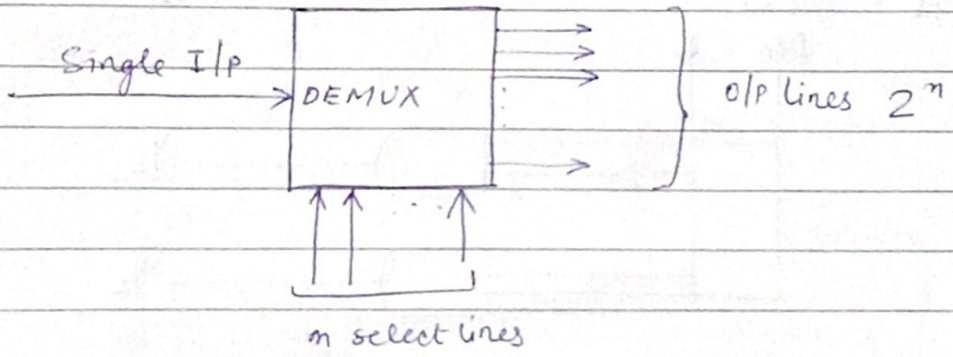
If C is used as input and AB are used as Select lines.

		00	01	10	11
		D0	D1	D2	D3
$\bar{C}$	0	0	2	4	6
C	1	1	3	5	7
		1	C	C	1



20/10/22

★ DEMULTIPLEXER:



→ A demultiplexer has a single input, multiple output i.e. 2^n and n select lines. A demultiplexer performs the reverse operation of multiplexer.

- Classification of Demultiplexer:

1) 1 X 2 DEMUX
2) 1 X 4 DEMUX
3) 1 X 8 DEMUX
4) 1 X 16 DEMUX

a) 1 X 2 DEMUX:

S (select line)	Y_1	Y_2
0	D_{in}	X
1	X	D_{in}

Din	S	Y_1	Y_2
0	0	0	0
1	0	1	0
0	1	0	0
1	1	0	1

K-Map Y_1

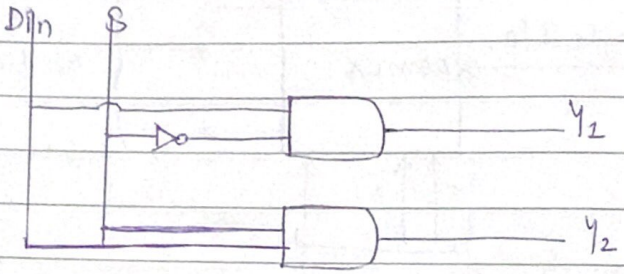
Din \ S	0	1
0		
1	1	

$Y_1 = Din\,\bar{S}$

K-Map Y_2

Din \ S	0	1
0		
1		1

$Y_2 = Din\,S$



b) 1 X 4 DEMUX:

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

CLASSMATE
Date
Page

K-Map For Y_0

D_{in} \ S_1S_0	00	01	11	10
0				
1	1			

$$Y_0 = D_{in}\,\bar{S}_1\,\bar{S}_0$$

K-Map For Y_1

D_{in} \ S_1S_0	00	01	11	10
0				
1		1		

$$Y_1 = D_{in}\,\bar{S}_1\,S_0$$

K-Map For Y_2

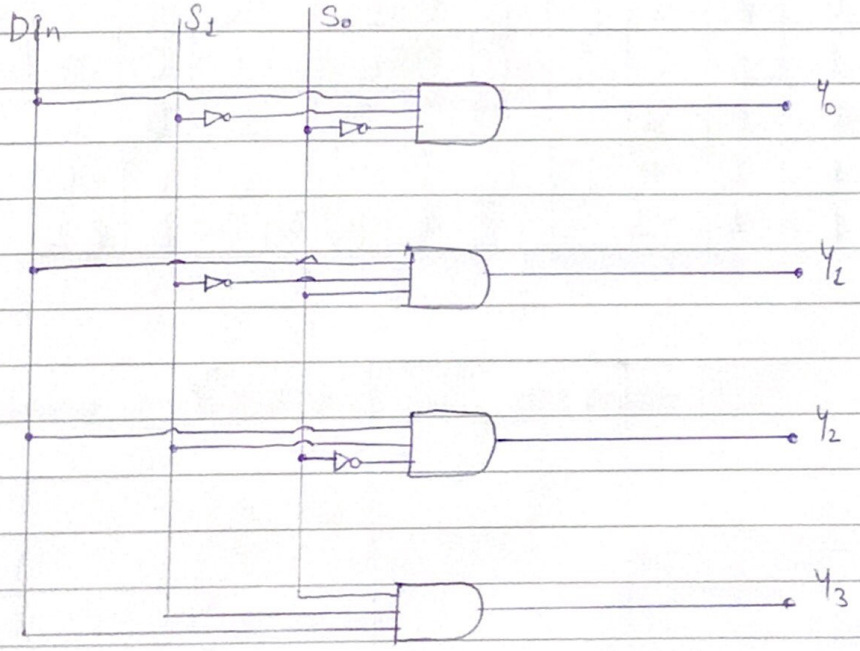
D_{in} \ S_1S_0	00	01	11	10
0				
1				1

$$Y_2 = D_{in}\,S_1\,\bar{S}_0$$

K-Map For Y_3

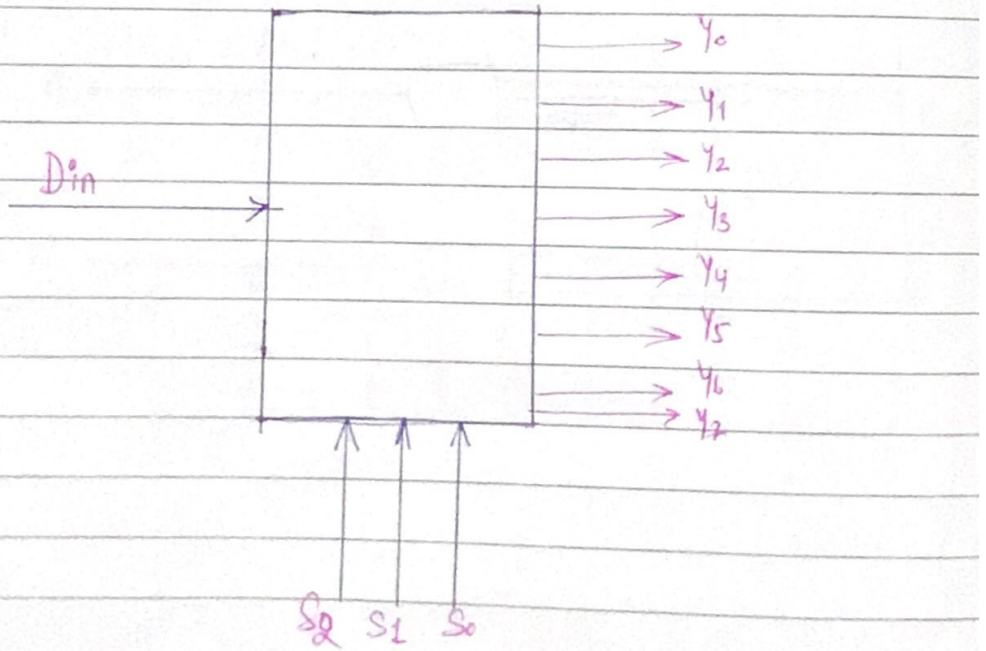
D_{in} \ S_1S_0	00	01	11	10
0				
1			1	

$$Y_3 = D_{in}\,S_1\,S_0$$



c) 1X8 DEMUX

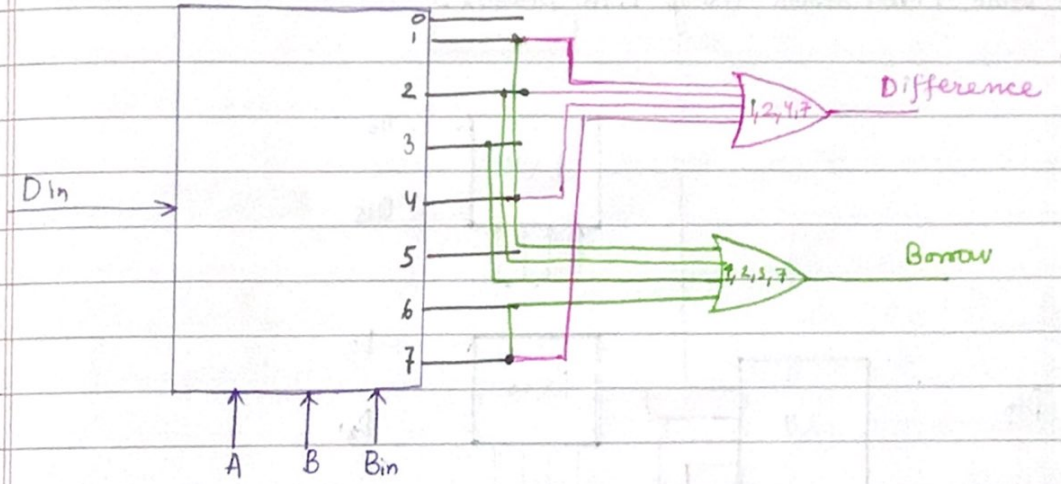
Din	S_0	S_1	S_2	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1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



27/10/22

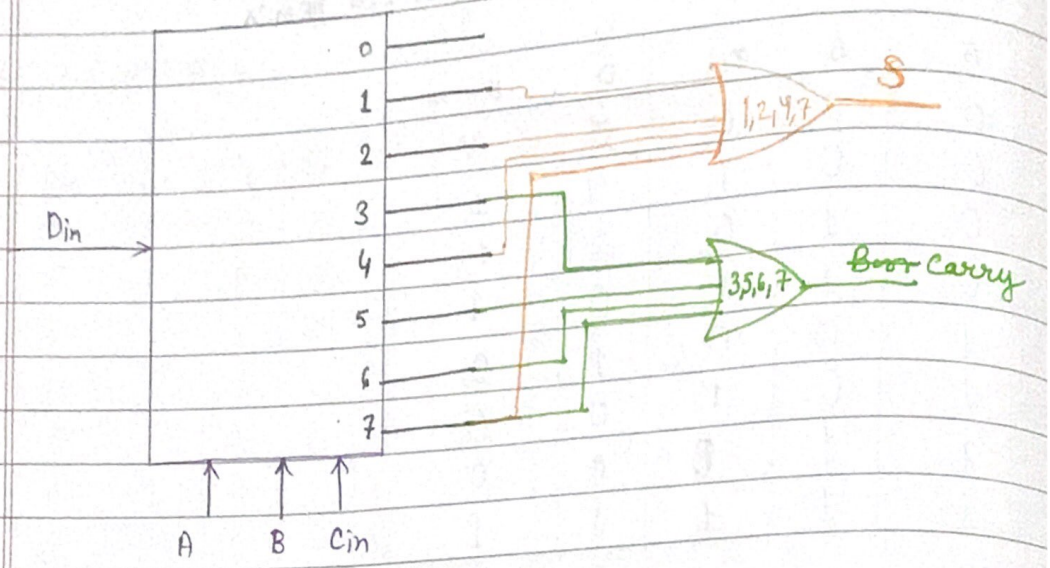
Q. Implement the Full Subtractor using a 1:8 DEMUX

A	B	Bin	D	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



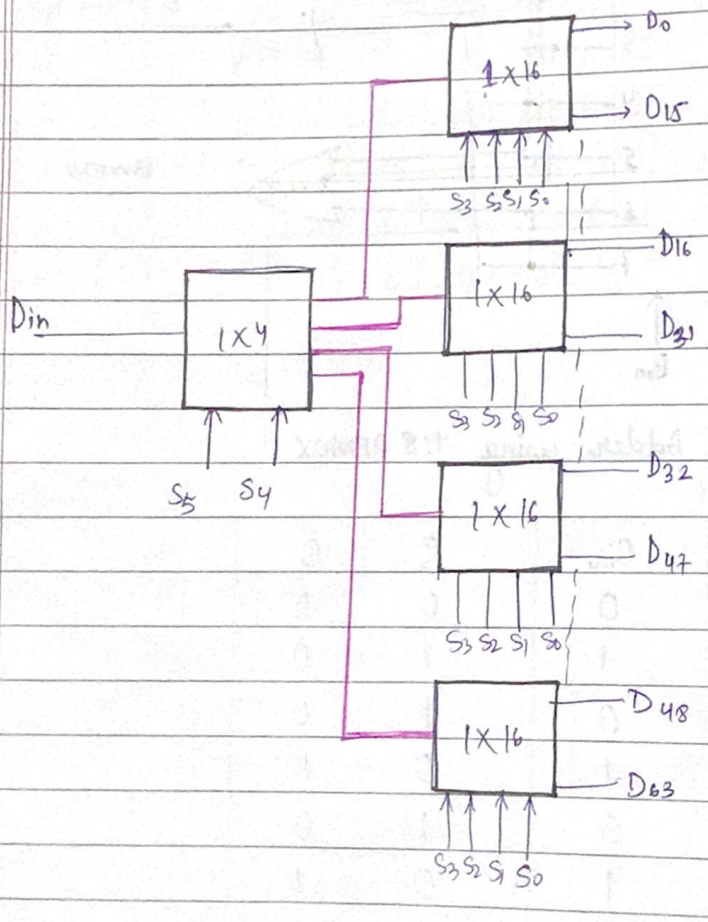
Q2. Implement a Full Adder using 1:8 DEMUX

A	B	Cin	S	C
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

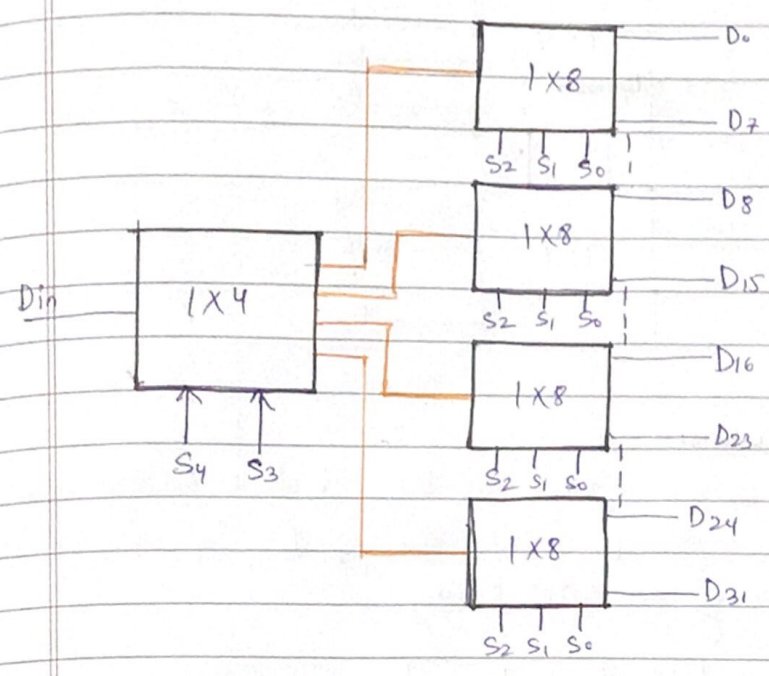


Q3. Draw 1:64 DEMUX Using 1:16 DEMUX.

Sol.



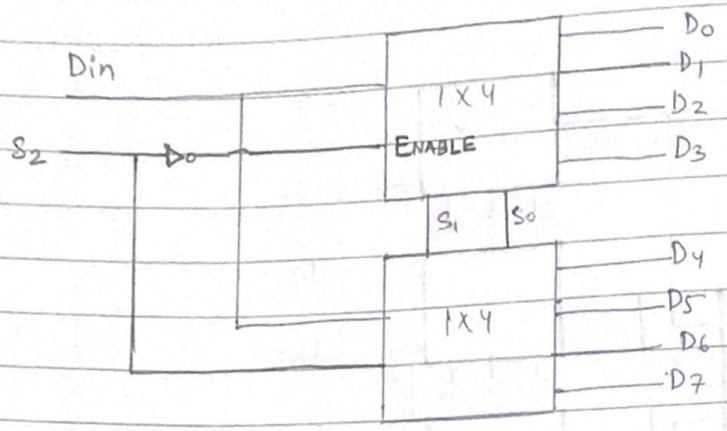
Q4. Draw 1X32 DEMUX using 1:8 DEMUX



Q5. Obtain 1X8 DEMUX using 2, 1:4 DEMUX

Sol:

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



28/10/22

* Parity generator / checker:-

→ A parity bit is an additional bit which is added to a binary word in order to make no. of 1's in new format with EVEN parity or ODD parity.

Ex: Message → | 1 | 0 | 1 | 0 | 1 | 1 |

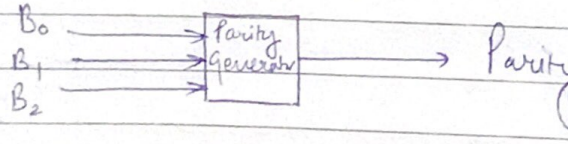
ODD → | 1 | 1 | 0 | 1 | 0 | 1 | 1 |

EVEN → | 0 | 1 | 0 | 1 | 0 | 1 | 1 |

→ Parity bits are added in order to detect any error occurs in the process of transmission, the data with parity bit are transmitted by the transmitter. At the receiver this data is checked for errors.

• Parity Generator:

→ A parity generator is a device which generates parity bits for even or odd parity.

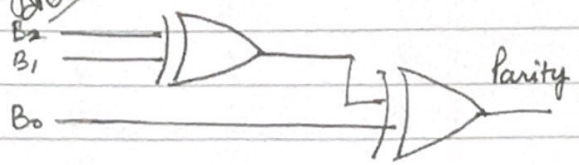


Q. Design a parity generator using basic gates to generate message with odd parity. Assume the input to be 3 bit binary words.

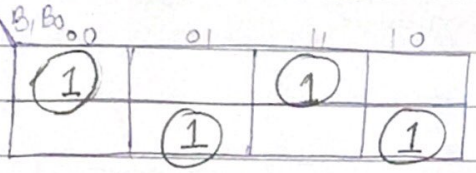
Sol.

B_2	B_1	B_0	ODD (Parity)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Logic diagram



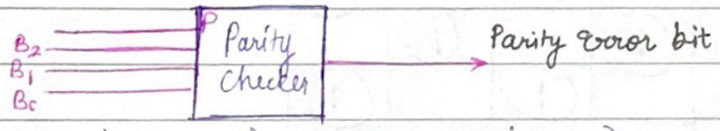
Karnaugh map



ODD Parity $= B_2 \oplus B_1 \oplus B_0$

- **Parity Checker:**

→ A parity checker is used at the receiver side it receives the data at the parity bit and compare it to check the error.



PEB → 0 (No error) PEB → 1 (error)

P	B_2	B_1	B_0	Parity (DDD)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

PB_2 \ B_1B_0	00	01	11	10
00	(1)		(1)	
01		(1)		(1)
11	(1)		(1)	
10		(1)		(1)

$$\text{Parity} = \bar{P}\bar{B}_2\bar{B}_1\bar{B}_0 + \bar{P}\bar{B}_2B_1B_0 + \bar{P}B_2\bar{B}_1B_0 + \bar{P}B_2B_1\bar{B}_0 + PB_2\bar{B}_1\bar{B}_0 + PB_2B_1B_0 + P\bar{B}_2\bar{B}_1B_0 + P\bar{B}_2B_1\bar{B}_0$$

$$= \bar{B}_1\bar{B}_0\left(\bar{P}\bar{B}_2 + PB_2\right) + B_1B_0\left(\bar{P}\bar{B}_2 + PB_2\right) + \bar{B}_1B_0\left(\bar{P}B_2 + P\bar{B}_2\right) + B_1\bar{B}_0\left(\bar{P}B_2 + P\bar{B}_2\right)$$

$$= \left(\bar{P}\bar{B}_2 + PB_2\right)\left(\bar{B}_1\bar{B}_0 + B_1B_0\right) + \left(\bar{P}B_2 + P\bar{B}_2\right)\left(\bar{B}_1B_0 + B_1\bar{B}_0\right)$$

$= (P \odot B_2)(B_1 \odot B_0) + (P \oplus B_2)(B_1 \oplus B_0)$

(1) $= \overline{(P \oplus B_2)}\,\overline{(B_1 \oplus B_0)} + (P \oplus B_2)(B_1 \oplus B_0) \rightarrow \bar{X}\bar{Y} + XY$

consider

$X = P \oplus B_2$

$Y = B_1 \oplus B_0$

therefore eq (1) becomes

$\Rightarrow \bar{X}\bar{Y} + XY$

$\Rightarrow X \odot Y$

$\Rightarrow \overline{X \oplus Y}$

$\Rightarrow \overline{(P \oplus B_2) \oplus (B_1 \oplus B_0)}$

logic design