MODULE - I

A) Binary, Octal, Hexadecimal to Decimal:-

$$(X)_{10} = (d_{n-1}\ d_{n-2} \ldots d_0 \bullet d_{-1}\ d_{-2}\ d_{-3} \ldots d_{-m})_b$$

integral part — fractional part

Example:-

$$\Rightarrow (101.101)_2 = 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$$

$$= 4 + 1 + \frac{1}{2} + \frac{0}{4} + \frac{1}{8}$$

$$= 5 + 0.625$$

$$= (5.625)_{10}$$

Binary to decimal conversion.

$$\Rightarrow \boxed{X = \sum_{i=-m}^{n-1} d_i b^i}$$

B) Decimal to Binary, Octal and Hexadecimal:-

$$\rightarrow (X.Y) = (p.q)_{b=2,8 \text{ or } 16}$$

i) to convert integer portion

b	X	
b	q_1	r_1
b	q_2	r_2
b	q_3	r_3
	0	r_4

$P = r_4 r_3 r_3 r_1$

ii) to convert fraction part

- $Y * b = c_1 \cdot P_1$
- $P_1 * b = c_2 \cdot P_2$
- $P_2 * b = c_3 \cdot P_3$

$q = c_1 c_2 c_3$

Example:-

Q. $(5.625)_{10}$.

→

2 | 5		0
2 | 2	1	↑
1 | 1	0	|
0	1	|

$b = 101$

and

$.625 \times 2 = 1.25$

$.25 \times 2 = 0.5$

$0.5 \times 2 = 1$

So $q = 101$.

thus,

$(6.625)_{10} = (101.101)_2$

1. $(101101)_2$

2. $(1001.0101)_2$

1. Sol) $(101101)_2 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2 + 1$

$= 32 + 0 + 8 + 4 + 1$

$= (45)_{10}$.

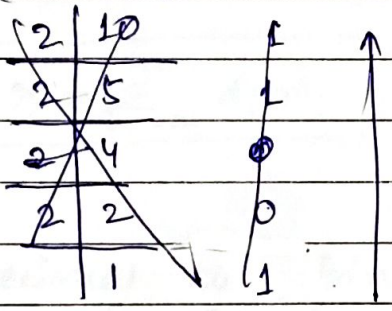
2. Sol) $(1001.0101)_2 = 1\times2^3 + 0 + 0 + 1 + 0 + \frac{1}{4} + 0 + \frac{1}{16}$

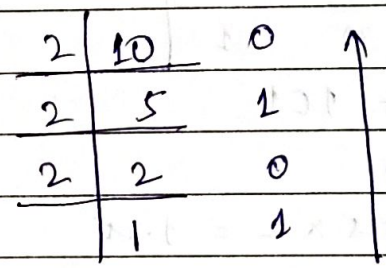
$= 8 + 1 + 0.25 + 0.0625$

$= 9.3125$

3. $(10.625)_{10}$

→ $(10.625)_{10}$





$p = 1010$

$0.625 \times 2 = 1.25$

$1.25 \times 2 = 0.5$

$0.5 \times 2 = 1.0$

$q = 101$

$(10.625)_{10} = (1010.101)_2$

⇒ Binary equivalent :-

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

→ In Hexadecimal

1

C → Binary, Octal and Hexadecimal Interconversion :-

(i) Octal to Binary.
→ 1 digit is represented in 3 bits.

(ii) Binary to Octal.
→ Grouping of 3 bits starting from binary point.

(iii) Hexadecimal to Binary.
→ 1 digit is represented by 4 bits.

iv) Binary to Hexadecimal.

→ Grouping of 4 digits starting from the binary point.

v) Octal to Hexadecimal.

→ Step 1:- Octal to Binary

Step 2:- Binary to Hexadecimal.

vi) Hexadecimal to Octal.

→ Step 1:- Hexadecimal to Binary.

Step 2:- Binary to Octal.

Example:-

Q. $(376)_8$ convert it into binary.

→ $(376)_8 = (011\ 111\ 110)_2$

Q $(2D5)_{16}$ convert to binary.

→ $(001011010101)_2$

Q.

Page No. :
Date: / /

Q Convert $(6327.4051)_8 = (\quad)_{10}$

→ $6\times8^3 + 3\times8^2 + 2\times8^1 + 7\times8^0 + 4\times\frac{1}{8} + 0\times\frac{1}{8^2} + 5\times\frac{1}{8^3} + 1\times\frac{1}{8^4}$

$= 3287 + 0.510$

$= (3287.510)_{10}$ · Ans

Q Convert $(247)_{10} = (\quad)_8$.

→

8	247	
8	30	7
	3	6

$(247)_{10} = (367)_8$.

Q Convert $(0.6875)_{10} = (\quad)_8$.

→ $0.6875 \times 8 = 5.5$

$0.5 \times 8 = 4$

So

$(0.6875)_{10} = (0.54)_8$.

Q. $(1001110)_2 = (\quad)_8$

→ $(01001110)_2 = (116)_8$

Q. $(0.10100110)_2 = (\quad)_8$

→ $(000.101001100)_2 = (0.514)_8$

Q. $(3A.2F)_{16} = (\quad)_{10}$

→ $3\times16^1 + 10\times16^0 + 2\times\frac{1}{16} + 15\times\frac{1}{16^2}$

$= (58.1835938)_{10}$ Ans.

Q $(95.5)_{10} = (\quad)_{16}$

→

16	95	
	5	15

$0.5\times16 = 8$

$(95.5)_{10} = (5F.8)_{16}$. Ans.

Q. $(111110001.100110011 01)_2 = (\quad)_{16}$

→ $(000111110001.100110011010)_2$

$= (1F1.99A)_{16}$. Ans

Q. $(A72E)_{16} = (\quad)_8$

→ $(A72E)_{16} = (1010\ 0111\ 0010\ 1110)_2$

$(001010\ 0111\ 0010\ 1110)_2 = (123456)_8$

Page No.:
Date: / /

Q. $(0.BF85)_{16} = (\quad)_8$

→ $(0.\ 1011\ 1111\ 1000\ 0101)_{2}$

⇒ $(0.101\ 111\ 111\ 000\ 010\ 100)_2 = (0.577024)_8$.

Q. $(247.36)_8 = (\quad)_{16}$.

→ $(247.36)_8 = (010\ 100\ 111 \cdot 011\ 110)_2$

⇒ $(0000\ 1010\ 0111 \cdot 0111\ 1000)_2 = (0A7.78)_{16}$.

⇒ Codes :-

1. Straight Binary code.
2. Natural BCD code.
3. Excess-3 code.
4. Gray code.

1. Straight Binary Code :-

$5 \rightarrow 101$

MSB (Most significant Bit)

LSB (Least significant Bit)

2. Natural BCD Code / BCD code :-

BCD → Binary coded Decimal. decimal

In this code, we represent each number in 4 bits.

Eg :-

5 → 0101

2 → 0010

9 → 1001

10 → 1010 → Binary.

0001 0000 → BCD

123 → 0001 0010 0011 (BCD)

→ It is also called 8-4-2-1 / 8,4,2,1 code.

→ It is also a weighted code.

3. Excess-3-Code :- It is another form of BCD code in which each decimal digit is coded into a four width binary code.

→ The code for each decimal digit is obtained by adding decimal 3 to the natural BCD code of the digit.

→ It is not a weighted code.

This code is a self complementing code which means one's complement of the coded number yields 9's complement of the number itself.

Eg :-

3 → 0011.

Excess-3 code of 2.

$$2 \rightarrow 0010$$
$$+0011$$
$$\overline{0101}$$

		Sum	Carry
→	0 + 0 = 0	0	0
→	0 + 1 = 1	1	0
→	1 + 0 = 1	1	0
→	1 + 1 = 0	0	1

Excess-3 code of 1:-

$$1 \rightarrow 0001$$
$$+0011$$
$$\overline{0100} \rightarrow 4 \text{ (Decimal)}$$

↓

~~Excess-3 code of 1.~~

4. Gray Code :- It is used in K-maps. In this the decimal number is represented in binary form in such a way, so that each gray code number differs from the preceding and succeding number by a single bit.
   → It is not a weighted code.
   → Gray code is a reflected code and can be constructed using the properties given below:-

i) A one bit gray code have two code words 0 and 1 respectively, representing decimal number 0 and 1 respectively.

ii) A n-bit (gray code)
A n-bit ($n \geq 2$) gray code will have first 2^{n-1} gray codes of $(n-1)$ bits written in order with a leading 0 appended.

iii) The last 2^{n-1} gray codes will be equal to the gray code words of an $(n-1)$ bit gray code written in reverse order with a leading 1 appended.

⇒ 1-bit Gray code :-

0 → 0
1 → 1

⇒ 2-bit Gray code :-

(n-1) bit Gray code.

2^{2-1}

0 → 0 0
1 → 0 1
- - - - -
3 → 1 1
2 → 1 0

→ There should be single bit change in succeeding and preceeding position.

Page No.:
Date: / /

3-bit gray code.

→ $2^{3-1} = 2^2 = 4$

0 → 0 0 0

1 → 0 0 1

3 → 0 1 1

2 → 0 1 0

- - - - - - -

6 → 1 1 0

7 → 1 1 1

5 → 1 0 1

4 → 1 0 0

⇒ Boolean Algebra :-

→ $A + 0 = A$

→ $A \cdot 1 = A$

→ $A + 1 = 1$

→ $A \cdot 0 = 0$

→ $A + A = A$

→ $A \cdot A = A$

→ $A + \bar{A} = 1$

→ $A \cdot \bar{A} = 0$

→ $A(B + C) = AB + AC$

→ $A + BC = (A + B)(A + C)$

} distributive laws.

→ $A + AB = A$ } Absorption law.

→ $A(A + B) = A$

→ $A \cdot B = B \cdot A$ } commutative law.

→ $A + B = B + A$

→ De Morgan's theorem :-

→ $\overline{A \cdot B} = \bar{A} + \bar{B}$

→ $\overline{A + B} = \bar{A} \cdot \bar{B}$

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

→ $\bar{F} = \overline{A\bar{B} + \bar{A}B}$ (By using De Morgan's theorem)

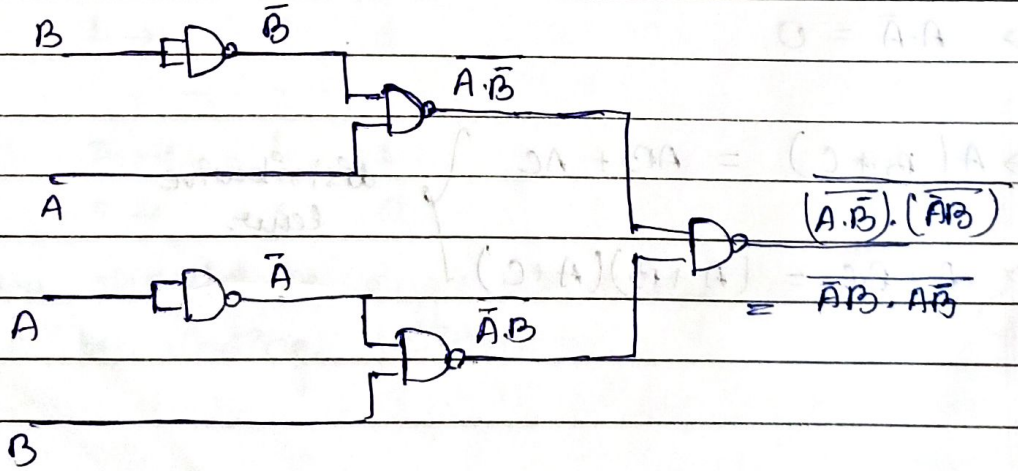
$= \overline{A\bar{B}} \cdot \overline{\bar{A}B}$

$= \bar{A}B \cdot A\bar{B}$

Now,

$F = \bar{\bar{F}} = \overline{\bar{A}B \cdot A\bar{B}}$

=



Page No.:
Date: / /

Q. $F = \bar{A}\bar{B} + AB$

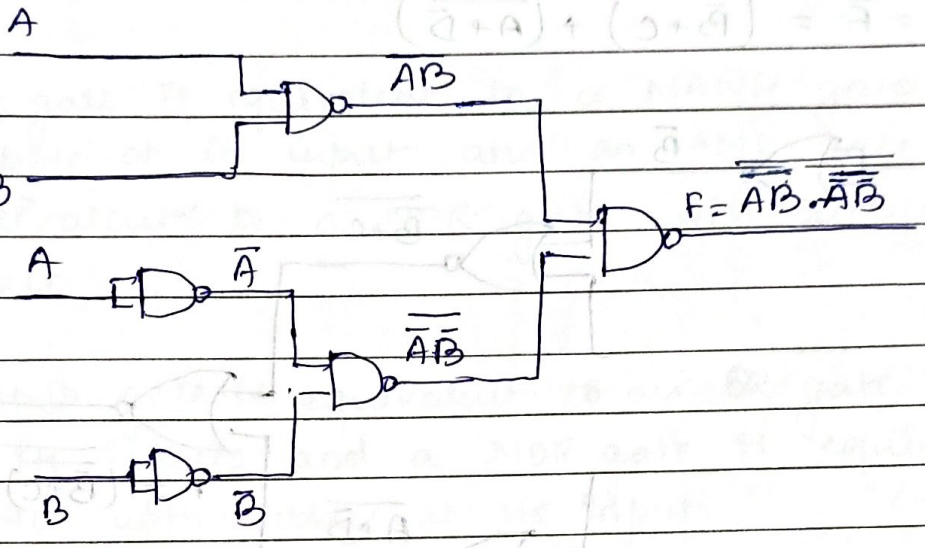
→ $\bar{F} = \overline{\bar{A}\bar{B} + AB}$ (Using DeMorgan's Law)

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

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

Now,

$F = \bar{\bar{F}} = \overline{\overline{AB} \cdot \overline{AB}}$



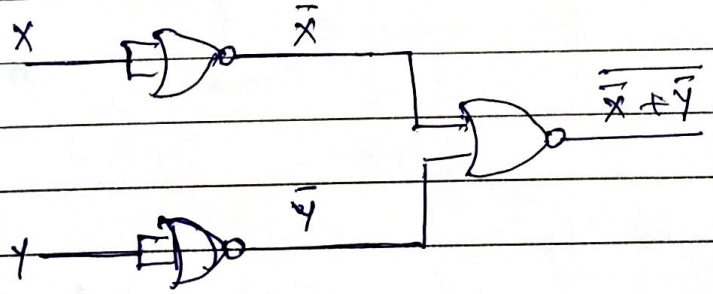
Q. $F = X \cdot Y$. Realize using NOR gate.

→ $F = XY$

$\bar{F} = \overline{XY}$ (By De-Morgan's Law)

$= \bar{X} + \bar{Y}$

$\bar{\bar{F}} = F = \overline{\bar{X} + \bar{Y}}$



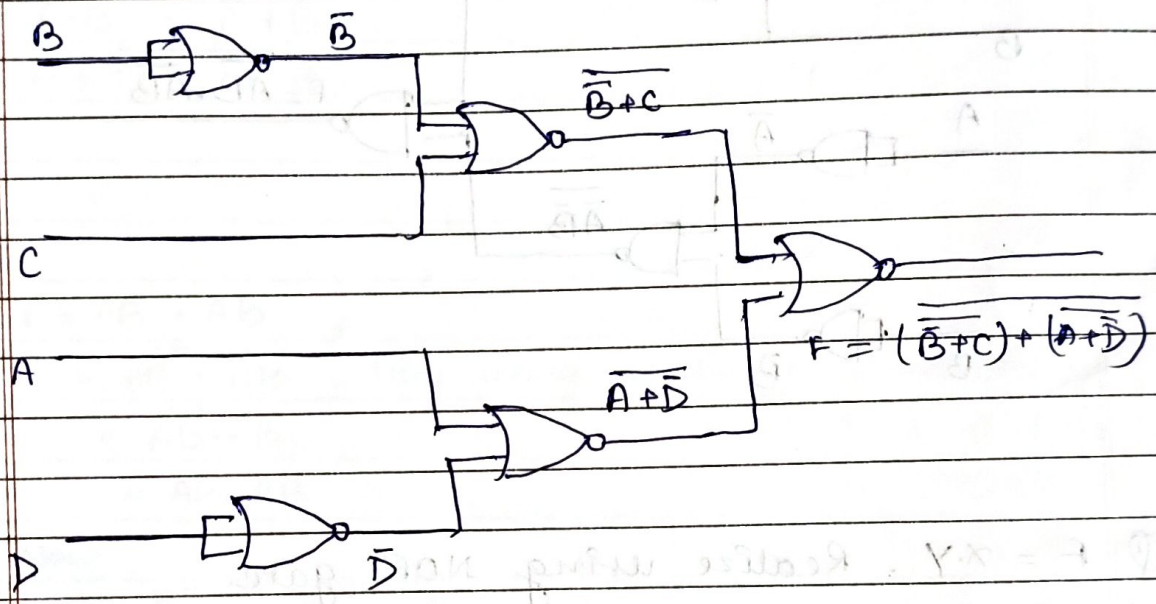
Q. $F = (\bar{B}+C)(A+\bar{D})$. Realize using only NOR gates.

→ $F = (\bar{B}+C)(A+\bar{D})$

$\bar{F} = (\bar{B}+C)(A+\bar{D})$ (By De-Morgan's Law)

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

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



→ Conversion to NAND-NAND and NOR-NOR gate networks :-

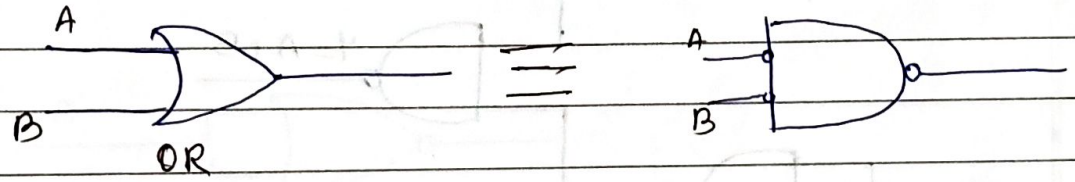
→ $\overline{A+B} = \bar{A} \cdot \bar{B}$, $\overline{AB} = \bar{A} + \bar{B}$

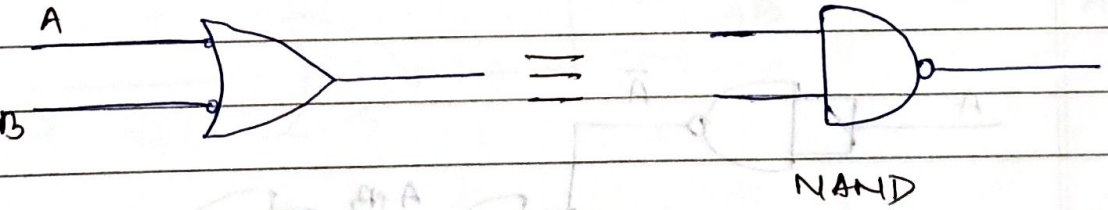
→ $A+B = \overline{\bar{A} \cdot \bar{B}}$

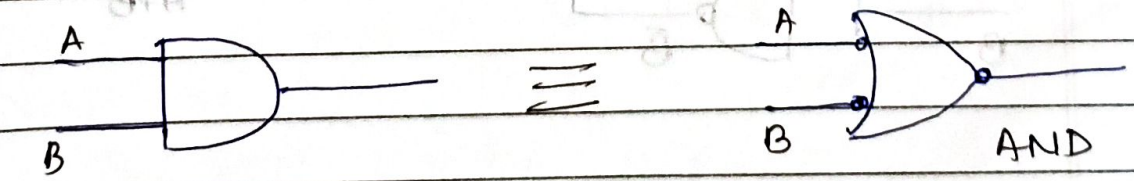
→ $A \cdot B = \overline{\bar{A} + \bar{B}}$

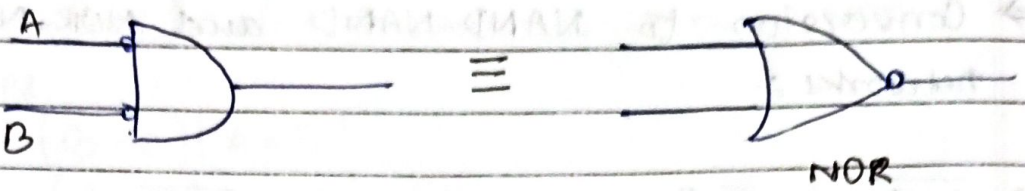
→ OR gate is equivalent to a NAND gate with bubble at its input and an AND gate is equivalent to a NOR gate with bubbles at its input.

→ NAND gate is equivalent to an OR gate with bubbles at its inputs and a NOR gate is equivalent to AND with bubbles at its input.

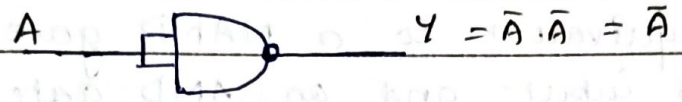


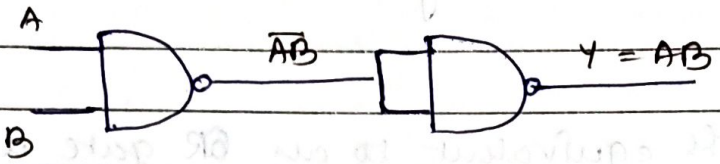


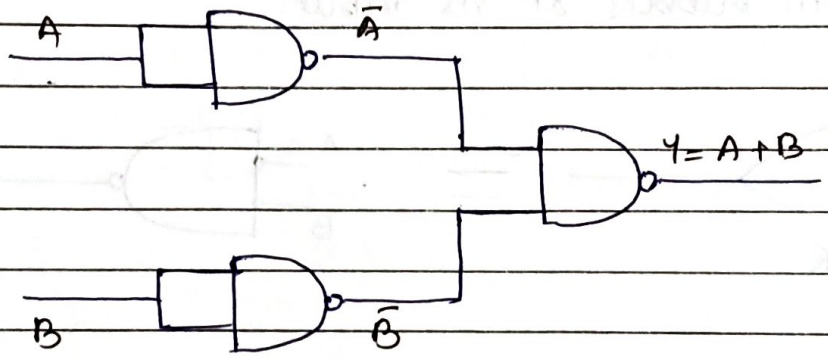


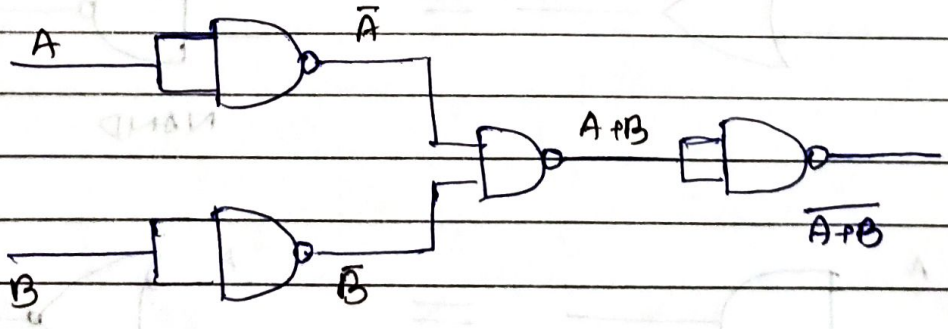


⇒ Realization of Basic Gates using NAND gate:-



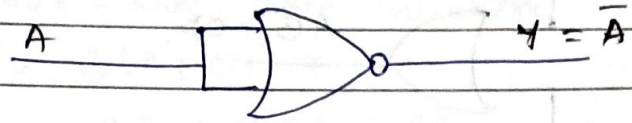


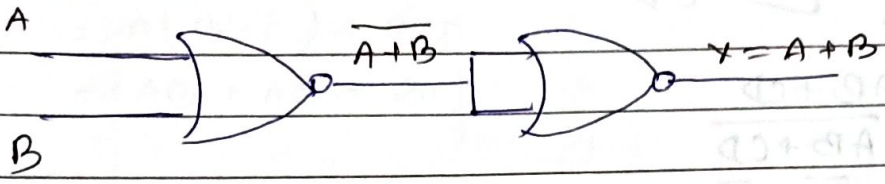


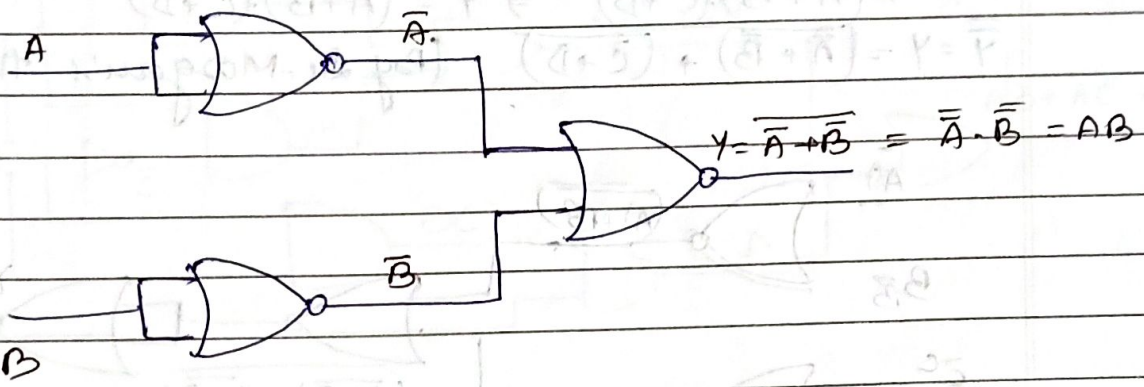


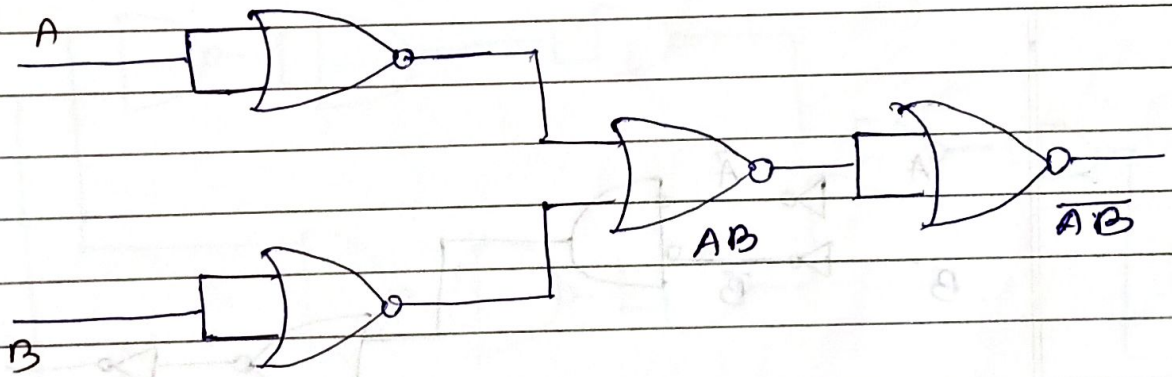
Page No.:
Date: / /

⇒ Realization of Basic Gate using NOR Gate :-

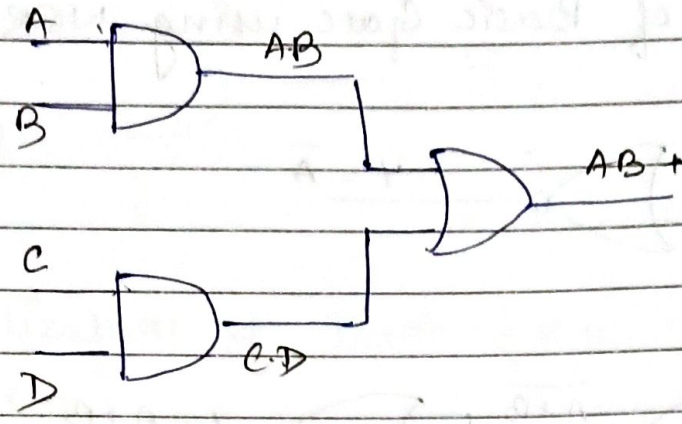








Q.



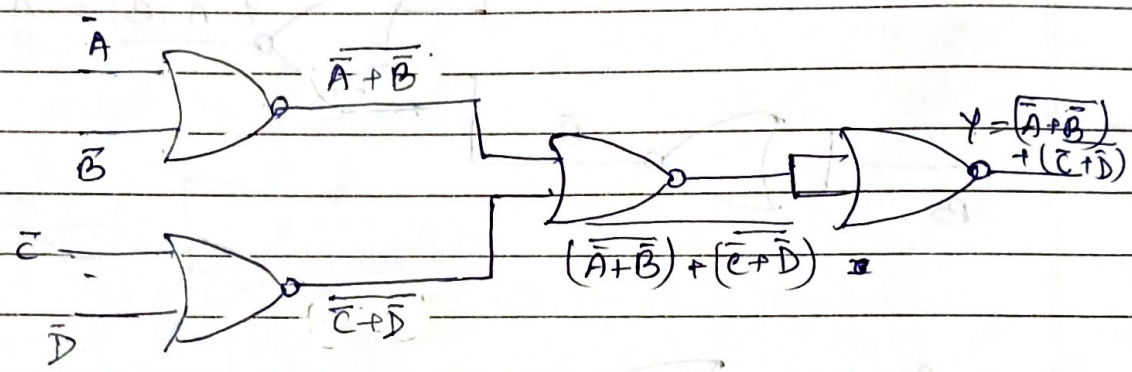
→ $Y = AB + CD$

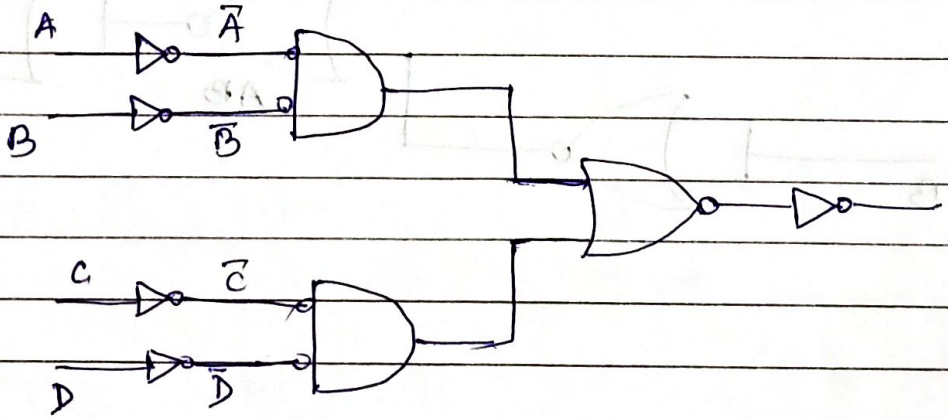
$\bar{Y} = \overline{AB + CD}$

$= \bar{A}\bar{B} \cdot \bar{C}\bar{D}$

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

$\bar{\bar{Y}} = Y = \overline{(\bar{A} + \bar{B})} + \overline{(\bar{C} + \bar{D})}$ (By De-Morgan's theorem)





Page No.:
Date: / /

Q. $f = B(A + CD) + A\bar{C}$, Realize this function using
i) NAND - NAND gate network.
ii) NOR - NOR gate network.

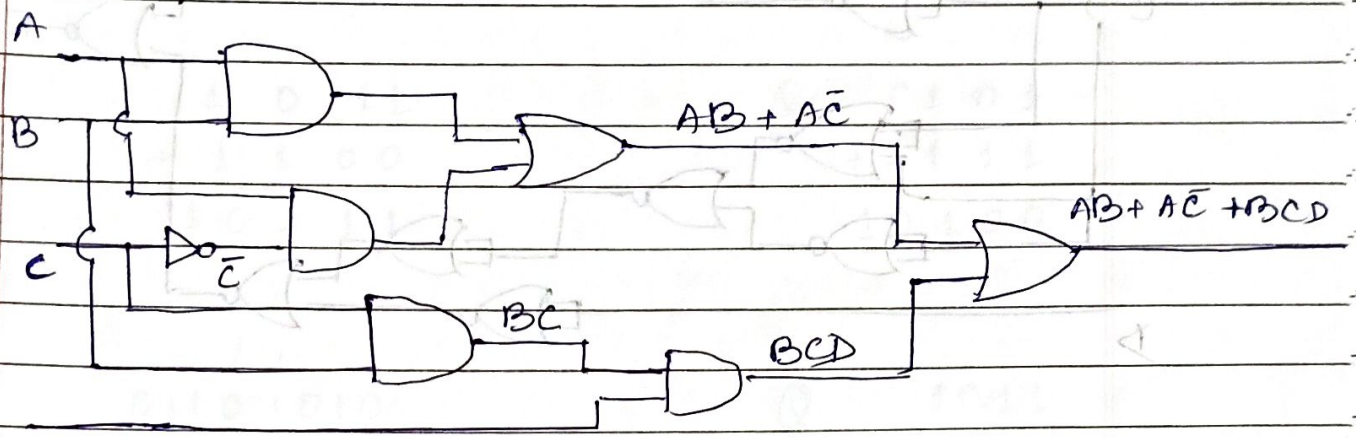
→ $f = B(A + CD) + A\bar{C}$

$= BA + BCD + A\bar{C}$

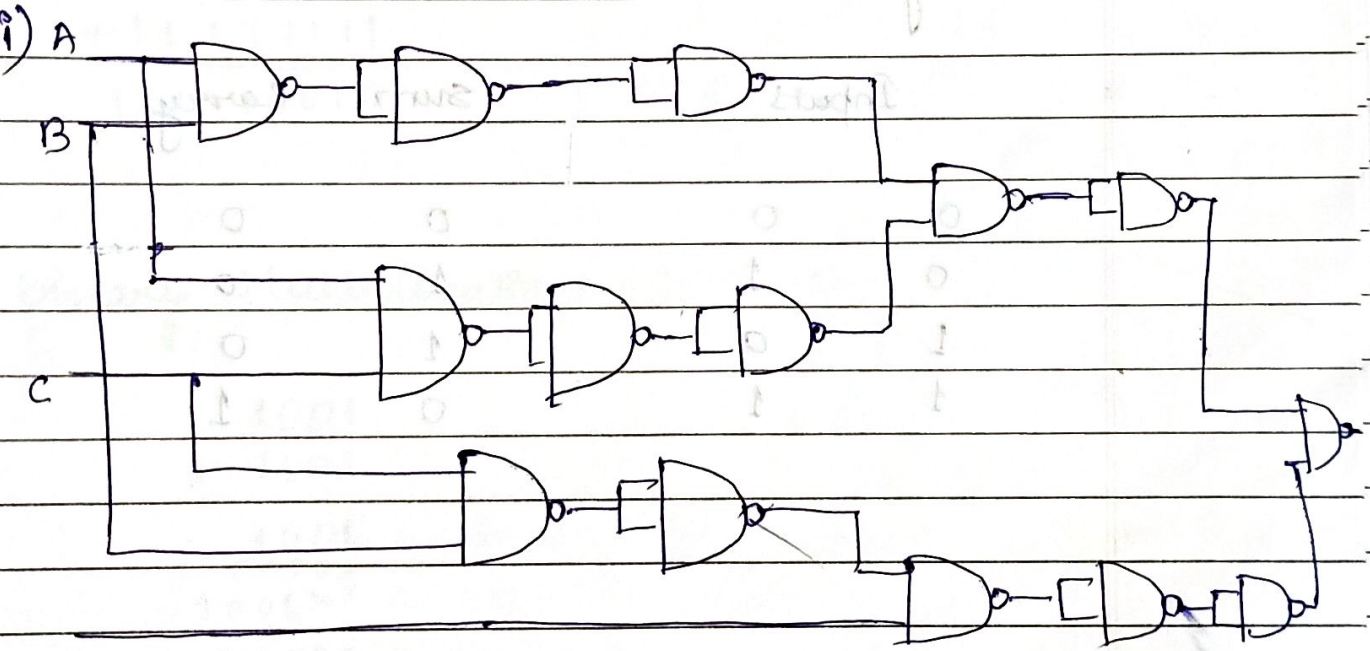
$= AB + A\bar{C} + BCD$

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

$= AB + A\bar{C} + BCD$

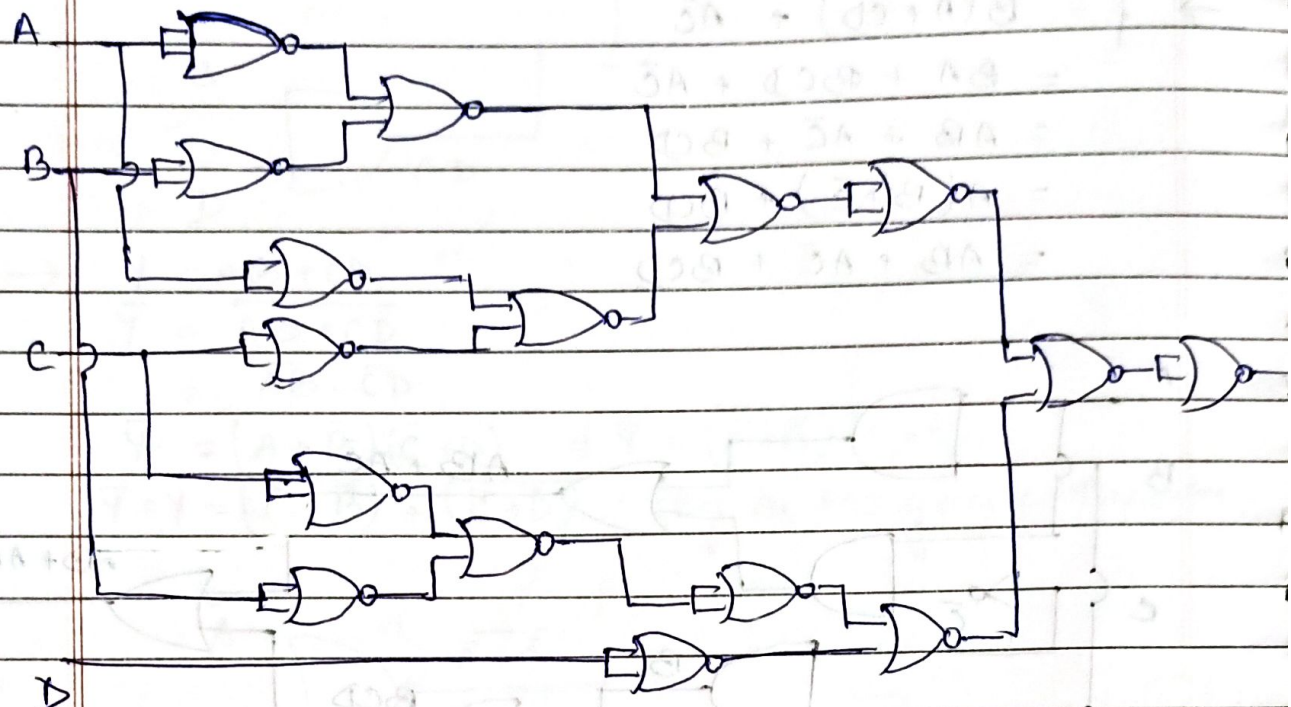


i)



NAND - NAND gate network.

ii) NOR-NOR gate network.



⇒ Binary Addition:

Inputs		sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

⇒ Binary Subtraction :-

Inputs		Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

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

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

Q.
```
  0 1 1 0 1 0 1 0
  0 0 0 0 1 0 0 0
  1 0 0 0 0 0 0 1
+ 1 1 1 1 1 1 1 1
-----------------
1 1 1 1 1 0 0 1 0
```

Q
```
  1 0 1 1
- 0 1 1 0
---------
  0 1 0 1
```

⇒ Binary Multiplication :-

```
    1 0 0 1
  x 1 1 0 1
-----------
    1 0 0 1
  0 0 0 0 x
1 0 0 1 x x
-----------
1 0 0 1 x x x
-----------
1 1 1 0 1 0 1
```

→ If the numbers of 1's to be added in a column is even the sum bit is 0 and if the no. of 1's to be added in column, then the sum bit is 1.

→ If every pair of 1's in a column produces a carry to be added to the next higher bit column.

⇒ Binary Division :-

Q Divide 1110101 by 1001.

→
```
        1101
1001 ) 1110101
     - 1001
       01001
       1001
       00001
          1001
          0000
```

Q Find one's complement of the following numbers:-

1. 01001111001

→ 10110000110. ← 1's complement.

→ For 1's complement invert the digits.

Page No.:
Date: / /

Q Find 2's complement of the following numbers:-

Q1. 0100111001.

→ Step 1:- First take one's complement.

Step 2:- Add 1 to the LSD of one's complement.

0100111001

→ 1011000110

+ 1

1011000111 ← 2's complement.

1. 01001110.
2. 00110101.

1. Sol 01001110

→ 10110001

+1

10110010 Ans.

2. Sol 00110101

11001010

+1

11001011. Ans.

⇒ Sign Bit :-

Q. -(2)

→ 1 (10)

↑ Sign bit

Q +2

→ 0 (10)

⇒ Sign Magnitude Representation :-

Q Find the decimal equivalent of the following binary numbers assuming sign magnitude representation :-

1. $0\overset{2}{1}\overset{1}{1}\overset{0}{1}$

→ $1\times2^2 + 1\times2^1 + 1\times2^0 = +(7)$

$= +7$. Ans

2. $\overset{2}{1}\overset{1}{1}\overset{0}{1}$

→ $2^2 + 2^1 + 2^0$

$= -7$. Ans.

3. 101100

4. 001000

Page No.:
Date: / /

Q Represent the following in one's complement form :-

a) +7 and -7.

→ 7 → 111

One's complement of +7 = 00000

One's complement of -7 = 10000

b) +8 and -8.

→ (+8) = 01000 one's complement → 10111.

(-8) = 10000

⇒ For an n-bit number the maximum positive number which can be represented in one's complement representation is $(2^{n-1} - 1)$, and the maximum negative number is $-(2^{n-1} - 1)$.

Q Represent (-17) in

i) Sign magnitude

ii) One's complement.

iii) Two's complement.

→ i) $(-17) = (110001)_2$

$(110001)_2$ ← Sign magnitude.

ii) $(101110)_2$ ← One's complement.

iii) $(101111)_2$ ← Two's complement.

⇒ For one's complement for a negative decimal number the one's complement should be taken of the positive value of that -ve number.

⓪ For an n bit number the maximum +ve number which can be represented in two's complement form is $(2^{n-1}-1)$ and the maximum -ve number which can be represented in two's complement form is -2^{n-1}.

⇒ Subtraction Using Two's Complement Form :-

→ Binary subtraction can be performed by adding the two's complement of the subtrahend to the minuhend.

→ If a final carry is generated, discard the carry and the answer is given by the remaining bits which is +ve (the minuhend is greater than subtrahend).

→ If the final carry is zero the answer is negative (the minuhend is smaller than the subtrahend) and is in two's complement form.

+7 = 0111 → 1000 → −7

Page No.:
Date: / /

Q Perform binary subtraction using two's complement representation of negative numbers.

1. (7 − 5) ← subtrahend, minuend

→ (+5) = 0101

1010 ← One's complement

+ 1

1011 ← Two's complement.

(+7) = 0111

+ 1011

(1)(0010)$_2$ ← Positive numbers.

final carry is discarded.

		S	B
0	0	0	X
1	0	1	X
0	1	1	1
1	1	0	X

1 0

−0 −1

1 1 B→

2. (5 − 7)

→ (+7) = 0111

1000 ← One's complement

+ 1

1001 ← Two's complement

		D	B
0	0		
0	1		
1	0		
1	0		

(+5) = 0101

0101

+ 1001

1110 ← Two's complement form −ve number.

Again taking 2's complement of the obtained number

1110

0001 ← 1's complement

+ 1

0010 ← 2's complement.

3. 48 - 23.
4. 23 - 48.
5. 48 - (-23)
6. -48 - 23.

Q Convert $(6327.4051)_8 = (\quad)_{10}$

Q Convert $(247)_{10} = (\quad)_8$.

Q Convert $(0.6875)_{10} = (\quad)_8$

3 48 → 00110000 ~~1's complement~~.

23 → 00010111

-23 → 11101000 → 1's complement

11101001 → 2's complement

00110000
11101111
1
11101000
00010111
11100111 → -25

→ VHDL :-

→ Verilog :-

It is an HDL (Hardware description language). The latest stable version of verilog is IEEE 1364 - 2005. It is a case sensitive language which only uses lowercase. It supports simulation. In other words it is possible to create a model of a function and simulate it before building the real system. The base language of verilog is C.

Module is a basic building block in verilog. It provides information about input and output codes, and hides the internal implementation details. The syntax of module is as follows :-

Every verilog program starts with the keyword "module", and ends with the keyword "end module".

```
module <module_name> (input, output);
<program logic>
end module
```

→ VHDL :-

VHDL is an HDL, that helps to describe circuits in digital systems. VHDL stands for Very High speed Integrated Circuit Hardware Description Language. It is a programming language used to model a digital system. It was first introduced in 1981 for the department of defence under Very High speed Integrated Circuit programs.

Three types of modelling in VHDL :-

i) Data-flow modelling :- Parallel signals represent the flow of data through an entity.

ii) Behavioural modelling :- Represents the behaviour of an entity as a set of statements to execute one after the other in a specified order.

iii) Structural modelling :- Represents an entity as a set of inter-connected components.

Q) What is Hardware module?

→ A Hardware module in VHDL is called an entity. The syntax is as follows. The entity starts with keyword "entity" and ends with keyword "end".

```
entity <entity_name> is
port declaration;
end entity_name;
```

There are other keywords like in, out:-

In	Port can be read
Out	Port can be written
Inout	Port can be read and written.
Buffer	Port can be read and written, it can have only one source.

→ Architecture :-
Architecture can be described using structural data flow, behavioural or mixed style.

architecture architecture_name
of entity_name

architecture_declarative_part;
begin
statements;
end architecture_name;

⇒ Differences between Verilog and VHDL :-

→ Verilog is an HDL used to model electronic systems. while VHDL is an HDL used in electronic design automation, to describe digital and mixed signal system such as field programmable gate arrays (FPGA) and integrated circuits.

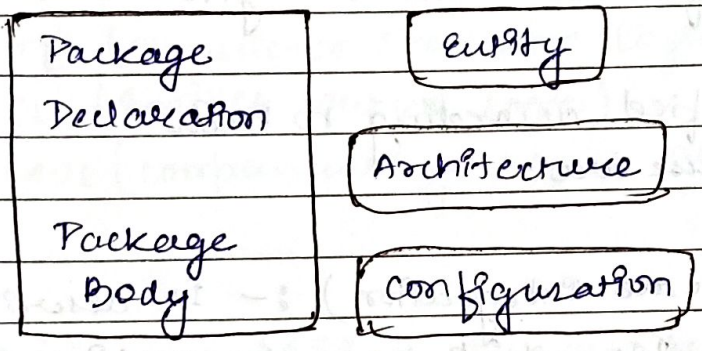
→ Verilog is based on C language while VHDL is based on Ada and Pascal languages.

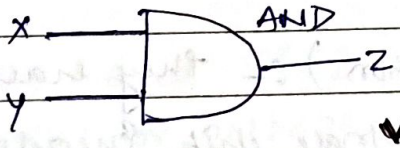
→ Verilog is case sensitive while VHDL is not case sensitive.

→ Verilog was introduced in 1984 while VHDL was introduced in 1981.

→ VHDL is more complex than Verilog.

⇒ VHDL Program Structure :-





VHDL Code

```
library ieee;
use ieee.std_logic_1164.all;
entity and1 is
port( x,y : in bit; z : out bit );
end and1;
architecture first of and1 is
```

```
begin
    z <= x and y;
end Brat;
```

→ The first two lines imports the ieee standard logic library which contains pre-defined logic functions and datatypes, such as std_logic and std_logic_vector.

→ The use statement determines which portions of a library file to use.

⇒ Classification of fixed function digital ICs:-

→ They are classified according to their complexities. These are:-

1. SSI (Small scale Integration) :- It describes fixed functions ICs which has about 12 gates on a single chip.

2. MSI (Medium Scale Integration) :- They have from 12-99 gates on a single chip. Encoder, decoder, multiplexer has all these ICs.

3. LSI (Large Scale Integration) :- They have about 100 to 9999 gates on a single chip.

3. VLSI (Very Larege Scale Integration) :- They have about 10000 to 99999 gates on a single chip.

4. ULSI (Ultra Large Scale Integration) :- It describes very large memories, large microprocessor and larger single chip computers. They have about more than 100000 gates on a single chip.

⇒ Integrated Circuit logic family :-

→ The three Integrated Circuit logic families are :-

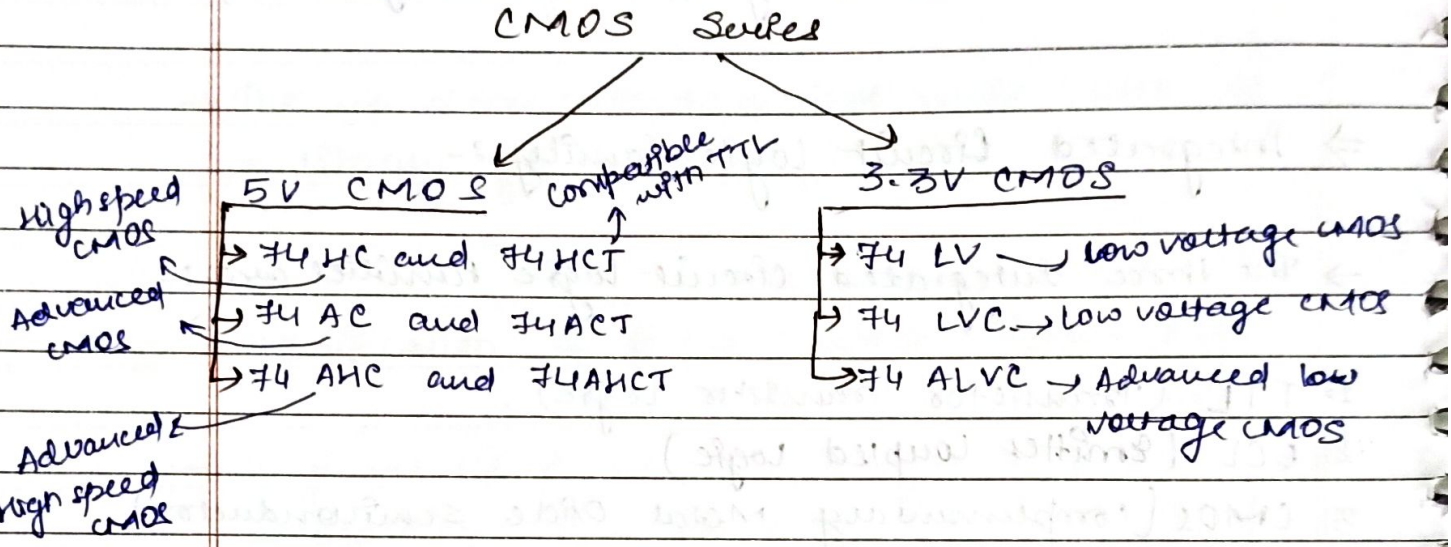
1. TTL (Transistor Transistor Logic)
2. ECL (Emitter Coupled Logic)
3. CMOS (Complementary Metal Oxide Semiconductor).

⇒ SSI and MSI :- TTL are present in both SSI and MSI.

→ LSI and VLSI and ULSI are implemented with CMOS or NMOS because it requires less area on chip and consumes less power.

⇒ Advantages of CMOS over TTL :-

→ CMOS ICs have faster switching speed, lower power dissipation, more noise immunity.



→ The prefix 74 indicates the commercial grade for general use

→ The prefix 54 is used for military grade.

→ In addition to pure CMOS, there is a series which combines both CMOS and TTL is called as BiCMOS.

→ BiCMOS

→ ① 74 BCT → BiCMOS.
→ ② 74 ABCT → Advanced BiCMOS
→ ③ 74 LVT → Low voltage BiCMOS
→ ④ 74 ALB → Advanced low voltage BiCMOS

⇒ TTL Series :-

→ ① 74 → Standard TTL
→ ② 74S → Schottky TTL
→ ③ 74AS → Advanced Schottky TTL
→ ④ 74 LS → Low Power Schottky TTL
→ ⑤ 74 ALS → Advanced low Power Schottky TTL
→ ⑥ 74F → Fast TTL

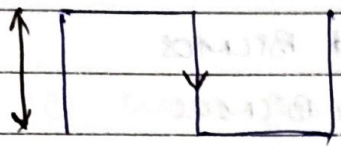
Example:-

74LS86 → XOR gate.
- Commercial
- Type
- Logic

⇒ Propagation Delay :- Propagation Delay Time of the logic gate is the time interval between the application of an input pulse and the occurence of the resulting output pulse. Denoted by t_p.

→ t_{PHL} :- The time between a specified reference point on the input pulse and the corresponding reference point on the output pulse, with the output changing

from high voltage level to low voltage level.



→ t_{PLH} :- The time between a specified reference point on the input pulse and the corresponding reference point on the output pulse, with the output changing from low level to high level.

→ DC supply voltage (V_{cc}) :-

→ CMOS → V_{cc} → +5V

→ +3.3 V.

→ TTL → DC → +5 V V_{cc}

→ Power dissipation, P_D :-

It is the product of the DC supply voltage and the average supply current.

The supply current when the gate output is low is greater than when the gate output is high. It is usually specified by the manufacturer.

$$P_D = V_{cc}\left(\frac{I_{CCH} + I_{CCL}}{2}\right)$$

(I_{CCH} → high output, I_{CCL} → low output)

→ The CMOS gates have very low power dissipation than the TTL series.

→ The power dissipation of CMOS is dependent on frequency of operation.

→ Power dissipation of TTL is independent of frequency.

⇒ Input and Output Logic Level

→ V_{IL} :- Low level input voltage.

→ V_{IH} :- High level input voltage.

The 5V CMOS accepts a maximum voltage of 1.5 V as low input level and 3.54V for high input level.

For TTL high voltage input level is 0.8 V and low level is 2.0 V.

~~For~~ TTL :-

V_{OL} → Low output voltage

V_{OH} → High output voltage.

For CMOS (5V) :- $V_{OL} = 0.33\ V$

$V_{OH} = 4.4\ V$

t_{PHL} → High to low

t_{PLH} → Low to High

For TTL :-

$V_{OL} = 0.4\ V$

$V_{OH} = 2.4$

⇒ Speed Power Product (SPP) :-

→ SPP of a logic circuit is product of propagation delay time and power dissipation.

$$SPP = t_p P_D$$

⇒ Fan-out :- fan-out of a logic gate is a maximum number of inputs of the same series of an IC family that can be connected to the output of a gate and still maintain the output voltage levels within specified limits.

→ It is specified in terms of unit-loads.

→ A unit load for a logic gate equals 1 input to a alike circuit.

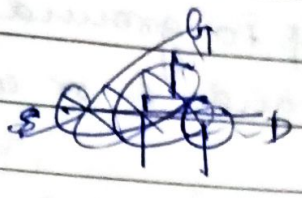
→ $$\text{Unit loads} = \frac{I_{OH}}{I_{IH}} = \frac{I_{OL}}{I_{IL}}$$

I_{OH} = High output current

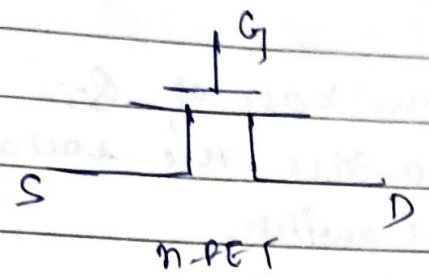
I_{OL} = Low output current

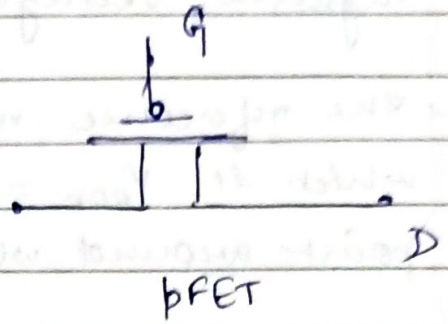
I_{IH} = High Input current

I_{IL} = Low input current.

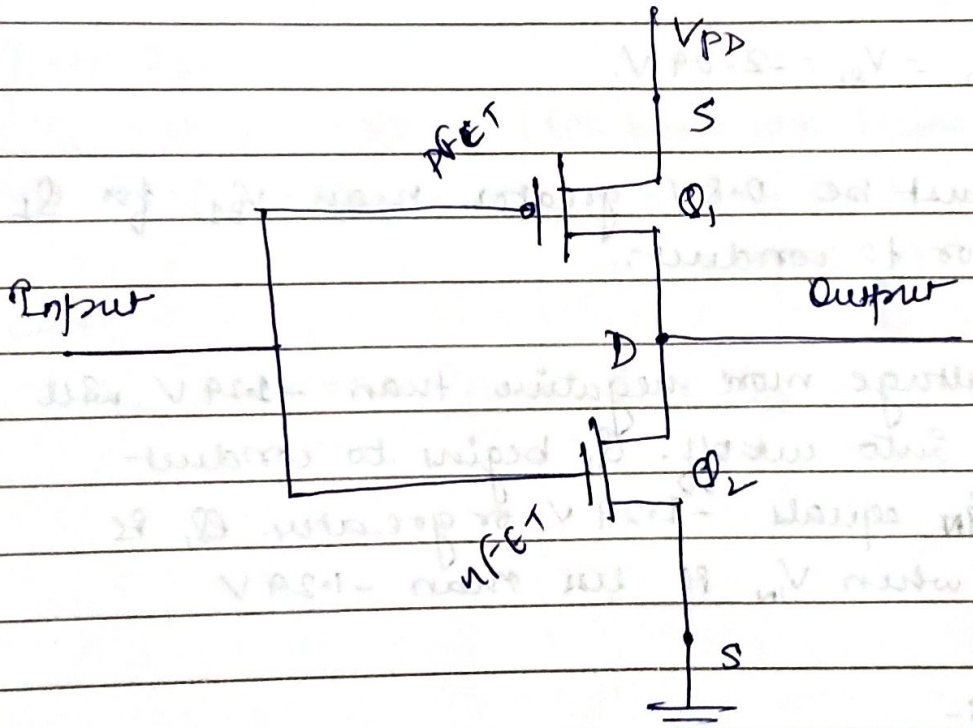


CMOS





(1) For positive gate voltage nFET is ON.	(1) +ve gate voltage pFET is OFF
(2) For Negative (or zero) Gate voltage, nFET is OFF	(2) '-ve' or zero gate voltage pFET is ON.



CMOS NOT gate

The collector resistors are pulled up to ground. The logic levels for both 0 and 1 are also negative voltages for ECL.

→ The reference voltage at the base of Q_2 which is $V_{REF} = -1.29\ V$ provides the voltage point around which Q_1 will switch.

→ V_{BE} equals 0.8 V for the transistor in ECL 10K family.

specified by the manufacturer

$$V_{E2} = V_{REF} - (V_{BE2})$$
$$= (-1.29\ V - 0.8\ V)$$
$$= -2.09\ V$$

Because both emitters are ~~not~~ at the same node.

$$V_{E2} = V_{E1} = -2.09\ V.$$

$V_{BE} = V_{B1} - V_{E1}$

→ V_{B1} must be 0.8 V greater than V_{E1} for Q_1 transistor to conduct.

→ Any voltage more negative than −1.29 V will pull Q_1 into cut-off. Q_1 begins to conduct when V_{IN} equals −1.29 V or greater. Q_1 is cut-off when V_{IN} is less than −1.29 V

Case 1 :-

$V_{IN} = 1 = -0.9\ V$ (ECL 10K high input voltage)

$V_{E1} = V_{IN} - V_{BE1}$

$= -0.9\ V - 0.8V$

$= -1.7\ V$

then,

$$-I_E = \frac{(V_{EE} - V_{E1})}{R_E}$$

$$= \frac{-5.2 - (-2.09)}{779}$$

$$= \frac{-3.11}{779}$$

$$= -3.99 \times 10^{-3}$$

$$\Rightarrow I_E = 3.99\ mA$$

let $I_E = I_{E1}$,

then,

$I_{E1} = 3.99\ mA$

$V_{C1} = -I_{E1} R_{E1} =$

Case 2:-

$V_{IN} = 0 = -1.75\ V$ (ECL 10K below input voltage)

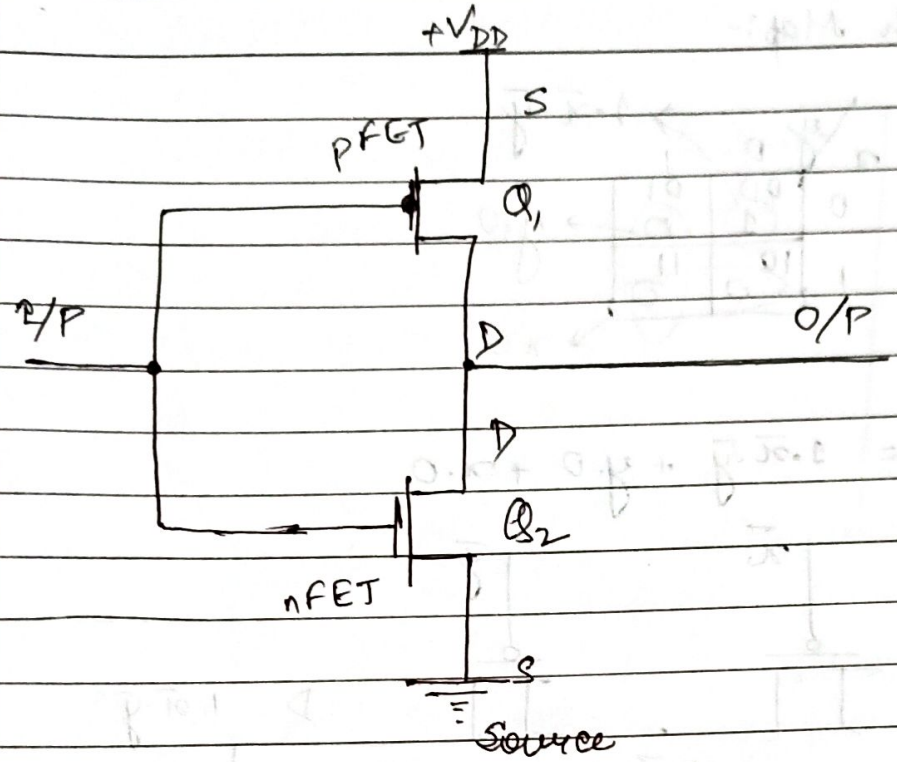
$V_{B1} = V_{B2} = V_{REF} - V_{BB2} = (1.29 - 0.8)V$

$= -2.09\ V.$

with

$V_{IN} = -1.75\ V$, then Q_1's $V_{BE} = -1.75 - (-2.09\ V)$

$= +0.34\ V.$



CMOS Inverter / CMOS NOT gate

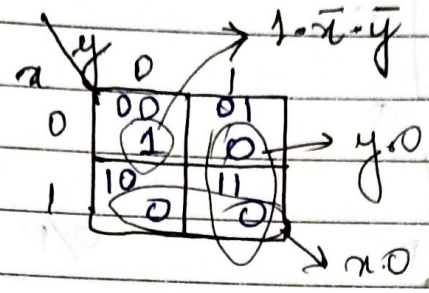
⇒ CMOS NOR Gate :-

LSB (~~least significant bit~~)

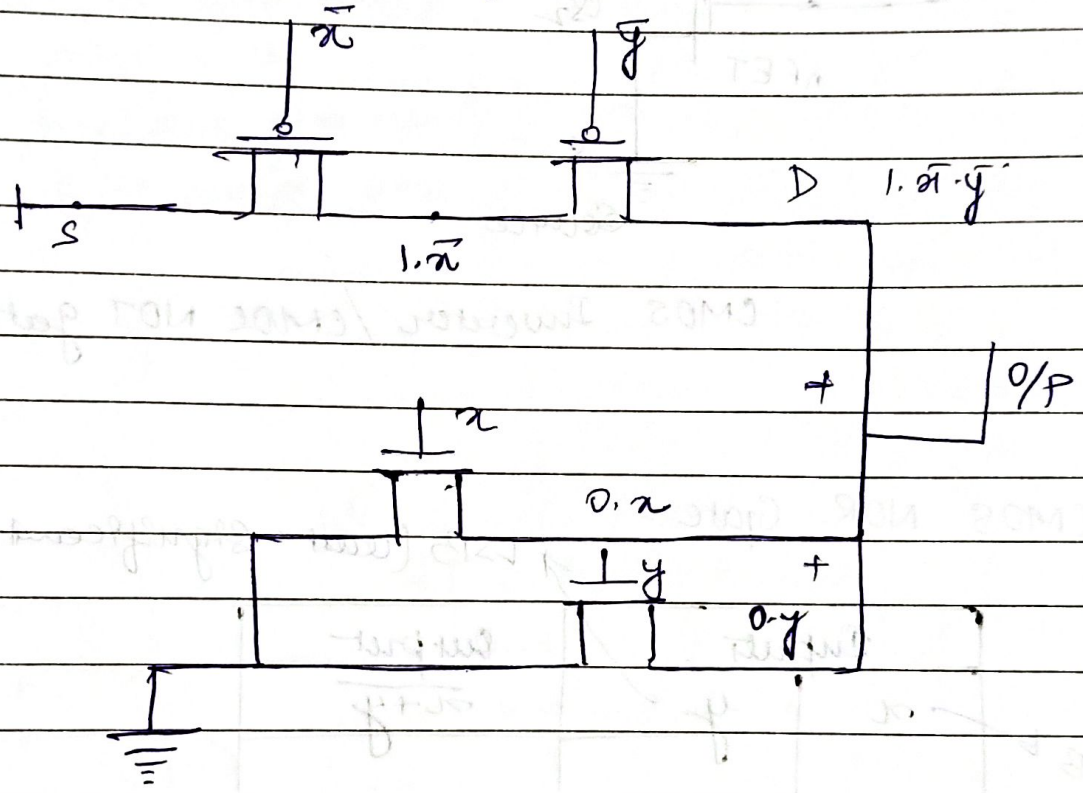
Inputs		Output
x	y	$\overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

MSB (Most significant bit)

⇒ Karnaugh Map:-



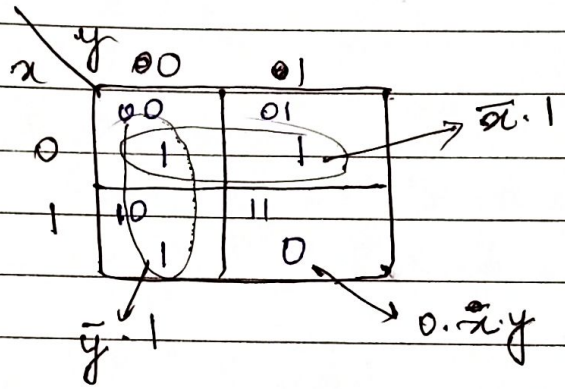
$$b(x,y) = 1.\bar{x}.\bar{y} + y.0 + x.0$$



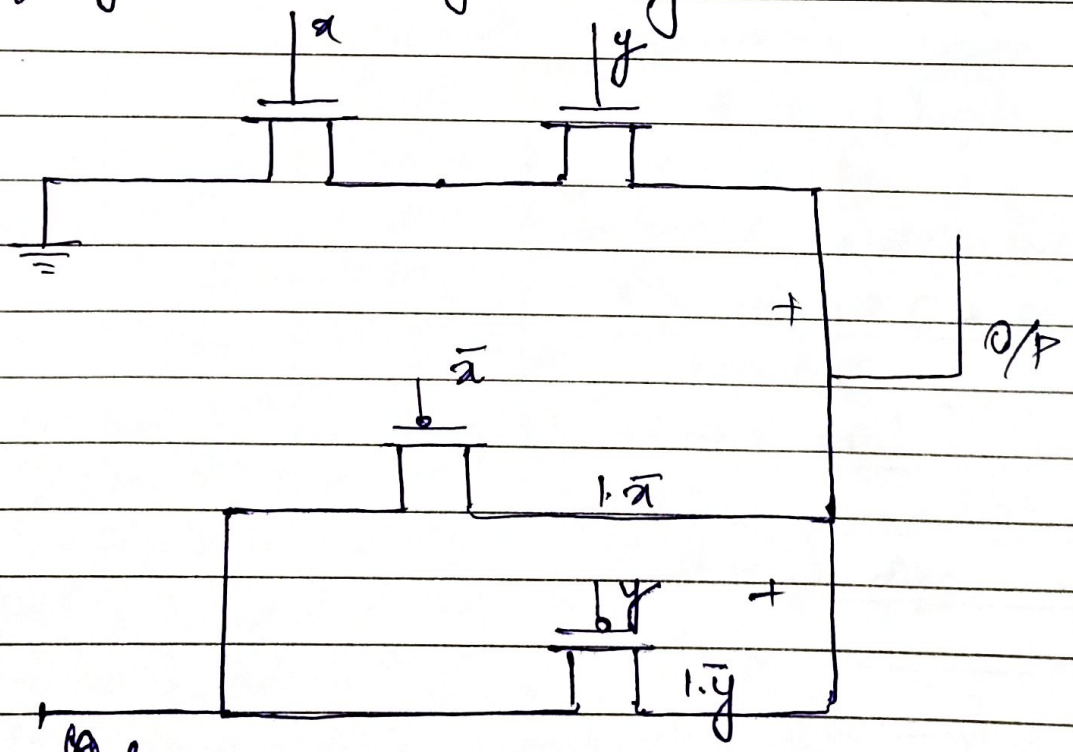
CMOS NOR

⇒ CMOS NAND Gate

Inputs		Output
x	y	$\overline{x.y}$
0	0	1
0	1	1
1	0	1
1	1	0



$f(x,y) = 1.\bar{x} + 1.\bar{y} + 0.x.y$

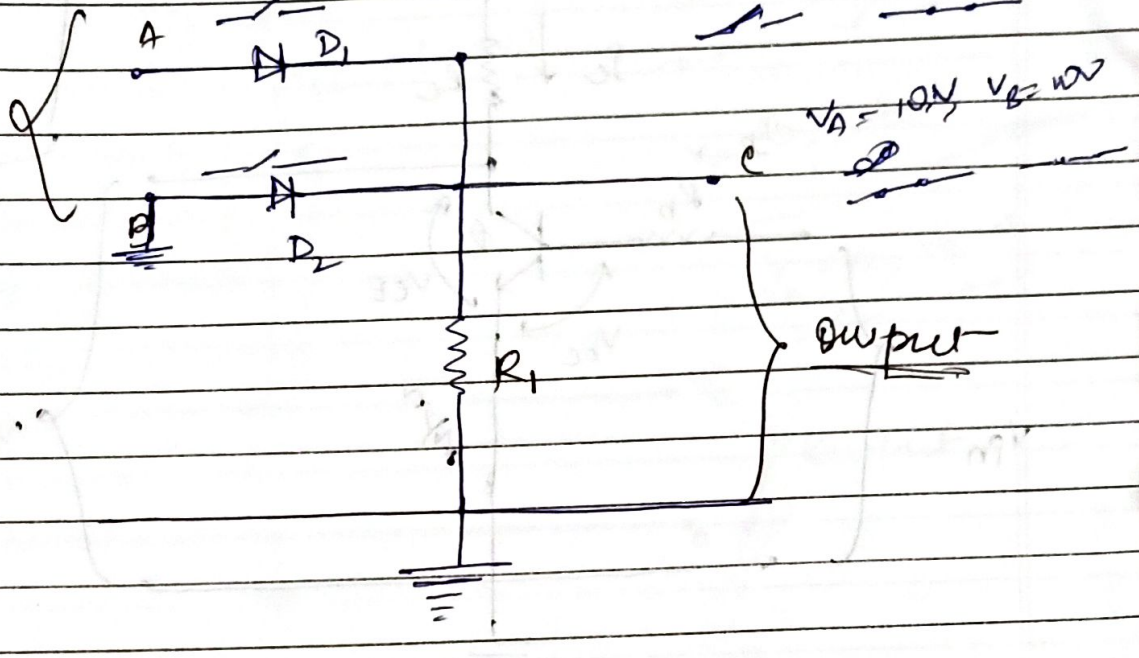


→ 2 input diodes OR gate.

$V_A = 10V$, $V_B = 0V$

$V_A = 0V$, $V_B = 10V$

$V_A = 10V$, $V_B = 10V$



If $V_A = V_B = 0$

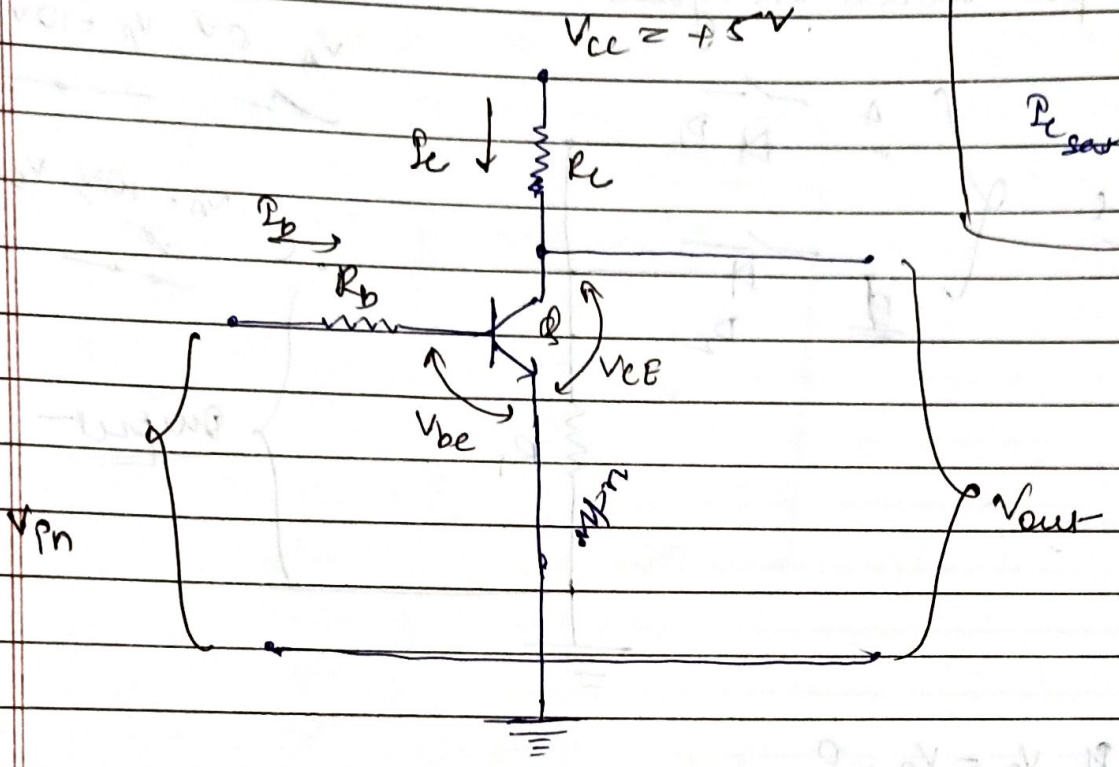
A	B	O/P
0	0	0
1	0	1
0	1	1
1	1	1

↳ OR gate

Transistor as an Inverter.

$$I_b = \frac{V_{in} - V_{be}}{R_b}$$

$$I_{c_{sat}} = \frac{V_{cc} - V_{CE(sat)}}{R_c}$$



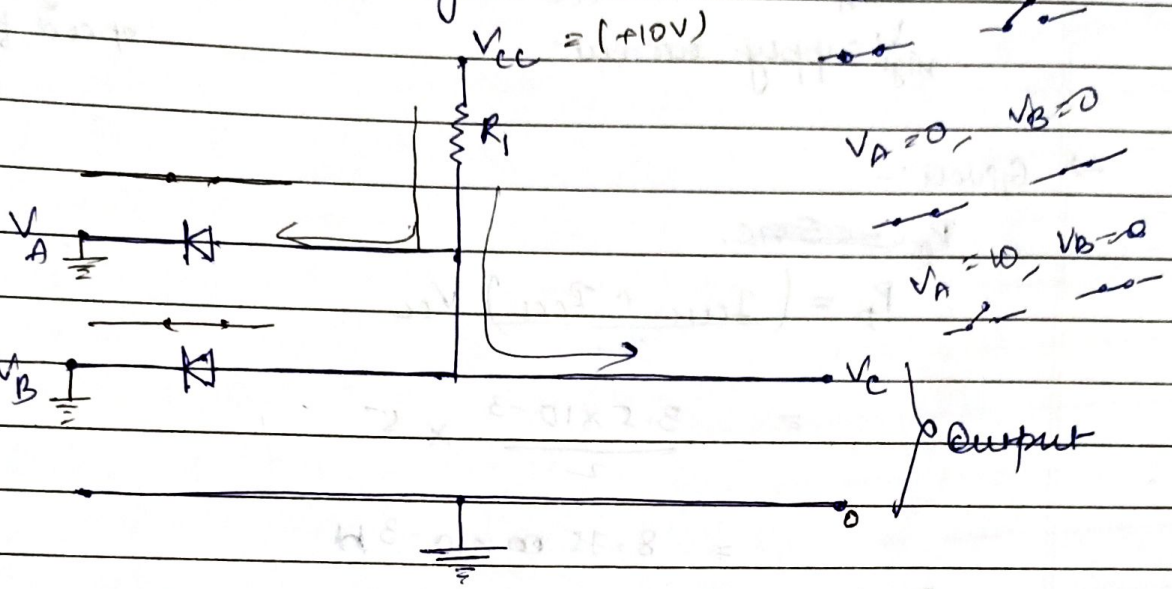
$V_{in} = 0\ V$, $I_b = 0$, $I_c = 0\ mA$

V_{in}	V_{out}
0	1
1	0

DTL → Diode Transistor Logic

Diode Gates:-

→ 2-input diode AND gate



For,
V_A 0, V_B = 10V

V_A = 0, V_B = 0

V_A = 10, V_B = 0

If $V_A = V_B = 0$ V

A	B	O/P
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate

Q. A certain gate has a propagation delay of 5ns. $I_{CCH} = 1$ mA, $I_{CCL} = 2.5$ mA, $V_{CC} = 5V$. SPP?
(high supply current; speed power product)

→ Given :-

$$P_D = \frac{(I_{CCH} + I_{CCL})}{2} V_{CC}$$

$$= \frac{3.5 \times 10^{-3}}{2} \times 5$$

$$= 8.75 \times 10^{-3} W$$

$t_p = 5$ ns.

$$SPP = 8.75 \times 10^{-3} \times 5 \times 10^{-9}$$

$$= 43.75 \times 10^{-12}$$

$$= 43.75 \text{ pJ}.$$

Q. $I_{IH} = 40\ \mu A$, $I_{IL} = 1.6$ mA, $I_{OH} = 400\ \mu A$ and $I_{OL} = 16$ mA. find the unit load.

→ Unit load $= \frac{I_{OH}}{I_{IH}} = \frac{I_{OL}}{I_{IL}}$

$$= \frac{400 \times 10^{-6}}{40 \times 10^{-6}}$$

$$= 10$$

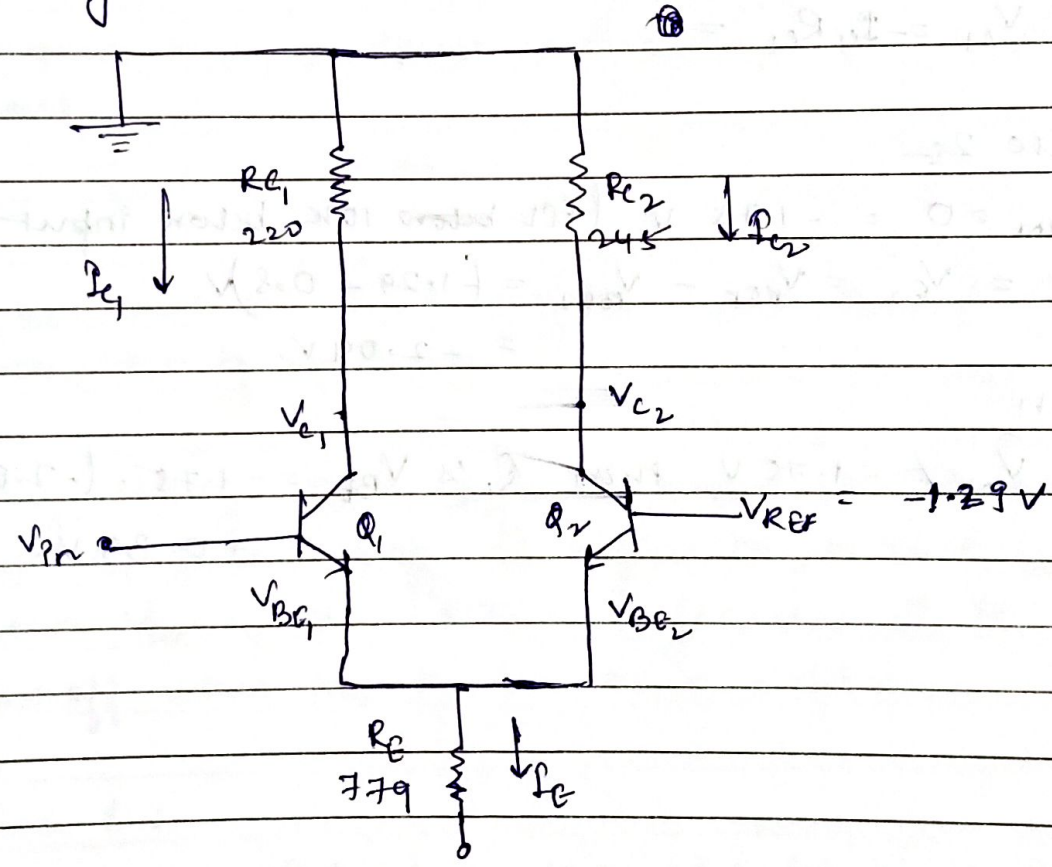
Page No. :
Date: / /

Emitter Coupled Logic (ECL):-

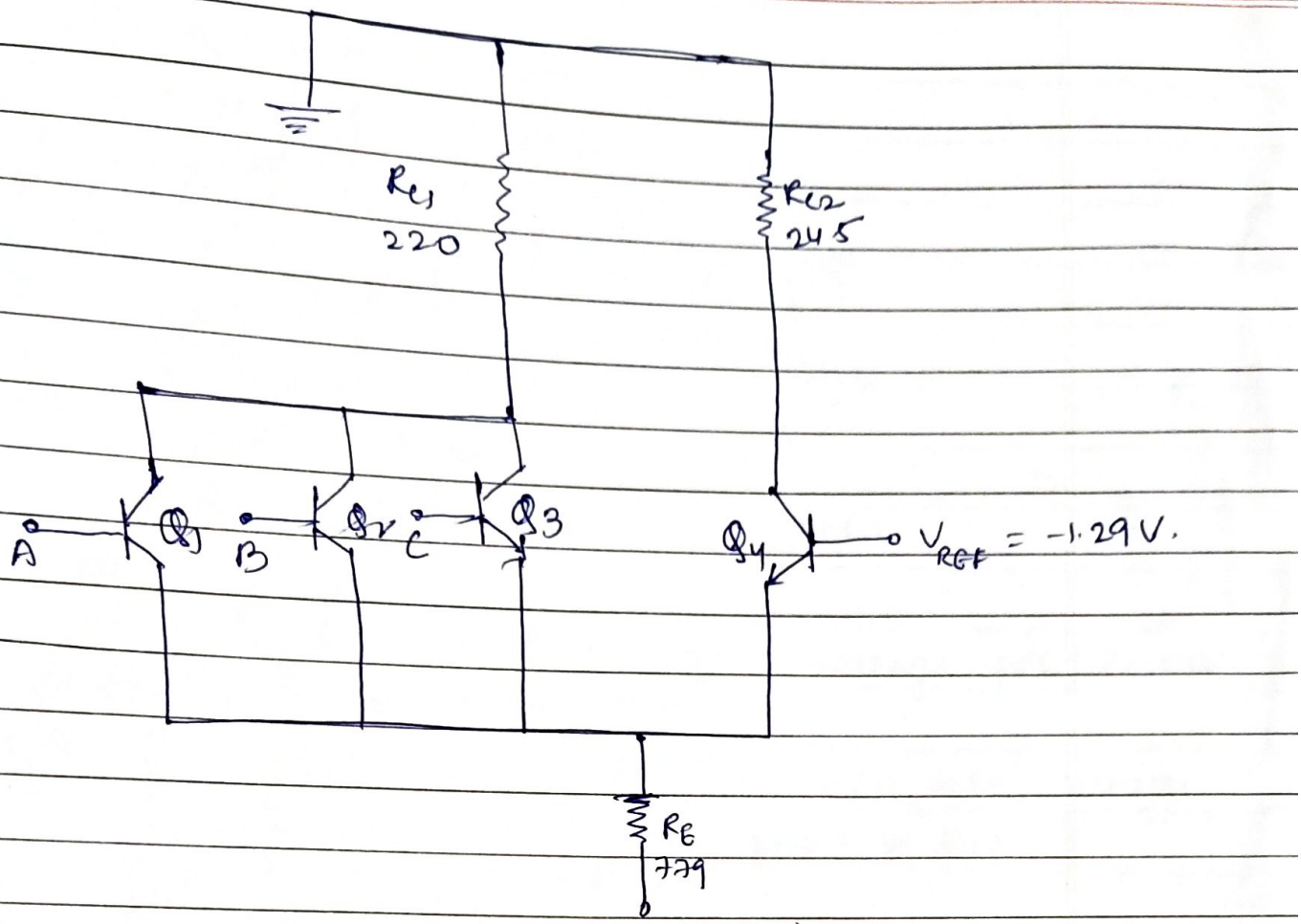
→ Because the output of the ECL switching transistor is taken from the emitter and coupled to a reference transistor emitter, the new logic family came to be known as emitter coupled logic (ECL).

→ It is not very widely used as compared to CMOS, and TTL,

↔ Because it is much more different logic family. the reason for this is the speed with which ECL operates and is much narrower noise margins.



ECL Differential Amplifier.



3. Input ECL circuit.

→ In analog applications the output signal is the voltage gain multiplied by the difference b/w the two input signal; → In switching applications, the inputs are restricted to a defined value for a logical 1 or 0.

→ The differential amplifier configuration provides a circuit that can compare the logic input level to a reference and generate output level. The input is provided to the base of Q_1 and the reference is provided to the base Q_2. Both transistor-emitter form a pair with a pull-down resistor to voltage V_{EE}. ECL circuits operate using a negative 5.2 V power supply.

MODULE 2

1. Minterms (Standard Product) → complemented
   Product of all variables either primed or unprimed. → uncomplemented

$$f(x, y, z) = xyz \xrightarrow{m_7},\ \bar{x}\bar{y}\bar{z} \xrightarrow{m_0},\ \bar{x}y\bar{z} \xrightarrow{m_2}$$

2. Maxterms (Standard sum)
   → Summation of all variables either primed or unprimed.

$$f(x, y, z) = (x+y+z) \xrightarrow{M_0} (\bar{x}+\bar{y}+\bar{z}) \xrightarrow{M_7} (\bar{x}+y+\bar{z}) \xrightarrow{M_5}$$

For Minterm →
- 1 → unprimed variable.
- 0 → primed variable.

⇒ Minterms represented by m.

⇒ Maxterms represented by M.

For Maxterms →
- 1 → primed variable
- 0 → unprimed variable.

$f(x,y) = \Sigma(0,1)$

x	y	Minterms	Maxterms	Output
0	0	$\bar{x}\bar{y}$	$x+y$	1
0	1	$\bar{x}y$	$x+\bar{y}$	1
1	0	$x\bar{y}$	$\bar{x}+y$	0
1	1	xy	$\bar{x}+\bar{y}$	0

$f(x,y) = \pi(2,3)$

→ A Boolean fn which is product of maxterms for which its value is 0

Q. Convert the Boolean Function into standard SOP form.

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

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

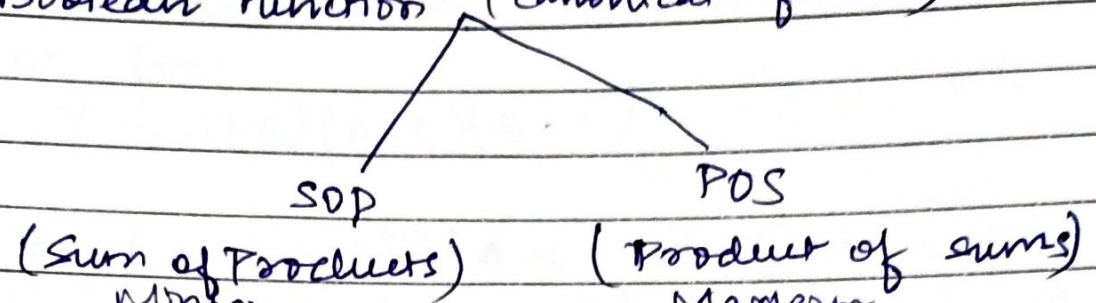
$Y = AB(C+\bar{C}) + A\bar{C}(B+\bar{B}) + BC(A+\bar{A})$

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

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

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

⇒ Boolean Function (canonical forms)



→ A Boolean function is a summation of minterms for which its value is 1.

~~f(x,y)~~

SOP form

$$f(X,Y) = \bar{X}\bar{Y} + \bar{X}Y \rightarrow SOP$$
$$= m_0 + m_1$$
$$= \Sigma(m_0 + m_1)$$
$$= \Sigma m(0,1)$$
$$= \Sigma(0,1)$$

POS form

$$f(X,Y) = (\bar{X}+Y).(\bar{X}+\bar{Y})$$
$$= M_2 \cdot M_3$$
$$= \pi M(2,3)$$
$$= \pi(2,3)$$

~~$f(\bar{x},y) = \Sigma(0,1)$~~

Q. Convert the Boolean function into standard POS form.

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

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

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

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

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

- Simplification :-

1. Algebraic Manipulation.
2. Map Method (K-Map)
3. Tabulation Procedure (Quine Mclausky method)

(1) $F = AB + AB' + A'B$

$= A(B + B') + A'B$

$= A \cdot 1 + A'B$

$= A + A'B$

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

$= A + B$

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

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

$= B \cdot 1 + \bar{A}\bar{B}$

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

$= \bar{A} + B$

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

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

$= A$

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

(5) $\bar{x}yz + xz + \bar{x}z$

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

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

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

$= 0$

(5) $\bar{x}yz + xz + \bar{x}z$

→ $\bar{x}yz + z(x + \bar{x})$

$\Rightarrow \bar{x}yz + z.1$

$= z(\bar{x}y + 1)$

$= z(\bar{x}y + 1)$

$= z.1$

$= z$

$1 + A = 1$

$\bar{x}(yz + z)$

(6) $F = \Sigma(1, 2, 4, 7)$

$= m_1 + m_2 + m_4 + m_7$

$= \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$

$= \bar{a}(\bar{b}c + b\bar{c}) + \bar{b}\bar{c}(a + \bar{a})$

$= \bar{a}(\bar{b}c + b\bar{c}) + \bar{b}\bar{c}$

$= \bar{a}\bar{b}c + bc + a\bar{b}\bar{c}$

$= c(\bar{a}\bar{b} + b) + a\bar{b}\bar{c}$

$= c(b + \bar{b})(\bar{a} + b) + a\bar{b}\bar{c}$

$= c(\bar{a} + b) + a\bar{b}\bar{c}$

EX-NOR

Ex. NOR

$=$

$= \bar{a}\bar{b}c + a\bar{b}c + \bar{a}b\bar{c} + abc$

$= \bar{b}\bar{c}(\bar{a} + a) + b(\bar{a}\bar{c} + ac)$

$= \bar{b}c' + b(\bar{a}\bar{c} + ac)$

$= \bar{b}c + \bar{a}b\bar{c} + abc$

$= ac(\bar{b} + ba) + \bar{a}b\bar{c}$

$= c(\bar{b} + b)(a + \bar{b}) + \bar{a}b\bar{c}$

$= ca + \bar{b}c + \bar{a}b\bar{c}$

$=$

1. $Y = \Sigma(2,4,6)$

2. $Y = \pi(1,3,5)$

1. Sol) $Y = \Sigma(2,4,6) = m_2 + m_4 + m_6$

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

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

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

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

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

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

2 Sol) $Y = \pi[1,3,5]$

$= M_1 + M_3 + M_5$

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

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

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

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

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

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

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

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

$= 0 + AB + A\bar{C} + \bar{A}\bar{C} + B\bar{C} + \bar{C}$

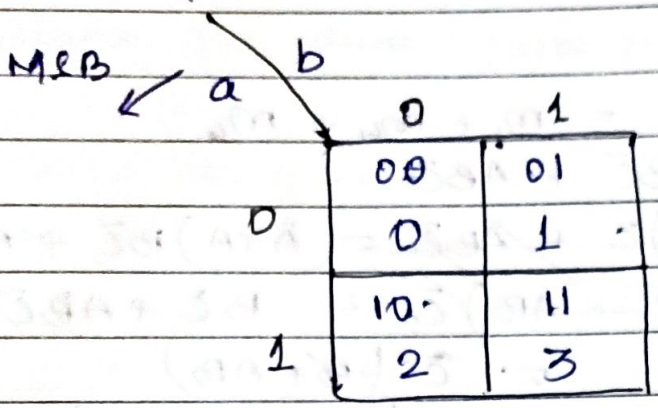
$= AB + \bar{C} + B\bar{C}$

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

$= AB + \bar{C}$

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

⇒ Karnaugh Map (K-Map) :-



Q. $Y(a,b) = \Sigma(0,1)$

→

a \ b	0	1
0	0=00	1=01
	1	1
1	2=10	3=11
	0	0

→ $\bar{a}$ (Here we have not taken b as it is not common in both the cells.)

Here, in this ques. grouping will be in SOP form:

Algebraically,

$Y(a,b) = \bar{a}\bar{b} + \bar{a}b$

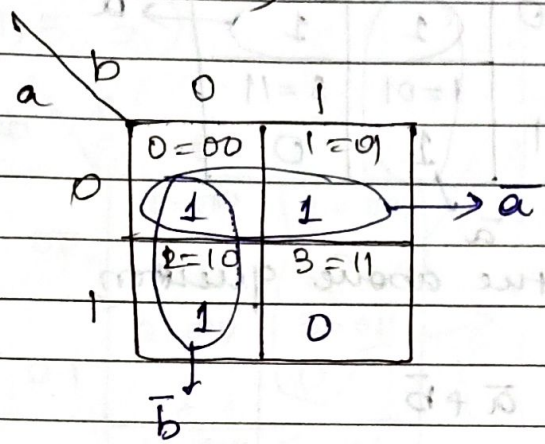
$= \bar{a}(\bar{b}+b)$

$= \bar{a}$.

→ Grouping can be formed only in the powers of 2, i.e., $2^1, 2^2, 2^3, \ldots$.

→ In K-map grouping of adjacent cells are done.

Q $Y(a,b) = \Sigma(0, 1, 2)$



So, by K-Map we have

$Y(a,b) = \bar{a} + \bar{b}$

and

Algebraically

$Y(a,b) = \bar{a}\bar{b} + \bar{a}b + a\bar{b}$

$= \bar{a}(\bar{b}+b) + a\bar{b}$

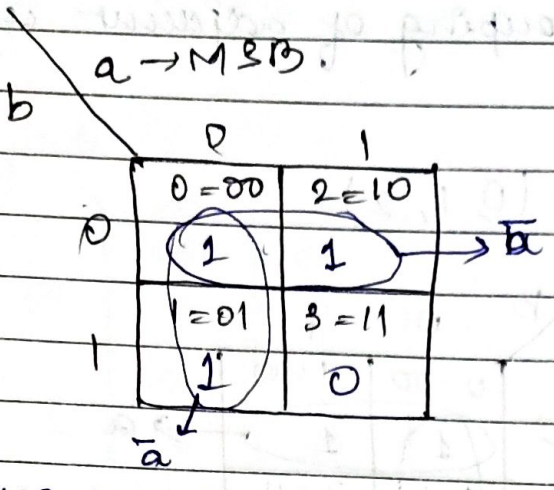
$= \bar{a}(\bar{a}+a) + \bar{a}b$

$= \bar{b} + \bar{a}b$

$= (\bar{a}+\bar{b})(b+\bar{b})$

$= (\bar{a}+\bar{b})$. Ans.

⇒ Pf:

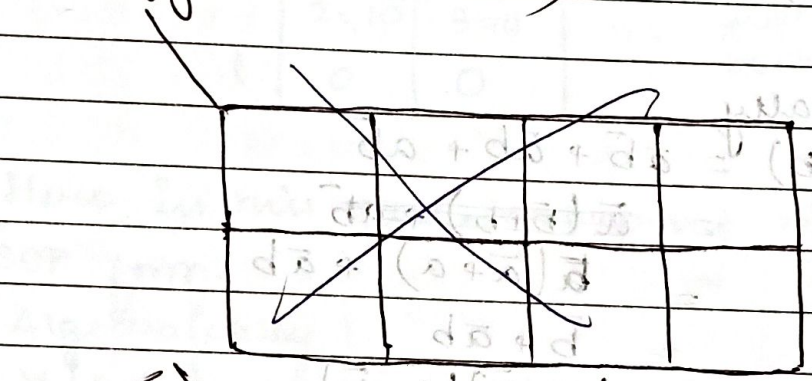


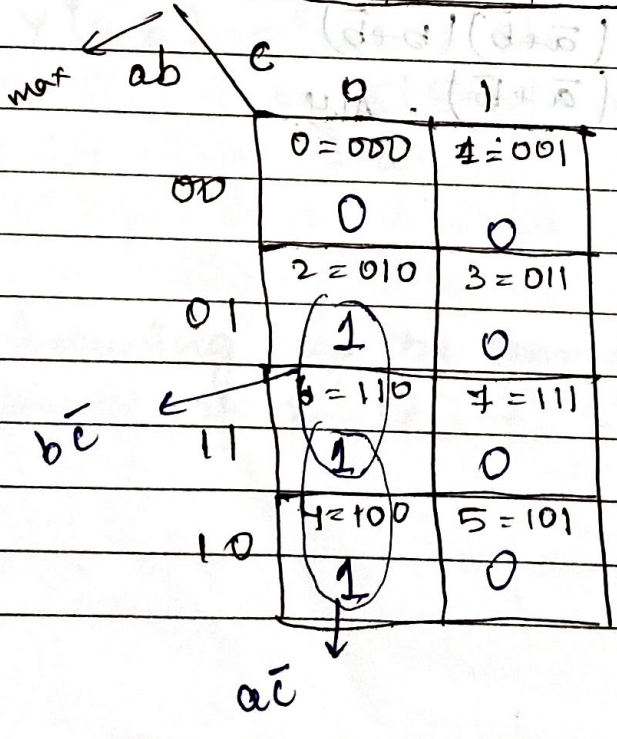
considering the above question, we have,

$Y(a,b) = \bar{a} + \bar{b}$

Q Simplify $Y = \Sigma(2,4,6)$

→





$Y = a\bar{c} + b\bar{c}$

$= (a+b)\bar{c}$ Ans.

Q $Y = \pi(1,3,5)$

Sol

ab \ c	0	1
00	0 = 000 → 1	1 = 001 → 0
01	2 = 010 → 1	3 = 011 → 0
11	6 = 110 → 1	7 = 111 → 1
10	4 = 100 → 1	5 = 101 → 0

$(a+\bar{c})$

$(b+\bar{c})$

So,

$Y = (a+\bar{c})(b+\bar{c})$

$= ab + \bar{c}$

Q. $f(x,y,z) = \Sigma(0,2,3,7)$

Q $f(x,y,z) = \Sigma(0,2,4,5,6)$.

$\frac{Y(s)}{U(s)} = C(sI-A)^{-1}B + D$ $\quad \dot{x} = Ax + Bu$, $y = Cx + Du$

Q. $f(x, y, z) = \Sigma(0, 2, 3, 7)$

→

xy \ z	0	1
00	0=000 1	1 0
01	2=010 1	3=011 1
11	6 0	7=111 1
10	4 0	5 0

$\bar{x}y$ → 3, 7

$\bar{x}\bar{z}$ → 0, 2

yz → 3, 7

$f(x, y, z) = \bar{x}\bar{z} + \bar{x}y + yz$ Ans

x \ yz	00	01	11	10
0	0=000 1	1 0	3=011 1	2=010 1
1	4 0	5 0	7=111 1	6 0

$\bar{x}y$ → 3, 7

$\bar{x}\bar{z}$ → 0, 2

Essential prime Implicant (EPI)

PI(0, 2) ; PI(3, 7) ; PI(2, 3). ← Prime Implicants

Q. $f(x, y, z) = \Sigma(0, 2, 4, 5, 6)$

→ $f(x, y, z) = x\bar{y} + y\bar{z} + \bar{y}\bar{z}$. Ans.

x \ yz	00	01	11	10
0	0=000 1	1 0	3 0	2=010 1
1	4=100 1	5=101 1	7 0	6=110 1

$y\bar{z}$ → 2, 6

$\bar{y}\bar{z}$ → 0, 4

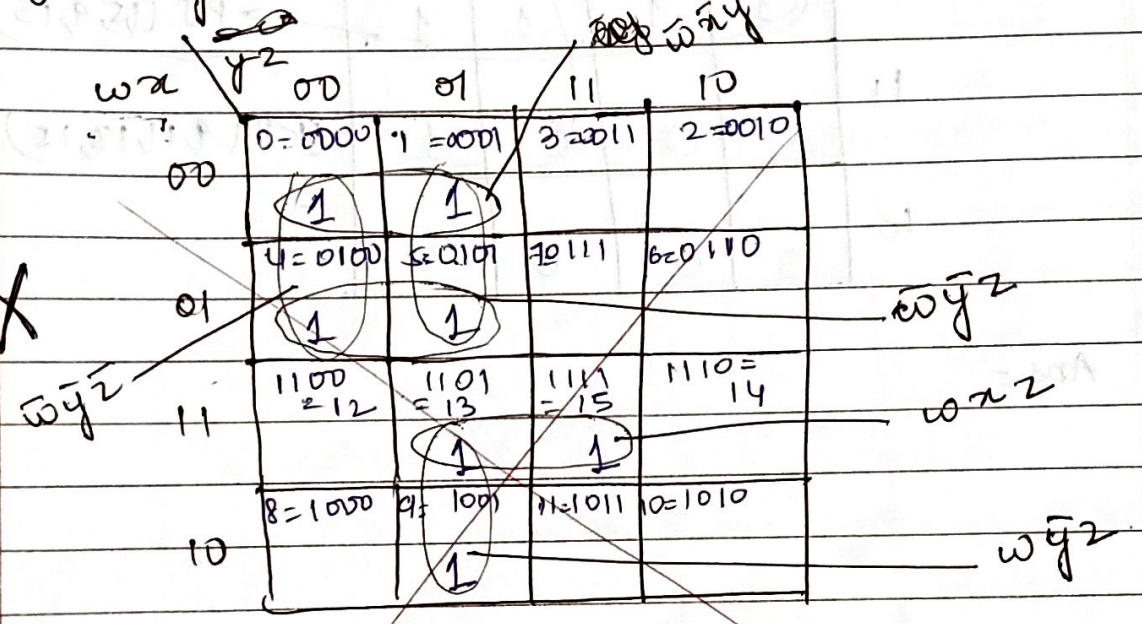
$x\bar{y}$ → 4, 5

⇒ Prime Implicant :- Any single minterm or permitted group of minterms is called an implicant of an output function. A prime implicant is a group of minterms that cannot be combined with any other minterms or groups.

⇒ Essential Prime Implicant :-
An essential prime implicant is a prime implicant in which one or more minterms are unique, ie. it contains atleast one minterm not contained in any other prime implicant.

Q. $f(w,x,y,z) = \Sigma(0,1,4,5,9,11,13,15)$

Ans →



⇒ ~~Number of~~ Order of variables should ~~be~~ must be taken into consideration.

$f(w,x,y,z) = \bar{w}\bar{x}\bar{y} + \bar{w}\bar{y}\bar{z} + \bar{w}\bar{y}z + wxz + w\bar{y}z$

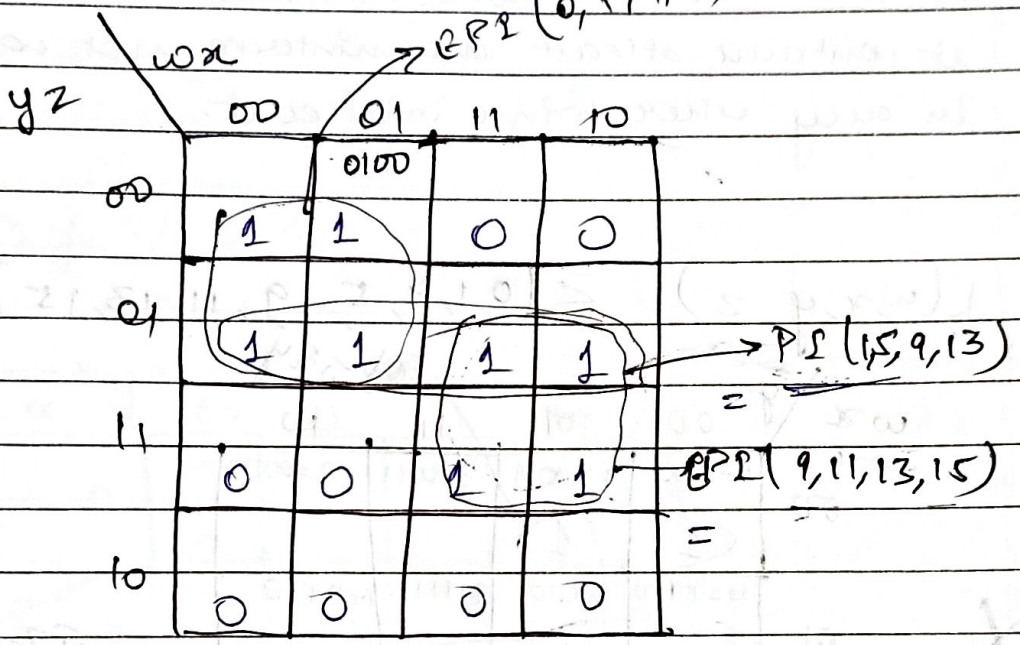
$= \bar{w}\bar{x}\bar{y} + \bar{w}\bar{y}(\bar{z}+z) + wxz + w\bar{y}z$

$= \bar{w}\bar{x}\bar{y} + \bar{w}\bar{y} + wxz + w\bar{y}z$

$= \bar{w}\bar{y}(\bar{x}+1) + wxz + w\bar{y}z$

$= \bar{w}\bar{y} + wxz + w\bar{y}z$

$= \bar{w}\bar{y}$



Ans =

Q. $f(w, x, y, z) = \Sigma(0, 2, 5, 7, 8, 10, 13, 15)$

→

yz \ wx	00	01	11	10
00	0 = 0000 1	4 = 0100 0	12 = 1100 0	8 = 1000 1
01	1 = 0001 0	5 = 0101 1	13 = 1101 1	9 = 1001 0
11	3 = 0011 0	7 = 0111 1	15 = 1111 1	11 = 1011 0
10	2 = 0010 1	6 = 0110 0	14 = 1110 0	10 = 1010 1

EPI (0, 8, 2, 10) → $\bar{z}\bar{x}$

zx

EPI (5, 7, 13, 15)

$f(w, x, y, z) = xz + \bar{x}\bar{z}$

$= x \odot z$ Ans

Q. $f(w, x, y, z) = \Sigma(1, 3, 4, 6, 9, 11, 12, 14)$

→

yz \ wx	00	01	11	10
00	0 0	4 = 0100 1	12 = 1100 1	8 0
01	1 = 0001 1	5 0	13 0	9 = 1001 1
11	3 = 0011 1	7 0	15 0	11 = 1011 1
10	2 0	6 = 0110 1	14 = 1110 1	10 0

zx

$z\bar{x}$

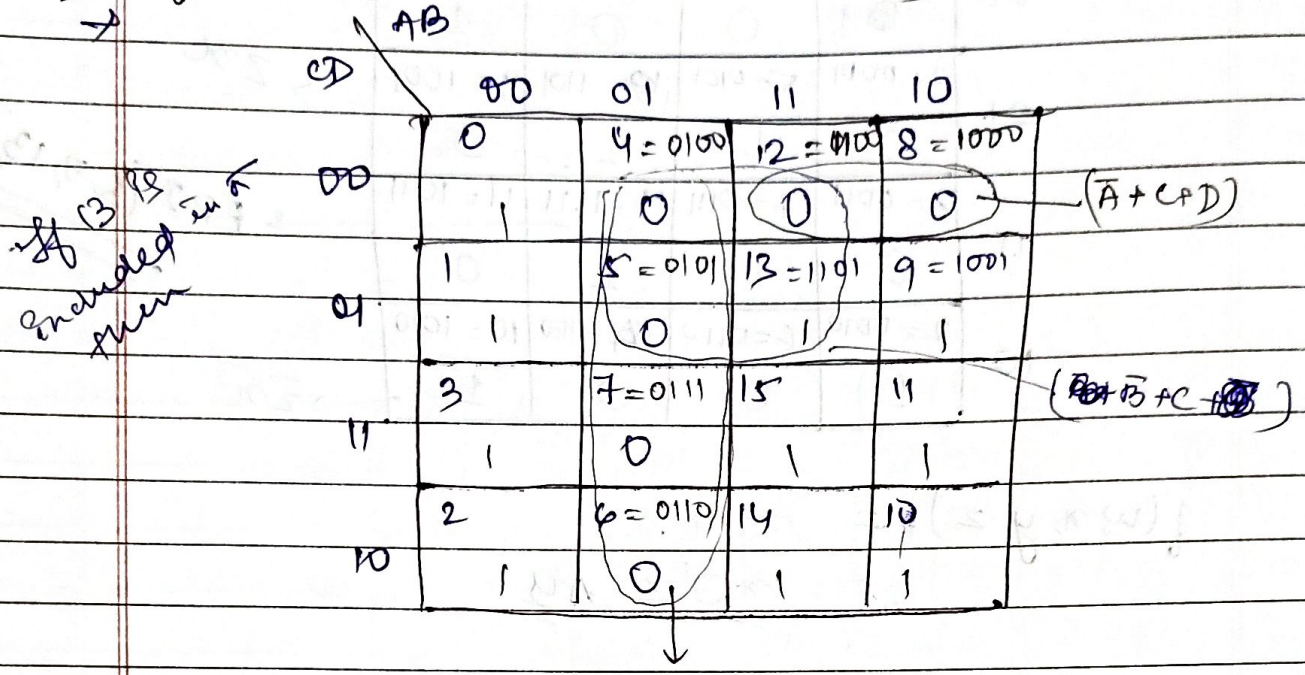
$f(w, x, y, z) = \bar{x}z + x\bar{z}$

$= x \oplus z$ Ans

Q. $f(a,b,c,d) = \Sigma(1,3,4,6,9,11,12,14)$.

Q. $f(A,B,C,D) = \pi(4,5,6,7,8,12)$

→



$f(A,B,C,D) = (A+\bar{B})(\bar{A}+C+D)$ Ans.

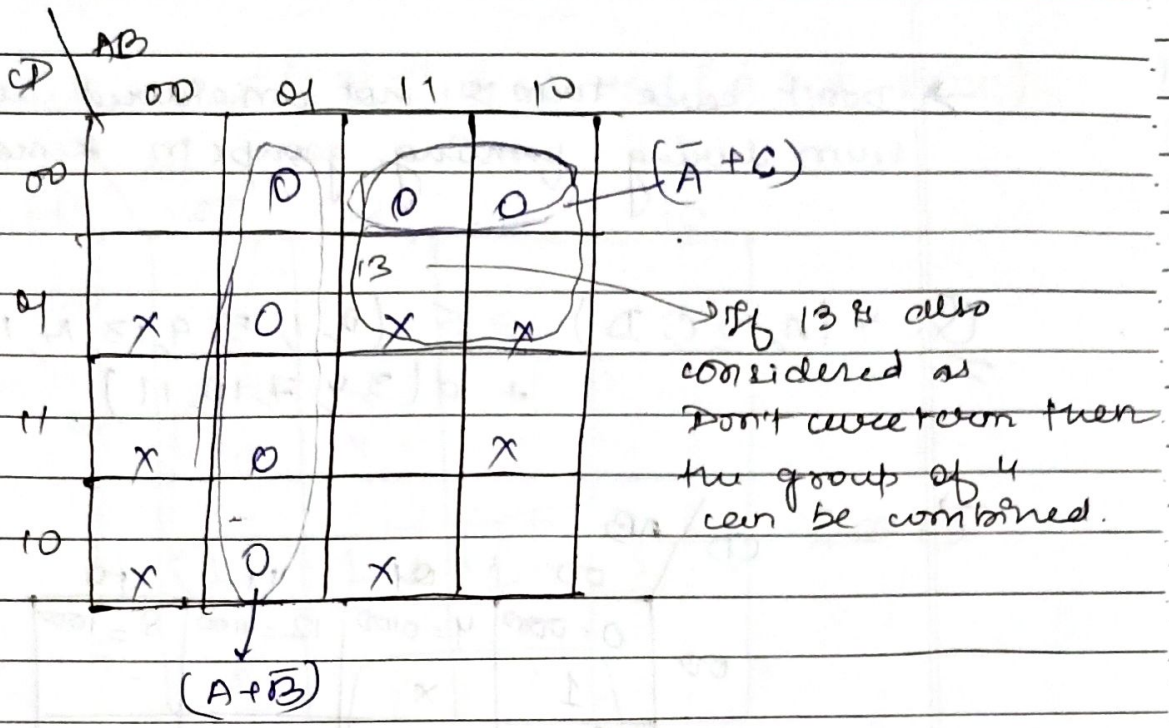
⇒ Don't Care Terms :-

Q. $d(1,2,3,9,11,14)$. For the previous ques.

↓ denotes don't care terms.

→ For this ques, don't care terms cannot be combined.

$f(A,B,C,D) = (A+B)(\bar{A}+C+D)(\bar{B}+C)$ Ans.



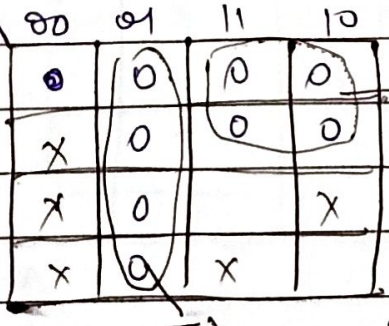
So,
For this case:-

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

→ If all the groups are combined and after that it will be grouped then it will be redundant.

→ If any one of the term is unique & not present in any group, then it is not considered as redundant.

Q $f(A,B,C,D) = \pi(4,5,6,7,8,12,13,9)$



$f(A,B,C,D) = (A+\bar{B})(\bar{A}+C)$. Ans

Ans → $\bar{D} + AC + \bar{A}C$

→ Don't cares term is not considered as unique term during forming groups in K-map.

Q. $Y(A,B,C,D) = \Sigma(0,1,5,9,13,14,15) + d(3,4,7,10,11)$

→

AB \ CD	00	01	11	10
00	0 = 0000, 1	4 = 0100, X	12 = 1100	8 = 1000
01	1 = 0001, 1	5 = 0101, 1	13 = 1101, 1	9 = 1001, 1
11	3 = 0011, X	7 = 0111, X	15 = 1111, 1	11 = 1011, X
10	2 = 0010	6 = 0110	14 = 1110, 1	10 = 1010, X

$\bar{A}\bar{C}$ $\bar{D}$ AC

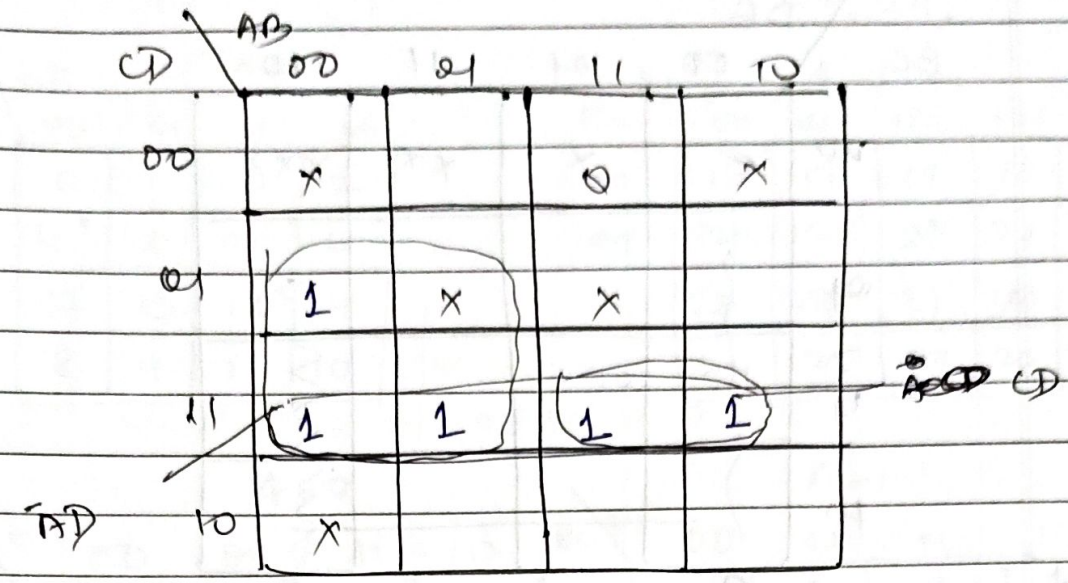
~~$Y(A,B,C,D) = \bar{A}\bar{C} + AB\bar{C} + AD$~~

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

Page No.:
Date: / /

Q $Y(A,B,C,D) = \Sigma(1,3,7,11,15) + d(0,2,5,8,14)$

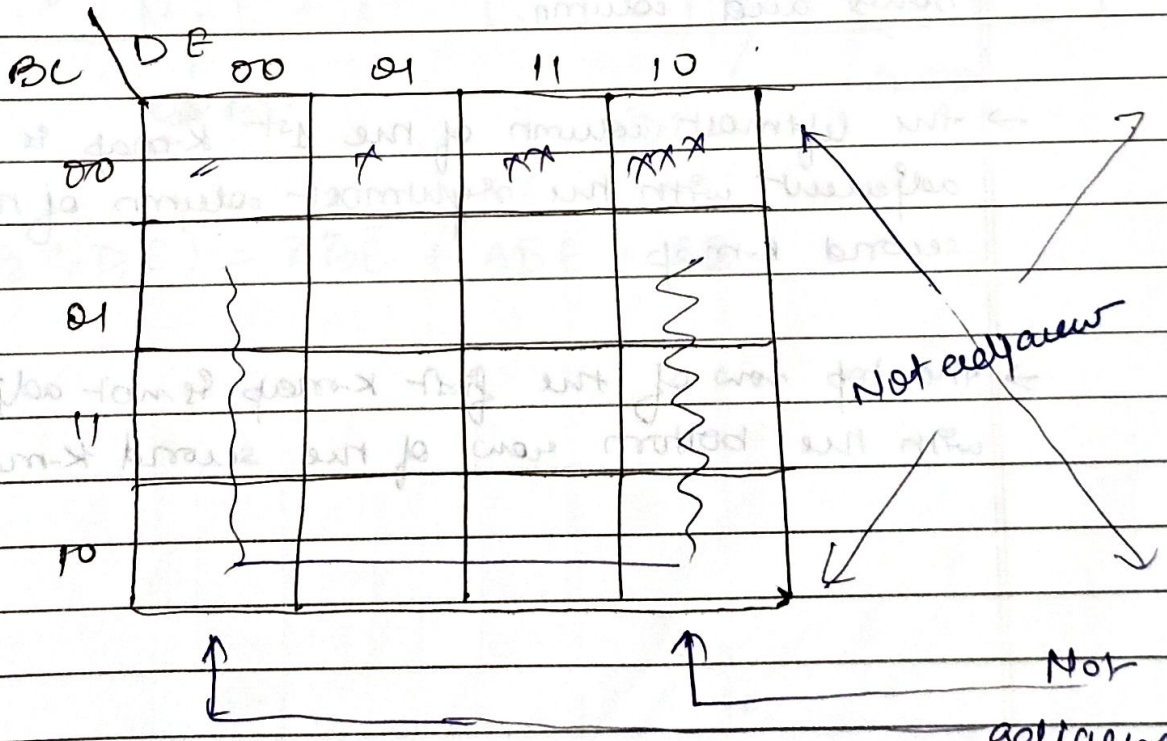
→



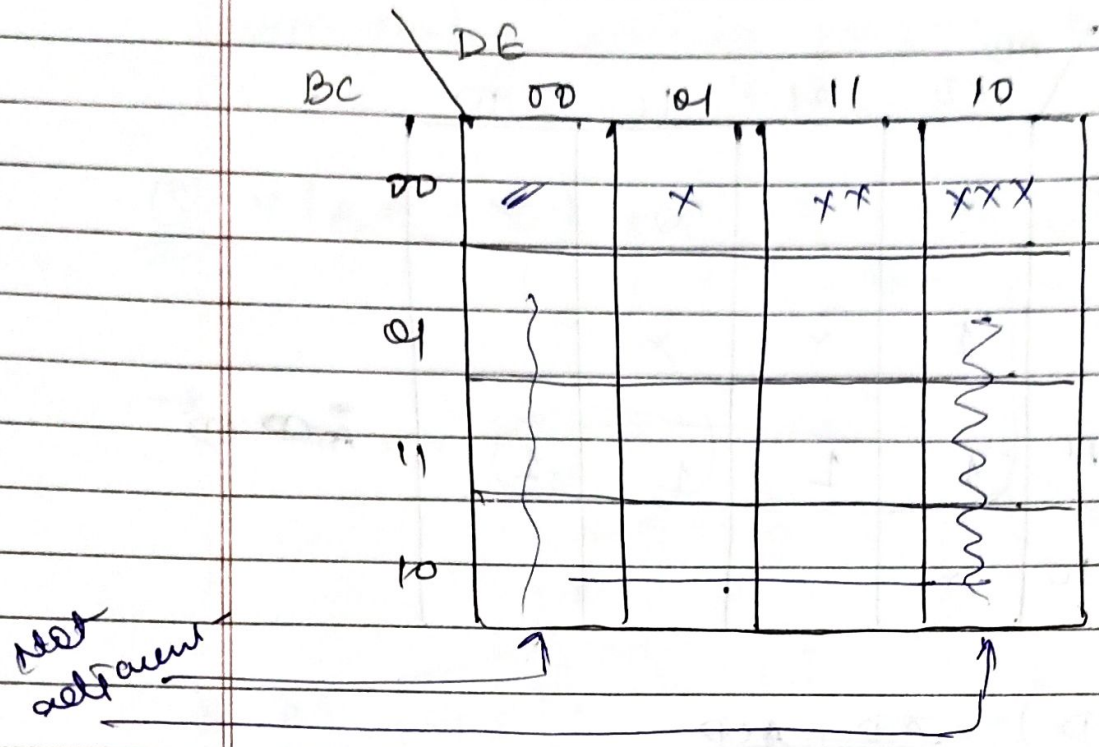
$Y(A,B,C,D) = \cancel{AD + ACD}$

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

A=0



A = 1



→ Every shell in 1st K-map is adjacent to the corresponding shell in the 2nd K-map.

→ Every row and every column in the first K-map is adjacent to the corresponding rows and column.

→ The leftmost column of the 1st K-map is not adjacent with the rightmost column of the second K-map.

→ The top row of the first K-map is not adjacent with the bottom row of the second K-map.

Page No.:
Date: / /

Q. $f(A,B,C,D,E) = \Sigma(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$

A = 0

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

A = 1

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

A = 0

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

$\bar{A}\bar{B}\bar{E}$

BE

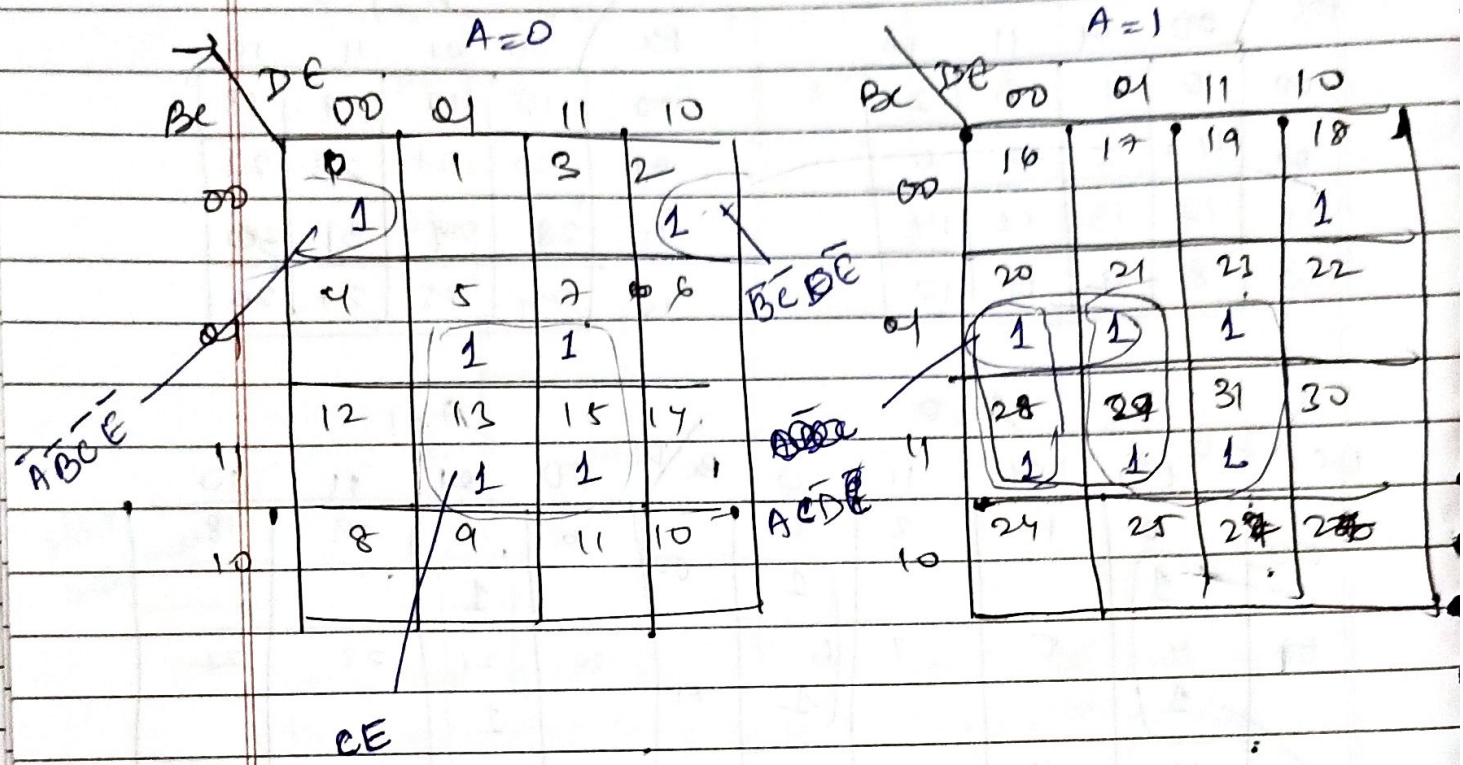
A = 1

BC \ DE	00	01	11	10
00	16	17, 1	19	18
01	20	21, 1	23	22
11	28	29, 1	31, 1	30
10	24	25, 1	27, 1	26

$A\bar{D}E$

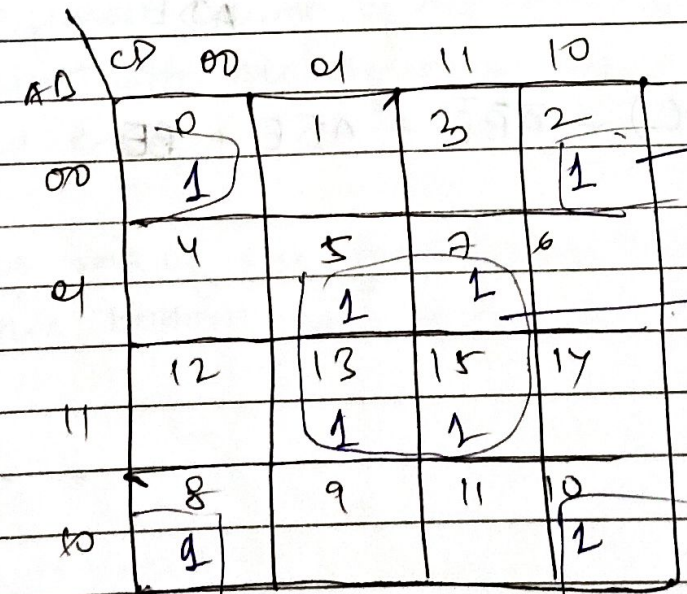
$f(A,B,C,D,E) = \bar{A}\bar{B}\bar{E} + A\bar{D}E + BE$

Q. $f(A,B,C,D,E) = \Sigma(0,2,5,7,13,15,18,20,21,23,28,29,31)$



$f(A,B,C,D,E) = \bar{A}B\bar{C}\bar{E} + \bar{B}C\bar{D}\bar{E} + CE + A\bar{C}\bar{D}\bar{E}$

Q. $f(A,B,C,D) = \Sigma(0,2,5,7,8,10,13,15)$



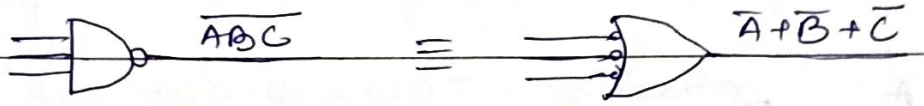
$f(A,B,C,D) = (BD + \bar{B}\bar{D}) = B \odot D$

Q. $f(a,b,c,d) = \Sigma(0,1,4,5,9,11,13,15)$

⇒ How to convert NAND-NAND / NOR-NOR implementation :-

→ NAND implementation.

Using De-Morgan's Theorem
NAND = Bubbled OR

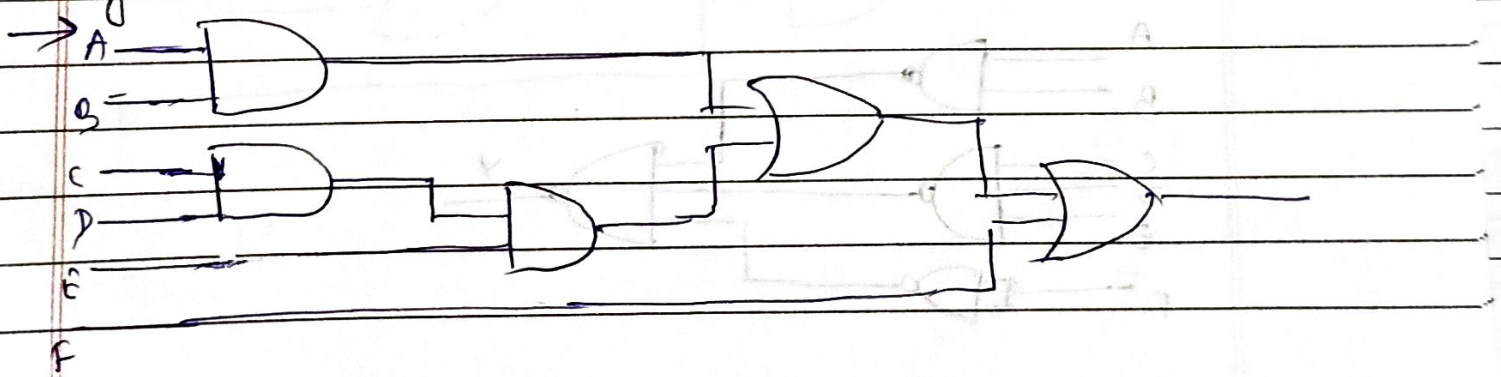


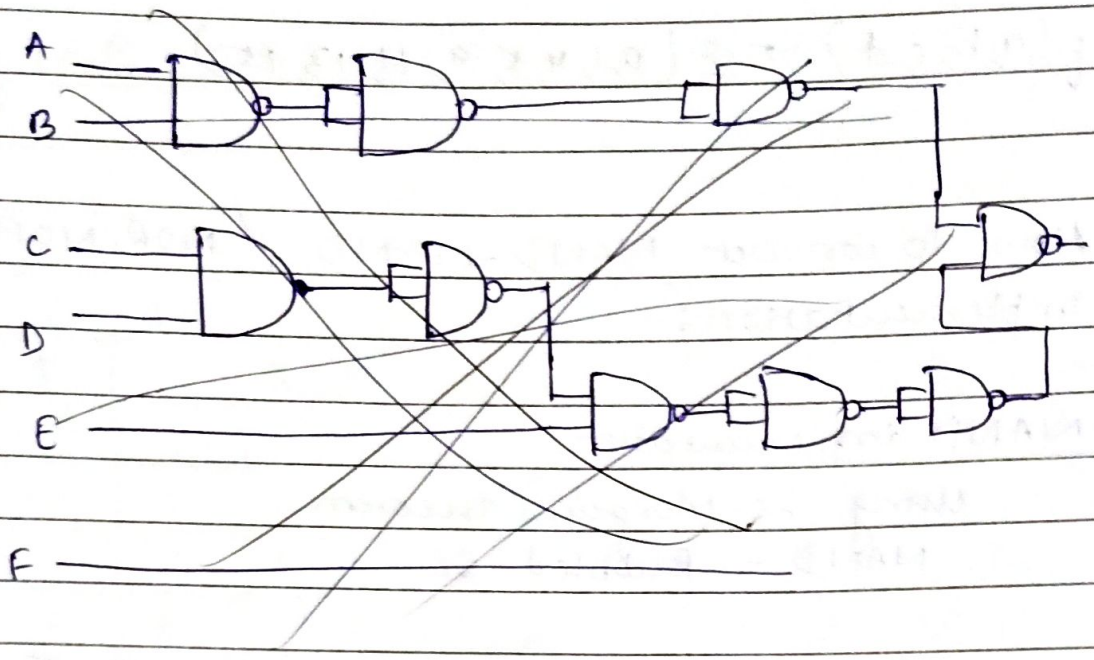
Procedure :-

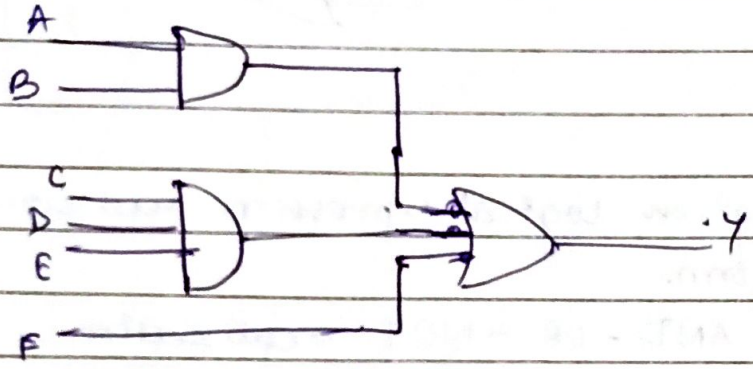
1. Simplify the given logical expression and convert it in the SOP form.
2. Draw the a AND-OR-NOT realization.
3. Then replace every AND gate by a NAND, every OR gate by a ~~NAND~~ bubbled or gate and NOT gate by a NAND inverter.
4. Finally draw the circuit using only NAND gates.

Q. $Y = AB + CDE + F$. Implement using only NAND gates.

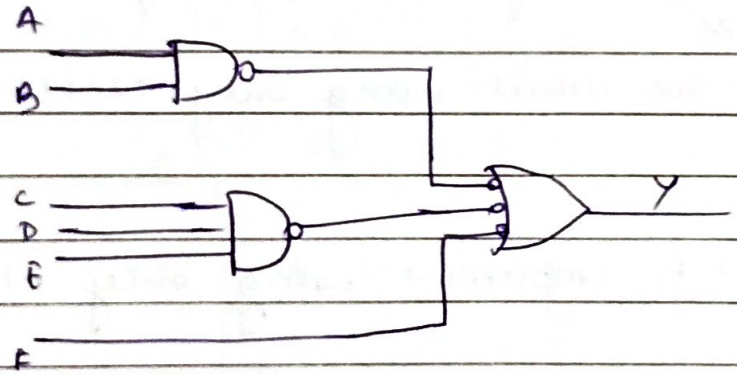
→

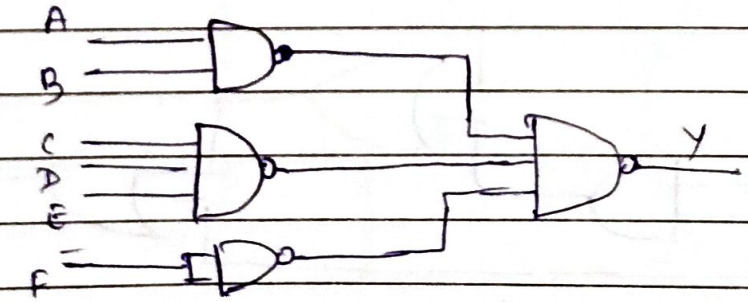






(1)

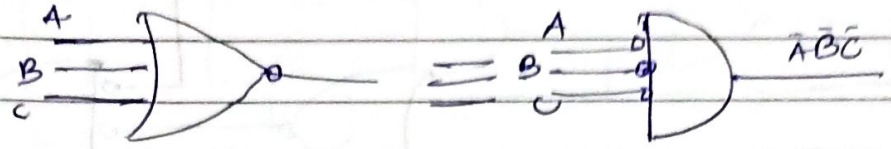




→ NOR. NOR implementation :-

Using De-Morgan's theorem

NOR ≡ Bubbled AND

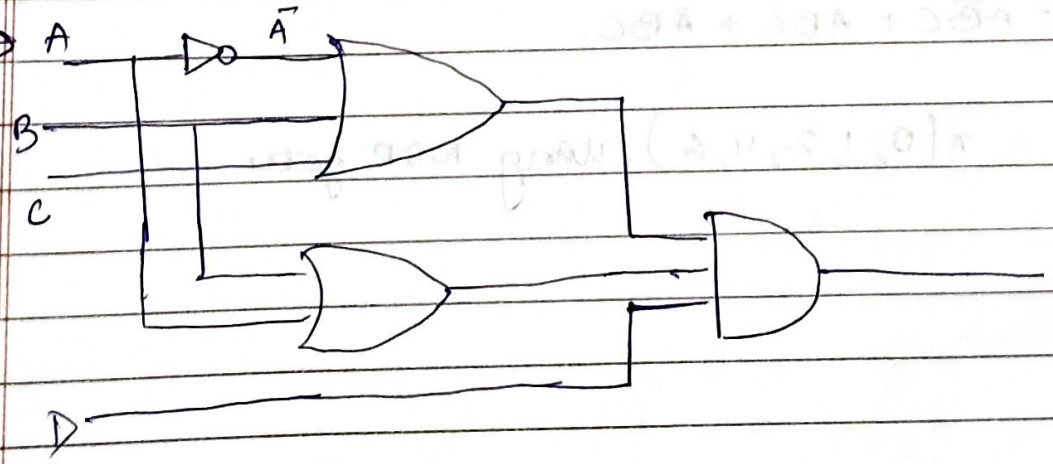


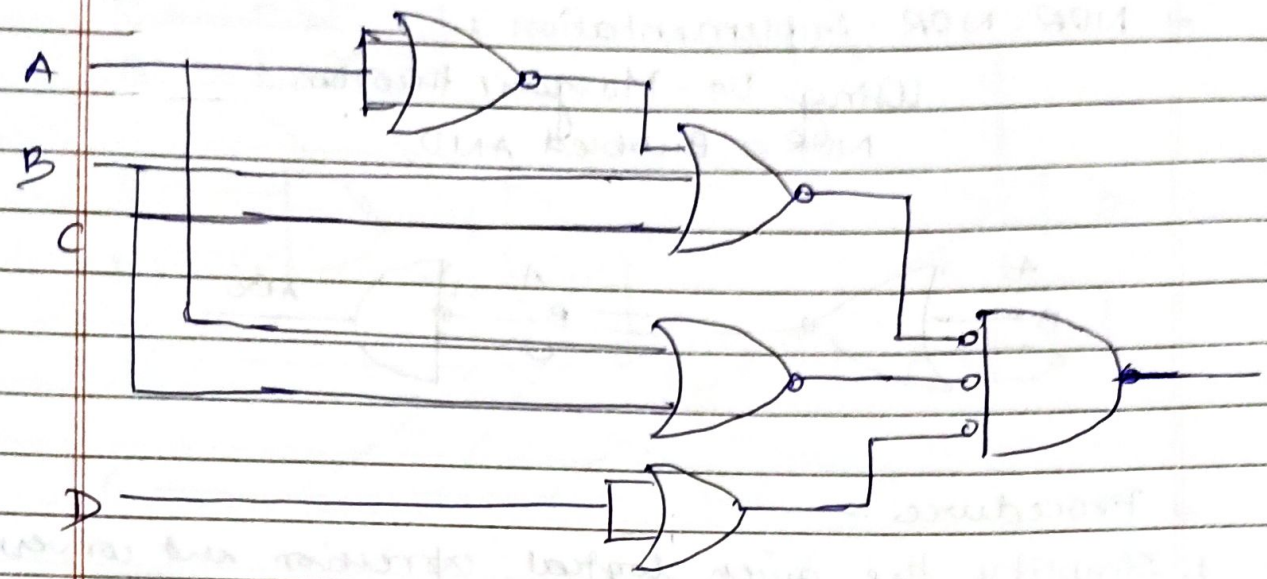
→ Procedure :-

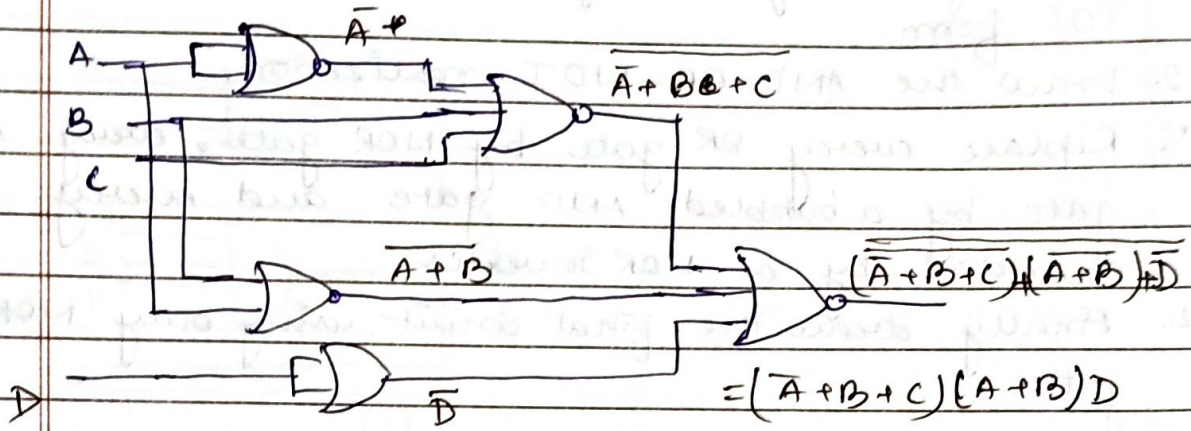
1. Simplify the given logical expression and convert into POS form.
2. Draw the AND-OR-NOT realization.
3. Replace every OR gate by NOR gate, every AND gate by a bubbled AND gate and every inverter by a NOR inverter.
4. Finally draw the final circuit using only NOR gates.

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

→





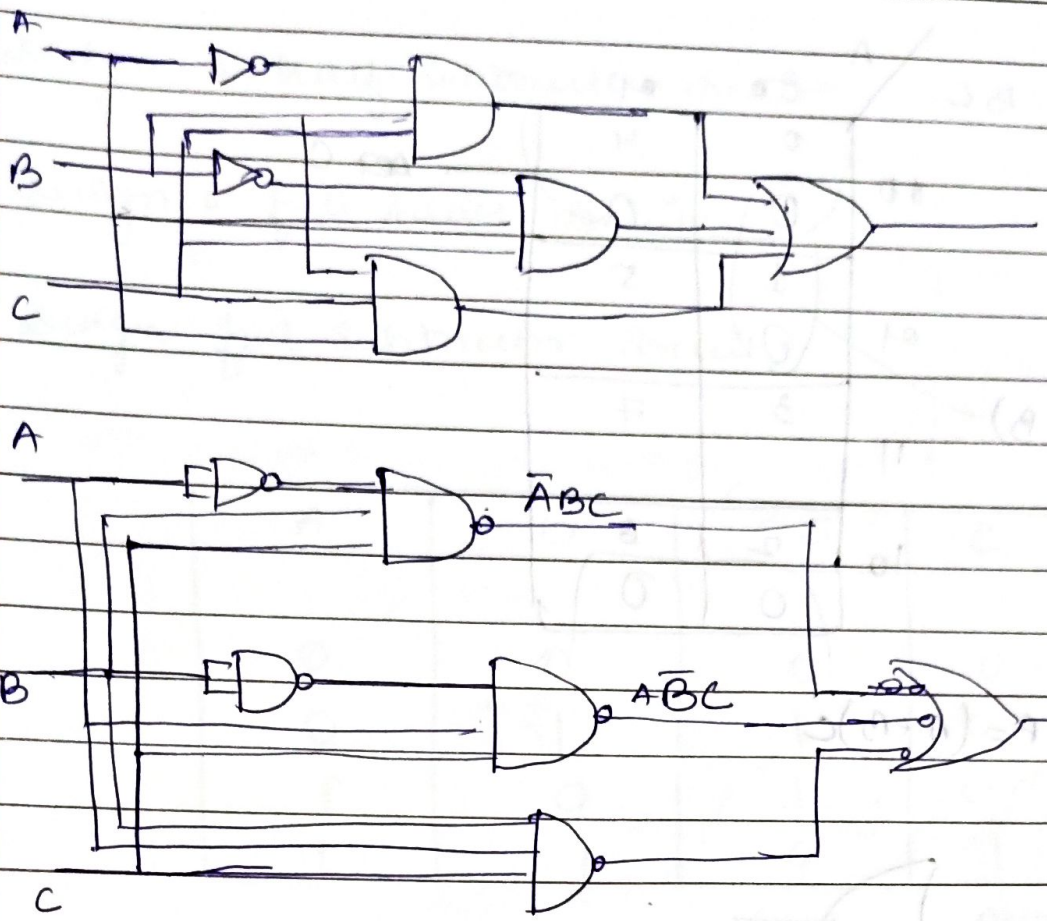


1. Q. $Y = A\bar{B}C + ABC + \bar{A}BC.$

2. Q. $F = \pi(0, 1, 2, 4, 6)$. Using NOR gates.

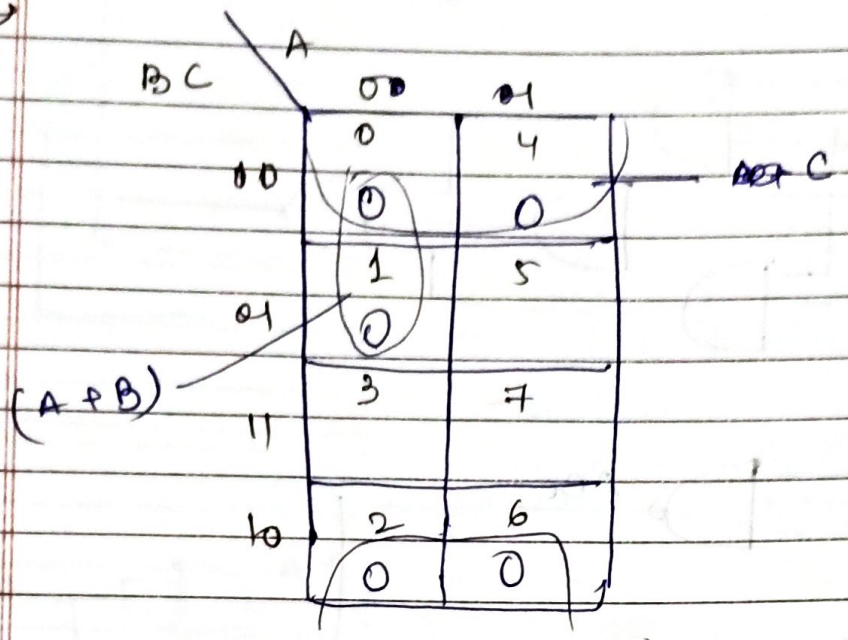
1. $Y = A\bar{B}C + ABC + \bar{A}BC$

→

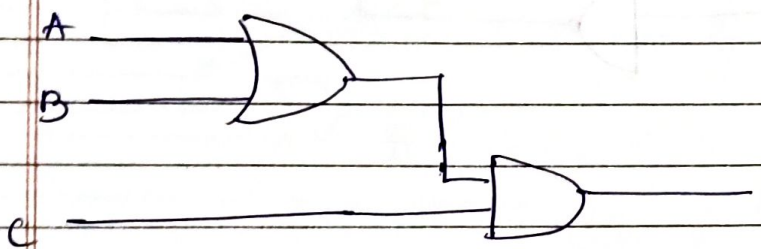


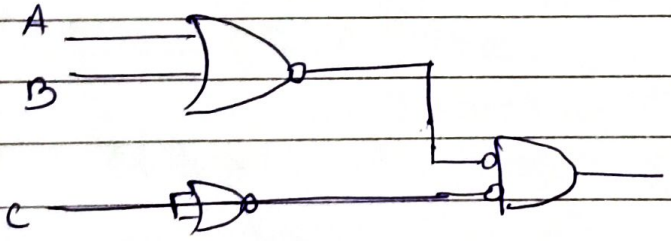
2. $F = \pi(0, 1, 2, 4, 6)$

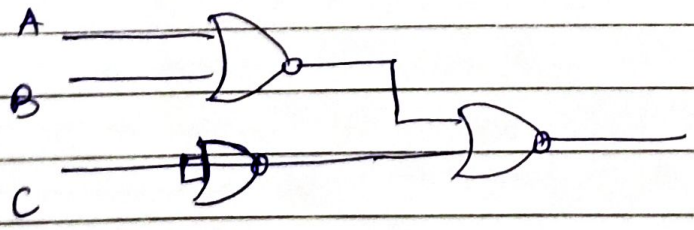
→



• $F = (A+B)C$







MODULE - 3

Logic Circuits

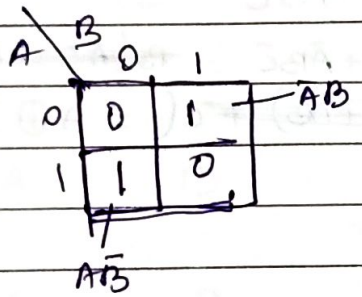
- Combinational Circuits
- Sequential Circuits

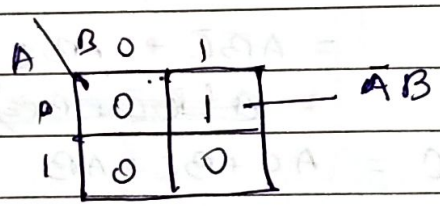
ALSO TRUTH TABLE FOR HALF SUBTRACTOR

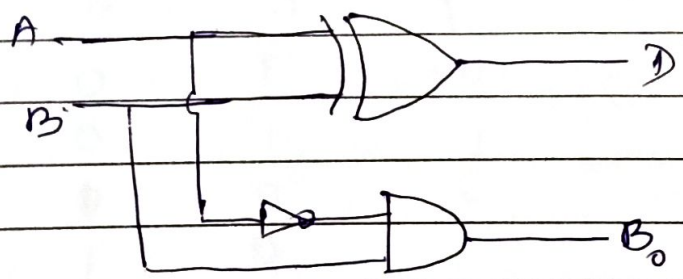
2 Sol)

Truth Table

A	B	D	B_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

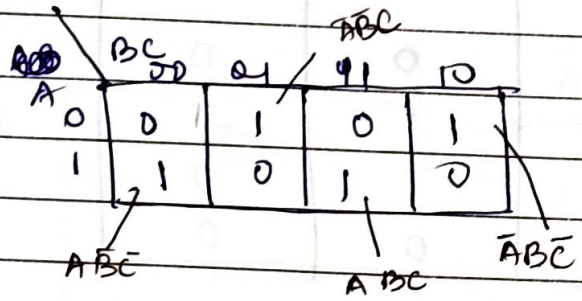


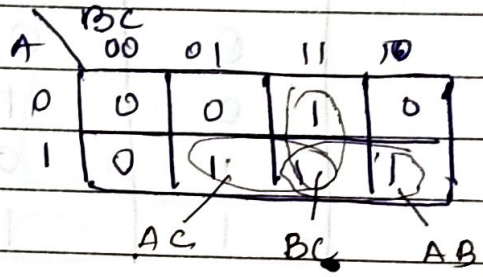




3. Sol,

A	B	C	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



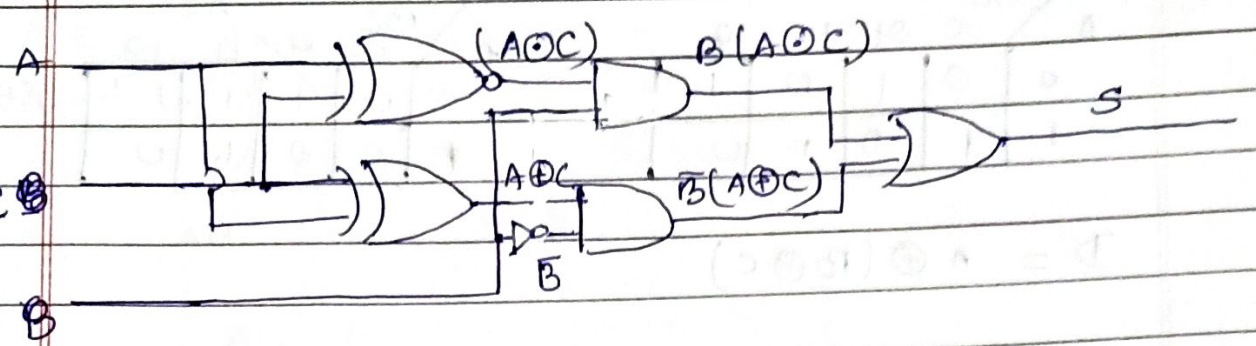


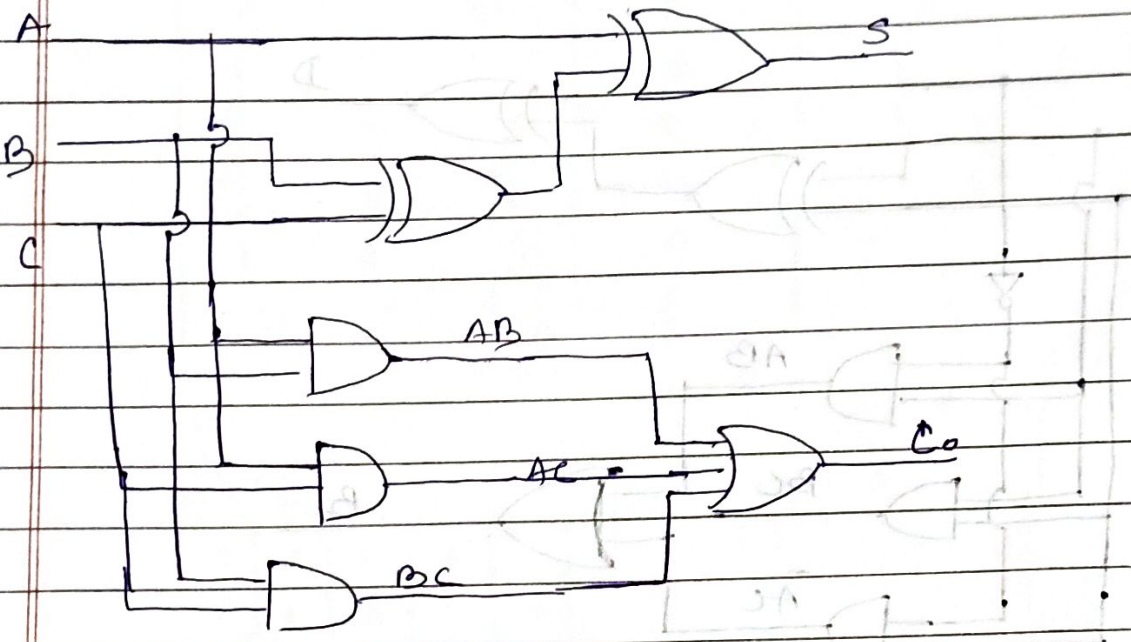
$S = A\bar{B}\bar{C} + ABC + \bar{A}B\bar{C} + \bar{A}\bar{B}C$

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

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

$C = AC + BC + AB$





ALSO TRUTH TABLE FOR FULL SUBTRACTOR

4. Sol

A	B	C	D	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

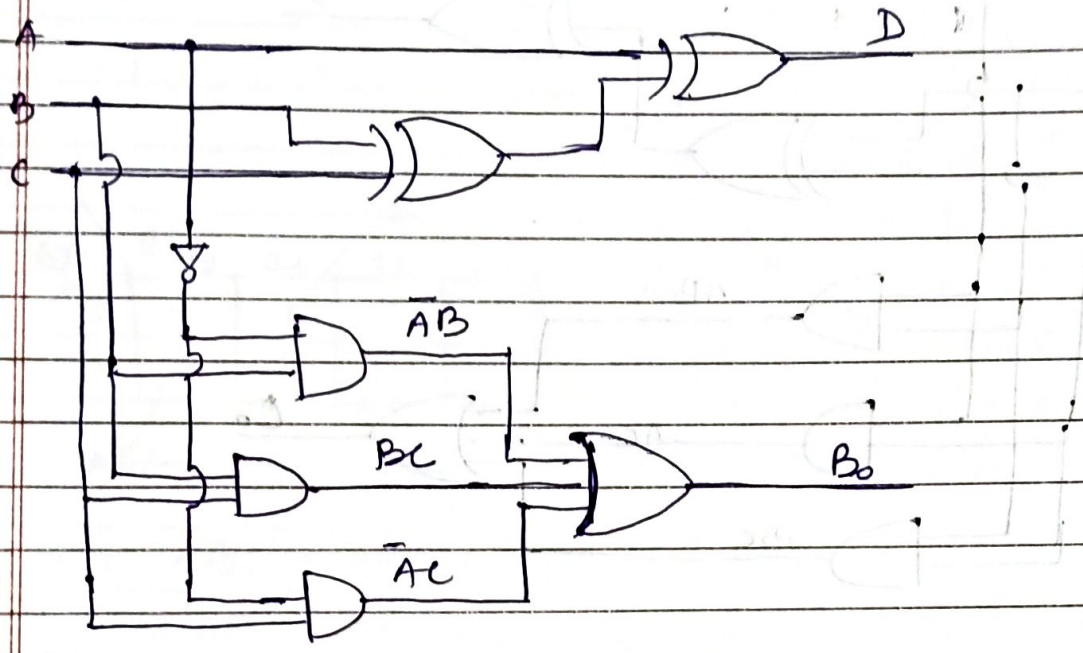
A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

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

$\bar{A}C$, $\bar{A}B$, BC

$D = A \oplus (B \oplus C)$

$B_0 = \bar{A}C + \bar{A}B + BC.$

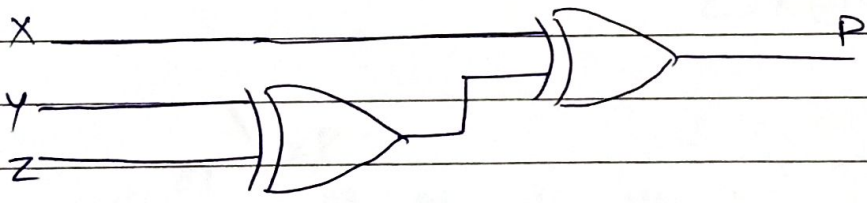


⇒ Even Parity Generator :-

Inputs			Output
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

X \ YZ	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$P = X \oplus (Y \oplus Z)$



⇒ Odd Parity Generators :-

Inputs			Output
X	Y	Z	P
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

X \ YZ	00	01	11	10
0	1	0	1	0
1	0	1	0	1

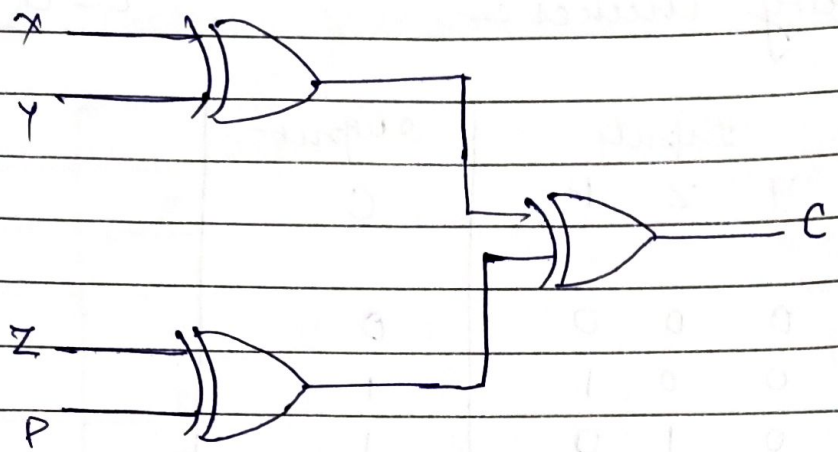
$C = 1 \rightarrow$ error
$C = 0 \rightarrow$ no error

⇒ Even Parity Checker :-

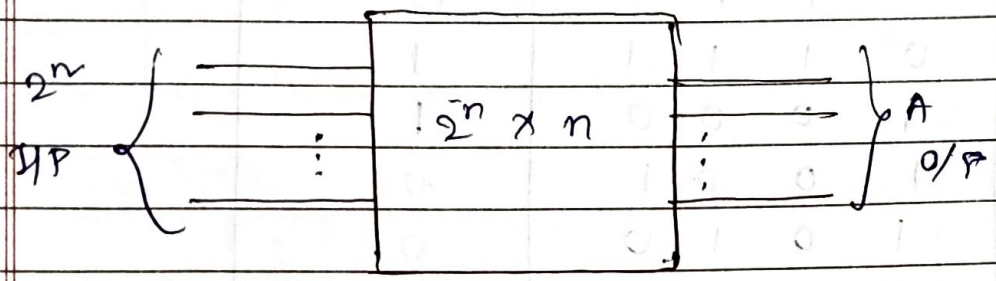
Inputs				output
X	Y	Z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

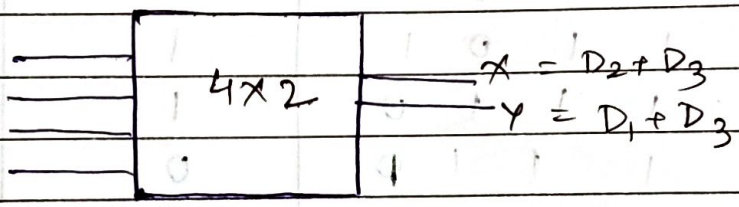
$C = X \oplus Y \oplus Z \oplus P$

XY \ ZP	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

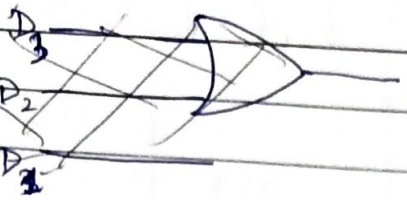


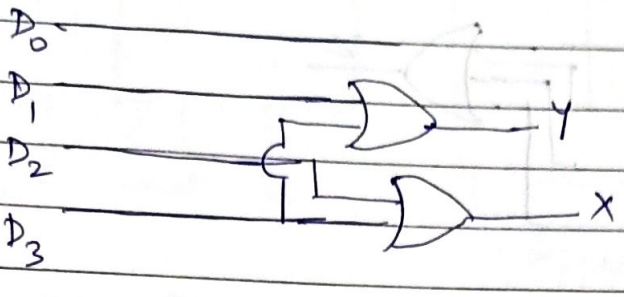
⇒ Encoder :-





D_0	D_1	D_2	D_3	X	Y
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1



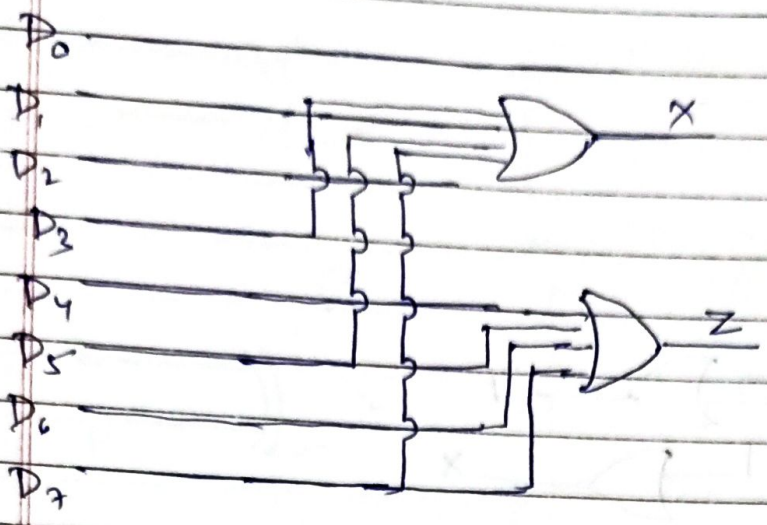


Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$X = D_4 + D_5 + D_6 + D_7$

$Y = D_2 + D_3 + D_6 + D_7$

$Z = D_1 + D_3 + D_7 + D_5$



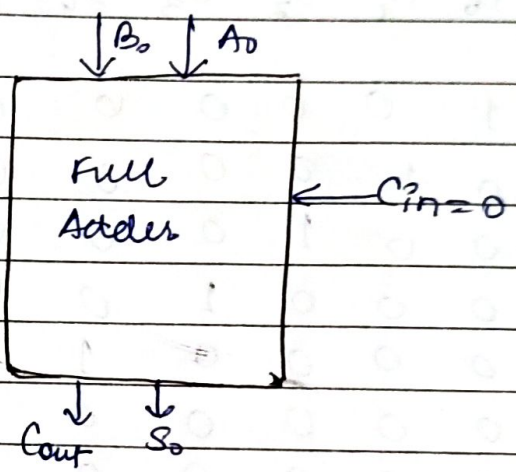
⇒ **Full Adder:-**

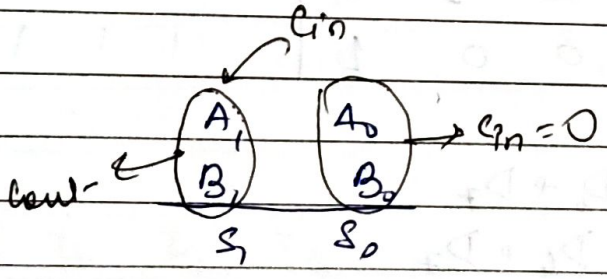
C_{in}

A — A_0

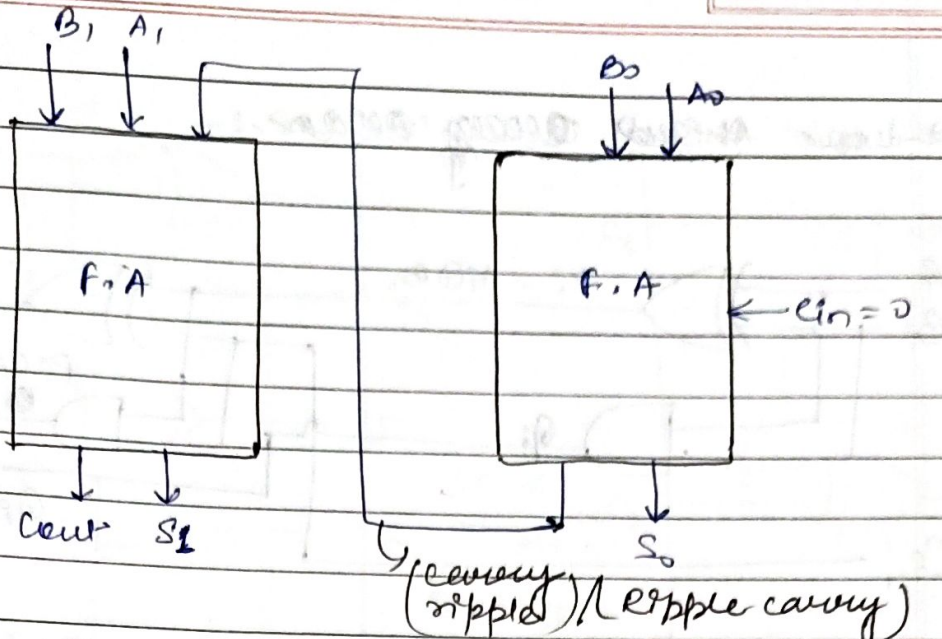
B — B_0

Cout S_0





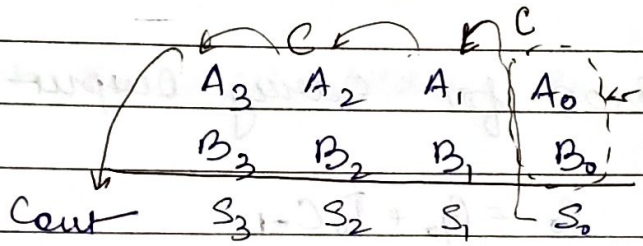
→ This is known as Binary Adder / Parallel Adder.



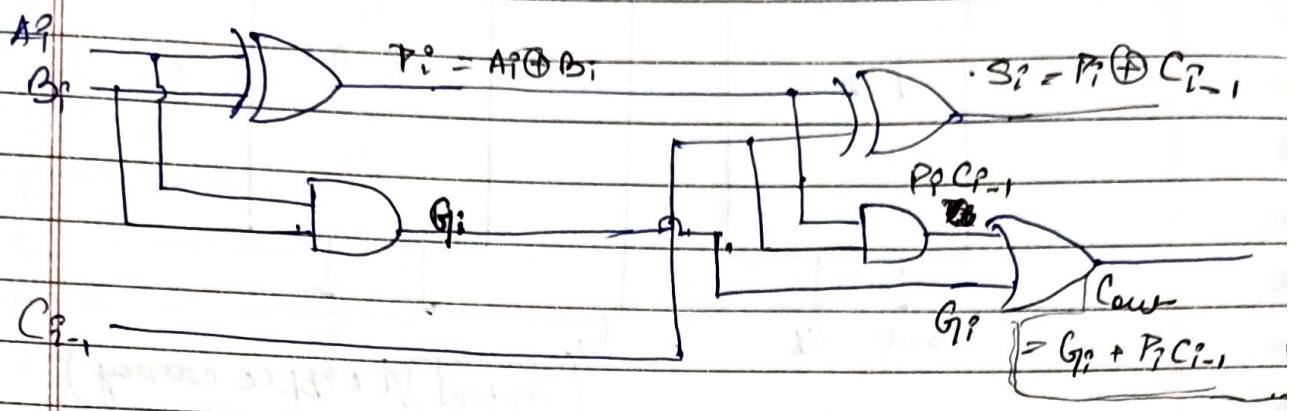
⇒ 2-bit Paeallel Adder → 2 full Adders are used.

⇒ For n-bit paeallel adder we require n number of full adder.

→ If we show it as half-adder, then for n-bit paeallel adder we require (n-1) number of full-adder.



⇒ Look Ahead Carry Adder :-



One full adder using two half-adders (for i^{th} order)

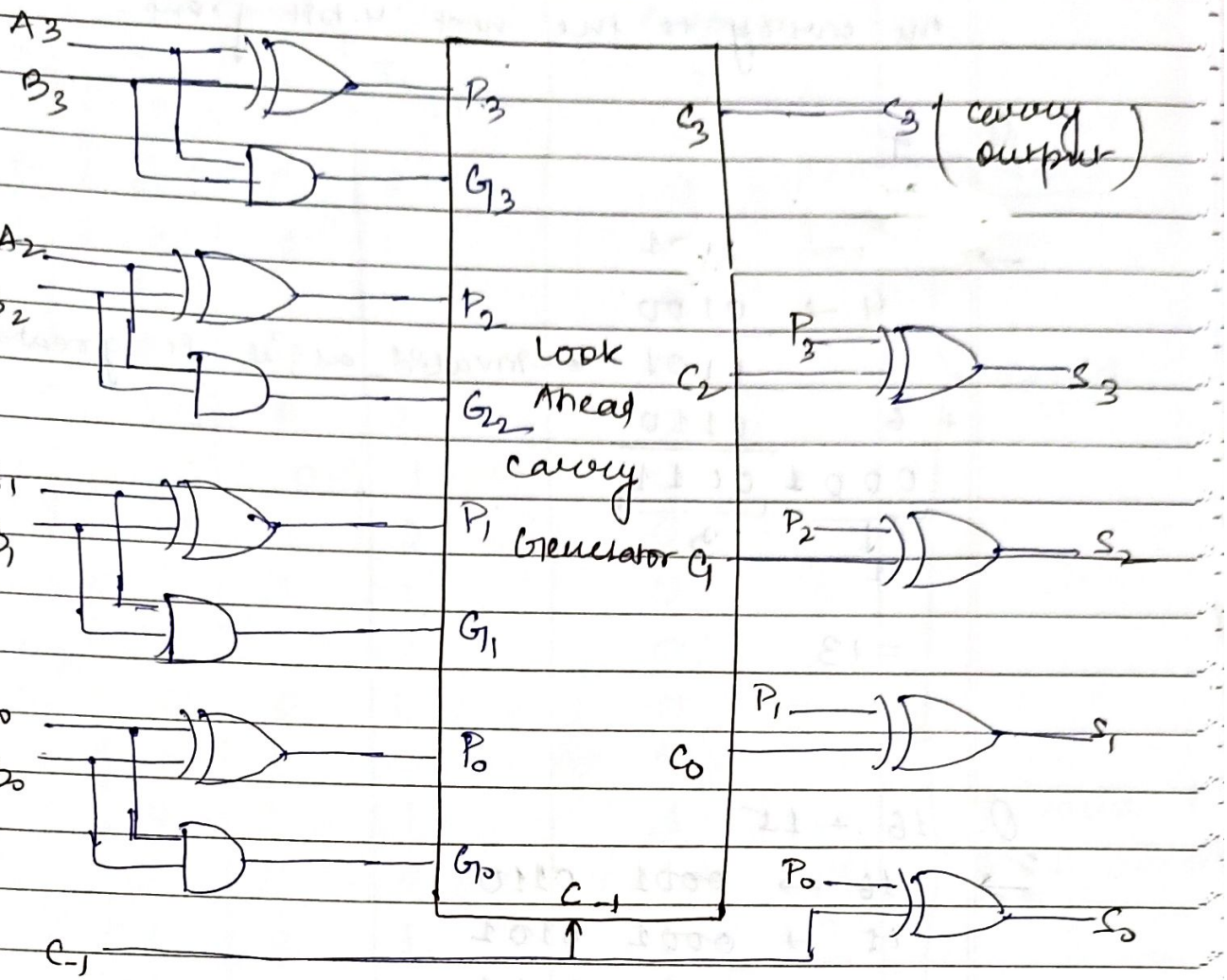
$P_i = A_i \oplus B_i$

$G_i = A_i B_i$

$S_i = P_i \oplus C_{i-1} = A_i \oplus B_i \oplus C_{i-1}$

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

Stage	Expn for carry output
0	$C_0 = G_0 + P_0 C_{-1}$
1	$C_1 = G_1 + P_1 C_0$ $= G_1 + P_1 (G_0 + P_0 C_{-1})$
2	$C_2 = G_2 + P_2 C_1$
3	$C_3 = G_3 + P_3 C_2$



4-bit Parallel Adder with Look-Ahead Carry.

⇒ Rules for BCD Addition:

If a 4-bit sum is equal to or less than 9 it a valid BCD number.

If a four bit sum is greater than 9, or if a carry out of the four bit-group is generated it is an invalid result. Then add 6 to the four bit sum. If a carry results when 6 is added, simply add

the carry to the next 4-bit group.

Q. 9
+4

→ 9 → 1001
4 → 0100
1101 ← invalid as it is greater than 9.
+6 0110
0001 0011
1 3

=13.

Q. 16 + 15

→ 16 → 0001 0110
15 → 0001 0101
0010 1011
0110
0011 0001
3 1

=31.

Inputs				Output	
S_3	S_2	S_1	S_0	Y	
1	1	1	0	1	← Invalid BCD.
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	Valid
0	1	0	0	0	
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	1	
1	0	1	1	1	Invalid BCD.
1	1	0	0	1	
1	1	0	1	1	
1	1	1	1	1	

Q 67 + 53.

67 → 0110 0111

53 → 0101 0011

1011 1010

0110 0110

0001 0010 0000

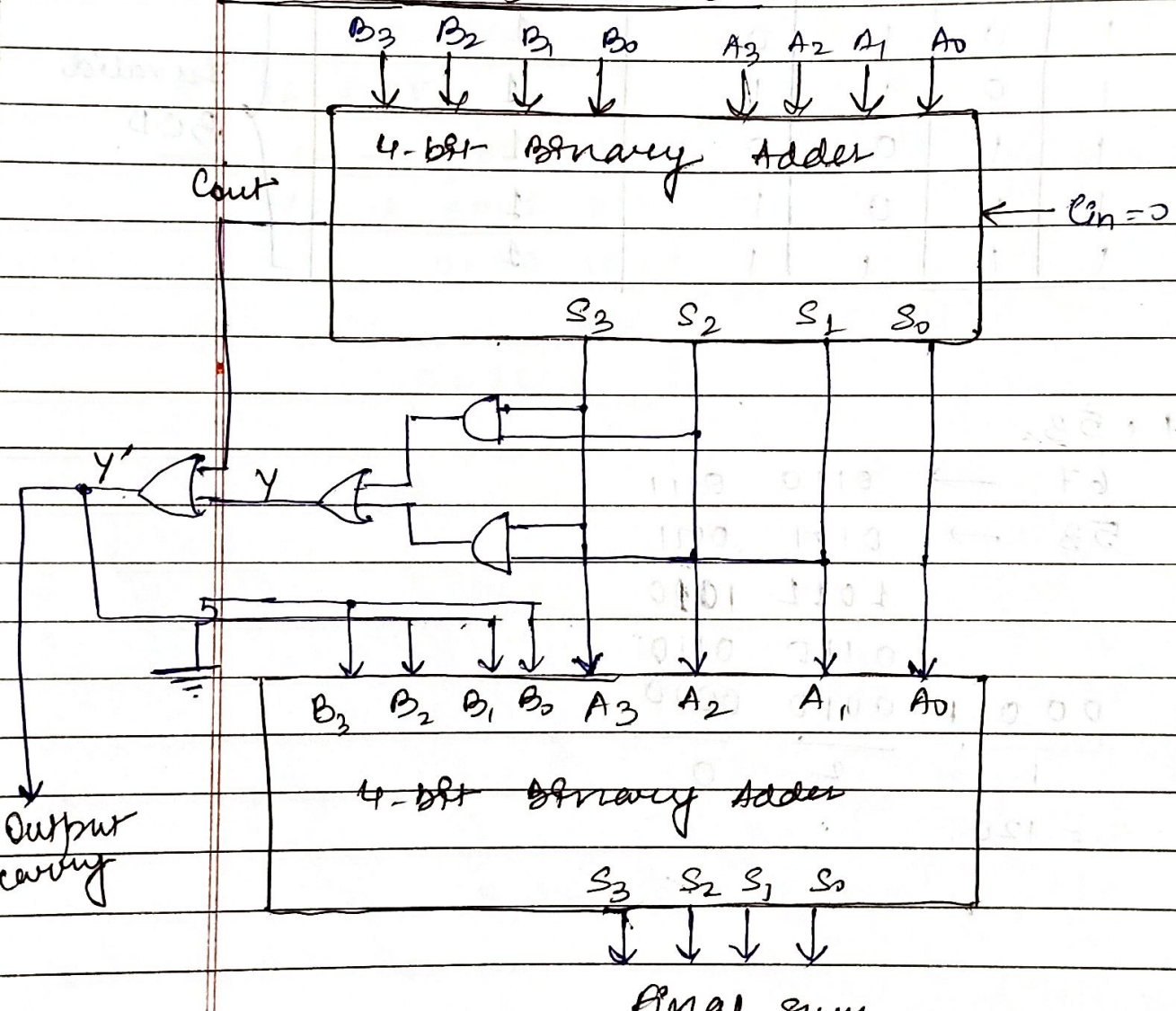
1 2 0

= 120.

For $Y = S_3S_2 + S_3S_1$

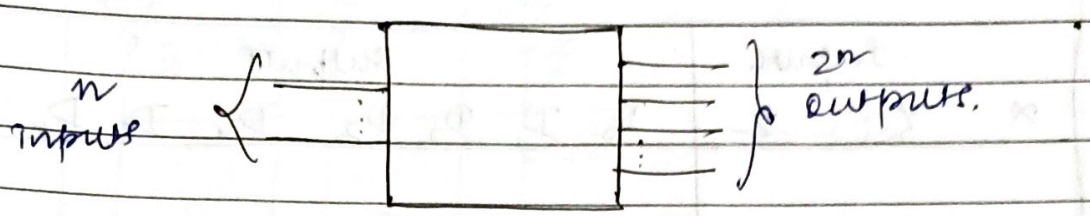
S_3S_2 \ S_1S_0	00	01	11	10
00	0 = 0000	4 = 0100	12	8
			1	
01	1 = 0001	5	13	9
			1	
11	3	7	15	11
			1	1
10	2	6	14	10
			1	1

⇒ 4-Bit BCD Adder :-



→ Magnitude comparator.
→ Priority Encoder.

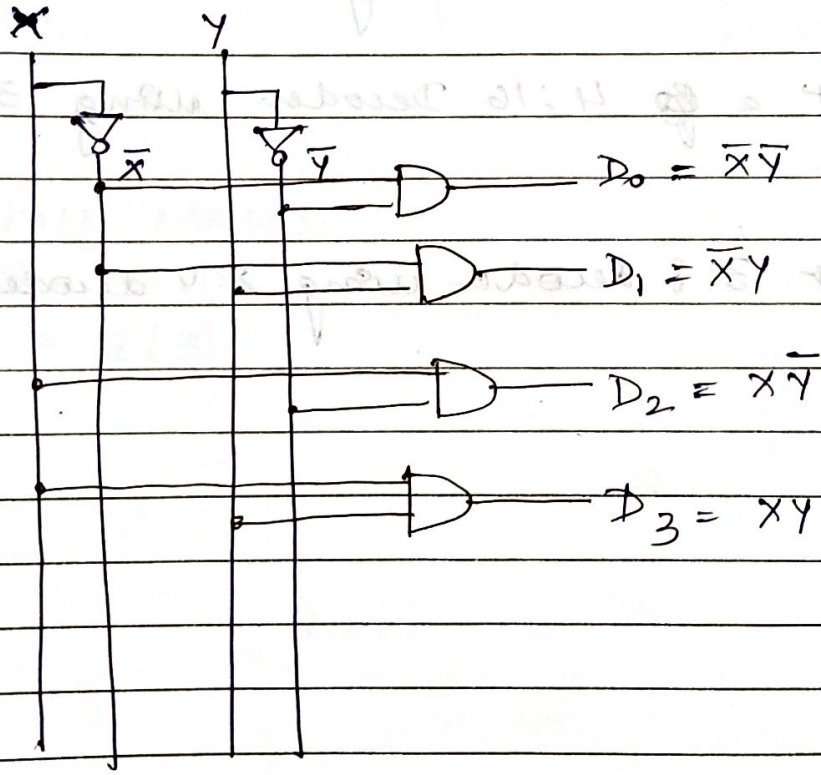
⇒ Decoder :-



→ For decoders we use AND Gates.

→ For encoder we use OR Gates.

I/Ps		O/Ps			
X	Y	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

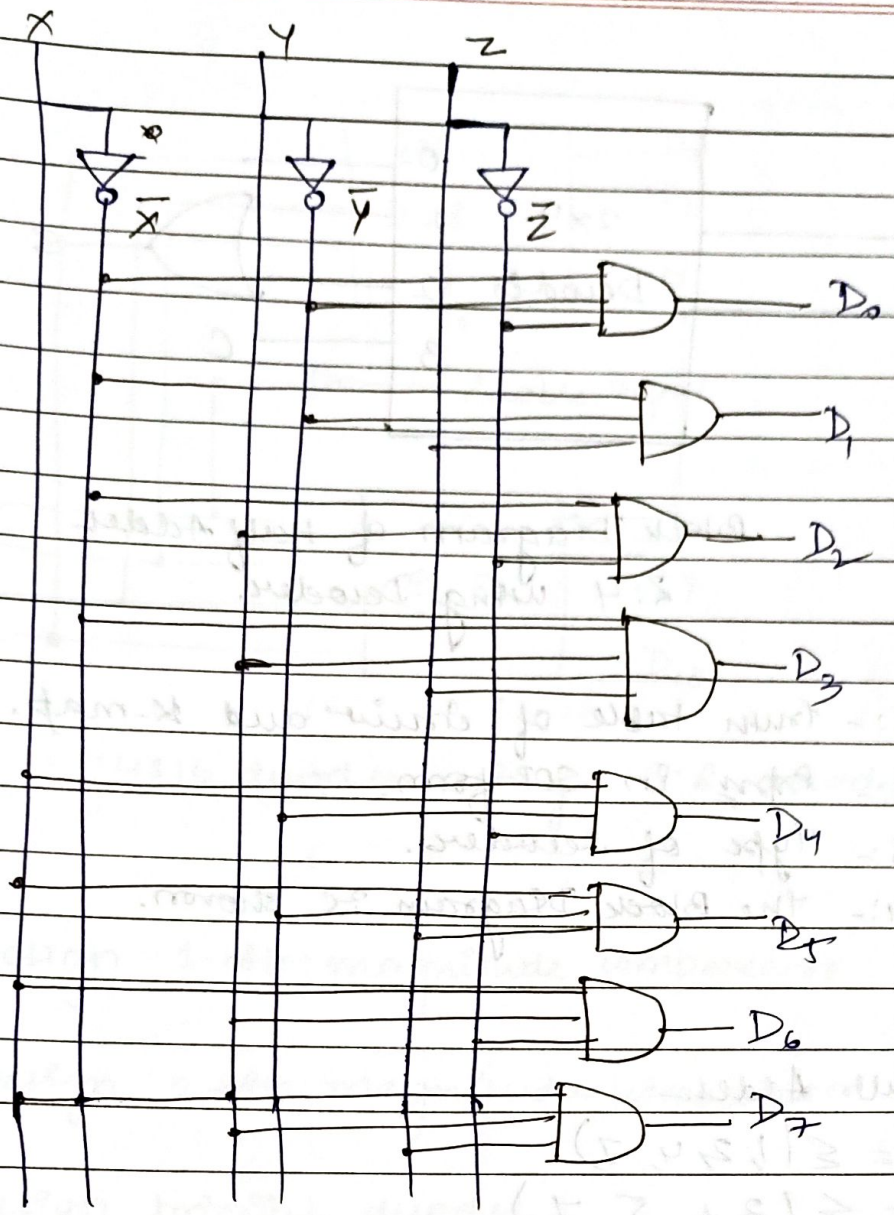


⇒ 3:8 Decoder :-

Inputs			Outputs							
X	Y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Sub:

1. Q. Design Half Adder using Decoder
2. Q. Design Full Adder using Decoder
3. Q. Construct a 4:16 Decoder using 3:8 decoder.
4. Q. Construct 3:8 decoder using 2:4 decoder.

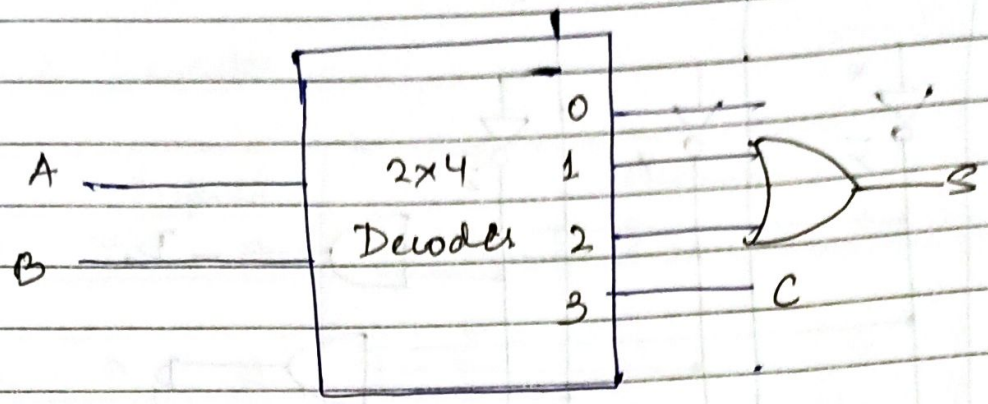


→ For Half Adder :-

$S = \Sigma(1,2)$

$C = \Sigma(3)$.

1. Sol →



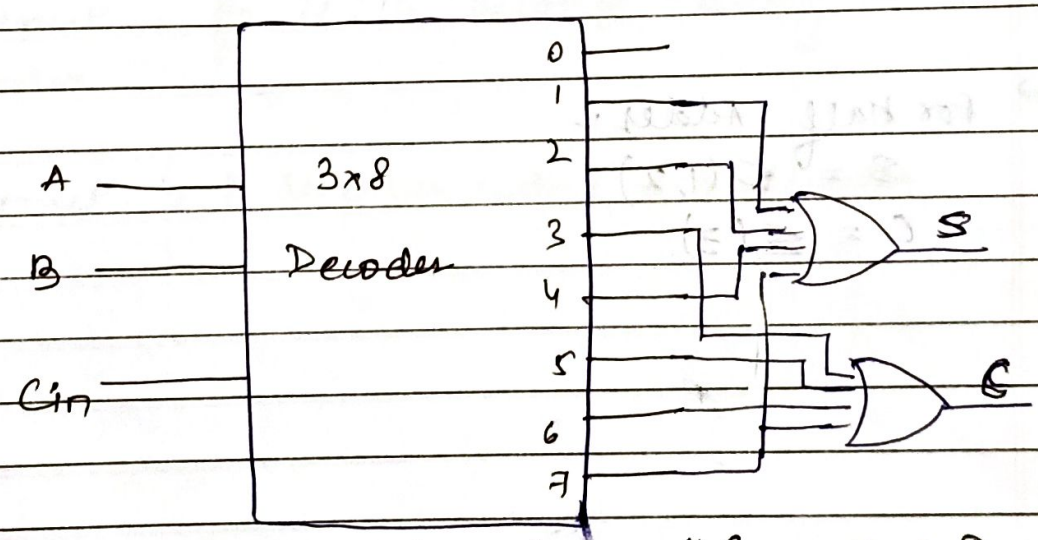
Block Diagram of Half Adder 2:4 using Decoder.

Steps: Step 1:- Truth table of circuit and K-map.
Step 2:- Expn in SOP form.
Step 3:- Type of decoder.
Step 4:- The Block Diagram is shown.

2. Sol → For Full Adder:-

$S = \Sigma(1, 2, 4, 7)$

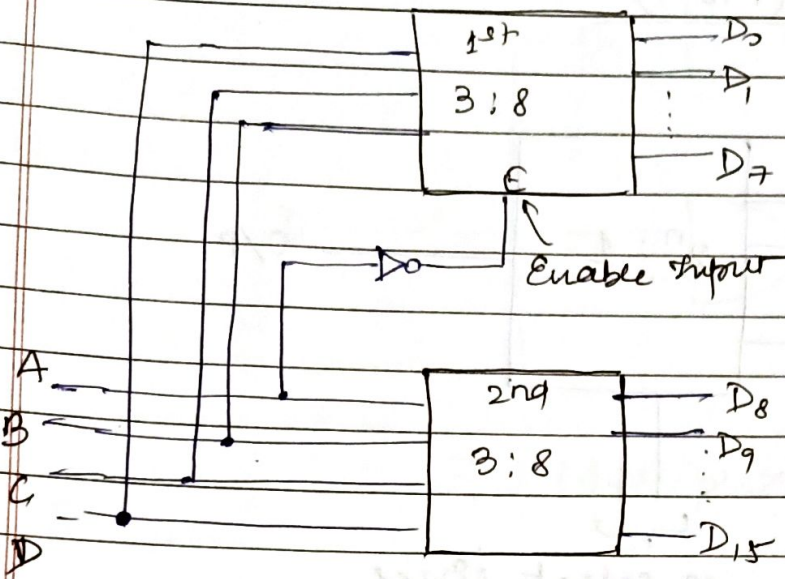
$C = \Sigma(3, 6, 5, 7)$



Block Diagram of Full Adder using 3:8 Decoder

Page No. :
Date: / /

3.Sol→



4:16 decoder using 3:8 decoder.

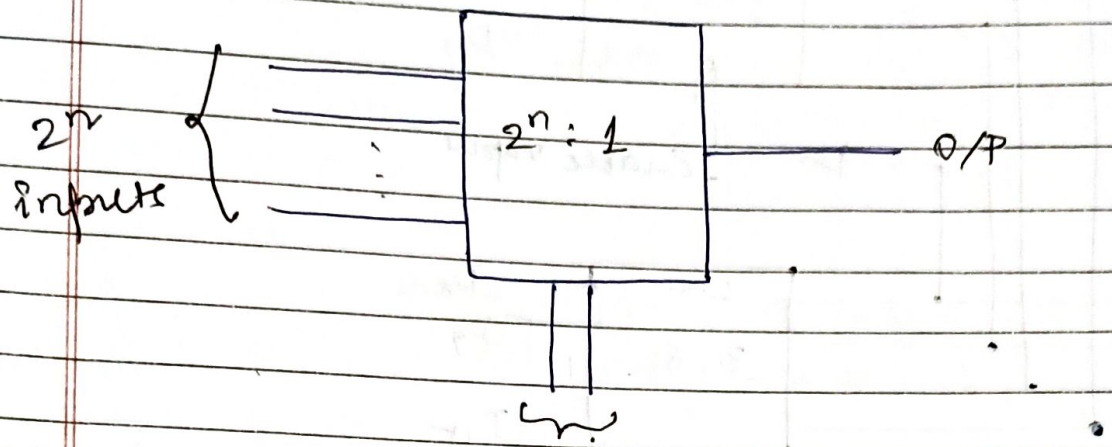
Q Design 1-bit magnitude comparator

Q. Design 2-bit magnitude comparator

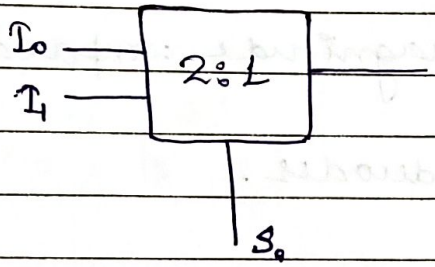
Q. Design priority decoder.

Page No.:
Date: 10/10/19

⇒ Multiplexer (MUX)

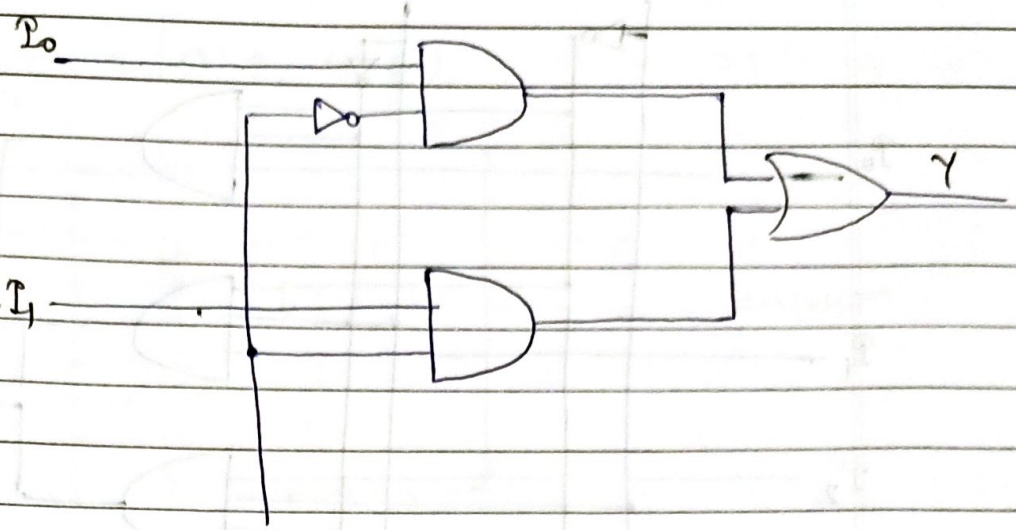


⇒ 2:1 MUX :-



function table

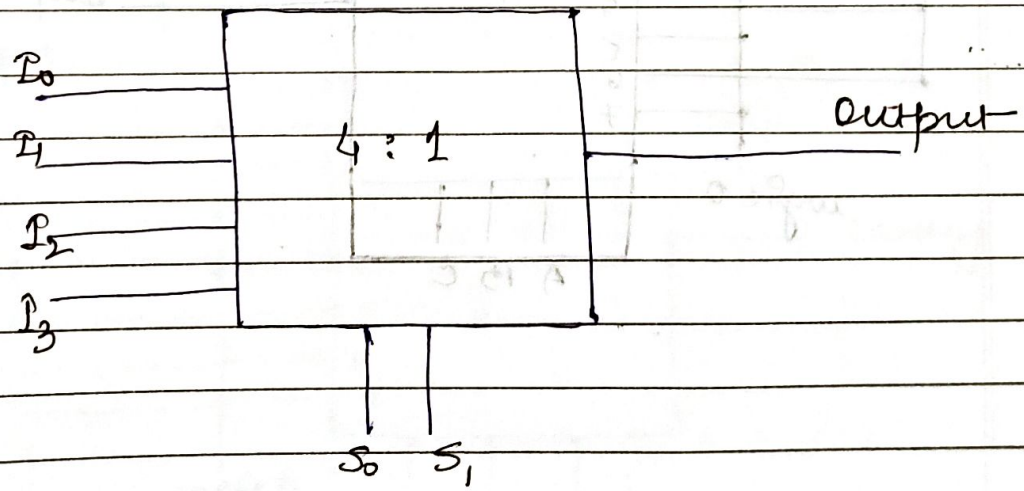
S_0	Y
0	I_0
1	I_1

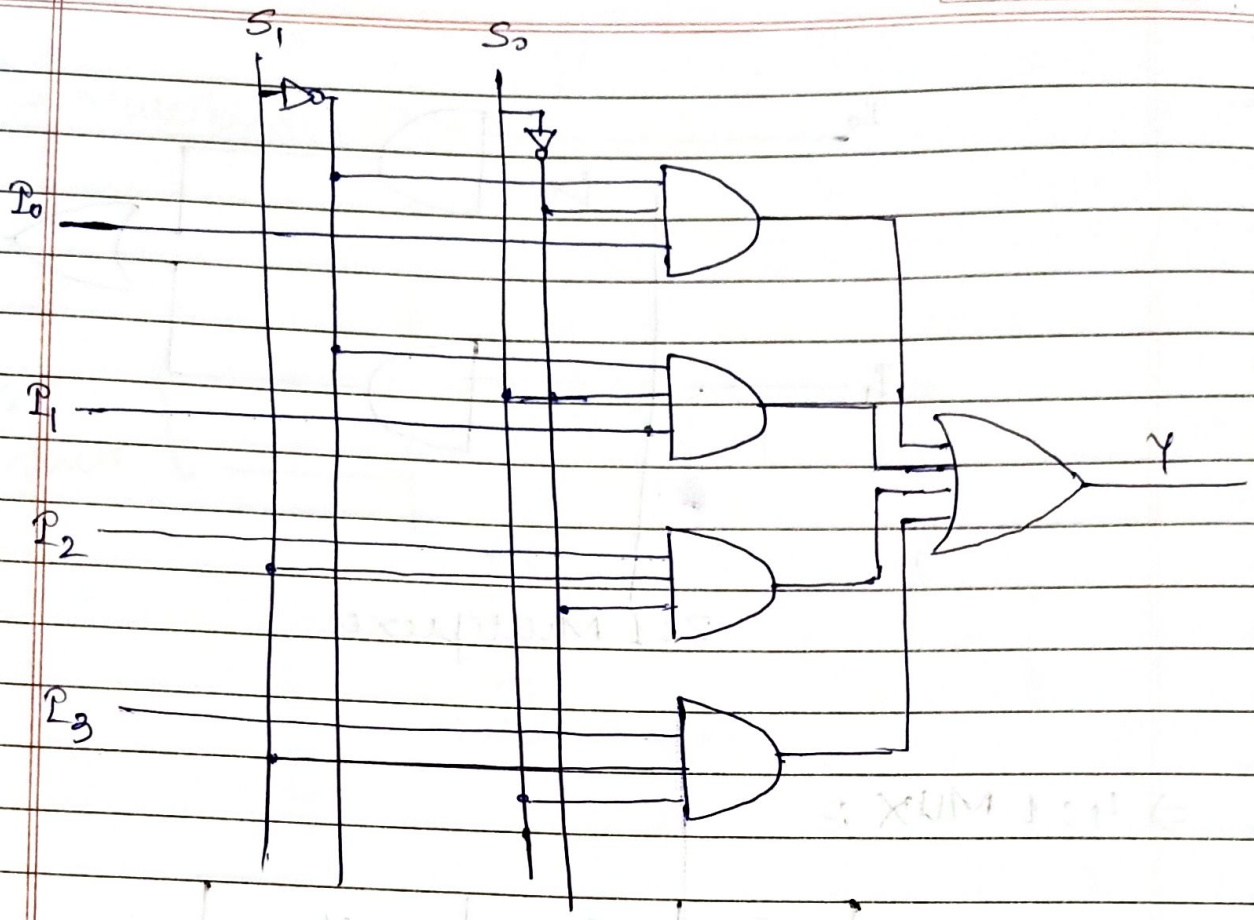


2:1 Multiplexer

⇒ 4:1 MUX :-

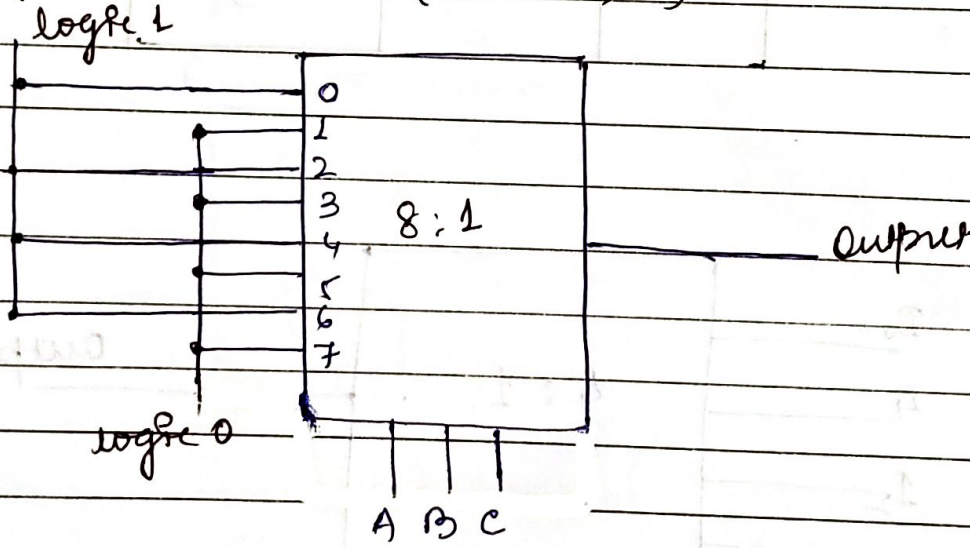
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



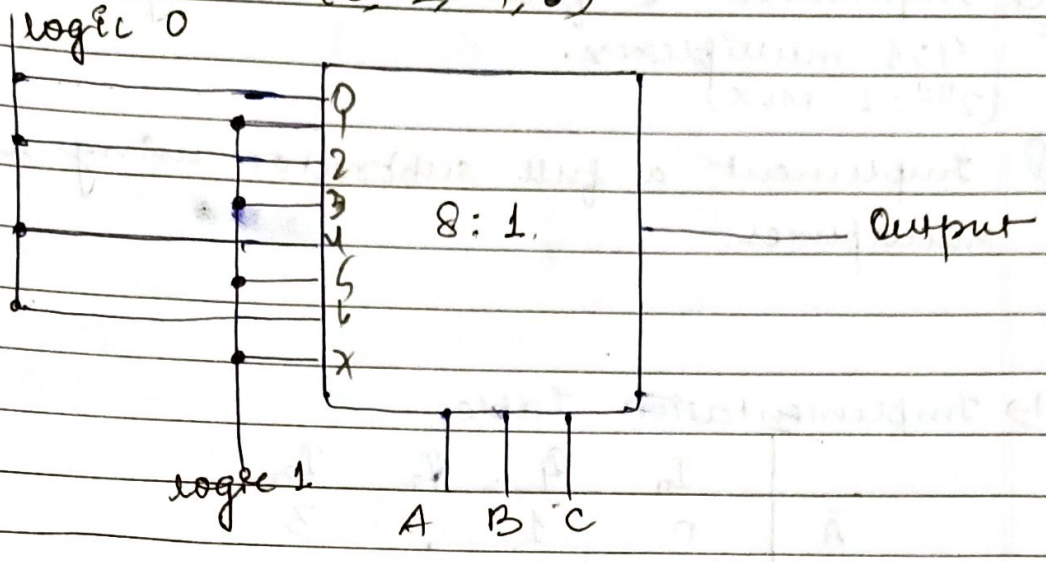


Q. Implement the following Boolean function using a multiplexer.

1. $f(A, B, C) = \Sigma(0, 2, 4, 6)$



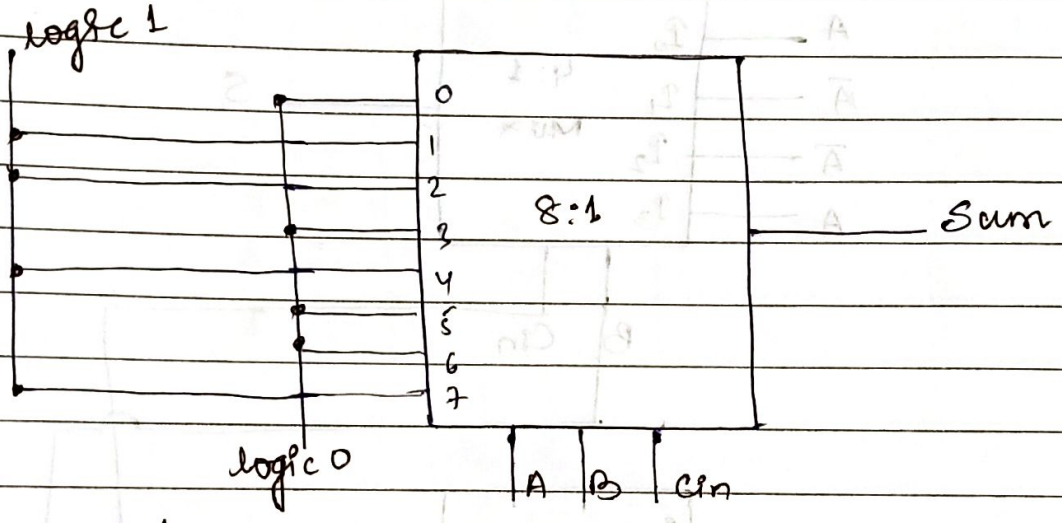
2. $f(A, B, C) = \pi(0, 2, 4, 6)$

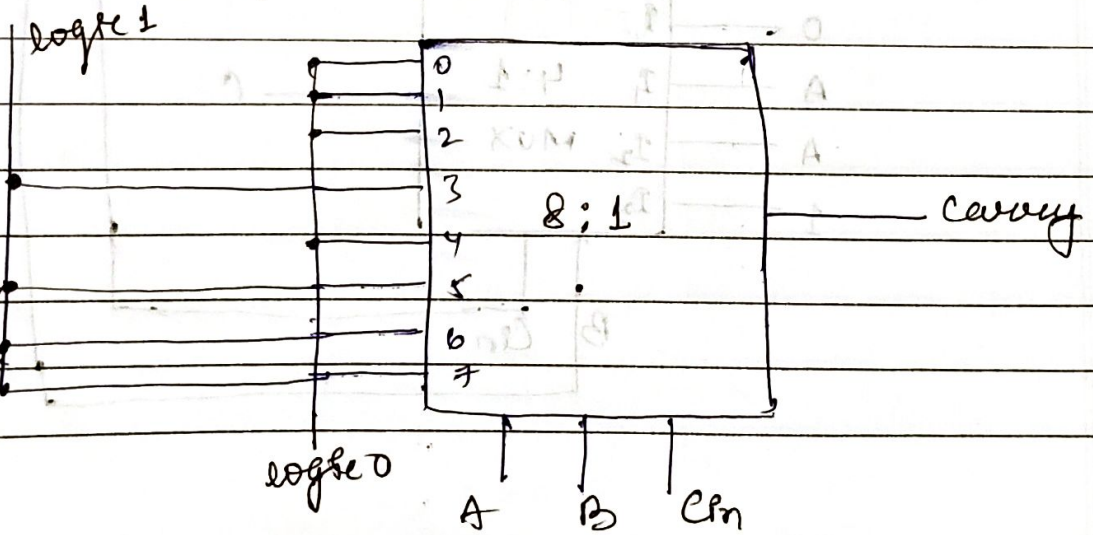


Q. Implement a full-adder using a multiplexer.

→ $S = \Sigma(1, 2, 4, 7)$

$C = \Sigma(3, 5, 6, 7)$





1. Q Implement a full adder using a 4:1 multiplexer.
($2^{n-1}:1$ MUX)

2. Q Implement a full subtractor using a 4:1 multiplexer.

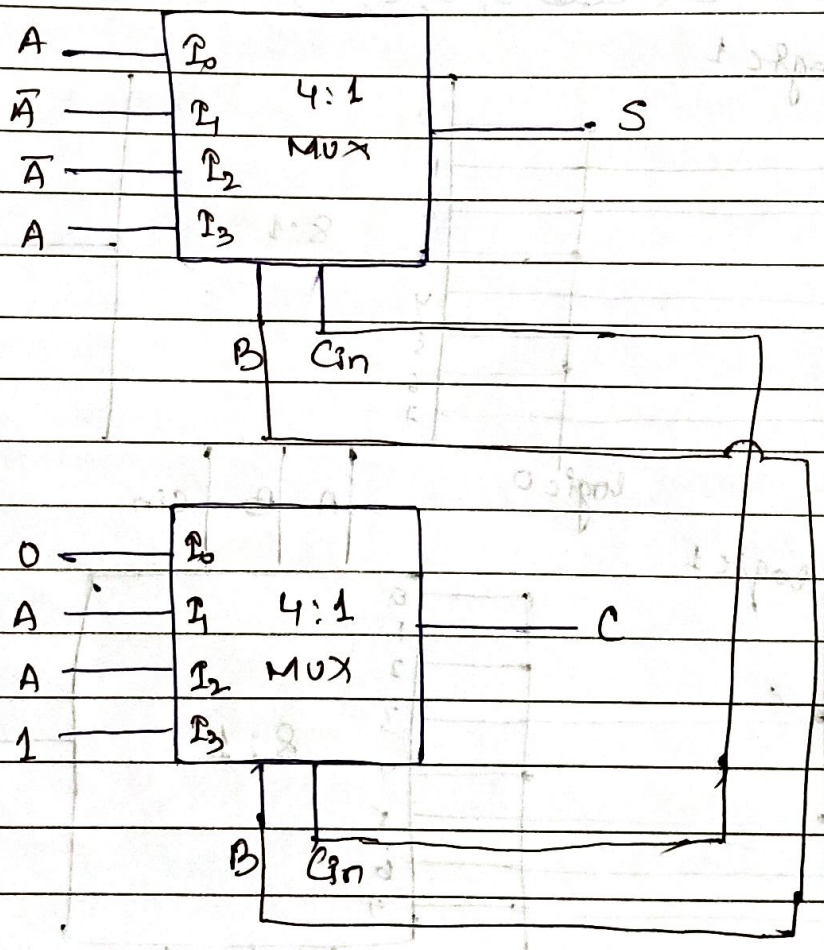
1. Sol → Implementation Table:-

	I_0	I_1	I_2	I_3
$\bar{A}$	0	1	2	3
A ←	4	5	6	7
	A	$\bar{A}$	$\bar{A}$	A

$S(A, B, Cin) = \Sigma(1, 2, 4, 7)$

$C(A, B, Cin) = \Sigma(3, 5, 6, 7)$

$\Sigma(1,2,4,7)$

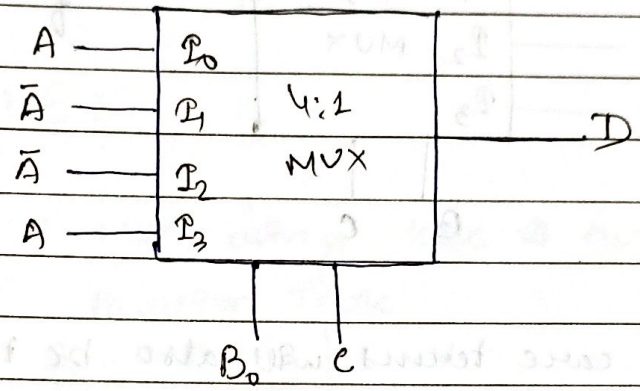


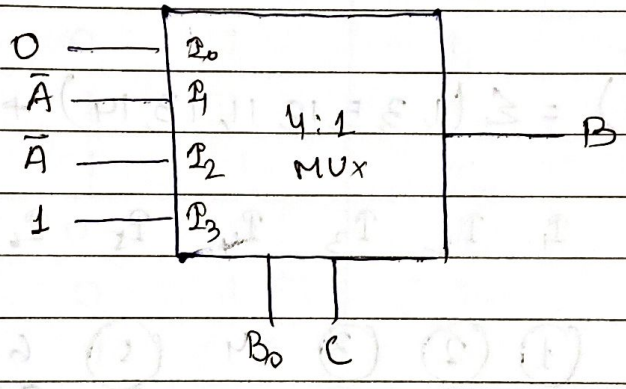
2.30) → Implementation table :-

	I_0	I_1	I_2	I_3
$\bar{A}$	0	(1)	(2)	(3)
A	(4)	5	6	(7)

$D(A, B, C) = (1, 2, 4, 7)$

$B(A, B, C) = (1, 2, 3, 7)$

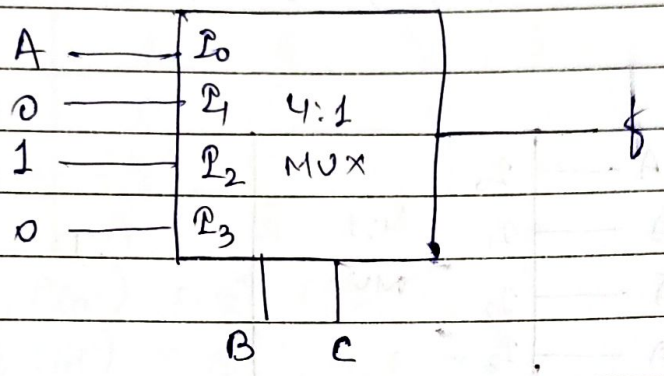




Q. $f(A,B,C) = \pi(0,1,3,5,7)$. Implement using 4:1 mux MUX.

→

	I_0	I_1	I_2	I_3
$\bar{A}$	0	1	(2)	3
A	(4)	5	(6)	7
	A	0	1	0

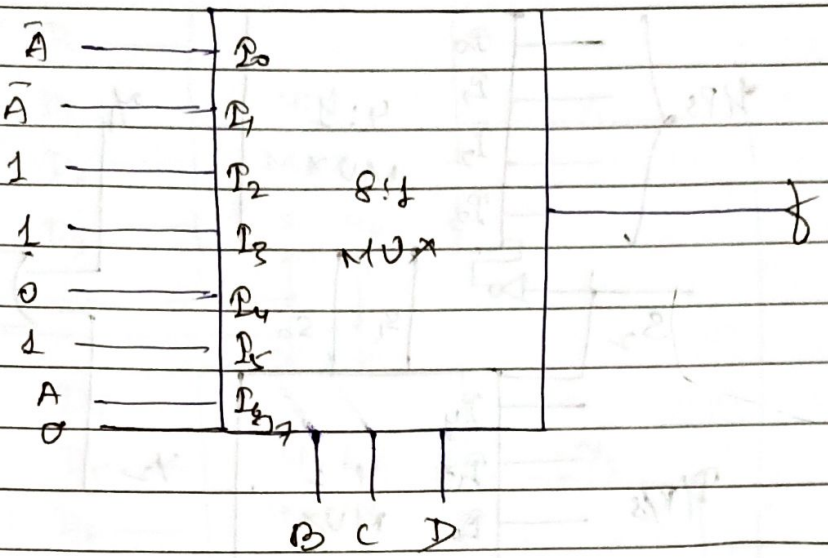


Note → the Don't care terms will also be treated as 1.

Q $f(A,B,C,D) = \Sigma(1,3,5,10,11,13,14) + d(0,2)$ {8:1}.

→

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



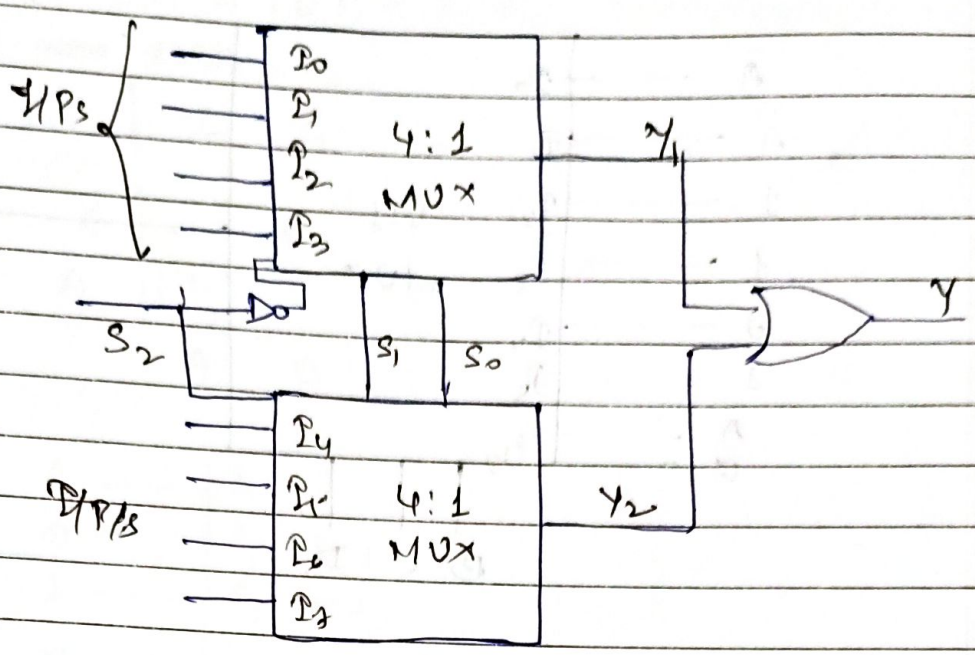
⇒ MUX TREE :-

Q. Obtain 8:1 MUX using two 4:1 MUX.

→

Function Table

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

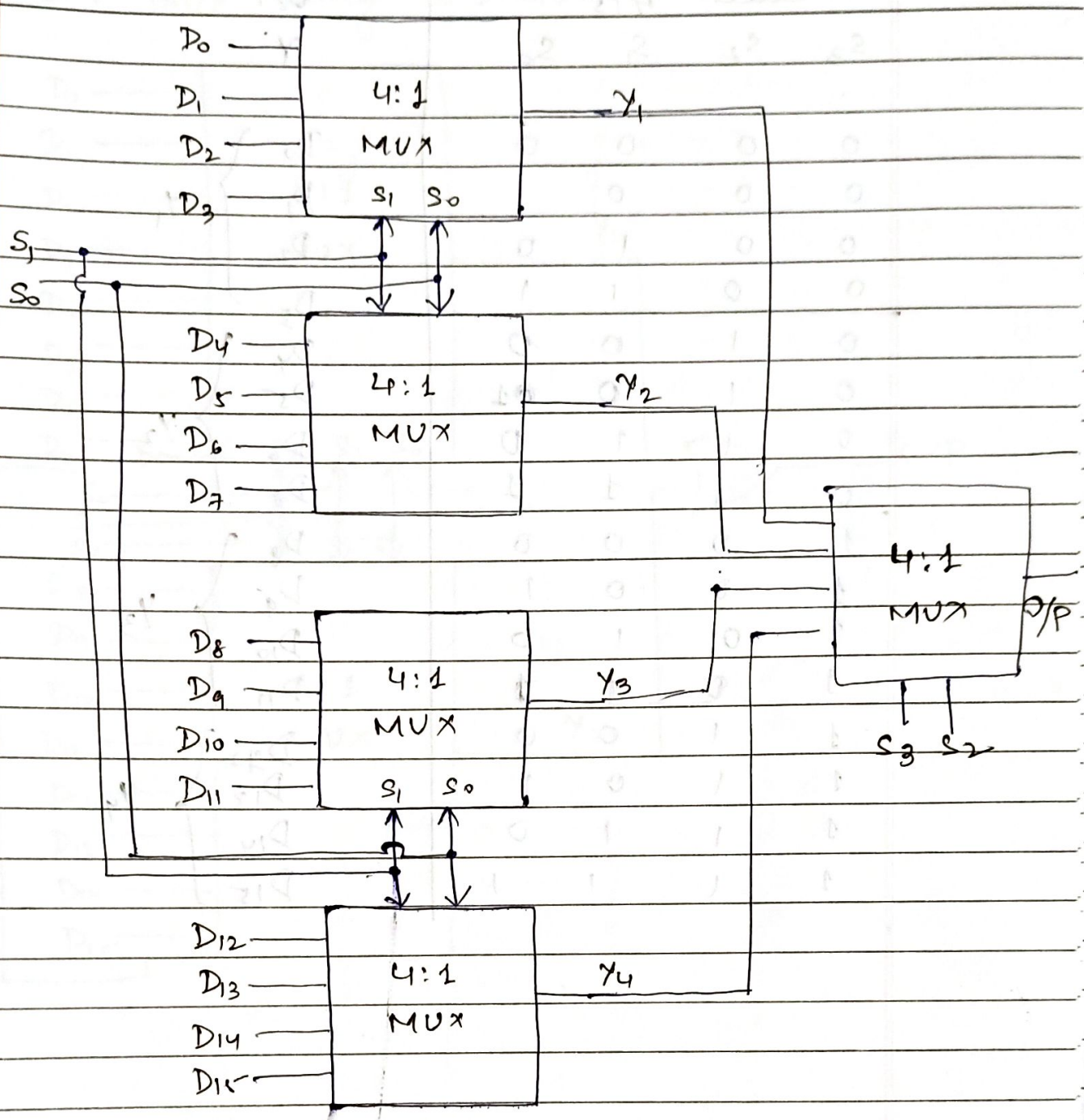


1. Q. Implement 4:1 MUX using 2:1 MUX.

2. Q. Implement 16:1 MUX using 4:1 MUX.

3. Q. Implement 32:1 MUX using 4:1 MUX.

2. Solⁿ

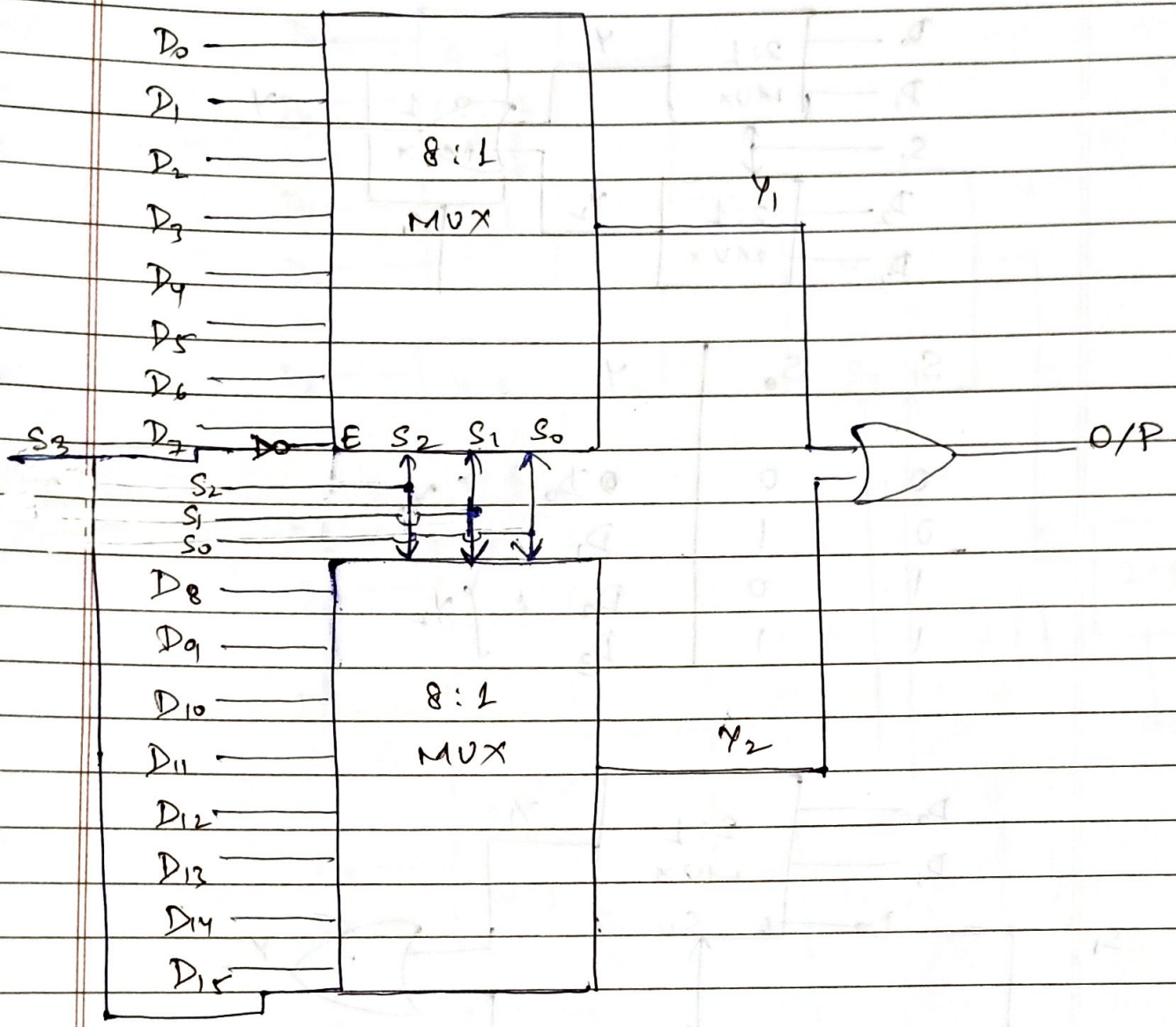

D0
D1
D2
D3
4:1
MUX
S1 S0
Y1
S1
S0
D4
D5
D6
D7
4:1
MUX
Y2
4:1
MUX
O/P
D8
D9
D10
D11
4:1
MUX
S1 S0
Y3
S3 S2
D12
D13
D14
D15
4:1
MUX
Y4

Select I/Ps				O/P	
S_3	S_2	S_1	S_0	Y	
0	0	0	0	D_0	Y_1
0	0	0	1	D_1	
0	0	1	0	D_2	
0	0	1	1	D_3	
0	1	0	0	D_4	Y_2
0	1	0	1	D_5	
0	1	1	0	D_6	
0	1	1	1	D_7	
1	0	0	0	D_8	Y_3
1	0	0	1	D_9	
1	0	1	0	D_{10}	
1	0	1	1	D_{11}	
1	1	0	0	D_{12}	Y_4
1	1	0	1	D_{13}	
1	1	1	0	D_{14}	
1	1	1	1	D_{15}	

Page No. :
Date: / /

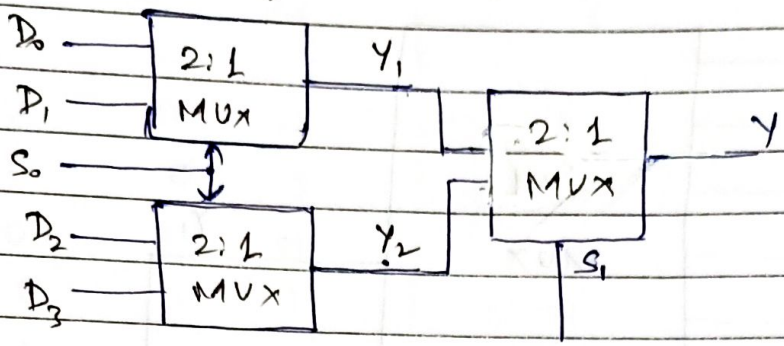
Q 16:1 MUX using 8:1 MUX.

→

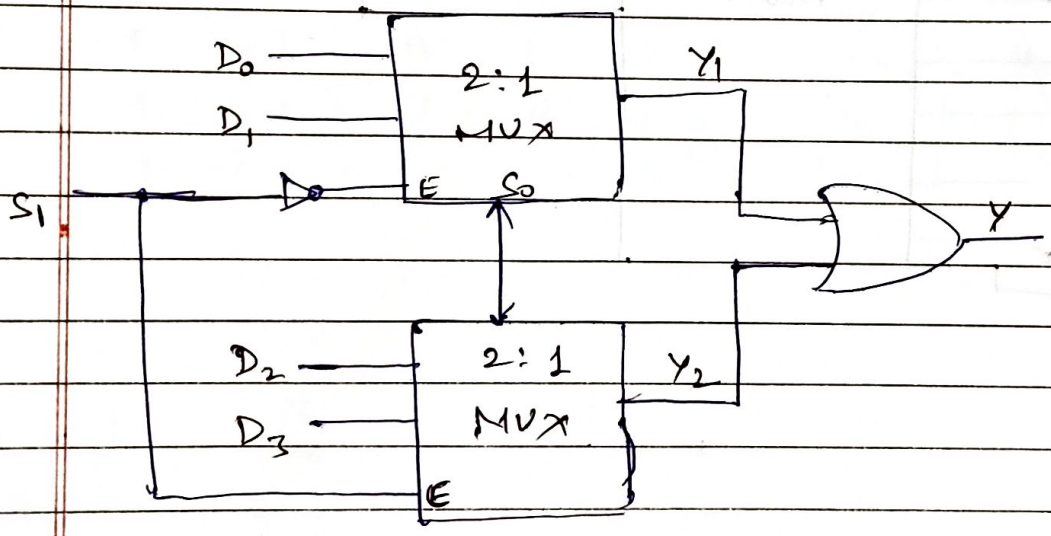


Q 4:1 MUX using 2:1 MUX

→



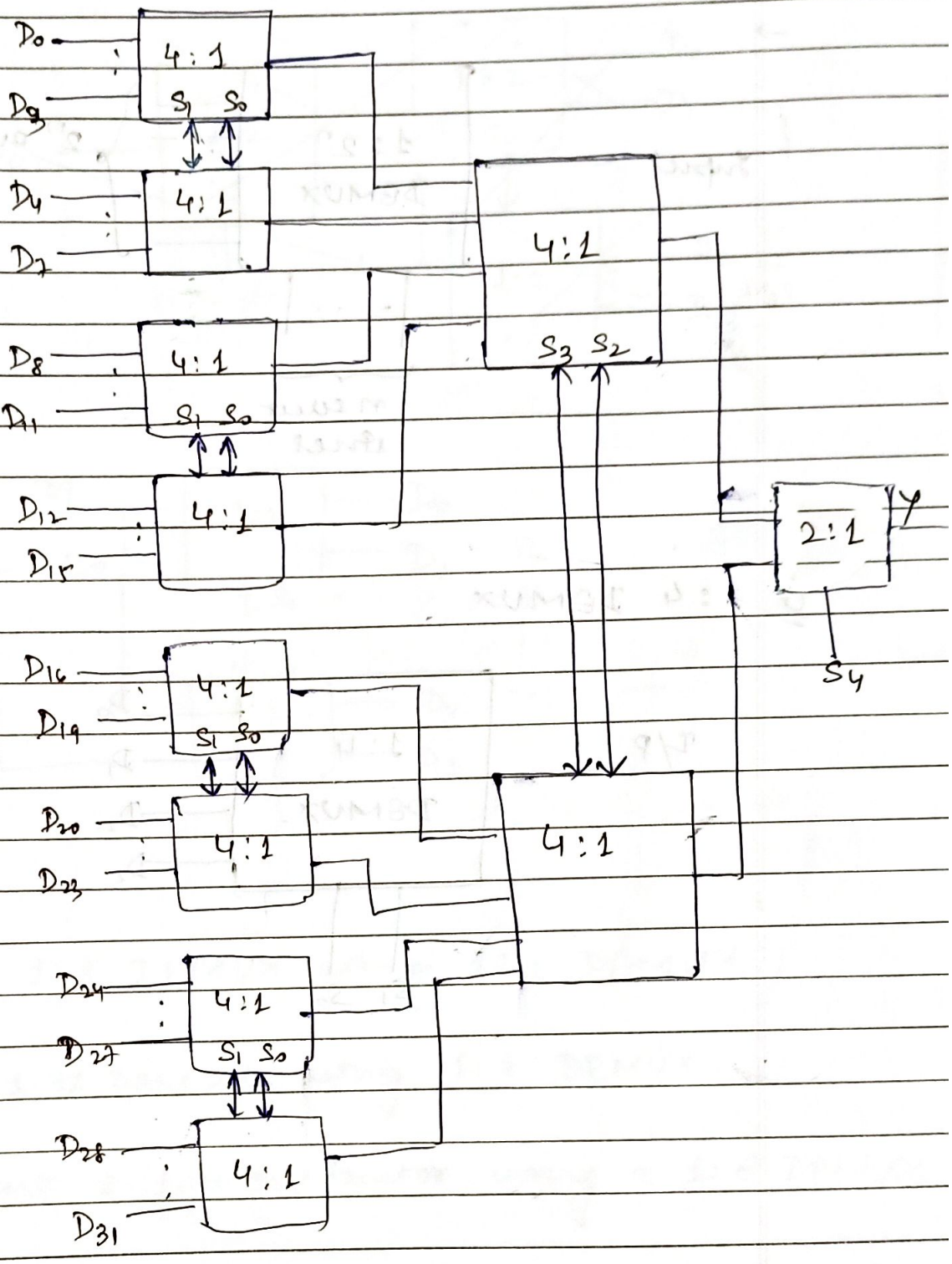
S_1	S_0	Y	
0	0	D_0	Y_1
0	1	D_1	
1	0	D_2	Y_2
1	1	D_3	



Page No. :
Date: / /

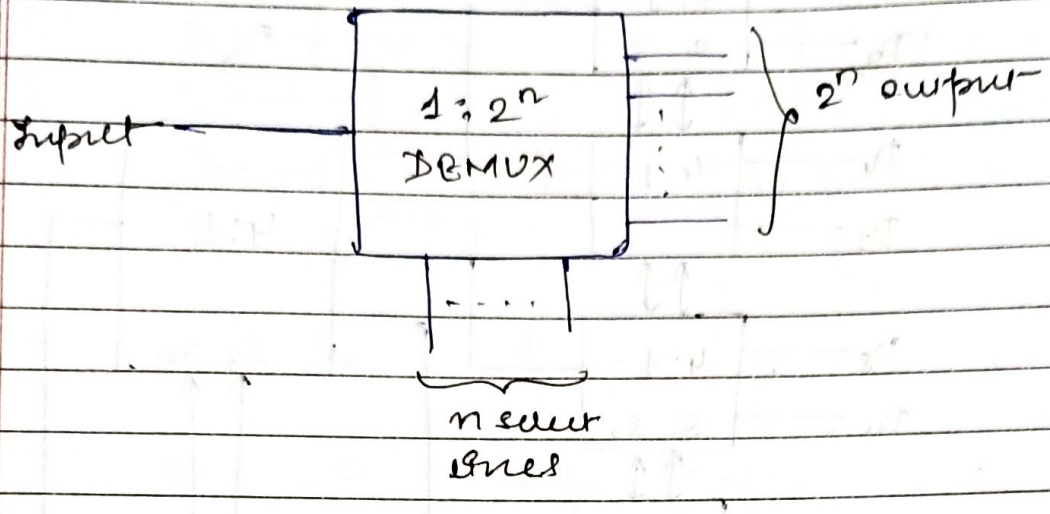
Q 32:1 MUX using 4:1 MUX.

→

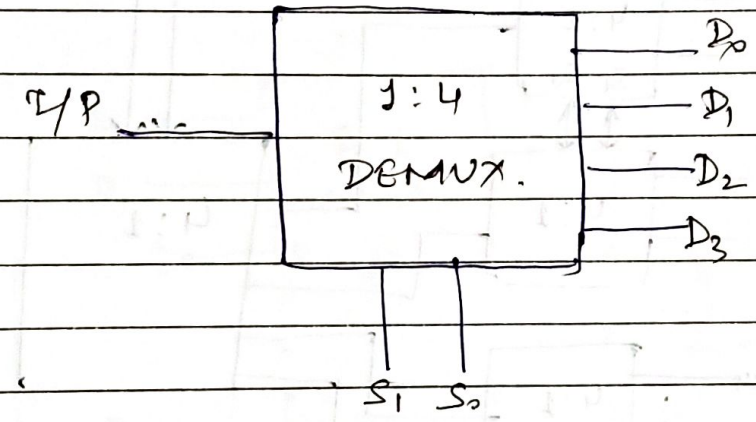


⇒ Demultiplexer (DEMUX) :-

→

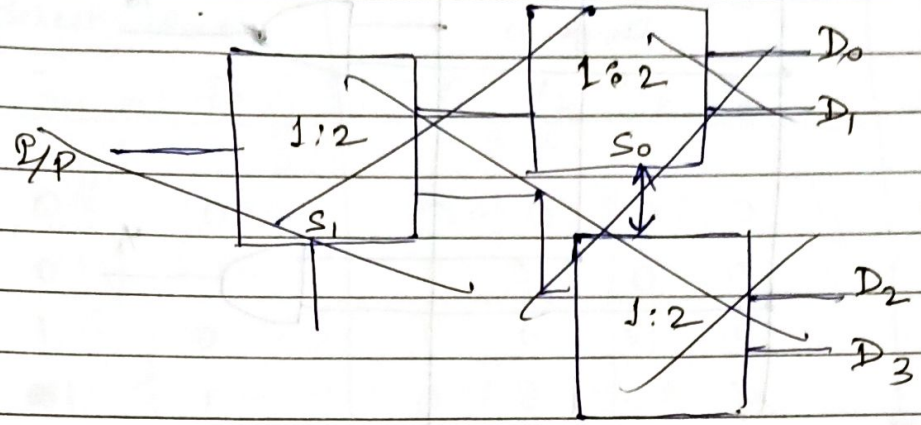


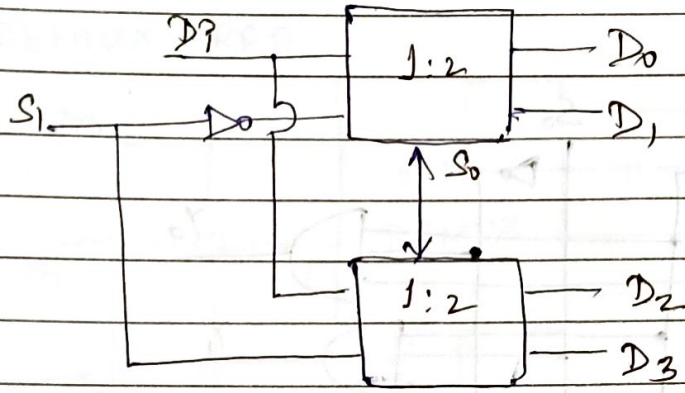
Q 1 : 4 DEMUX.



Q 1:4 DEMUX using 1:2 DEMUX.

→





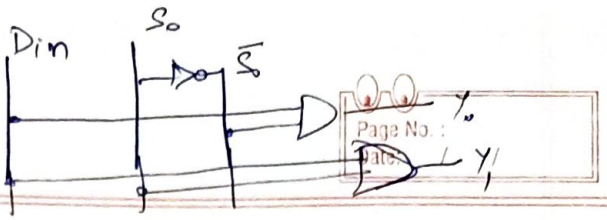
Q Obtain 1:8 DEMUX using 1:4 DEMUX.

Q Obtain 1:32 DEMUX using 1:8 DEMUX.

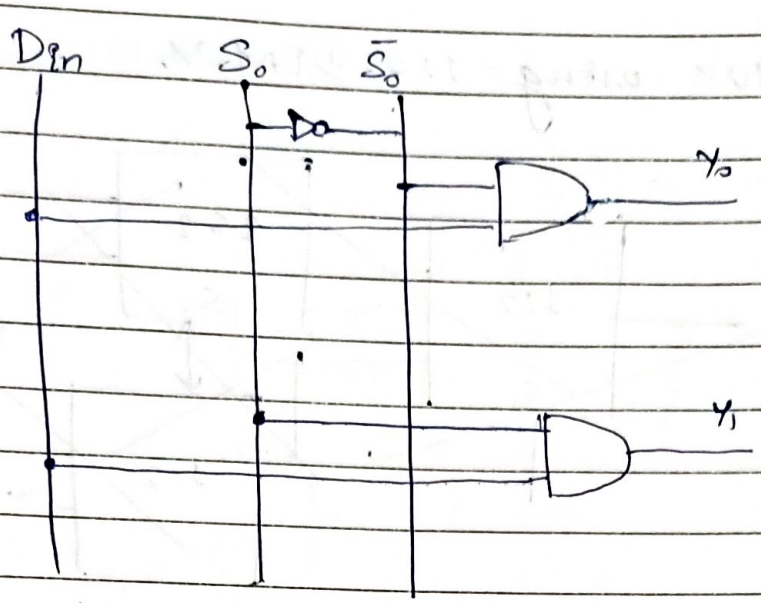
Q Implement a full-subtractor using a 1:8 DEMUX.

Q Implement a full-adder using DEMUX.

Q Draw a 1:64 DEMUX tree using 1:16 DEMUX.

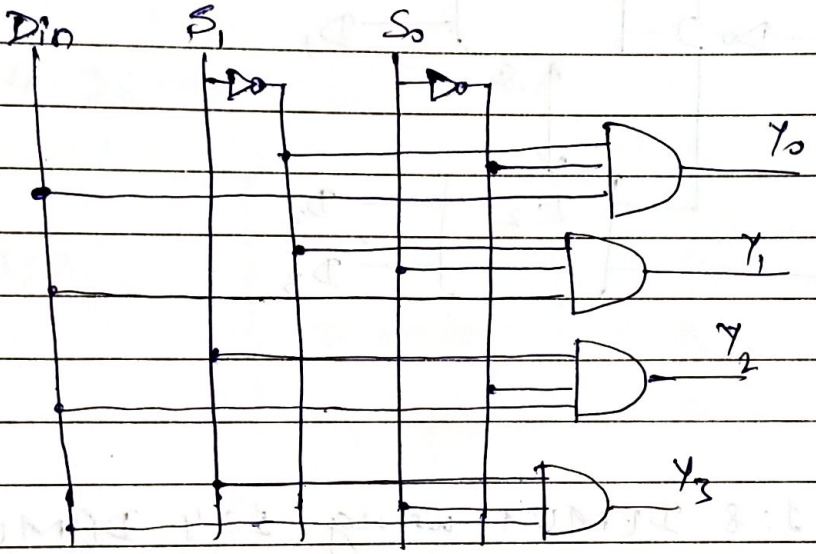


→



1 : 2 DEMUX

→

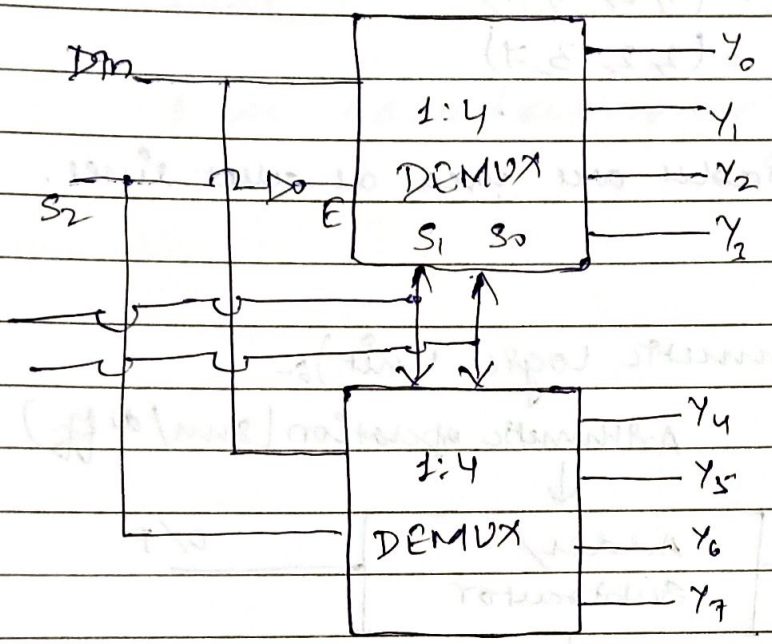


S_1	S_0	Y
0	0	Y_0
0	1	Y_1
1	0	Y_2
1	1	Y_3

1. Q 1:8 DEMUX using 1:4 DEMUX.

Select lines		Outputs			
S_1	S_0	Y_1	Y_2	Y_3	Y_4
0	0	D_{in}	0	0	0
0	1	0	D_{in}	0	0
1	0	0	0	D_{in}	0
1	1	0	0	0	D_{in}

1. DEMUX TREE

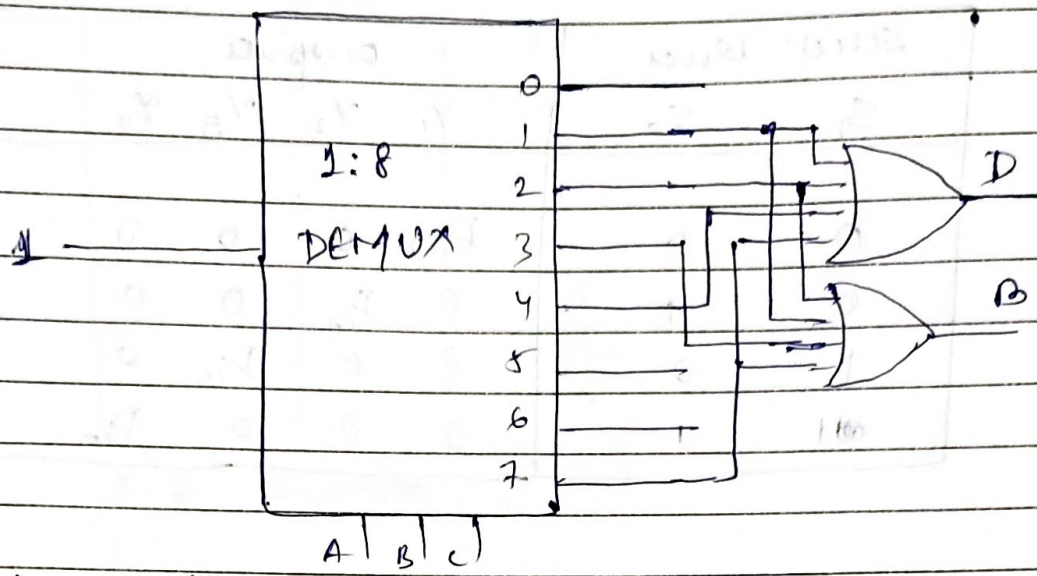


S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	D_{in}	0	0	0	0	0	0	0
0	0	1	0	D_{in}	0	0	0	0	0	0
0	1	0	0	0	D_{in}	0	0	0	0	0
0	1	1	0	0	0	D_{in}	0	0	0	0
1	0	0	0	0	0	0	D_{in}	0	0	0
1	0	1	0	0	0	0	0	D_{in}	0	0
1	1	0	0	0	0	0	0	0	D_{in}	0
1	1	1	0	0	0	0	0	0	0	D_{in}

Page No.:
Date: / /

Q Implement full-subtractor using 1:8 DEMUX.

→

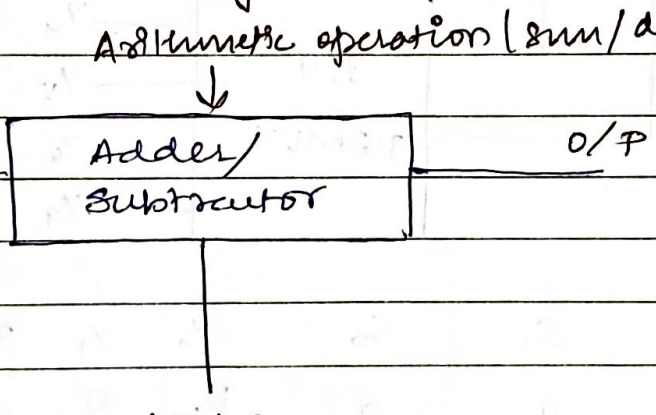


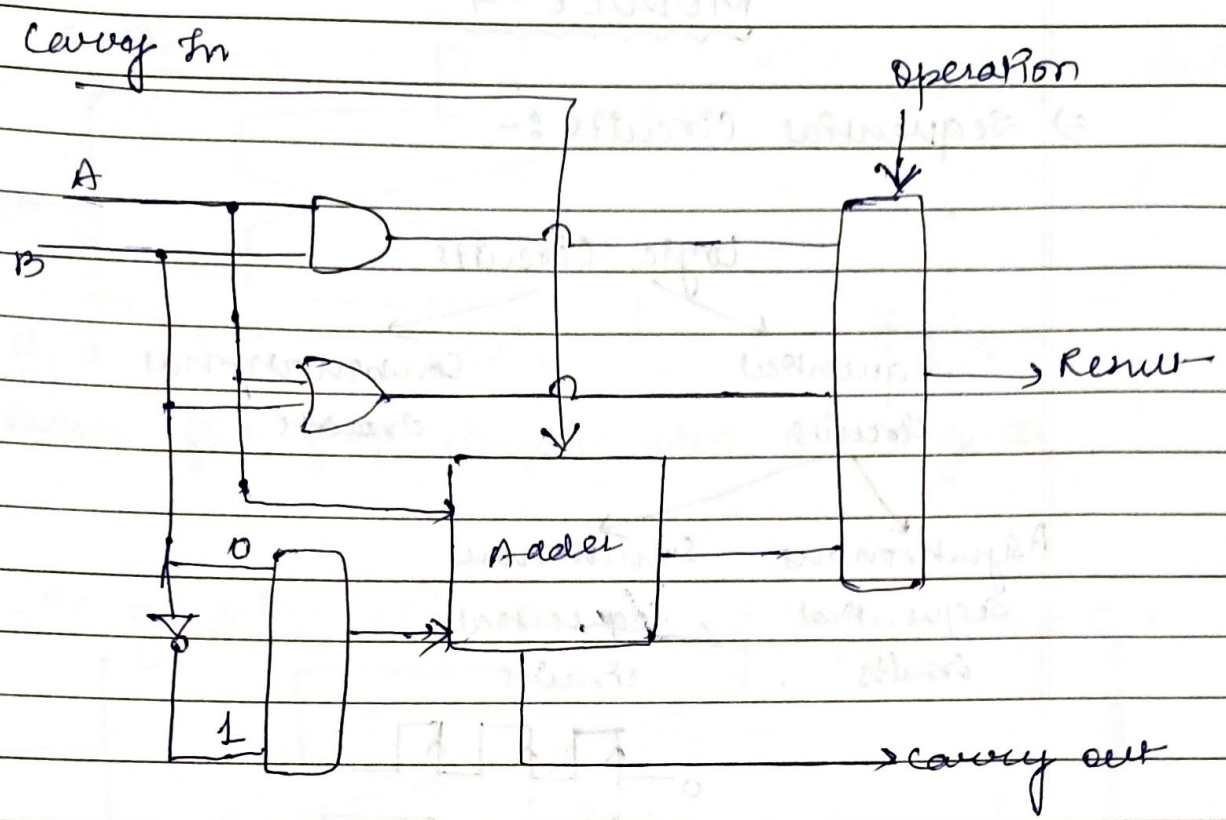
$D(A, B, C) = (1, 2, 4, 7)$

$B(A, B, C) = (1, 2, 3, 7)$

→ Input variables are used as select lines.

- ALU (Arithmetic Logic Unit):-

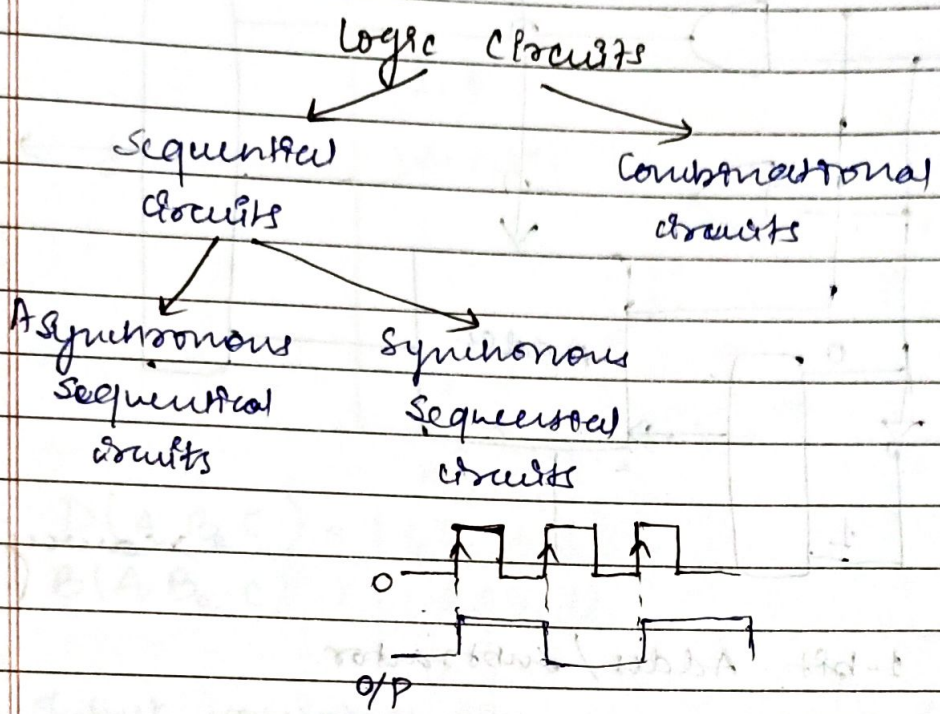




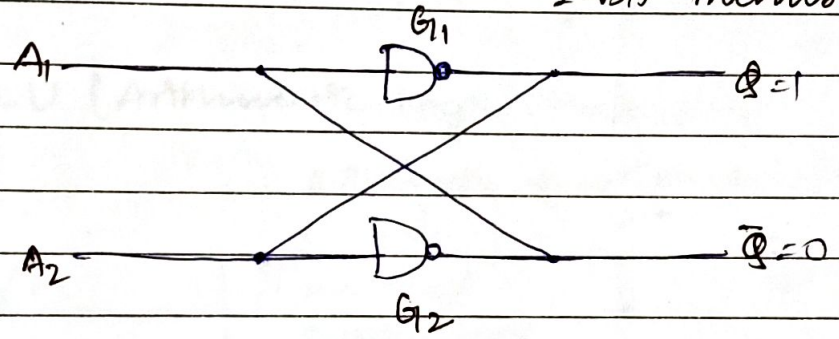
1-bit Adder / subtractor.

MODULE - 4

⇒ Sequential Circuits :-

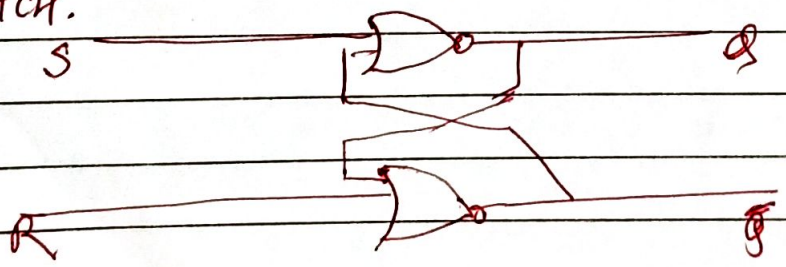


→ 1-bit memory cell / simple latch.



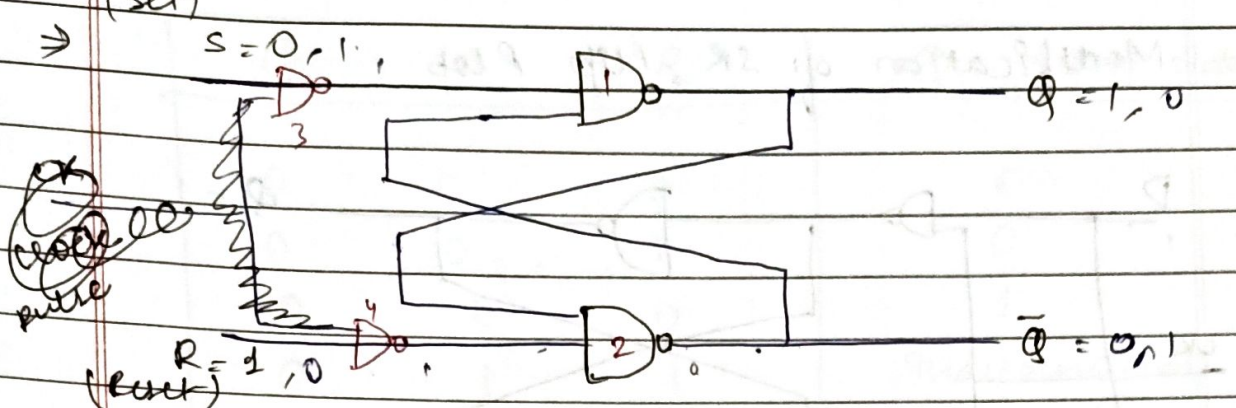
cross coupled inverters as a memory element

⇒ Latch :- Basic storage element is called LATCH.

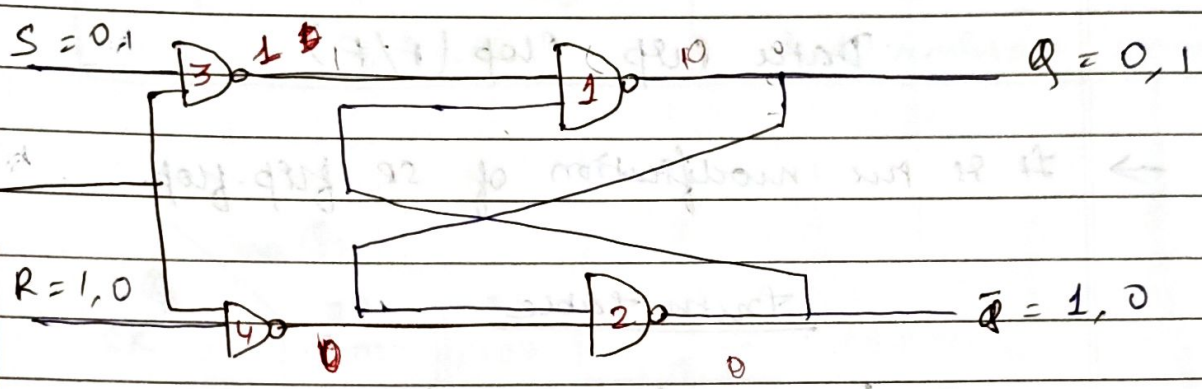


(set)

⇒



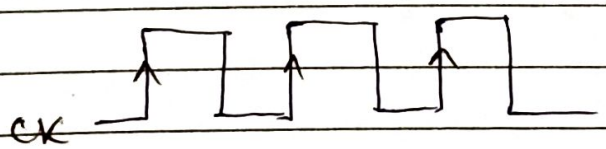
Basic flip-flop circuit with NAND gates



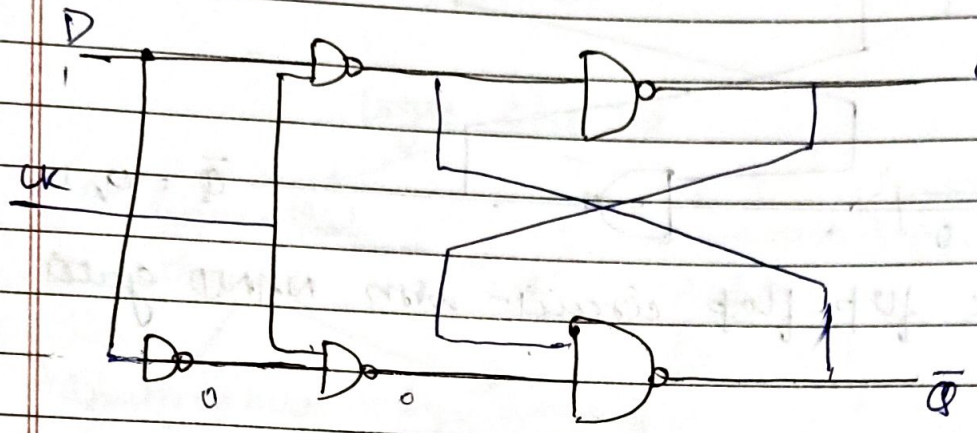
S-R Flip-flop (Asynchronous)

Truth table :-

S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	?



⇒ Modification of SR Flip Flop



Data Flip-Flop (F/F)

→ It is the modification of SR flip-flop.

Truth table

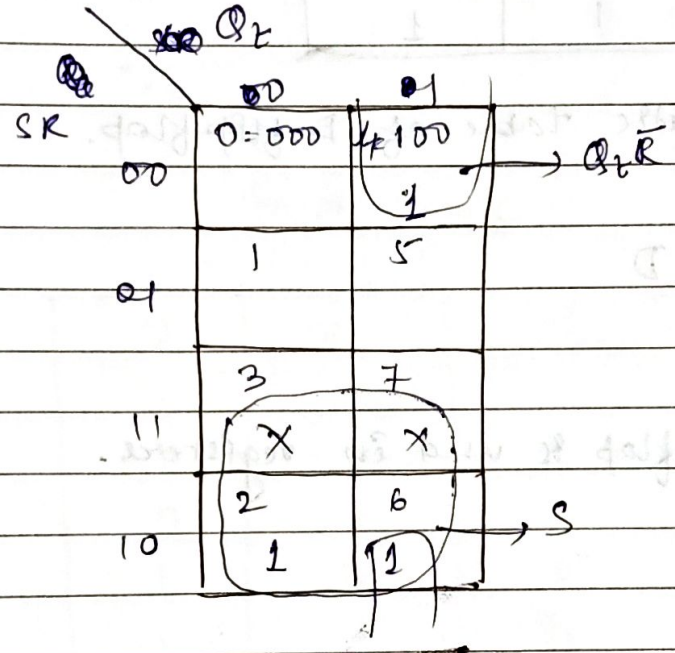
D	Q_{t+1}
0	0
1	1

⇒ Characteristic table :-

→ Here three inputs are taken.

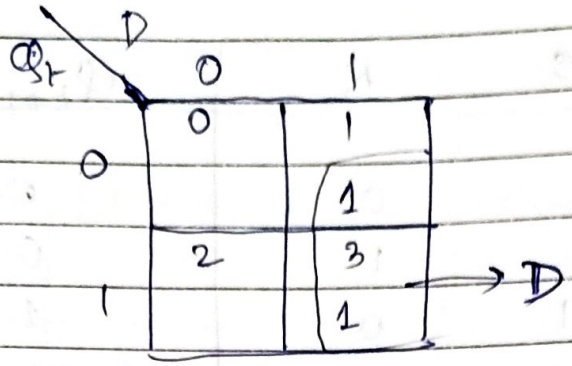
→ Q_t → Previous state output.

Q_t	S	R	Q_{t+1} (Next state output)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indeterminate state
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indeterminate state



$Q_{t+1} = S + Q_t\bar{R}$

characteristic eqn of SR flip-flop.



Q_t	D	Q_{t+1}
0	0	0
0	1	1
1	0	0
1	1	1

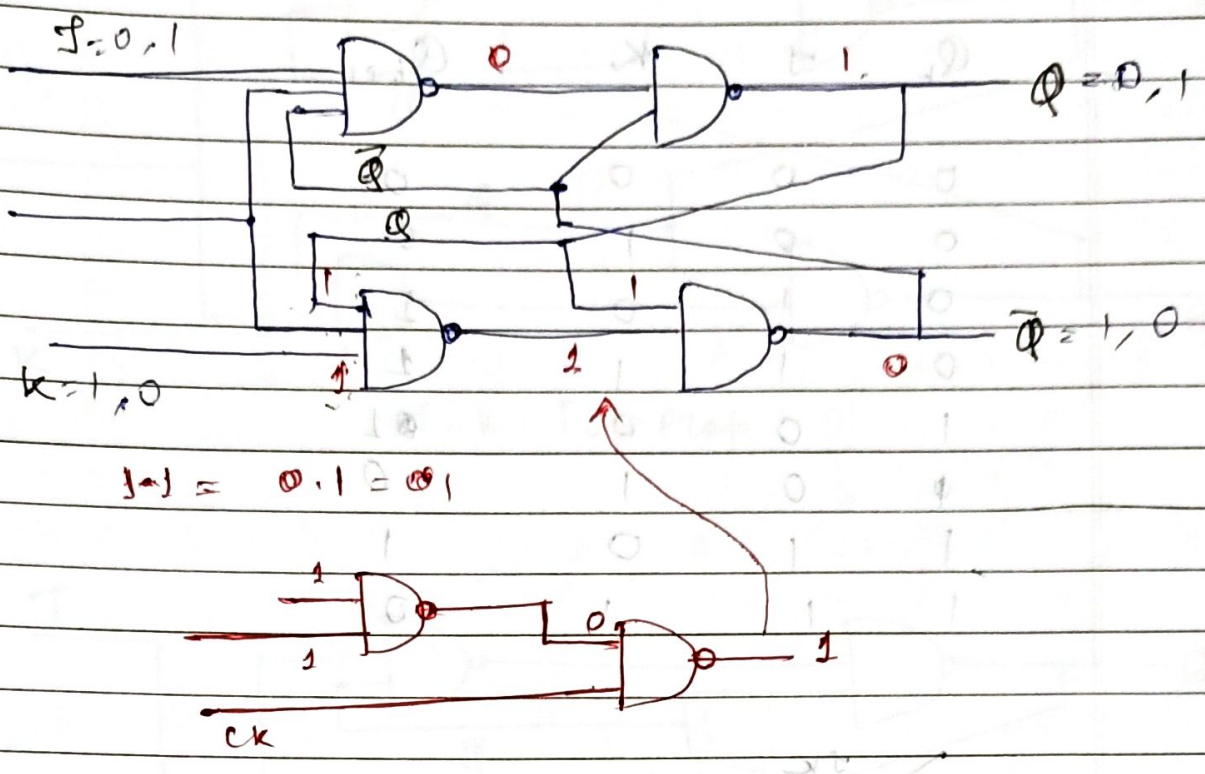
characteristic table of D-flip-flop.

So,

$$Q_{t+1} = D$$

→ this D flip-flop is used in registers.

⇒ J-K Flip Flop :-



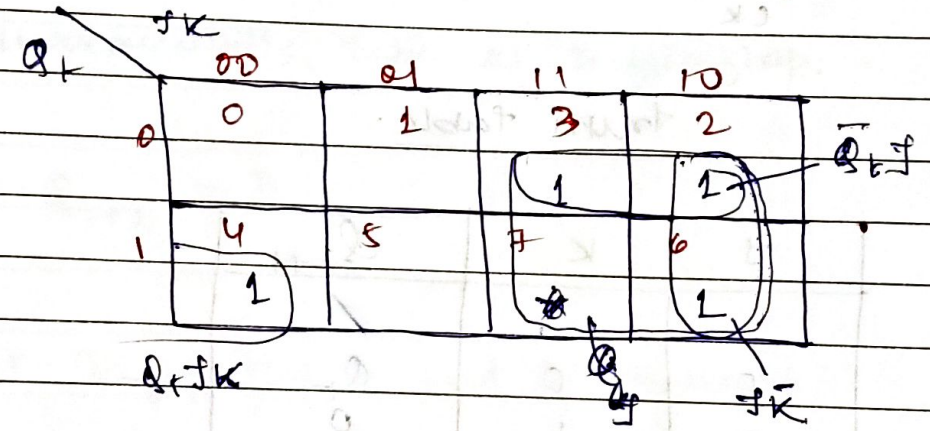
Truth Table.

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\bar{Q}_t$

Page No. :
Date: / /

Characteristic table:

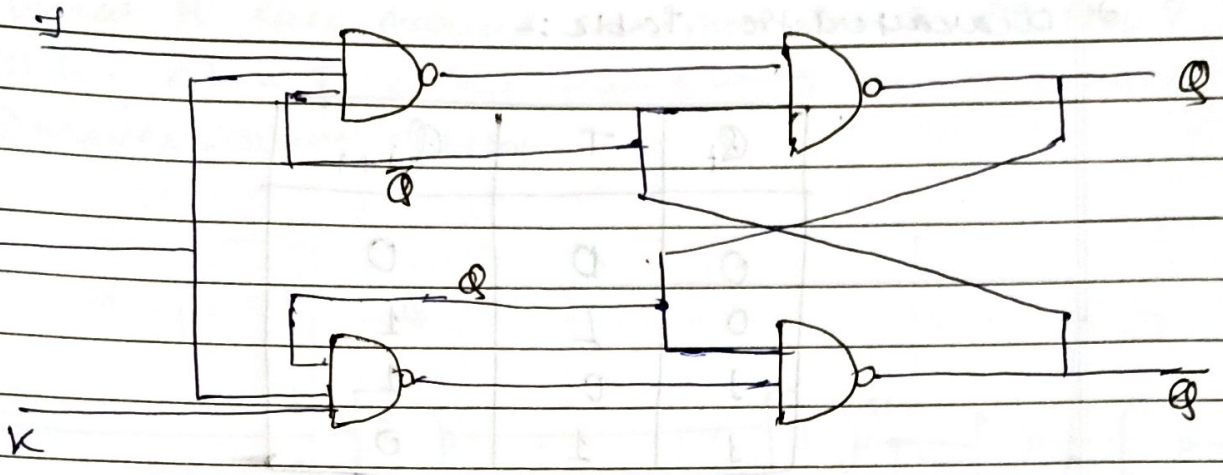
Q_t	J	K	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



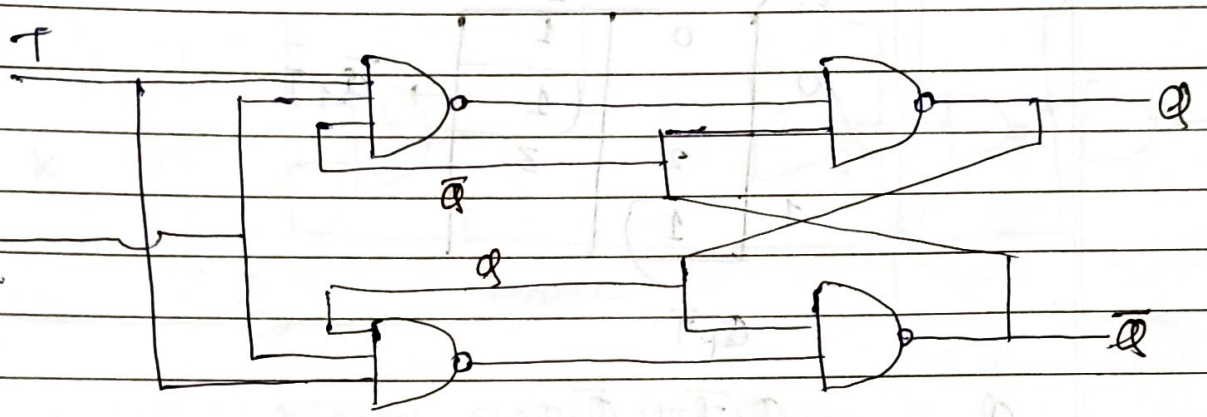
$$Q_{t+1} = J\bar{Q}_t + \bar{K}Q_t$$

Toggle → T

Page No. :
Date: / /



J-K Flip Flop.

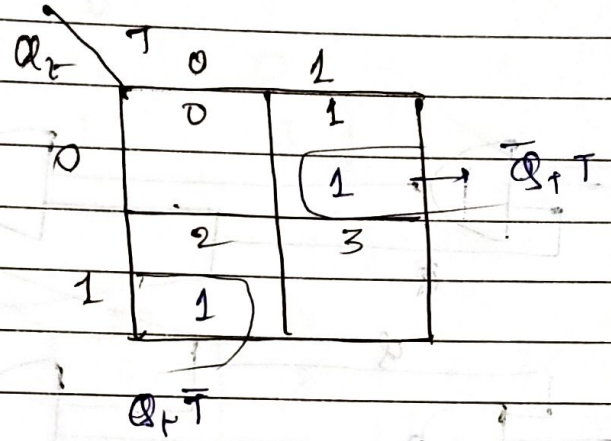


T Flip-Flop.

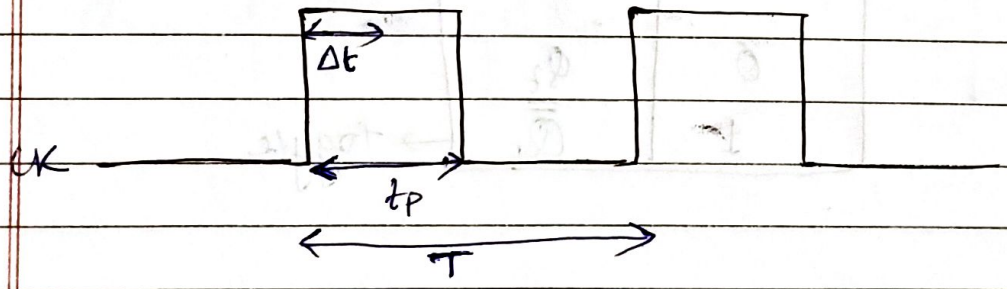
T	Q_{t+1}
0	Q_t
1	$\bar{Q}_t$ → Toggle

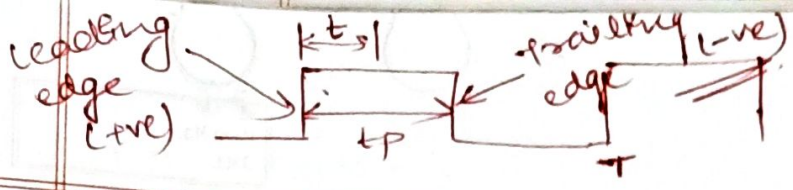
⇒ Characteristic table :-

Q_t	T	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0



$$Q_{t+1} = Q_t \bar{T} + \bar{Q}_t T$$
$$= Q_t \oplus T$$

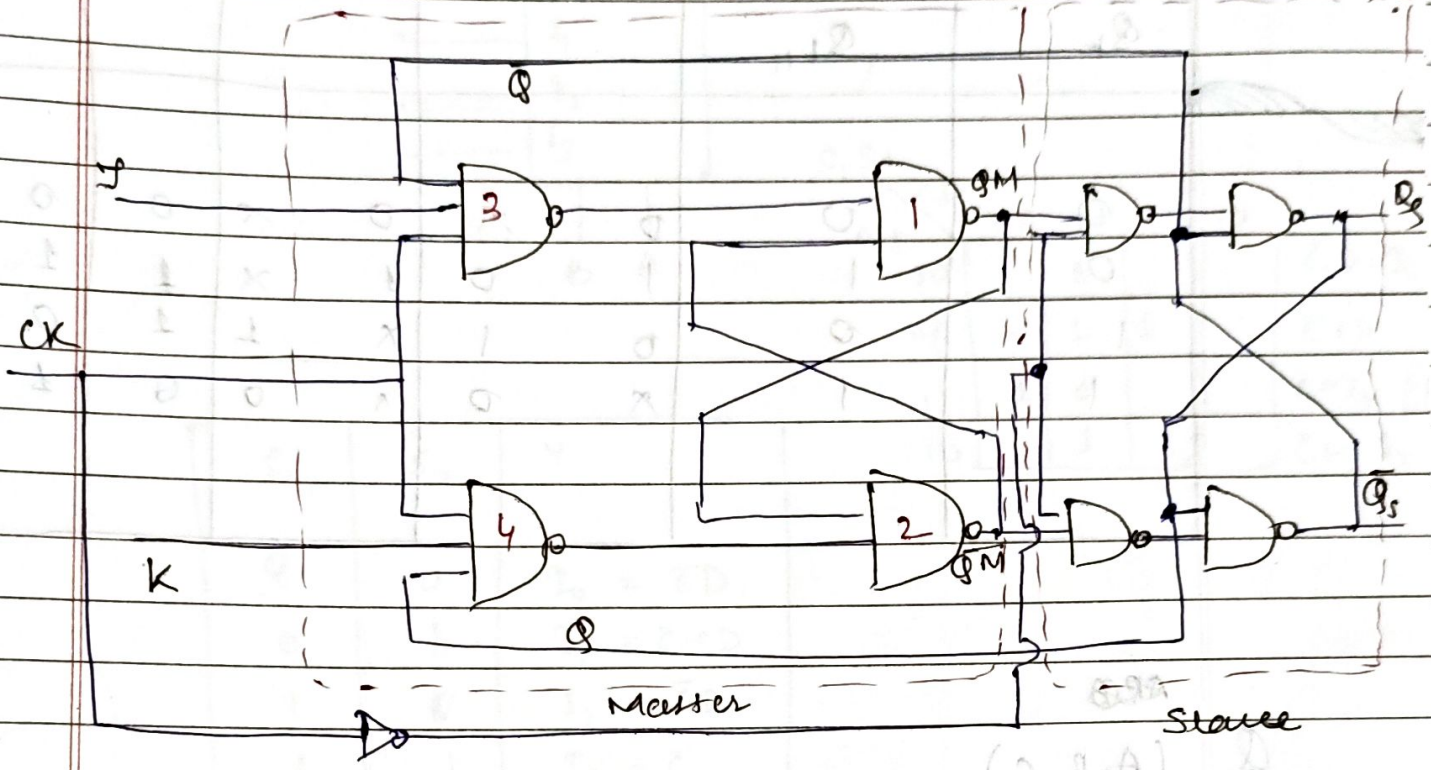




Q What is Race Around Condition in JK Flip-Flop?

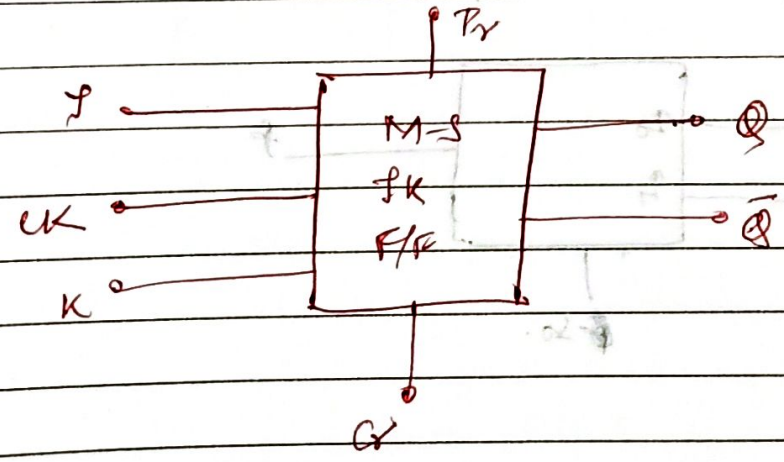
① $t_p < \Delta t < T$ ← Race around condn can be avoided

② Master-Slave Flipflop.



Master-slave Flop-flop.

→ In this master-slave flip-flop the inputs to the gates of master flip-flop gate 3 and 4 do not change during the clock-pulse. therefore the race-around condition does not exist.



→ Excitation Table :-

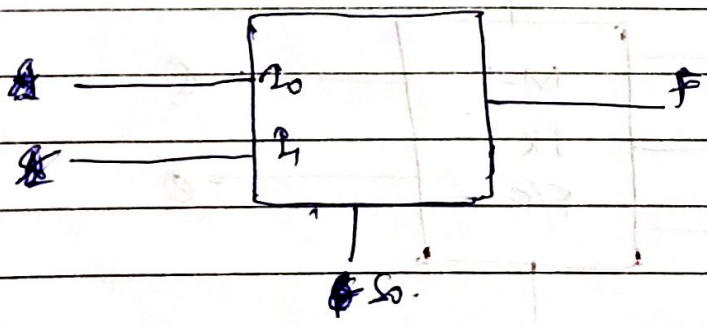
Present State Q_t	Next state Q_{t+1}	S-R F/F S_n	S-R F/F R_n	J-K F/F J_n	J-K F/F K_n	T F/F T_n	D F/F D_n
0	0	0	x	0	x	0	0
0	1	1	0	1	x	1	1
1	0	0	1	x	1	1	0
1	1	x	0	x	0	0	1

Q. (A, B, C)
$S = \Sigma(1, 2, 4, 7)$ using 2:1 MUX.

→

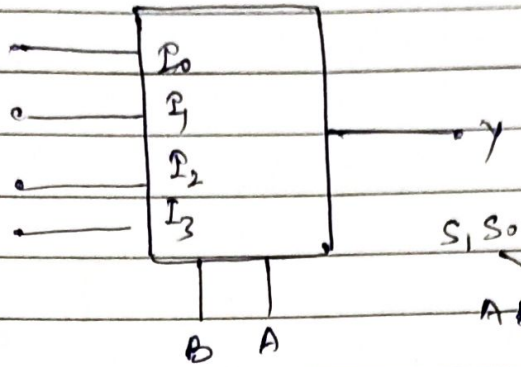
	I_0	I_1
$A\bar{B}$	0	(1)
$\bar{A}B$	(2)	3
$A\bar{B}$	(4)	5
AB	6	(7)

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



Page No. :
Date: / /

Q. $F(A, B, C, D) = \Sigma m(1, 4, 5, 7, 9, 12, 13)$ using 4:1 MUX.

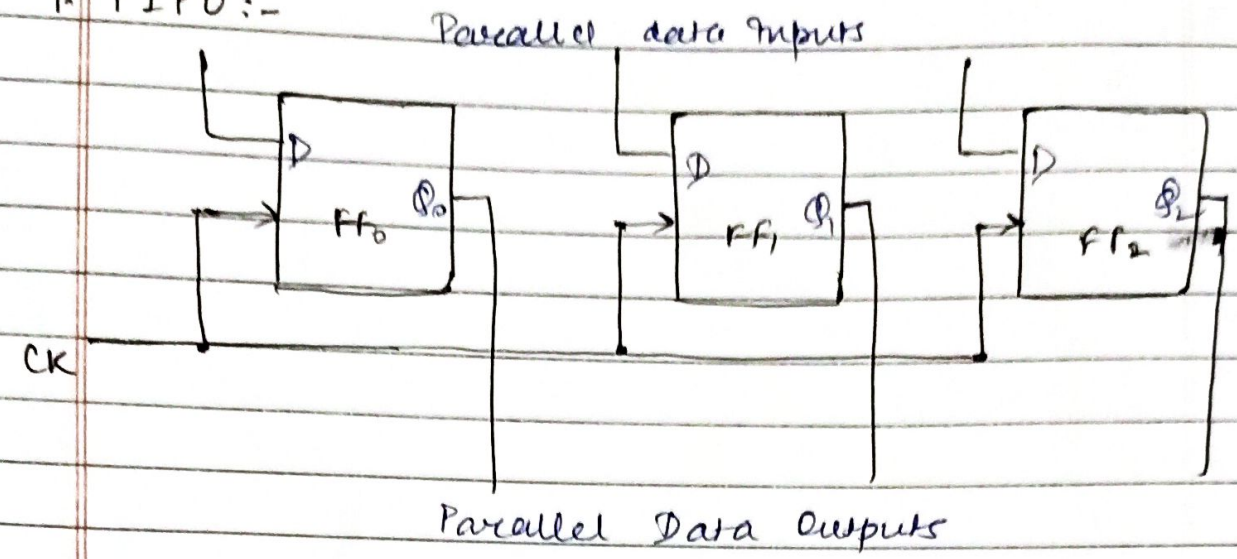


$S_1 S_0$ = CD

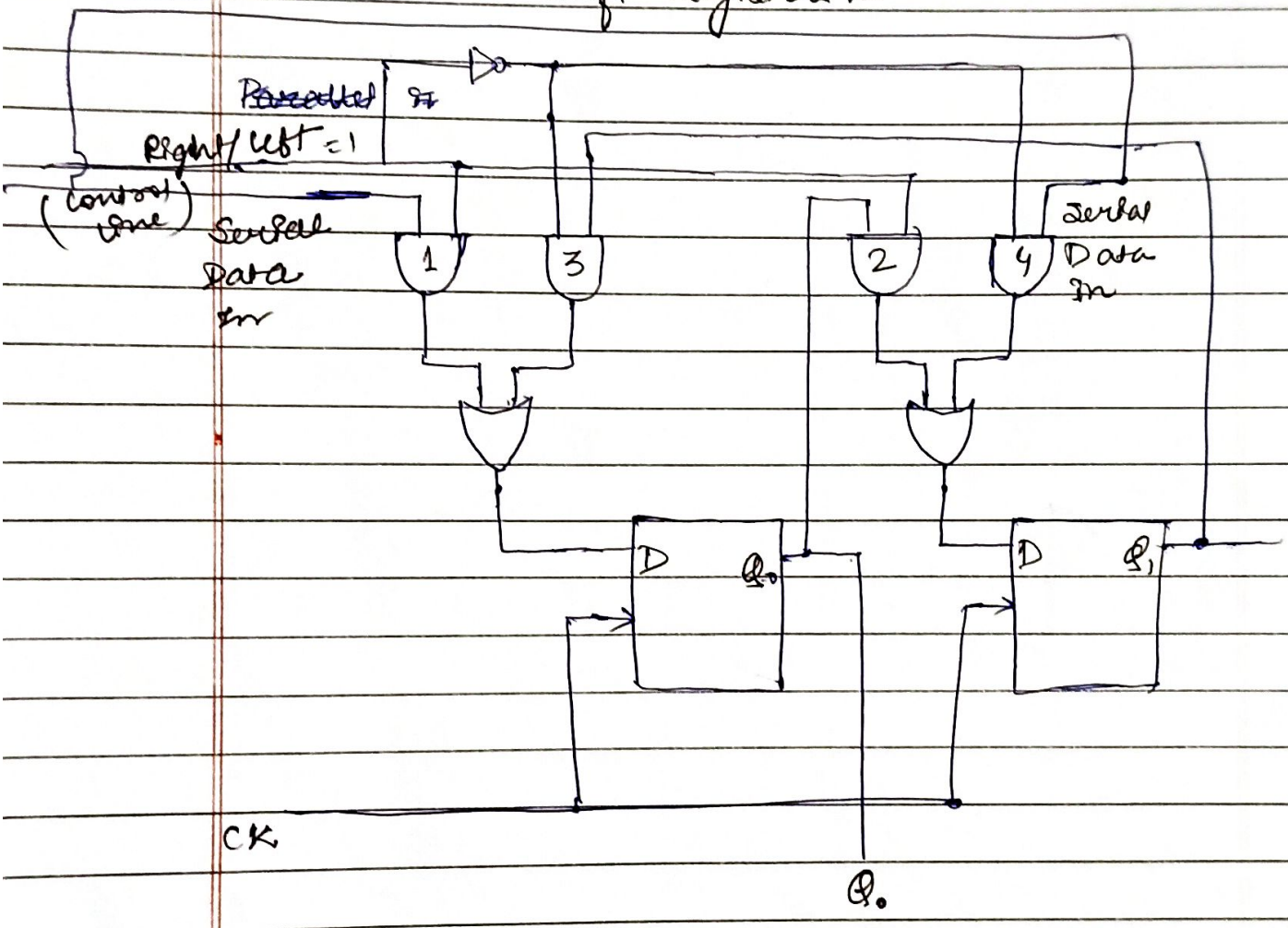
AB \ CD	00	01	11	10	
00		1			$\bar{C}D = I_0$
01	1	1	1		$\bar{C} + D$
11	1	1			$\bar{C} = I_3$
10		1			$\bar{C}D = I_2$

S_1	S_0	Y
0	0	$I_0 = \bar{C}D$
0	1	$I_1 = \bar{C} + D$
1	0	$I_2 = \bar{C}D$
1	1	$I_3 = \bar{C}$

4. PIPO :-



⇒ Bidirectional Shift Register :-



Page No. :
Date: / /

Assignment Ques.

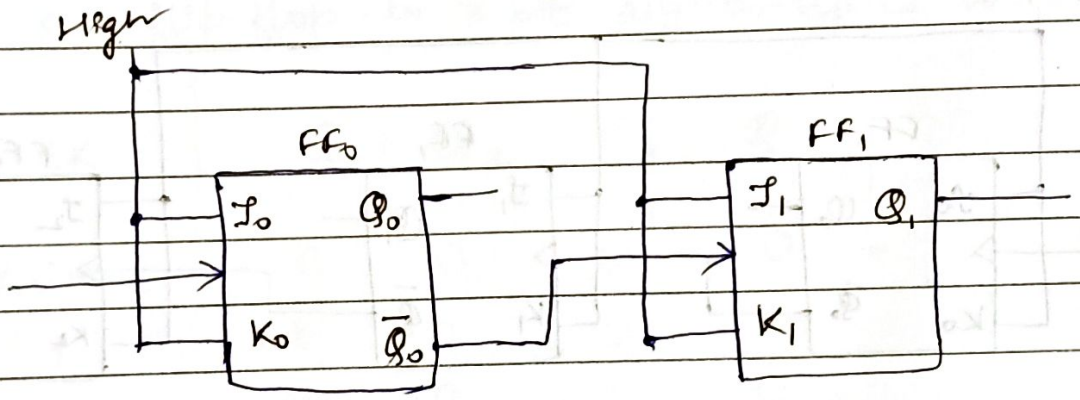
1. Write the application of shift registers.
2. Discuss, how a shift register counter differs from a basic shift register.
3. Explain the operation of a Johnson counter and a Ring counter, and specify the sequence for any number of bits in both the Johnson counter and Ring counter.

⇒ Shift Register Applications.

⇒ **Counters :-**

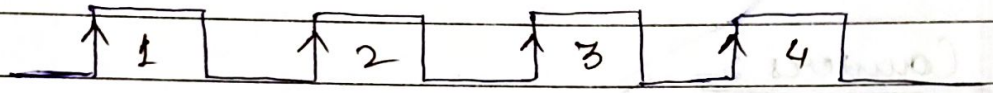
→ Asynchronous counter.
→ Synchronous counter.

1. Asynchronous Counter Operation :-



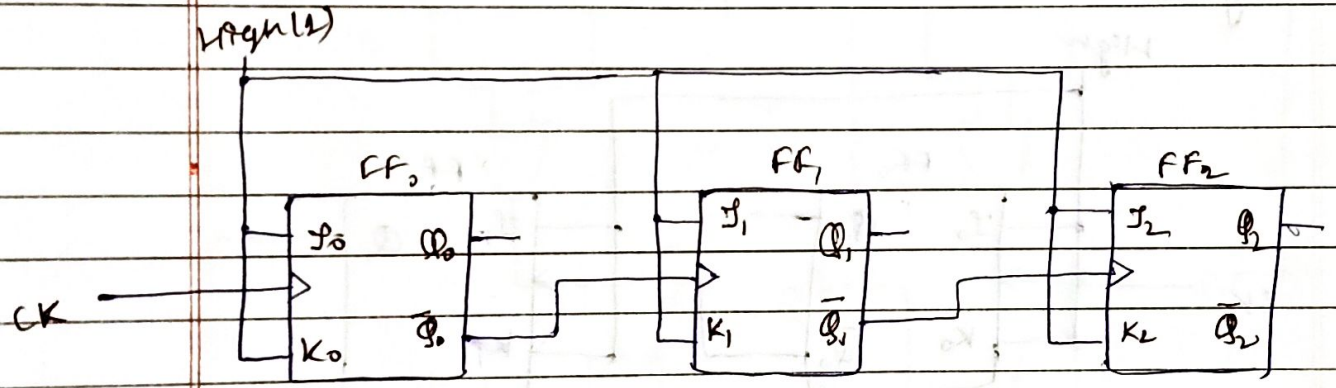
2-bit Asynchronous counter.

Clock Pulse	Q_1	Q_0
Initially	0	0
1	0	1
2	1	0
3	1	1
4	0	0 ← recycles to the initial state.



Q_0

Q_1



3-bit Asynchronous counter.

2^n ← maximum no. of states
where, n = no. of flip flops

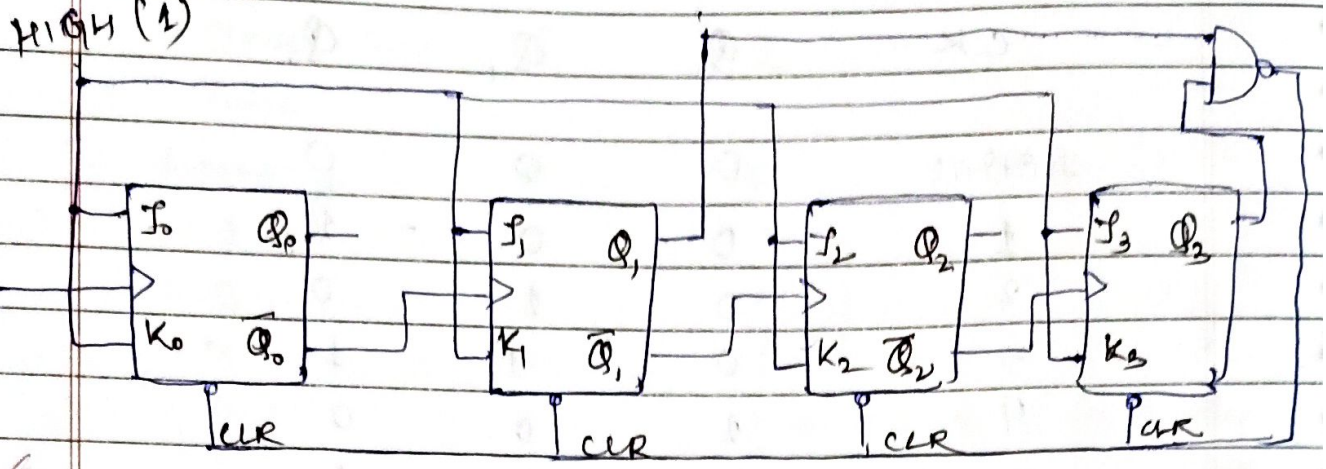
CK	Q_2	Q_1	Q_0	
Initial	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	
8	0	0	0	← recycle to the initial state.

Here,
$2^3 = 8$ states, $N = 4$

⇒ Asynchronous Decade Counters :- (10 states)
(MOD 10 counter)

→ Add one flip-flop to 3-bit asynchronous counter.

CK	Q_3	Q_2	Q_1	Q_0	
Initial	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	0	0	0	0	← recycle

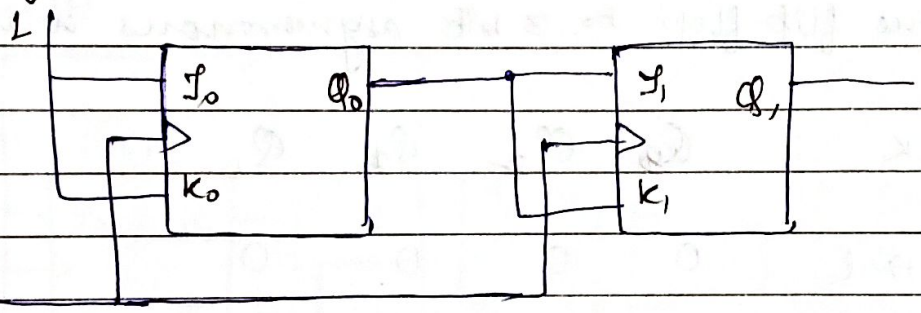


$Q_3 Q_2 Q_1 Q_0$
1100

Q Design a MOD 12 counter.

Q Design a MOD 6 asynchronous counter. Also draw timing diagram.

⇒ Synchronous counter:-

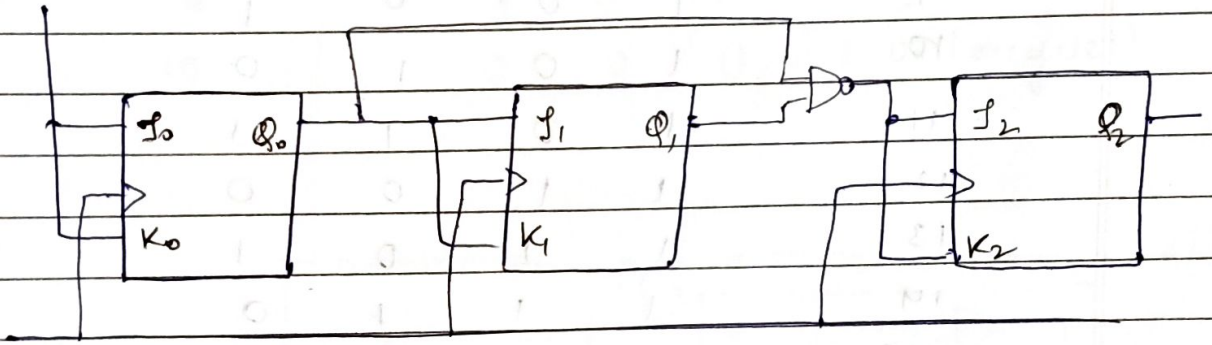


2- bit synchronous counter

CK	Q_1	Q_0	
Initial	0	0	
1	0	1	
2	1	0	
3	1	1	
4	0	0	← recycles.

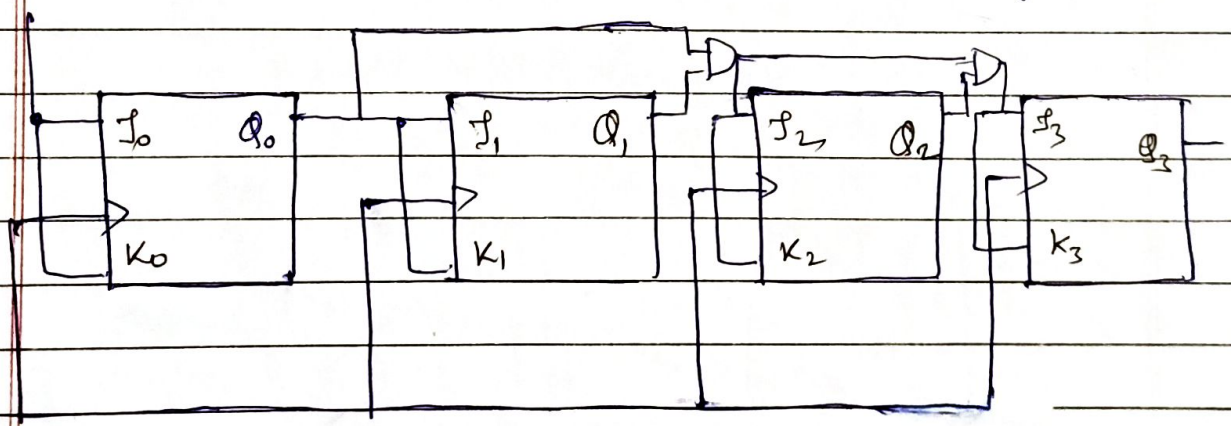
⇒ 3-bit Synchronous Counter :-

CK	Q_2	Q_1	Q_0	
Initial	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	
8	0	0	0	(recycle)



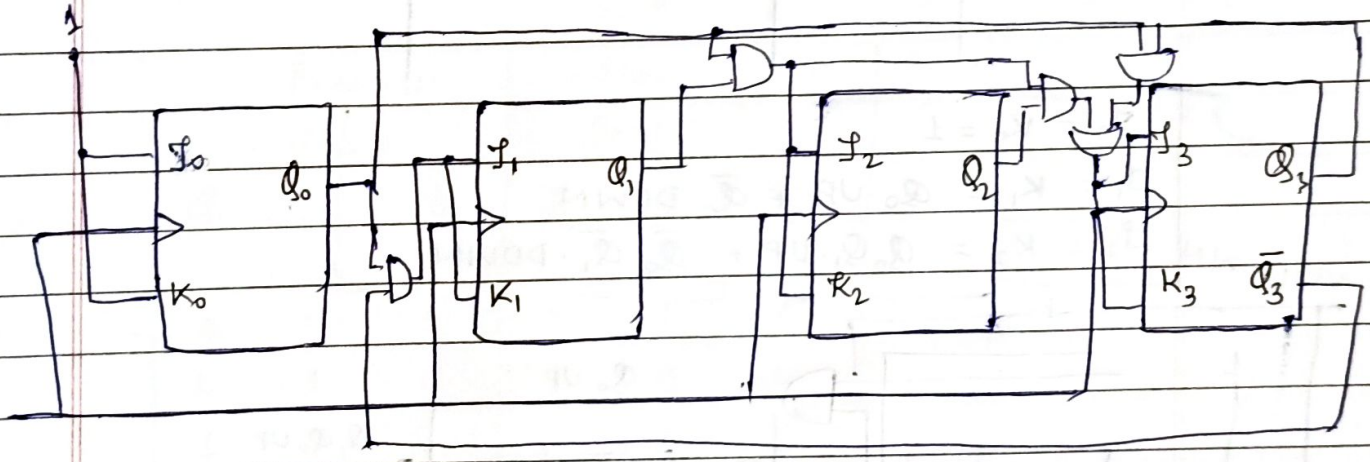
→ 4-Bit Synchronous counter :-

CK	Q_3	Q_2	Q_1	Q_0
Initial	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
16	0	0	0	0 (recycle)



⇒ Decade Synchronous Counter :-

CK	Q_3	Q_2	Q_1	Q_0
Initially	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	0	0	0	0 (recycles)



$J_0 = K_0 = 1$

$J_1 = K_1 = Q_0 \bar{Q}_3$

$J_2 = K_2 = Q_1 Q_0$

$J_3 = K_3 = Q_2 Q_1 Q_0 + Q_3 Q_0$

Page No.:
Date: / /

Imp

2. Design up-down synchronous counter.
3. Design up-down asynchronous counter.

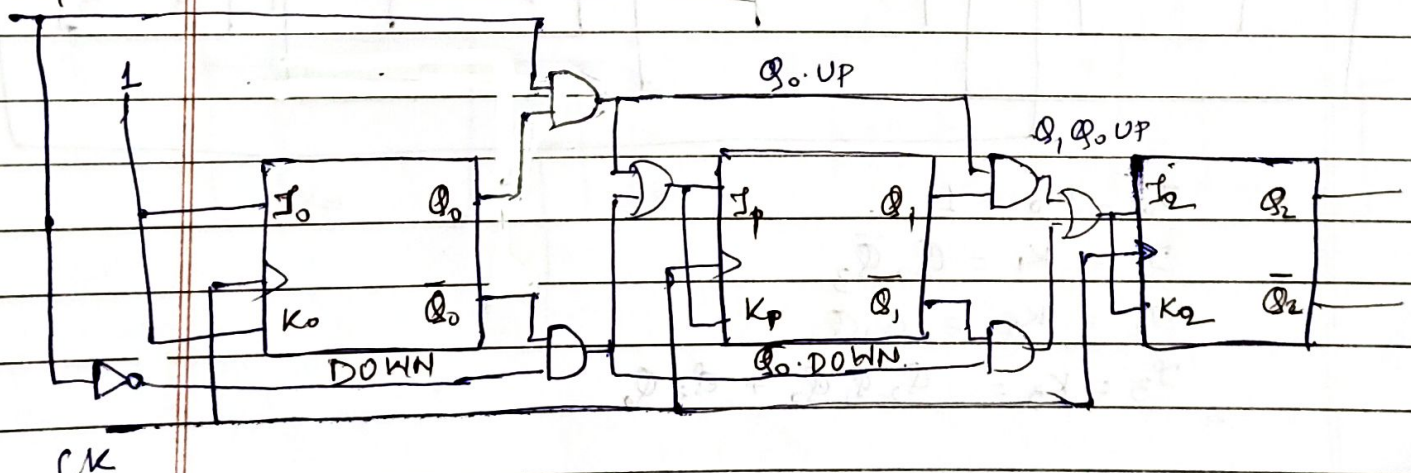
→ 3-bit up-down synchronous counter.

CK	UP	Q_2	Q_1	Q_0	Down
Initial.		0	0	0	
1		0	0	1	
2		0	1	0	
3		0	1	1	
4		1	0	0	
5		1	0	1	
6		1	1	0	
7		1	1	1	

$J_0 = K_0 = 1$

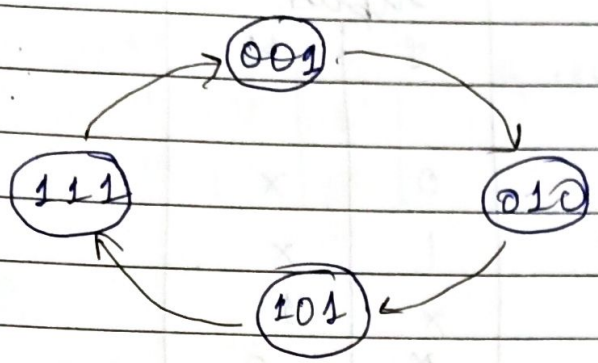
$J_1 = K_1 = Q_0 \cdot UP + \bar{Q}_0 \cdot DOWN$

$J_2 = K_2 = Q_0 Q_1 \cdot UP + \bar{Q}_0 \cdot \bar{Q}_1 \cdot DOWN$



Page No. :
Date: / /

Q Design a counter with the irregular Binary count sequence as shown in the state diagram.



State Diagram.

→ It is represented in Binary form.

$1 \rightarrow 2 \rightarrow 5 \rightarrow 7$.

⇒ Next State Table :- (Synchronous)

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	1	0	1	0
0	1	0	1	0	1
1	0	1	1	1	1
1	1	1	0	0	1

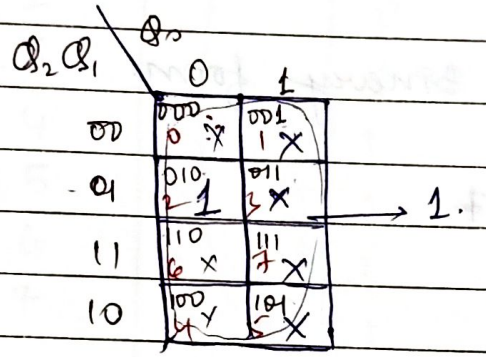
→ It is obtained by state diagram.

Q Design using J-K flip-flop.

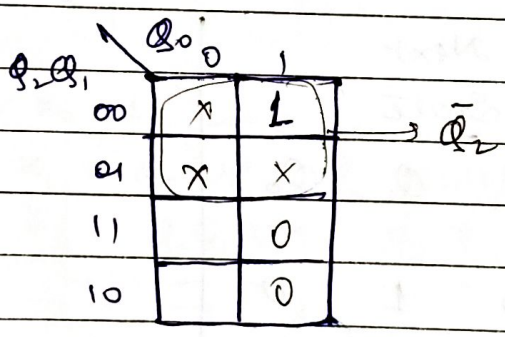
→ Excitation table :-

Output		Inputs	
Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

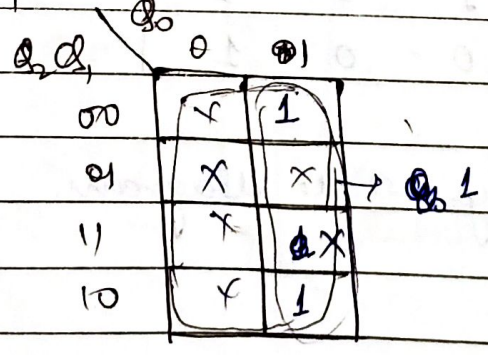
For J_0



For K_0



For J_1



For K_1

Q_2Q_1 \ Q_0	0	1
00	X	X
01	1	X
11	X	1
10	X	X

→ Q_1

For J_2

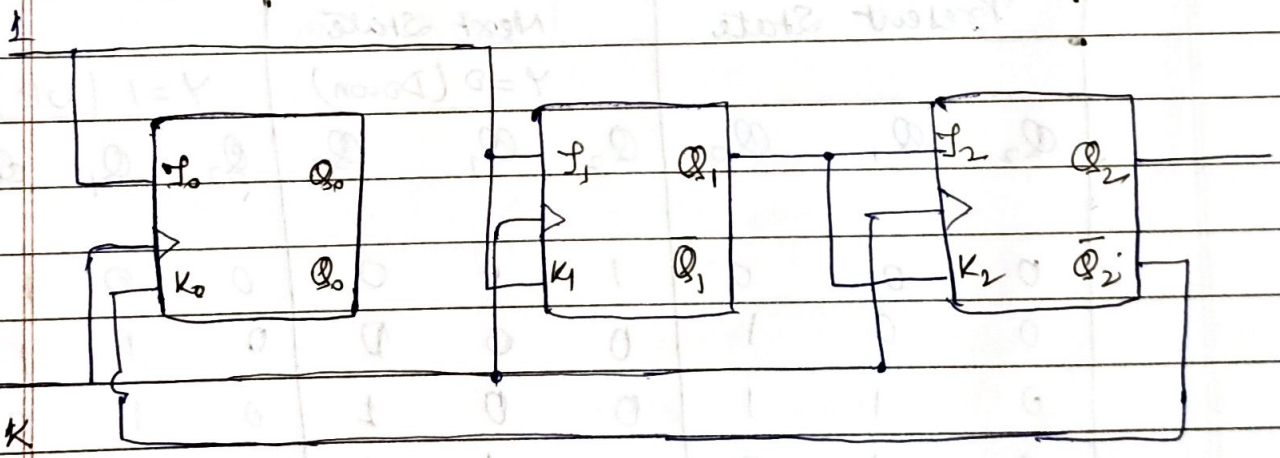
Q_2Q_1 \ Q_0	0	1
00	X	0
01	1	X
11	X	X
10	X	X

→ Q_1, $\bar{Q}_0$

For K_2

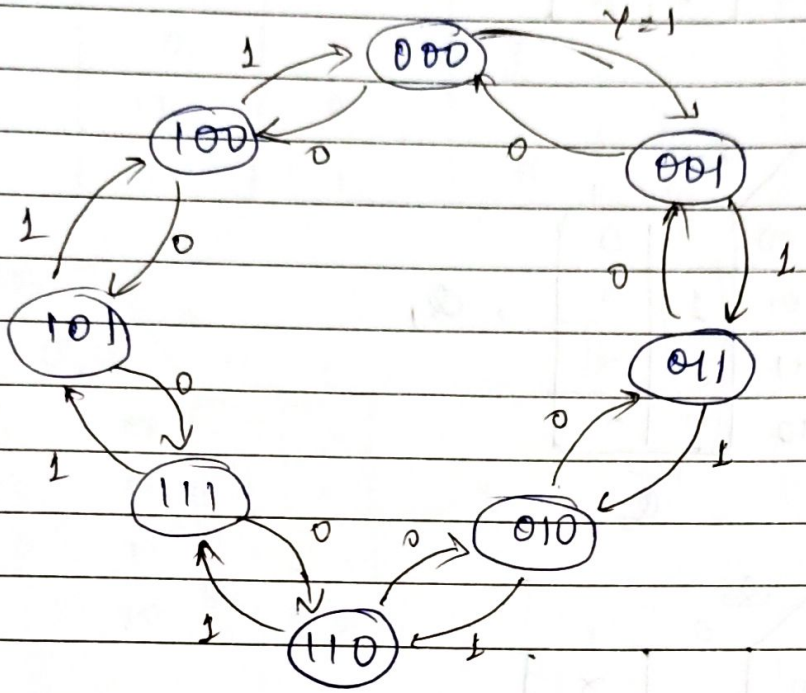
Q_2Q_1 \ Q_0	0	1
00		X
01	X	X
11	X	1
10		0

→ Q_1



Q. Develop a synchronous 3-bit up-down counter with a gray code sequence. The counter should count UP when the control I/P is 1 and DOWN when the control I/P is 0. And the counter should count the DOWN sequence, when the control I/P is 0.

→



State diagram for 3-bit up-down gray code counter.

Present State			Next State					
			Y=0 (Down)			Y=1 (UP)		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	1	0
1	1	0	0	1	0	1	0	1
1	1	1	1	1	0	1	0	0
1	0	1	1	1	1	1	0	0
1	0	0	1	0	1	0	0	0

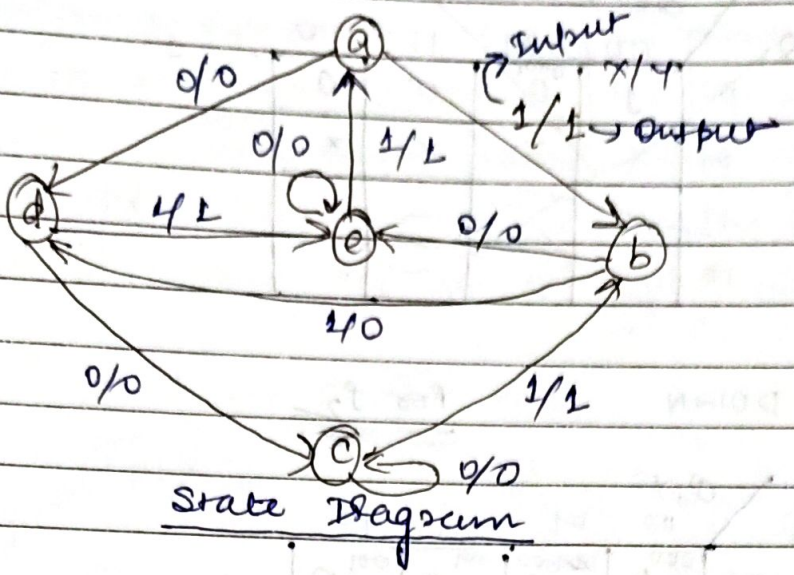
Y = UP / DOWN For J_2

Q_2Q_1 \ Q_0Y	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	X	X	X	X
10	X	X	X	X

$$J_2 = \bar{Q}_2\bar{Q}_1\bar{Q}_0\bar{Y} + Q_2\bar{Q}_1Q_0\bar{Y}$$
$$= \bar{Q}_2\bar{Q}_1(\bar{Q}_0\bar{Y} + Q_0Y)$$
$$= \bar{Q}_2\bar{Q}_1(Q_0 \oplus Y)$$

$$J_2 = \bar{Q}_1\bar{Q}_0\bar{Y} + Q_1\bar{Q}_0Y$$
$$= \bar{Q}_0(\bar{Q}_1\bar{Y} + Q_1Y)$$
$$= \bar{Q}_0(Q_1 \oplus Y)$$

Q.



State Diagram

Present State	Next State		Output (Y)	
	x = 0	x = 1	x = 0	x = 1
a	d	b	0	1
b	e	d	0	0
c	c	b	0	1
d	c	e	0	1
e	e	a	0	1

→ Equivalent State :-

Two states are said to be equivalent if the output and the next state are same for each input condy. Once the equivalent states are found, one of them may be eliminated without

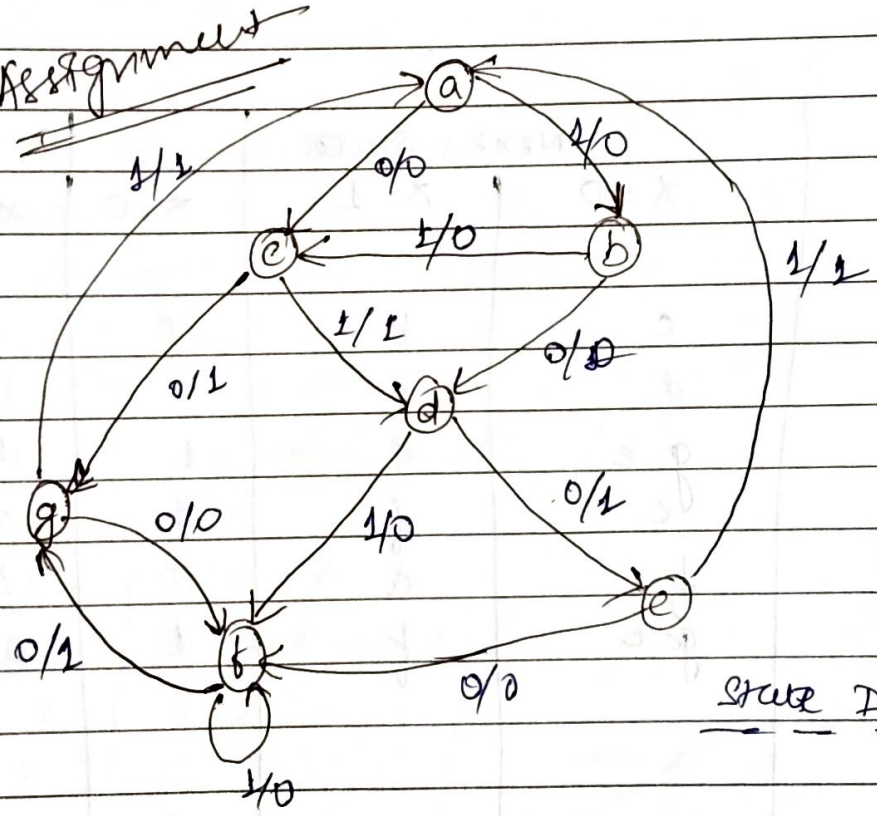
altering the I/P and O/P relations.
State Reduction.

⇒ State Assignment:-

→ The no. of bits used to assign binary values to alphabets depends on no. of alphabets used. If the no. of alphabets is M, then ~~M bits~~ n bits are required to represent the alphabets in binary such that $2^n \geq M$.

→ The selection of n is as many as possible.

Q. Assignment



State Diagram

Draw next state! Then equivalent state is replaced,

suppose $a \approx b$

then $b \to a$

if e and g are equivalent

$e \to g$

g is replaced by e

Page No. :
Date: / /

Next State table

Present State	Next State		Output	
	x = 0	x = 1	x = 0	x = 1
a	c	b	0	0
b	d	e	0	0
c	g	d	1	1
d	e	f	1	0
e	f	a	0	1
f	g	f	1	0
g	f	a	0	1

Present State	Next State		Output	
	x = 0	x = 1	x = 0	x = 1
a	c	b	0	0
b	d	c	0	0
c	e	d	1	1
d	e	f	1	0
e	f	a	0	1
f	e	f	1	0

Present State	Next State		Output	
	$x=0$	$x=1$	$x=0$	$x=1$
a	c	b	0	0
b	d	c	0	0
c	e	d	1	1
d	e	d	1	0
e	d	a	0	1

⇒ State Assignments –

Here,

No. of alphabets, $M = 5$

Now,

we have,

$$2^n \geq M$$

$$\Rightarrow 2^n \geq 5$$

So, we can take $n = 3$.

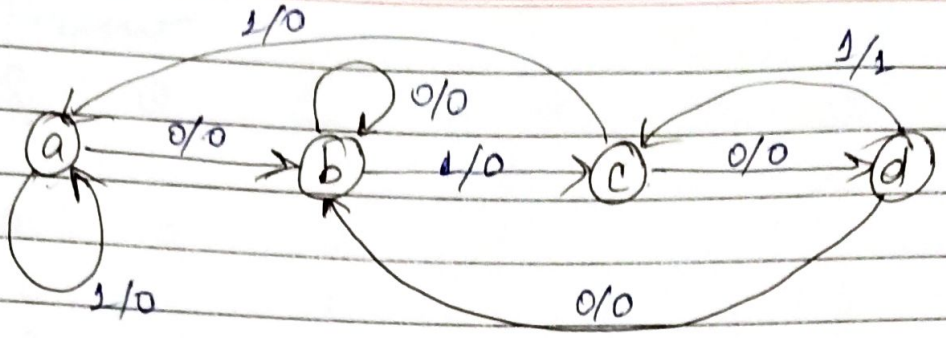
Here,

No. of bits, $n = 3$.

Variable	Assigned value
a	0 0 0
b	0 0 1
c	0 1 0
d	0 1 1
e	1 0 0

Page No.:
Date: / /

Q



→ Next state table

Present State	Next State		Output	
	x = 0	x = 1	x = 0	x = 1
a	b	a	0	0
b	b	c	0	0
c	a	d	0	0
d	c	b	0	1

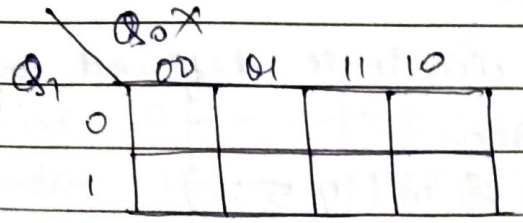
$2^n \geq 4$

then value of $n = 2$.

a	0 0
b	0 1
c	1 0
d	1 1

Q_t	Q_{t+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Q_1	Q_0	Y = 0 Q_1	Y = 0 Q_0	Y = 1 Q_1	Y = 1 Q_0
0	0	0	1		
0	1	1	0		
1	0	1	1		
1	1				

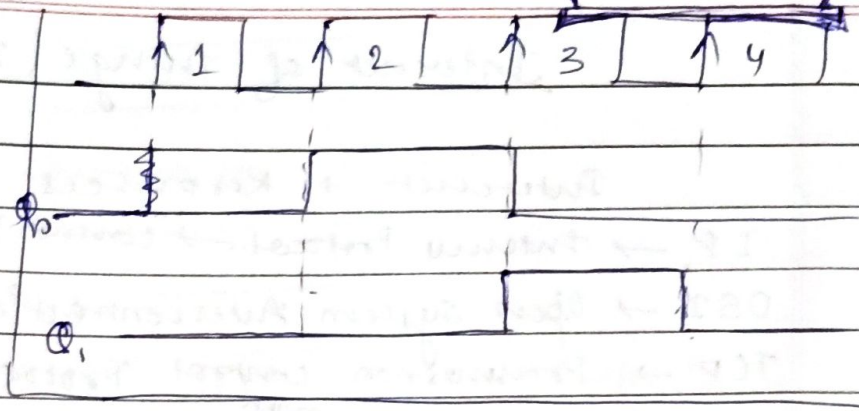


Present State	Next State		Output	
	$X=0$	$X=1$	$X=0$	$X=1$
a 00	01	00	0	0
b 01	01	10	0	0
c 10	00	11	0	0
d 11	10	01	0	1

D.

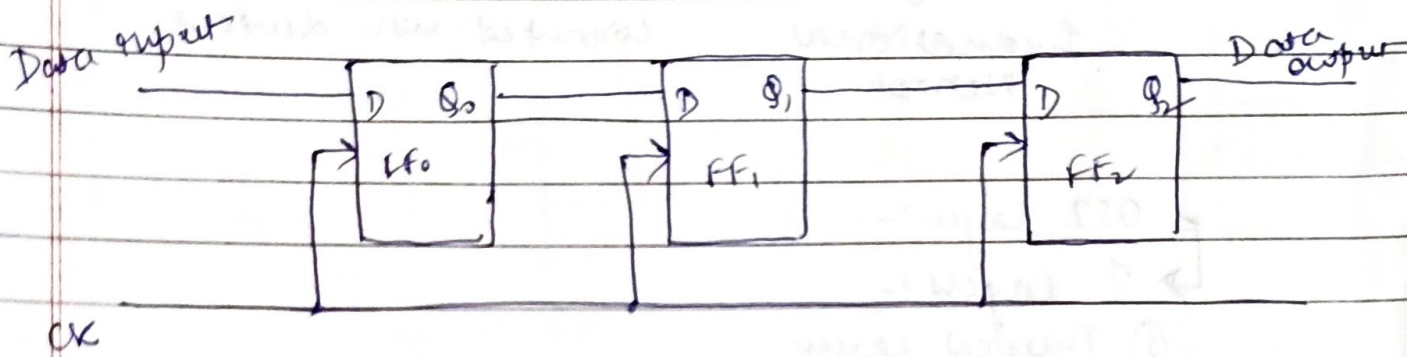
Registers:-

1. SISO
2. SIPO
3. PIPO
4. PISO

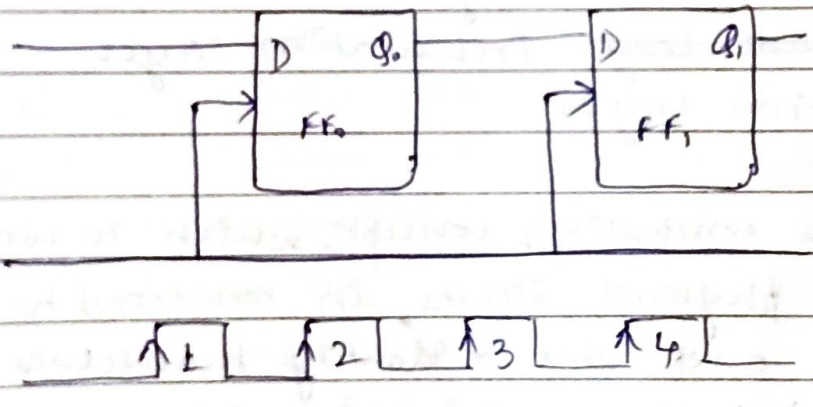


→ n bit Registers will have n number of flip-flop.

1. SISO.



3 bit right shift registers.



CK	Q_0	Q_1	
	0	0	Initial
1	0	0	
2	1	0	
3	0	1	
4	0	0	(reset)

Page No.:
Date: / /

Q A combinational circuit is defined by the following function.

$F_1(A,B,C) = \Sigma m(4,5,7)$

$F_2(A,B,C) = \Sigma m(3,5,7)$.

Implement this circuit with a PLA having 3 I/P, 3 product terms and 2 O/P terms.

→ $F_1(A,B,C) = A\bar{B}\bar{C} + A\bar{B}C + ABC$

$F_2(A,B,C) = \bar{A}BC + A\bar{B}C + ABC$

$F_1(A,B,C) = A\bar{B}\bar{C} + A\bar{B}C + ABC$

$= A\bar{B} + ABC$

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

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

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

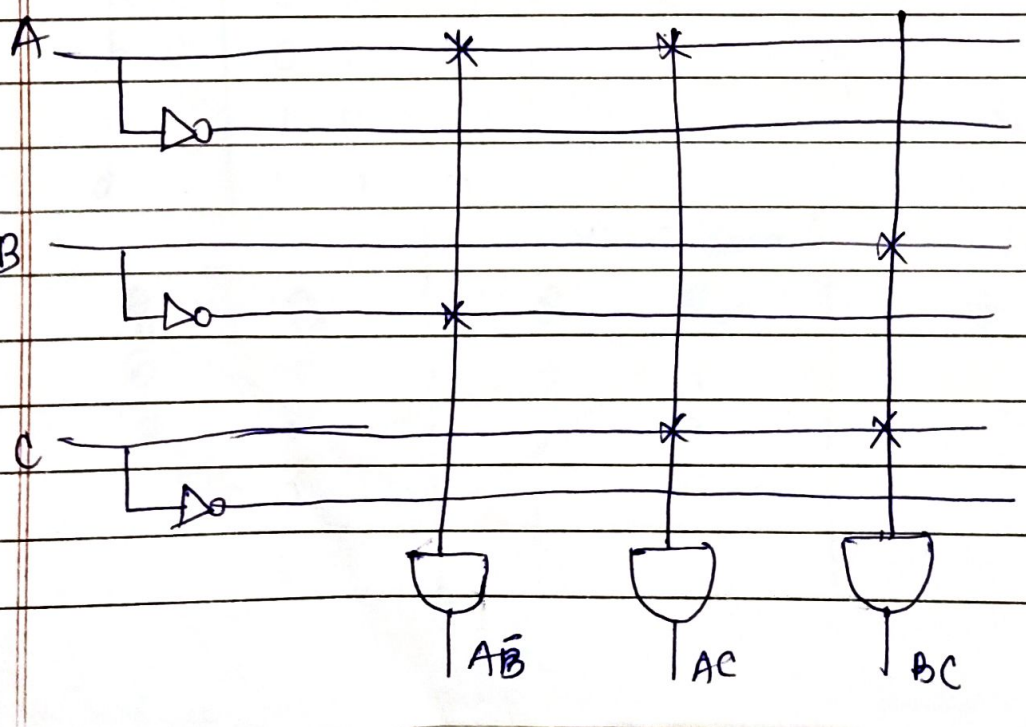
$F_2(A,B,C) = \bar{A}BC + A\bar{B}C + ABC$

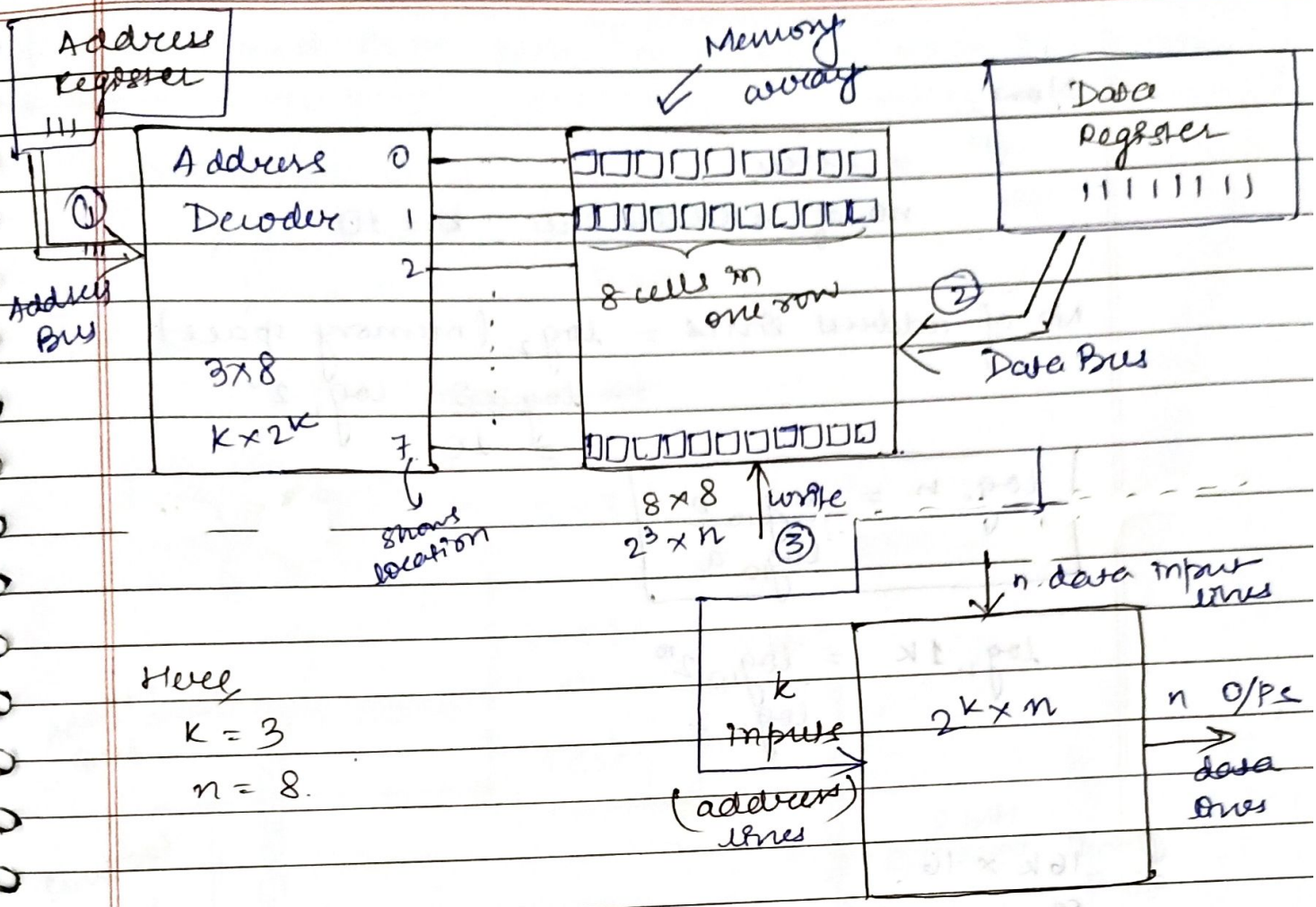
$= BC + A\bar{B}C$

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

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

$= BC + AC$





Here,
$k = 3$
$n = 8$.

Block Diagram of a RAM

- Write → means data will store in this register.
- n → denotes no. of bits in each word.
- 2^K → indicates no. of words the RAM is having.

Q. Consider a memory with capacity 1K × 16. Find the no. of address lines. No. of bits in each word.

→

1 KB = 2^{10} Bytes.
1 MB = 2^{20} Bytes
1 GB = 2^{30} Bytes

1 TB = 2^{40} Bytes.

2^{10} Bytes

1 Byte = 8 bits. $2^{10} \times 8$ 10

$= 1024$

→ Now,

2^{10} → ~~1111~~

no. of address lines, $k = 10$

No. of address lines = $\log_2$ (memory space)

$= \log_2 2^{10}$

$= 10$.

$$\log_a b = \frac{\log_{10} b}{\log_{10} a}$$

$$\log_2 1K = \frac{\log_{10} 2^{10}}{\log_{10} 2}$$

Q 16k × 16.

Find no. of address lines.

→ No. of address lines = $\log_2 (2^4 \times 2^{10})$

$= \log_2 2^{14}$

$= 14$

⇒ **Word length expansion:-**

Q 65,536 × 4 to 65,536 × 8

→ No. of ROM chips required = $\frac{65,536 \times 8}{65,536 \times 4} = 2$.

⇒ RAM and ROM both have same block Diagram.

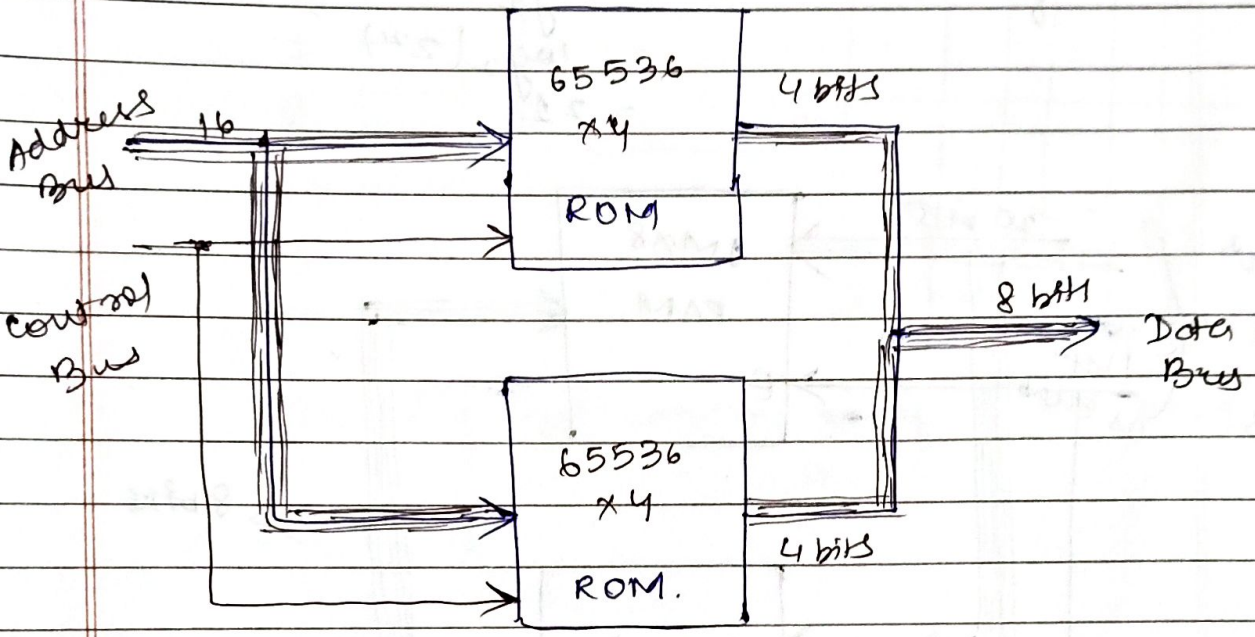
Construction:-

$$\frac{\log_{10} 65,536}{\log_{10} 2} = \frac{11.09}{0.69} = 16$$

20,

~~$2^{16} \times 16 \Rightarrow$~~

$16 \times 2^{16} \Rightarrow k \times 2^{k}$.



65,536 × 8 ROM from two 65536 × 4 ROMs.

⇒ Word capacity expansion:-

→ To increase the capacity, the number of address lines is to be increased.

k = no of address lines

Page No.:
Date: / /

Q Use 1M × 8 to construct 2M × 8 RAMs.

→ No. of RAMs / chips required = $\frac{2M \times 8}{1M \times 8}$

= 2.

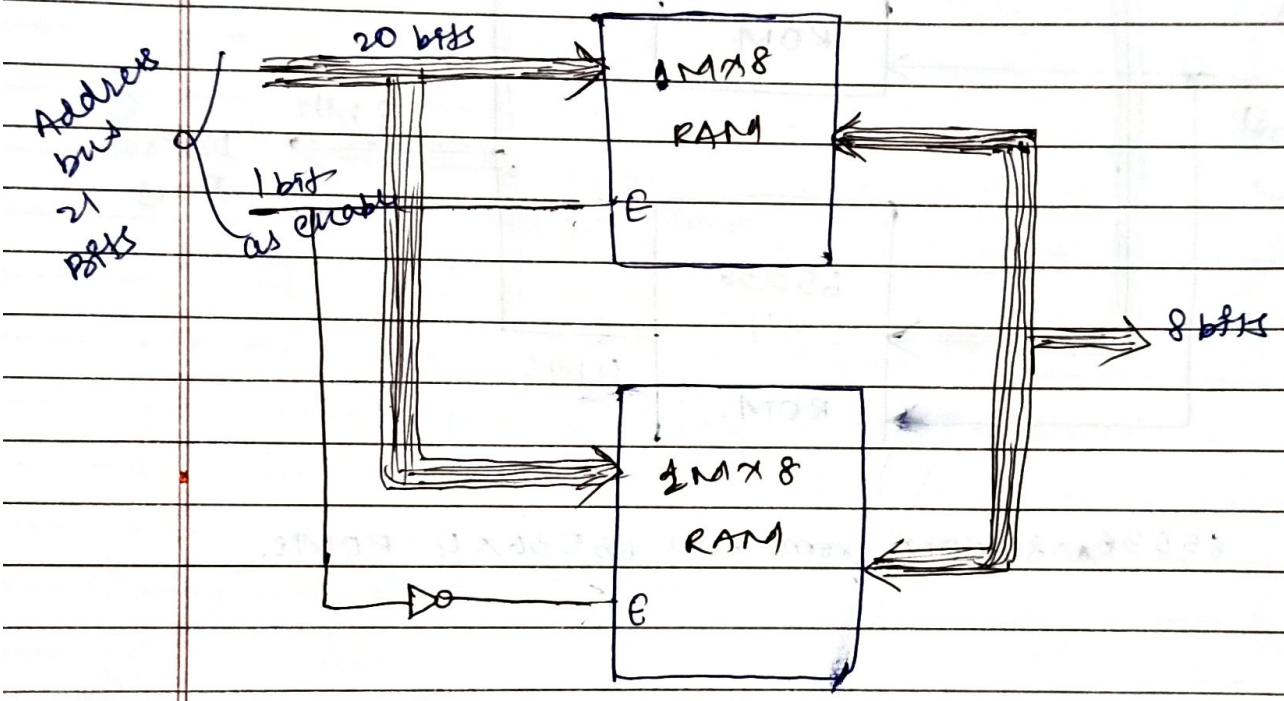
No. of address lines = $\log_2(2^{20})$

= 20.

No. of address lines = $\log_2(2 \times 2^{20})$

= $\log_2(2^{21})$

= 21.

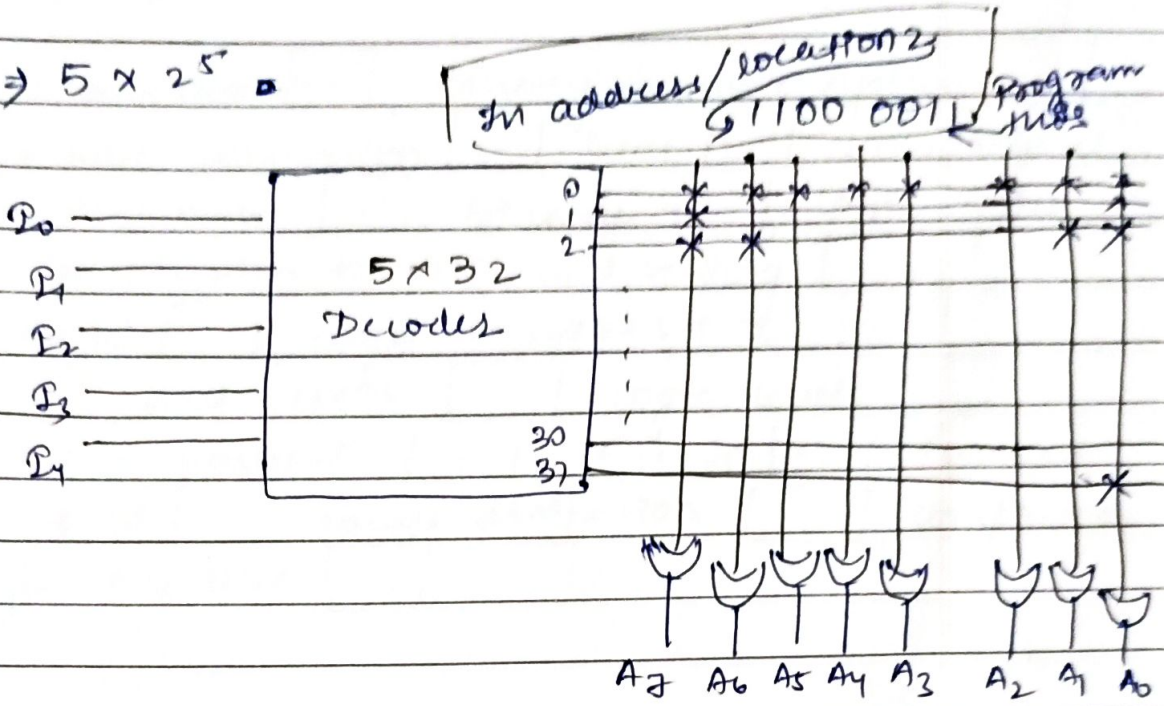


2M × 8 RAM using 1M × 8 RAMs.

Q Construct a 32 × 8 ROM.

→ No. of address lines = $\log_2(2^5)$
= 5

$k \times 5^k \Rightarrow 5 \times 2^5$ ∎



Internal construction of 32 × 8 ROM

× ⟶ denotes 1

PROM
(Programmable ROM)

Inputs					Outputs								No. of bits / data lines
P_0	P_1	P_2	P_3	P_4	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	0	0	1	1	1	1	1	1	1	1	
0	0	0	0	1	0	0	0	0	0	0	0	1	
0	0	0	1	0	1	1	0	0	0	0	1	1	
				32 bits									
1	1	1	1	1	0	0	0	0	0	0	0	1	