Date :
Page :

Syllabus

Course of Contents

Unit 1 :- Introduction

1.1 Digital Signals and Wave Forms
1.2 Digital Logic and Operation
1.3 Digital Computer and Integrated Circuits (IC)
1.4 Clock Wave Form

Unit 2 - Number System

2.1 Binary, Octal and Hexadecimal Number Systems and Their Conversion
2.1.1 Representation of Signed Numbers - Floating point Number
2.1.2 Binary Arithmetic
2.2 Representation of BCD - ASCII - Excess 3 - Gray code - Error Detecting and Correcting Codes

Unit 3 - Combinational Logic Design

3.1 Basic Logic Gates NOT, OR and AND
3.2 Universal Logic Gates NOR and NAND
3.3 EX-OR and EX-NOR Gates
3.4 Boolean Algebra :
3.4.1 Postulates and Theorems
3.4.2 Canonical Forms - Simplification of Logic Functions
3.5 Simplification of Logic Functions using Karnaugh Map.
3.5.1 Analysis of SOP And POS Expression
3.6 Implementation of Combinational Logic Functions
3.6.1 Encoders and Decoders
3.6.2 Half Adder, & Full Adder
3.7 Implementation of Data processing Circuits
3.7.1 Multiplexers and De-Multiplexers
3.7.2 Parallel Adder - Binary Adder - parity Generator/checker - Implementation of Logical Functions using multiplexers.

3.8 Basic concepts of programmable logic
3.8.1 PROM
3.8.2 EPROM
3.8.3 PAL
3.8.4 PLA

Unit 4 - Counters & Registers
4.1 RS, JK, JK Master - Slave, D & T flip flops
4.1.1 Level Triggering and Edge Triggering
4.1.2 Excitation Tables
4.2 Asynchronous and Synchronous counters
4.2.1 Ripple Counter: Circuit and State Diagram and Timing Waveforms
4.2.2 Ring Counter: Circuit and State Diagram and Timing Waveforms
4.2.3 Modulus 10 Counter: Circuit and State Diagram and Timing waveforms
4.2.4 Modulus Counters (5, 7, 11) and Design principle, Circuit and State Diagram
4.2.5 Synchronous Design of Above Counters, Circuit Diagrams and State Diagrams
4.3 Application of Counters
4.3.1 Digital Watch
4.3.2 Frequency Counter
4.4 Registers
4.4.1 Serial in parallel out Register
4.4.2 Serial in Serial out Register
4.4.3 parallel in Serial out Register
4.4.4 Parallel in parallel out Register
4.4.5 Right Shift, Left Shift Register

Date: .....
Page: .....

Unit 5 – Sequential Logic Design

5.1 Basic Models of Sequential Machines
- Concept of state
- State Diagram

5.2 State Reduction through partitioning and Implementation of Synchronous Sequential Circuits

5.3 Use of Flip-Flops in Realizing the Models

5.4 Counter Design

Evaluation

Examination scheme				
Internal Assessment		External Assessment		
Theory	practical	Theory	practical	Total
20	20	60	–	100

Unit - 1 Introduction

Date: .....
Page: .....

1. Signal

A signal is a function that represents various of a physical quantity with respect to time, pressure, distance etc. A signal is an electromagnetic or electrical current that carries data from one system or network to another.

In electronics, a signal is often a time-varying voltage that is also an electromagnetic wave carrying information, though it can take on other forms such as current.

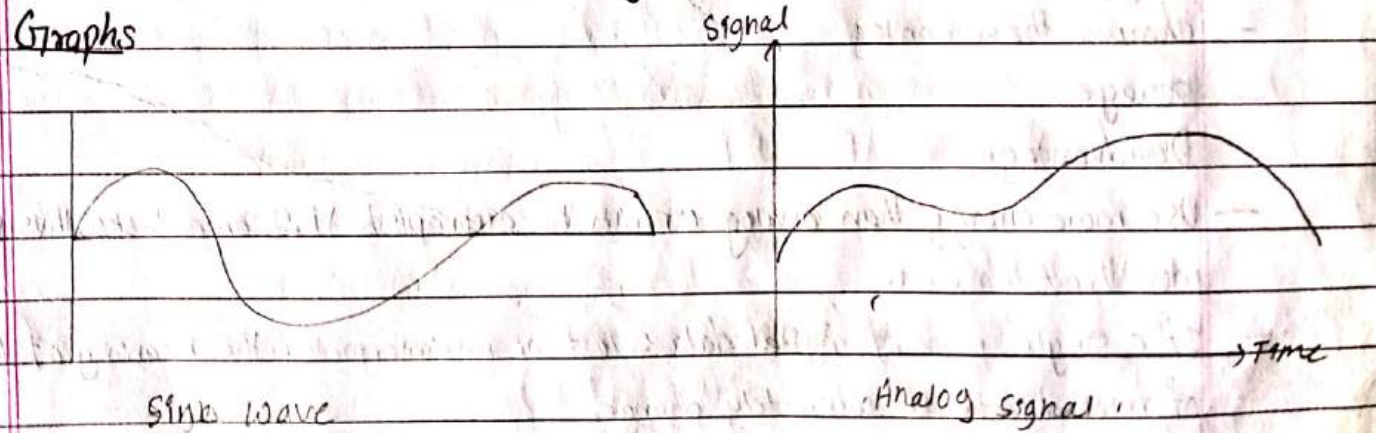
Mainly, There are two types of signals. They are:-

a. Analog Signal b. Digital Signal

a. Analog Signal

- Continuous ~~Signals~~ Signals
- Infinity range of values
- More exact value, but more difficult to work with.
- Represented by sine waves.
- Only be used in analog devices
- Human voice, natural sound, analog electronic devices are example of ~~dev~~ ~~ana~~ it

Graphs

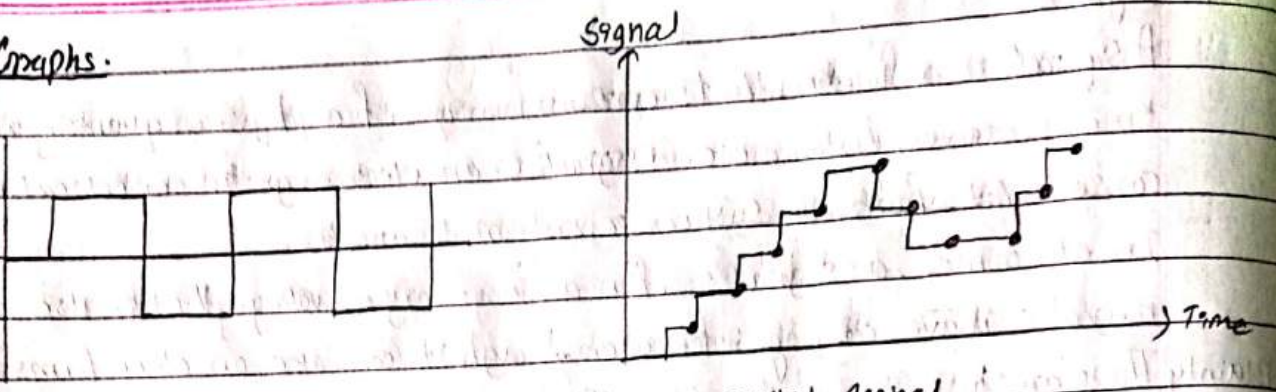


b. Digital Signal

- Discrete signals
- Finite range of values
- Not as exact as analog but easier to work with.
- Represented by square waves
- Used in digital electronics devices like mobile, computer etc.
- Computer, optical drives, mobile etc are example of it.

Date: .....
Page: .....

Graphs.



Square Wave

Digital Signal

Discrete Time Signal (DTS)

The signal which is defined for the discrete intervals of time is called discrete time signal.

2. Digital ~~Waveforms~~ Signal advantages and Disadvantages

Advantages over Analog signal

- Noise (unwanted voltage fluctuations) doesnot affect digital ~~signal~~ data
- More secure and reliable
- Long distance transfer reliable
- cheaper than analog
- Storage

Disadvantages

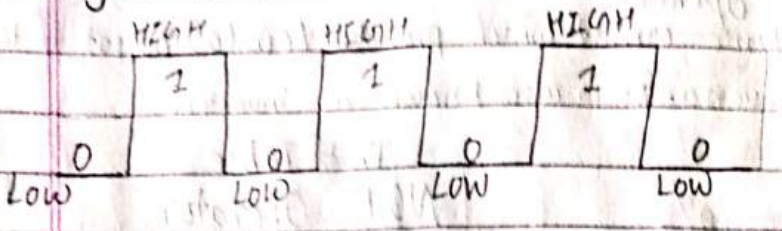
- Use more energy than analog circuits to accomplish the same tasks, thus producing more heat as well.
- If a single piece of digital data is lost or misinterpreted the meaning of large blocks of related data can completely change.
- Digital computer manipulates discrete elements of information by means of a binary code.

3. Digital Wave forms

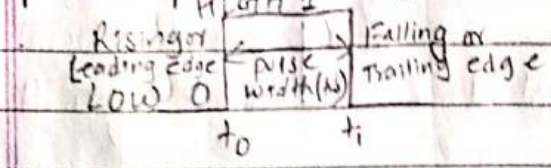
Digital waveforms consist of voltage levels that are changing back and forth between the HIGH and LOW levels or states. It is made of up series of pulses and carrys binary information.

Date:
Page:

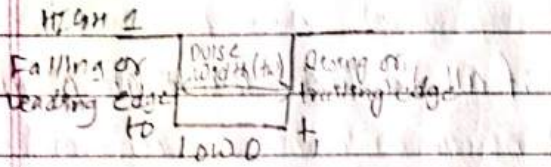
digital waveforms



a. positive pulse going pulse



b. Negative Going pulse

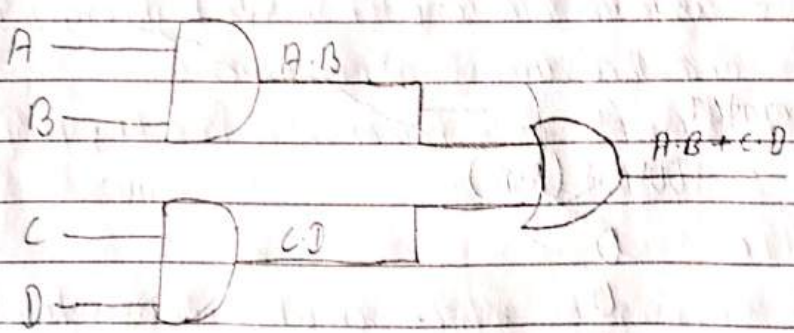


4. Digital Logic and Operation

Digital logic is the representation of signal and sequence in a digital circuit through numbers.

Logic operation is applied to digital circuits used to implement logic functions.

$F = A \cdot B + C \cdot D$



Logic Operator

There are three basic logic operator. They are:

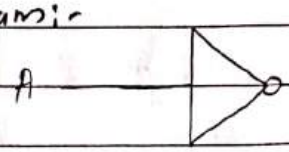
a. NOT Operator
b. AND Operator
c. OR Operator

Date: .....
Page: .....

a. NOT Operator

NOT Operator is an ~~electronic circuit~~ Operator which provides Low (0) output if the input is HIGH (1) and vice-versa. It is also known as inverter.

Diagram:-



∴ NOT Operator.

∴ Function (F) = $\bar{A}$

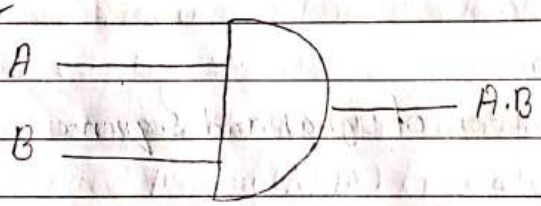
Truth Table

NOT Operator	
Input (A)	Output ($\bar{A}$)
0	1
1	0

b. AND Operator

AND Operator provides HIGH Output if all the input are HIGH otherwise it provides LOW output.

Diagram:-



∴ Function (F) = A.B

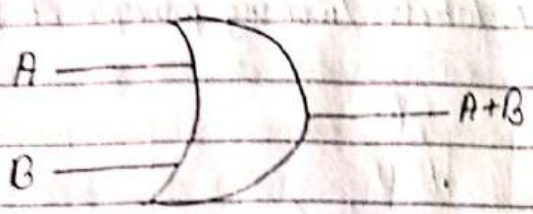
Truth Table

AND Operator		
Input		
A	B	Output (A.B)
0	0	0
0	1	0
1	0	0
1	1	1

c. OR Operator

OR operator provides HIGH output when one of the input is HIGH and provides LOW output if all the input is LOW.

Diagram



∴ Function (F) = A + B

Truth table

OR Operator		
Input		Output (A+B)
A	B	
0	0	0
0	1	1
1	0	1
1	1	1

5. Digital Computer

A digital computer can be considered as a digital system that performs various computational tasks. The digital computer use the binary number system which has two digits: 0 and 1. A binary digit is called bit.

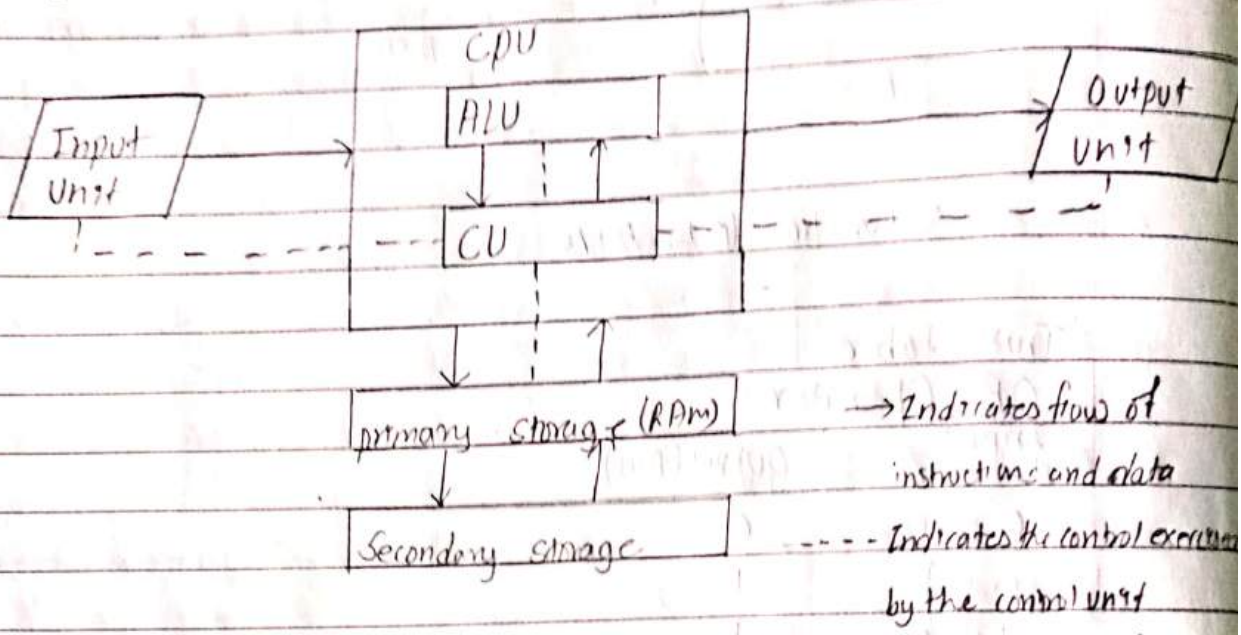
A digital computer system is divided into two functional entities: Hardware and software.

The hardware consists of all the electronic components and electromechanical devices that consist of the physical entity of the device.

The software of the computer consists of the instructions and data that the computer manipulates to perform various data-processing tasks.

Date: .....
Page: .....

Building Block / Anatomy / Architecture / working principle of Digital Computer



- The central processing unit (CPU) contains an arithmetic and logic unit for manipulating data, a number of registers for storing data, and a control unit for fetching and executing instructions.
- The memory unit of a digital computer contains storage for instructions and data.
- The Random Access Memory (RAM) for real-time processing of the data.
- The input-output devices for generating inputs from the user and displaying the final results to the user.
- The Input-output devices connected to the computer include keyboard, mouse, terminal magnetic disk drives, and other communication devices.
- The Secondary Storage stores the data permanently for a long period of time.

6. Integrated Circuits (IC)

Integrated Circuits (IC) is a microchip ~~which~~ on which thousands and hundreds of electrical components, such as registors, capacitors and transistors, are fabricated. An IC functions as an oscillator, amplifier, microprocessor, timer or as computer memory. An electronic circuit is constructed entirely on a single chip of silicon.

Date: .....
Page .....

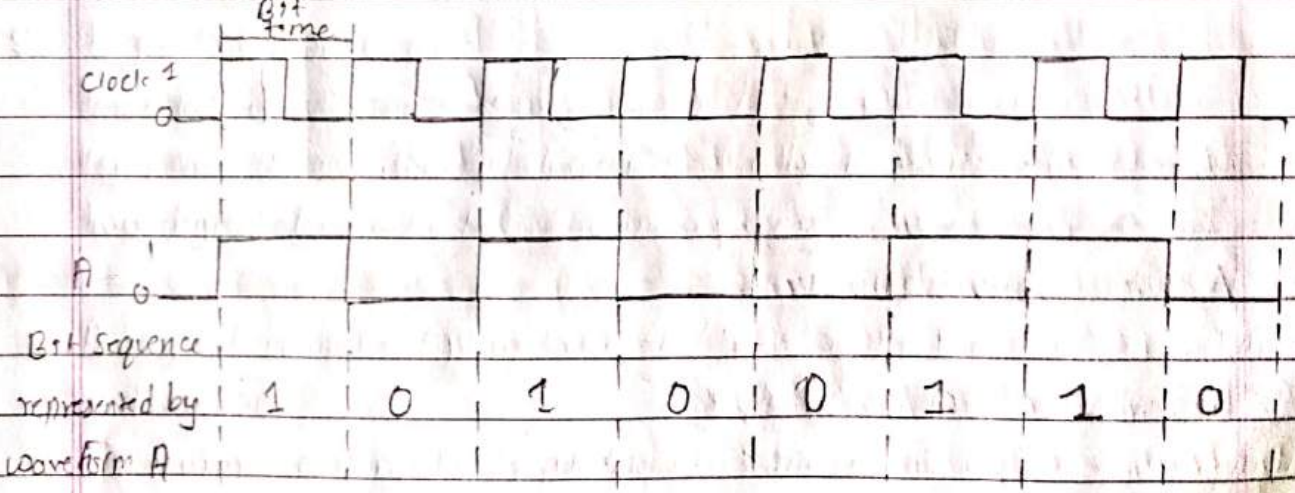
The digital IC can be divided into two:

a- Fixed-function logic :- Logic function have been fixed by the manufacturer. Need to refer specification before using it/ cannot be changed.

b- Programmable Logic :- Logic function can be changed based on the program that we write into the IC.

7. Clock Wave form

- Digital wave form carries binary information.
- Binary information that is handle by digital system appears as wave forms that represent sequence bit
- When the waveform is high binary 1 is present. When the waveform is low 0 is present.
- Each bit in a sequence occupies a define time interval called a bit time.
- In digital system, all wave forms are synchronized with a basic timing waveform called the clock.



Example of a clock waveform
Synchronized with a waveform
representation of a sequence of bits

Date: .....
Page: .....

Unit:-2 Number Systems

1. Number Systems

Number systems defines a set of value used to represent the quantity. There are four types of number system. They are:-

a. Binary Number System

A number system that have base or radix of two 0 is called binary number system. It has only two digits i.e. 0 and 1. Every number is represented by 0 and 1 in binary number system.

b. Decimal Number System

A number system that have base or radix ten and contains 10 digits is called decimal number system. It has 0 to 9 numbers.

c. Octal Number System

A number system that have base of eight and contains 8 digits is called Octal number system. It has 0 to 7 numbers.

d. Hexadecimal Number System

A number system which have radix of sixteen and contains 16 numeric and alphabetic character. It contains 0 to 9 numbers and A to F alphabetic character where A = 10, B = 11, C = 12, D = 13, E = 14 and F = 15.

Example:-

Binary Number :- $(11010101)_2$, $(10010011)_2$

Decimal Number :- $(9864)_{10}$, $(890107)_{10}$

Octal Number :- $(1674)_8$, $(40365)_8$

Hexadecimal Number :- $(10CF9)_{16}$, $(ACF9658)_{16}$

Number System

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

2. Number System Conversion

a. Decimal to other Number System

i. $(64)_{10} = (?)_2$

2 | 64 -- 0
2 | 32 -- 0
2 | 16 -- 0
2 | 8 -- 0
2 | 4 -- 0
2 | 2 -- 0
1

$\therefore (64)_{10} = (1000000)_2$

ii) $(53.625)_{10} = (?)_2$

$(53)_{10}$

2 | 53 — 1
2 | 26 — 0
2 | 13 — 1
2 | 6 — 0
2 | 3 — 1
1

$(0.625)_{10}$

$0.625 \times 2 = 1.25 = 1$ ↓
$0.25 \times 2 = 0.5 = 0$
$0.5 \times 2 = 1.00 = 1$
$0.00 \times 2 = 0.00 = 0$

$(53.625)_{10} = (110101.1010)_2$

iii. $(0.456)_{10} = (?)_2$

$(0.456)_{10}$

$0.456 \times 2 = 0.912 = 0$
$0.912 \times 2 = 1.824 = 1$
$0.824 \times 2 = 1.648 = 1$
$0.648 \times 2 = 1.296 = 1$
$0.296 \times 2 = 0.592 = 0$
$0.592 \times 2 = 1.184 = 1$
$\therefore (0.456)_{10} = (0.011101)_2$

iv. $(64)_{10} = (?)_8$

8 | 64 — 0
8 | 8 — 0
1

$\therefore (64)_{10} = (100)_8$

v. $(0.456)_{10} = (?)_8$

$(0.456)_{10}$

$0.456 \times 8 = 3.648 = 3$
$0.648 \times 8 = 5.184 = 5$
$0.184 \times 8 = 1.472 = 1$
$0.472 \times 8 = 3.776 = 3$
$0.776 \times 8 = 6.208 = 6$
$\therefore (0.456)_{10} = (0.35136)_8$

Date: .....
Page: .....

vii. $(444.456)_{10} = (?)_8$

$(444)_{10}$

8 | 444 — — 4
8 | 55 — — 7
6

$(0.456)_{10}$

$0.456 \times 8 = 3.648 = 3$
$0.648 \times 8 = 5.184 = 5$
$0.184 \times 8 = 1.472 = 1$
$0.472 \times 8 = 3.776 = 3$
$0.776 \times 8 = 6.208 = 6$

$\therefore (444.456)_{10} = (674.35136)_8$

viii. $(12345)_{10} = (?)_{16}$

$(12345)_{10}$

16 | 12345 — — 9
16 | 771 — — 3
16 | 48 — — 0
3

$\therefore (12345)_{10} = (3039)_{16}$

ix. $(0.625)_{10} = (?)_{16}$

$(0.625)_{10}$

$0.625 \times 16 = 10.00 = 10 = A$
$0.00 \times 16 = 0.00 = 0$
$\therefore (0.625)_{10} = (0.A0)_{16}$

x. $(444.625)_{10} = (?)_{16}$

$(444)_{10}$

16 | 444 — — 12
16 | 27 — — 11

$(0.625)_{10}$

$0.625 \times 16 = 10.00 = 10 = A$
$0.00 \times 16 = 0.00 = 0$

$\therefore (444.625)_{10} = (1BC.A)_{16}$

Date :
Page :

b. Other number System to Decimal

i. $(10101.1011)_2 = (?)_{10}$

$= 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$

$= 16 + 0 + 4 + 0 + 1 + 0.5 + 0 + 0.125 + 0.0625$

$= (21.6875)_{10}$

ii. $(10111.1101)_2 = (?)_{10}$

$= 1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 1\times2^{-4}$

$= 32 + 0 + 8 + 4 + 2 + 1 + 0.5 + 0.25 + 0 + 0.0625$

$= (47.8125)_{10}$

iii. $(200.71)_8 = (?)_{10}$

$= 2\times8^2 + 0\times8^1 + 0\times8^0 + 7\times8^{-1} + 1\times8^{-2}$

$= 128 + 0 + 0 + 0.875 + 0.015625$

$= (128.890625)_{10}$

iv. $(444.625)_8 = (?)_{10}$

$= 4\times8^2 + 4\times8^1 + 4\times8^0 + 6\times8^{-1} + 2\times8^{-2} + 5\times8^{-3}$

$= 256 + 32 + 4 + 0.75 + 0.03125 + 0.009765625$

$= (292.790625)_{10}$

v. $(2D5)_{16} = (?)_{10}$

$= 2\times16^2 + D\times16^1 + 5\times16^0$

$= 2\times256 + 13\times16 + 5\times1$

$= 512 + 208 + 5$

$= (725)_{10}$

vi. $(1BC.A)_{16} = (?)_{10}$

$= 1\times16^2 + B\times16^1 + C\times16^0 + A\times16^{-1}$

$= 1\times256 + 11\times16 + 12\times1 + 10\times1/16$

$= 256 + 176 + 12 + 0.625$

$= (444.625)_{10}$

c. Binary to Octal and Hexa Decimal Number System

i. $(100\ 101\ 111\ 011\ 011\ 0)_2 = (?)_8$

Since each octal digit corresponds to three binary bits, grouping three bit each from right to left we get

001 001 011 101 101 110

1 1 3 5 5 6

$(113556)_8$

$\therefore (1001011110110110)_2 = (113556)_8$

ii. $(111\ 010\ 110\ 001\ 01 . 100\ 101)_2 = (?)_8$

Since each octal digit corresponds to three binary bits, grouping three bit each from right to left before point and from left to right after decimal point, we get

011 101 011 000 101 . 100 101 010

3 5 3 0 5 . 4 5 2

$(35305.452)_8$

$\therefore (111010110001.10010101)_2 = (35305.452)_8$

iii. $(1101011000101.100101010101) = (?)_{16}$

Since each Hexa digit corresponds to four binary bits, grouping four bit from Right to left from point and left to right after point. we get,

0011 1010 1100 0101 . 1001 0101 0100

3 10 12 5 . 9 5 4

3 A C 5 . 9 5 4

$(3AC5.954)_{16}$

$\therefore (1101011000101.100101010)_2 = (3AC5.954)_{16}$

d. Octal and Hexa Decimal to Binary Number System.

i. $(325.25)_8 = (?)_2$

Converting each octal digit into group of three binary bits.

3 2 5 . 2 5

111 010 101 . 010 101

Combining the group

$(111010101.010101)_2$

$\therefore (325.25)_8 = (111010101.010101)_2$

ii. $(AB2C.B)_{16} = (?)_2$

Converting each Hexa digit into group of four binary bits

A B 2 C . B

1010 1011 0010 1100 1011

Combining the group

$(1010101100101100.1011)_2$

$\therefore (AB2C.B)_{16} = (1010101100101100.1011)_2$

e. Octal to Hexa decimal

i. $(5672.25)_8 = (?)_{16}$

Converting each octal digit into group of three binary bit

5 6 7 2 . 2 5

101 110 111 010 . 010 101

Combining the group

$(101110111010.010101)_2$

Since each Hexa digit corresponds to four binary bit, grouping four bits each from right to left before point and left to right after point, we get:

1011 1011 1010 . 0101 0100

B B A . 5 4

$(BBA.54)_{16}$

$\therefore (5672.25)_8 = (BBA.54)_{16}$

Date
Page

4. Hexa decimal to octal

i. $(7B5E.A29)_{16} = (?)_8$

Converting each hexa digit into group of four binary bit

7	B	5	E	.	A	2	9
0111	1011	0101	1110	.	1010	0010	1001

combining the group

$(0111101101011110.1010001010011)_2$

Since each octal digit corresponds to three binary bits, grouping three bits each from right to left before point and left to right after point, we get :-

111	100	110	101	110	.	101	000	101	001
7	4	6	5	6	.	5	0	5	1

$(74656.5051)_8$

$\therefore (7BE.A29)_{16} = (74656.5051)_8$

3. Binary Arithmetic Operation

a. Addition

0+0 = 0

0+1 = 1

1+0 = 1

1+1 = 10 (write carry 1)

i. 101 + 101

```
  1 1
  101
+ 101
-----
 1010
```

ii. 1111 + 1111

```
  1111
+ 1111
------
 11110
```

iii. 111 + 111 + 111

```
   1 1
   111
 + 111
 + 111
-------
 10101
```

b. Subtraction

0 - 0 = 0

1 - 0 = 1

1 - 1 = 0

0 0 - 1 = 1 (with Borrow 1)

i.
$$\begin{array}{r} 111 \\ -101 \\ \hline 010 \end{array}$$

ii. $1111 - 1011$

$$\begin{array}{r} 1111 \\ -1011 \\ \hline 0100 \end{array}$$

c. Multiplication

0 × 0 = 0

0 × 1 = 0

1 × 0 = 0

1 × 1 = 1

i. 11×11

$$\begin{array}{r} 11 \\ \times 11 \\ \hline 11 \\ 11\times \\ \hline 1001 \end{array}$$

ii. 1011×1111

$$\begin{array}{r} 1011 \\ \times 1111 \\ \hline 1011 \\ 1011\times \\ 1011\times\times \\ 1011\times\times\times \\ \hline 10100101 \end{array}$$

d. Division

i. $110 \div 11$

$$\begin{array}{r|l} 11\overline{)110} & 10 \\ -11 & \\ \hline 00 & \\ -00 & \\ \hline \times\times & \end{array}$$

$\therefore 110 \div 11 = 10$

ii. $111\overline{)111110}\ (10010$

$$\begin{array}{r} -111 \\ \hline 000110 \\ -111 \\ \hline 000 \end{array}$$

$\therefore 111110 \div 111 = 10010$

Date: .....
Page: .....

4. Complements

Complements are used in digital computers for simplifying subtraction operation. It is the process of repeated addition.

There are two types of complements. They are:-

a. r's complements :- 10's complement and 9's complement
b. (r-1)'s complements :- 9's complement and 1's complement

a. 10's complement

It is calculated by $r^n - N$. example:-

10's complement of 123

r = radix/base = 10, n = 3, N = 123

$10^3 - 123 = 1000 - 123 = 877$.

i. $(57520)_{10}$

$= 10^5 - 57520 = 100000 - 57520 = 42480$

ii. $(0.3267)_{10}$

$= 10^0 - 0.3267 = 1 - 0.3267 = 0.6733$

iii. $(25.639)_{10}$

$= 10^2 - 25.639 = 100 - 25.639 = 74.361$

b. 9's Complement.

It is calculated by subtracting each digit in the number from 9. Example:-

9's complement of 123

$999 - 123 = 876$

i. 19

$$\begin{array}{r} 99 \\ -19 \\ \hline 80 \end{array}$$

Date: .....
Page: .....

ii. 469

999
-469
530

iii. 4397

9999
-4397
5602

c. 1's complement

1's complement is calculated by replacing each digit by 1 to 0 and 0 to 1.

Example :-

1's complement of 100101 -

100101 = 011010

i. 101101

101101 = 010010

ii. 11111001

11111001 = 00000110

d. 2's complement

2's complement is calculated by adding 1 to the least significant bit of the 1's complement.

Example

2's complement of 101101

1's complement = 010010
+ 1
2's complement 010011

i. 11111011

11111011 = 00000100 +1 = 00000101

ii. 101100

101100 = 010011 + 1 = 010100

Subtraction Using r's complement

Steps

Step 1: Add the minuend to the r's complement of the subtrahend.

Step 2: Inspect the result obtained in step 1 for an end around carry.

(i) If an end around carry occurs, discard it

(ii) If an end around carry doesn't occurs, take the r's complement of the number obtained in step 1 and place the negative sign infront.

Subtraction Using (r-1)'s complement

Steps

Step 1: Add the minuend to the (r-1)'s complement of the subtrahend.

Step 2: Inspect the result obtained in step 1 for an end around carry

(i) If an end carry occurs, Add 1 to the least significant digit.

(ii) If an end carry doesn't occurs, take (r-1)'s complement of the number obtained in step 1 and place negative sign infront.

Subtraction using 10's complement method

a. 72532 - 3250

Minuend = 72532

Subtrahend = 03250 (∵ Minuend digit = Subtrahend digit)

Taking the 10's complement of subtrahend 03250

100000 - 3250 = 96750

Adding the minuend with 10's complement of subtrahend

$$\begin{array}{r} 72532 \\ +\ 96750 \\ \hline \text{carry bit} \rightarrow\ 69282 \end{array}$$

∴ The result have carry. So, discarding the carry, we get 69282

∴ The required Answer is 69282.

Date:
Page:

b. 72532 from 3250

Minuend = 03250

Subtrahend = 72532

Taking the 10's complement of subtrahend 72532

100000 − 72532 = 27468

Adding the minuend with 10's complement of subtrahend

```
  03250
+ 27468
-------
  30718   ← No carry Bit
```

The result have no carry. So, Taking the 10's complement of the result and placing the negative sign infront

100000 − 30718 = −69282

∴ The required answer is (−69282).

Subtraction Using 9's Complement Method

a. 72532 − 3250

Minuend (M) = 72532

Subtrahend (N) = 03250

Taking the 9's complement of subtrahend 03250 is

```
  99999
- 03250
-------
  96749
```

Adding the minuend with 9's complement of subtrahend

```
  72532
+ 96749
-------
1 69281   ← carry bit
      +1  ← Adding carry bit
-------
  69282
```

∴ The result have carry bit, Adding the carry to the LSB. We get 69282.

∴ The required Answer is 69282

b- 72532 from 03250

Minuend = 72532

Subtrahend = 03250

Taking the 9's complement of subtrahend 72532

$$\begin{array}{r} 99999 \\ -\ 72532 \\ \hline 27467 \end{array}$$

Adding the minuend with 9's complement of subtrahend

$$\begin{array}{r} 03250 \\ +\ 27467 \\ \hline 30717 \end{array} \leftarrow \text{No carry}$$

∴ The result have no carry So, Taking the 9's complement of the result and placing the negative sign

$$\begin{array}{r} 99999 \\ -\ 30717 \\ \hline -\ 69282 \end{array}$$

∴ The required answer is (-69282)

Subtraction using 2's complement method

a- $(1101)_2$ from $(1111)_2$

Minuend = 1111

Subtrahend = 1101

Taking the 2's complement of the subtrahend 1101 is

$0010 + 1 = 0011$

Adding the minuend with 2's complement of subtrahend

$$\begin{array}{r} 1111 \\ +\ 0011 \\ \hline 10010 \end{array}$$

carry bit → 1

∴ The result have carry. So, Discarding the carry. we get $(0010)_2$

∴ The required answer is $(10)_2$

Date: .....
Page: .....

b. $(1110)_2$ from $(1011)_2$

Minuend = 1011

Subtrahend = 1110

Taking the 2's complement of subtrahend 1110 is

0001 + 1 = 0010

Adding the minuend with 2's complement of subtrahend

$$\begin{array}{r} 1011 \\ +\ 0010 \\ \hline 1101 \end{array}$$ ← No carry

∴ The result have no carry. So, taking the 2's complement of the result and placing the negative sign:

$$1101 = \begin{array}{r} 0010 \\ +\ 1 \\ \hline -0011 \end{array}$$

∴ The required Answer is $(-11)_2$.

Subtraction using 1's complement

a. $(100)_2$ from (1010)

Minuend = 1010

Subtrahend = 1001

Taking the 1's complement of subtrahend 1001 is 0110

Adding the minuend with 1's complement of subtrahend

$$\begin{array}{r} 1010 \\ +\ 0110 \\ \hline 10000 \\ +\ 1 \\ \hline 0001 \end{array}$$

carry bit → ← Adding carry bit

∴ The result have carry. So, Adding the carry with the least significant bit. we get 0001.

∴ The required answer is $(1)_2$

Date: .....
Page: .....

b. $(1111)_2$ from $(1010)_2$

Minuend = 1010

Subtrahend = 1111

Taking the 1's complement of subtrahend 1111 is 0000

Adding the minuend with the 1's complement of subtrahend.

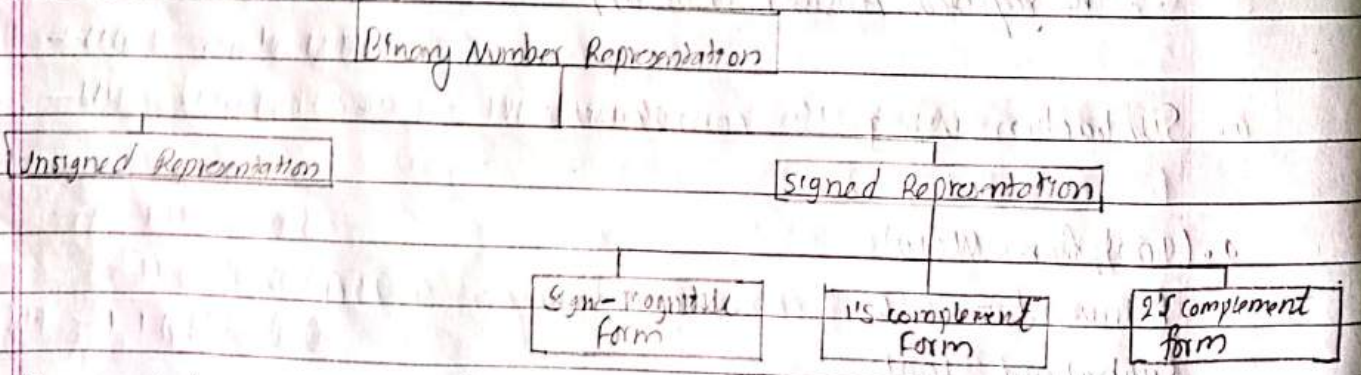
$$\begin{array}{r} 1010 \\ +0000 \\ \hline 1010 \end{array}$$ ← no carry

∴ The result have no carry. So, Taking the 1's complement of the result and placing the negative sign in front

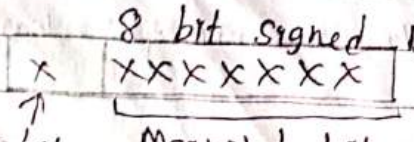
$1010 = -(0101)$

∴ The required answer is (-101).

5. Representation of signed Binary Number



A signed binary consists of both signed and magnitude information, the sign indicated whether the number is positive or negative and the magnitude is the binary value of the number. When a number is represented in signed binary. The left most bit is sign and remaining are magnitude bits. The magnitude bits are in the true binary for both positive and negative number. 0 represent positive and 1 represent negative sign.



Date: .....
Page: .....

Arithmetic operation with signed numbers.

Addition

[When you have to convert negative number in binary form. First convert the number into Binary form and take the 2's complement.]

a. +29 and +19

+29 = 00011101

+19 = 00010010

```
  00011101
  00010010
 ----------
  00110000
```

Sign | magnitude

∴ The required solution is 00110000

b. +32 and -22

+32 = 00100000

+22 = 00010110

Taking the 2's complement of +22, we get -22.

-22 = (11101001 + 1) = 11101010

```
  00100000
  11101010
 ----------
1 00001010
```

carry bit →

∴ Discard the carry bit. We get : 00001010

∴ The required solution is 00001010

Date: .....
Page: .....

c. -47 and +29

+47 = 00101111

-47 = (11010000+1) = 11010001

+29 = 00011101

```
  11010001
  00011101
----------
  11101110
```

∴ The required solution is 11101110

d. -32 and -44

+32 = 00100000

-32 = (11011111 + 1) = 11100000

+44 = 00101100

-44 = (11010011 + 1) = 11010100

```
  11100000
  11010100
----------
1 11011000
```
carry bit →

∴ Discarding the carry bit. We get 11011000

∴ The required solution is 11011000.

M = +19 = 00010

N = +8 = 0001

= (11100 + 1) = 1

M + 2's

00010011
+ 11100100
11110111

Subtraction

[In subtraction, Takes the 2's complement of subtrahend]

a. +28 and +19

M = +28 = 00011100

N = +19 = 00010011

= (11100 + 1) = 11101101 2's complement of subtrahend

00011100
+ 11101101
Discard carry bit → 100001001

∴ Discarding carry bit we get 00001001

∴ The required solution is 00001001

b. +39 and -21

M = +39 = 00100111

N = -21

= 00010101 2's complement of subtrahend

00100111
+ 00010101
00111100

∴ The required Solution is 00111100

Date: .....
Page: .....

c. +19 and -43

M = +19

N = -43

+19 = 00010011

+43 = 00101011 2's complement of subtrahend

```
  00010011
+ 00101011
----------
  00111110
```

∴ The required solution is 00111110

d. -57 and -33

M = -57

N = -33

+57 = 00111001

-57 = 11000111

+33 = 00100001 2's complement of subtrahend

```
  11000111
+ 00100001
----------
  11101000
```

∴ The required solution is 11101000

∴ Note: To subtract two sign numbers, take the 2's complement of subtrahend and add. Discard any final carry bit.

Date :
Page :

6. Floating point Numbers

A floating point Numbers System is based on scientific notation which is capable of representing very large and very small numbers without an increase in the number of bits and also for representing numbers that have both integer and fractional components.

A floating-point number consists of two parts plus a sign i.e. Mantissa and exponent. The mantissa is the part of floating point number that represents the magnitude of the number. The Exponent is the part of a floating point numbers that represent the number of places that the decimal point or binary point is to be moved. Example :-

$$\underbrace{0.24567890}_{\text{Mantissa}} \times 10^{9} \leftarrow \text{Exponent}$$

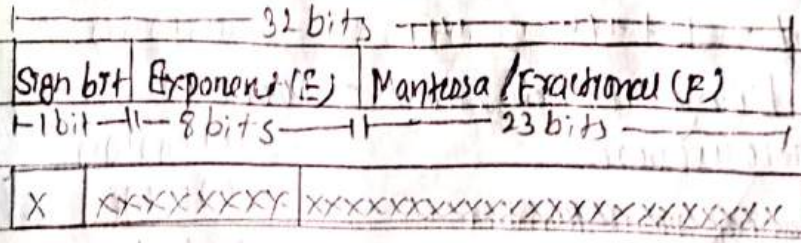
$$\underbrace{1.01110110}_{\text{Mantissa}} \times 10^{8} \leftarrow \text{Exponent}$$

For Binary floating-point Number, the format is defined by ANSI/IEEE (Institute of electrical and electronic engineering) standard 754-1985 in three forms :-

- Single-precision 32 bit
- Double-precision 64 bit
- Extended-precision 80 bit

Single-precision Floating-point Binary numbers.

In the standard format for a single-precision binary number, the sign bit (S) is the left-most bit, the exponent (E) includes the next eight bits and the mantissa or fractional part (F) includes the remaining 23 bits, as shown :-



Date: .....
Page: .....

Representing the numbers in Single-precision floating-point Numbers.

a. 1011010010111

Normalize the given numbers

$1.01101001011 \times 10^{12}$

Assuming that this is a positive number. So,

Sign bit = 0

Exponent (E) = 12

Biased Exponent (BE) = 127 + 12 = 139

Biased Exponent binary representation = 10001011

Mantesa = .01101001011

The compete floating point Number:-

0	10001011	01101001011100000000000

b. 3.248×10^4

$3.248 \times 10^4 = 32480$

Converting 32480 in binary form

32480 = 111111011100000

Normalize the numbers

$1.11111011100000 \times 10^{14}$

Assume that this is positive number. So,

Sigh bit = 0

Exponent (E) = 14

Biased Exponent = 127 + 14 = 141

Biased Exponent binary representation = 10001101

Mantessa = .11111011100000

The complete floating point Number:-

0	10001101	11111011100000000000000

DATE _____ PAGE _____

7. Binary codes

- Weighted code
  - * Binary coded Decimal (BCD)
  - * Binary
- Non-Weighted code
  - * Excess 3
  - * Grey
- Alphanumeric code
  - * ASCII
  - * EBCDIC
- Error detecting/correct code
  - * parity
  - * Hamming

code

Code is a symbolic representation of a discrete information which may be presen in the form of number, letters or the physical quantities. The symbols used by th binary digits are 0 and 1 which are arrange according to the rules of code.

Binary coded Decimal (BCD)

Binary coded Decimal (BCD) is the way to express each of the decimal digi with a binary code.

BCD is the combination of 4 binary digits that represent decimal numbers 0 to 9.

BCD means that each decimal digits 0 to 9 is represented by a code of 4 bi Simply to express any decimal number in BCD, Each decimal digit should be replaced by appropriate 4 bit code.

The 8421 BCD code.

The 8421 code is a type of BCD code. BCD means that each decimal digit, 0 through 9, is represented by a binary code of four bits. The designation 8421 indicates the binary weights of the four bits ($2^3, 2^2, 2^1, 2^0$). The 8421 code is the predominant BCD code, and when we refer to BCD, we always mean the 8421 code unless otherwise stated.

Decimal BCD conversion table

Decimal	Binary	Binary coded Decimal	
0	0000	0000	Valid BCD code
1	0001	0001	
2	0010	0010	
3	0011	0011	
4	0100	0100	
5	0101	0101	
6	0110	0110	
7	0111	0111	
8	1000	1000	
9	1001	1001	
10	1010	1010	Invalid BCD code (Don't care)
11	1011	1011	
12	1100	1100	
13	1101	1101	
14	1110	1110	
15	1111	1111	

Conversion of the decimal number to Binary coded Decimal.

a. 37

Since BCD is 8421 code. So replace each digit with 4 bit binary code.

3 7

0011 0111

00110111

∴ BCD code of 37 = 00110111

b. 83

Since BCD is 8421 code. So replace each digit with 4 bit binary code.

8 3

1000 0011

10000011

∴ BCD code of 83 = 10000011

c. 98

Since BCD is 8421 code. So replace each digit with 4 bit binary code

9 8

1001 1000

10011000

∴ BCD code of 98 = 10011000

d. 2479

Since BCD is 8421 code. so replace each digit with 4 bit binary code.

2 4 7 9

0010 0100 0111 1001

0010010001111001

∴ BCD code of 2479 = 0010010001111001

Conversion of the Binary coded Decimal to Decimal Number.

a. 100100110101

BCD is a 4 bit binary code. So grouping 4 bit each from left to right

1001 0011 0101

9 3 5

∴ 935

∴ (100100110101) to decimal is 935.

b. 1010000100100100

BCD is a 4 bit binary code. So grouping 4 bit each from left to right

0010 1000 1001 0100

2 8 9 4

2894

∴ 1010000100100100 decimal is 2894.

Binary Coded Decimal Addition

Rules of adding two BCD numbers

Step 1: Add the two BCD numbers, using the rules for binary addition.

Step 2: If a 4-bit sum is equal to or less than 9, it is a valid BCD number.

Step 3: If a 4-bit sum is greater than 9, or if a carry out of the 4-bit group is generated, it is a invalid result. Add 6 (0110) to the 4-bit sum in order to skip the six invalid state and return the code to 8421. If a carry results when 6 is added, simply add the carry to the next 4-bit group.

a. 0101 + 0100

$$\begin{array}{r} 0101 \\ +0100 \\ \hline 1001 \end{array}$$

Valid BCD because sum is not greater than 9.

b. 01110110 + 00010010

$$\begin{array}{r} 0111\ \ 0110 \\ +\ 0001\ \ 0010 \\ \hline 1000\ \ 1000 \end{array}$$

Valid BCD because sum are not greater than 9.

c. 3 + 4

3 = 0011

4 = 0100

$$\begin{array}{r} 0011 \\ +0100 \\ \hline 0111 \end{array}$$

Valid BCD because sum is not greater than 9.

d. 61 + 37

6 = 0110 3 = 0011

1 = 0001 7 = 0111

0010 0001

+ 0011 0111

0101 1000 Valid BCD because both are less than or equal to nine.

e. 1001 + 0101

1001

+ 0101

1110 Invalid BCD because the sum is greater than 9.

+ 0110 Add 6 (0110) to make valid BCD

10100

0001 0100 Valid BCD

f. 00010110 + 00010101

0001 0110

+ 0001 0101

0010 1011 Invalid BCD because right group is greater than 9.

+ 0110 Add 6 (0110) to invalid group.

0011 0001

0011 0001 Valid BCD.

g. 68 + 79

6 = 0110 7 = 0111

8 = 1000 9 = 1001

0110 1000

+ 0111 1001

1101 10001 Invalid BCD because right group consist carry bit and left group is greater than 9.

+ 0110 0110 Add 6 (0110) to both group

10100 0111

0001 0100 0111 Valid BCD.

DATE

Non-Weighted code

Excess-3 code (XS-3 code)

Excess-3 code is also known as XS3 which is non-weighted code used to express binary code in a number greater than 3.

An Excess-3 code is obtained by adding 3 to a decimal number and convert it into binary form.

The key features of excess-3 code is that it is self complementing code.

The 1's complement of an excess-3 numbers is the excess 3 code for 9's complement of the corresponding decimal number.

Decimal and Excess-3 conversion Table.

Decimal number	BCD code	Excess-3 code (BCD + 0011)
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Converting of the numbers to Binary Excess-3 code.

$(6)_{10}$

6

+3 Added by 3

9

Convert the above sum into binary code

9

1001

∴ The excess-3 code for 9 is 1001.

b. $(25)_{10}$

2 5

+3 +3 Add by 3

5 8

Converting the above sum into Binary code

0101 1000

∴ The excess-3 code for 25 is 01011000

c. $(643)_{10}$

6 4 3

+3 +3 +3 Add by 3

9 7 6

Converting above sum into Binary code.

1001 0111 0110

∴ The excess-3 code for 643 is 100101110110.

d. $(14.53)_{10}$

1 4 . 5 3

+3 +3 +3 +3 Add by 3

4 7 . 8 6

Converting above sum into Binary code

0100 0111 . 1000 0110

∴ The excess-3 code for 14.53 is 01000111.10000110

e. $(34.57)_{10}$

3 4 . 5 7

+3 +3 +3 +3 Add by 3

6 7 . 8 10

Converting above sum into Binary code.

0110 0111 . 1000 1010

∴ The excess-3 code for 34.57 is 01100111.10001010

f. $(10010111)_2$

Grouping the 4-bits from left to right

1001	0111	
+ 0011	+ 0011	Add by 0011
1100	1010	

∴ The excess-3 code for 10010111 is 11001010.

g. $(1000100100011.00100011)_2$

Grouping the 4-bits from left to right before point and right to right after point

1000	1001	0011 .	0010	0011	
+ 0011	+ 0011	+ 0011 .	+ 0011	+ 0011	Add by 0011
1011	1100	0110 .	0101	0110	

∴ The excess-3 code for (1000100100011.00100011) is 10111100011 0.01010110

b. Grey code

Grey code is an unweighted code which have no specific weights assigned to the bit positions.

The Grey code belongs to a class of codes called minimum change codes in wh only one bits in the code group changes when moving from one stage to next.

The Two successive values differs in only one bit.

Binary Numbers are converted into grey code in order to reduce switching operations.

Grey code is also known as unweighted code, Minimum error code, Unit distan code and Reflected Binary code.

The Grey code can have any number of bits.

Four-Bit Grey code Table

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

Binary to Grey Conversion

Step 1: The most significant bit (left-most) in the Grey code is the same as the corresponding MSB in the binary number.

Step 2: Going from left to right, add each adjacent pair of binary code bits to get the next Grey code bit. Discard carries.

Step 3: Repeat the process

Example:-

1011 0

MSB

1 0 1 1 0 → Binary number

1 1 1 0 1 → Grey code

DATE _____ PAGE _____

Convert the following Binary number to Grey code.

i. 1011

1 0 0 1
↓ ↘ ↓ ↘ ↓ ↘ ↓
1 1 1 0

∴ Grey code = 1110

ii. 1111

1 1 1 1
↓ ↘ ↓ ↘ ↓ ↘ ↓
1 0 0 0

∴ Grey code = 1000

iii. 1110

1 1 1 0
↓ ↘ ↓ ↘ ↓ ↘ ↓
1 0 0 1

∴ Grey code = 1001

iv. 11011

1 1 0 1 1
↓ ↘ ↓ ↘ ↓ ↘ ↓ ↘ ↓
1 0 1 1 0

∴ Grey code = 10110

v. 11011011

1 1 0 1 1 0 0 1
↓ ↘ ↓ ↘ ↓ ↘ ↓ ↘ ↓ ↘ ↓ ↘ ↓ ↘ ↓
1 0 1 1 0 1 0 1

∴ Grey code = 10110101

Grey to Binary conversion

Step 1 :- The most significant bit (left-most) in the binary code is the same as the corresponding bit in the Grey code.

Step 2 :- Add each binary code bit generated to the Grey code bit in the next adjacent position. Discard carries.

Step 3 :- Repeat process

Example :-

11101 → Grey code

1 1 1 0 1
↓ ↗ ↓ ↗ ↓ ↗ ↓ ↗ ↓
1 0 1 1 0 → Binary Number

DATE _____ PAGE _____

Convert the following Gray code to Binary number.

i. 1000

1 0 0 0

↓ ↗ ↗ ↗

1 1 1 1

∴ Binary Number = 1111

iv. 10101111

1 0 1 0 1 1 1 1

↓ ↗ ↗ ↗ ↗ ↗ ↗ ↗

1 1 1 1 0 1 0 0

∴ Binary Number = 11110100

ii. 111010

1 1 1 0 1 0

↓ ↗ ↗ ↗ ↗ ↗

1 0 1 1 0 1

∴ Binary Number = 101101

v. 111001100

1 1 1 0 0 1 1 0 0

↓ ↗ ↗ ↗ ↗ ↗ ↗ ↗ ↗

1 0 1 1 1 1 0 1 1

∴ Binary Number = 101111011

iii. 111011

1 1 1 0 1 1

↓ ↗ ↗ ↗ ↗ ↗

1 0 0 1 0 1

∴ Binary Number = 100101

9. Error Detecting and Correcting Codes

a. Parity Bit

If a single error transforms of a valid code word into an invalid code, it is said to be a single error detecting code.

Parity bit is simply a error detecting extra bit for 1 bit error.

The most simple and commonly use error detecting method is parity check in which an extra parity bit is included with the binary message to make the total number of 1's either even or odd resulting in two methods.

Parity bit is an extra bit added to the orginal message used to detect error.

There are two types of parity bit. They are:

i. Even parity
If the total number of 1 in original data is even, parity bit will be 0 else the parity bit will be 1.

ii. Odd parity
If the total number of 1 in original data is odd, parity bit will be 0 else the parity bit will be 1.

When digital signals (group of 0's and 1's) are transmitted from one circuit or system to another circuit, error may occur. The change of one to zero or zero to one during transmission is known as error. So, it is necessary to detect and correct error to obtain the correct message.

Parity Method for Error Detection
This is a concept to detect error.
Single bit error is detected by this method.
parity is an additional bit added to the original data at the transmission before transmitting the data.
Before adding the parity bit, number of the 1's or 0's is calculated in the data. Based on this calculation extra (parity) bit is defined.
The circuit which adds a parity bit to the data and then transmit is called parity generator and the parity bits are transmitted and are checked at the receiver.
If the parity bits sent from the transmitter and the parity bit received from the receiver are not equal, then the error is detected.
The circuit which checks the parity at the receiver is called parity checker.

4-bits Data

A	B	C	D	Data	Even parity	odd parity
0	0	0	0	0000	0	1
0	0	0	1	0001	1	0
0	0	1	0	0010	1	0
0	0	1	1	0011	0	1
0	1	0	0	0100	1	0
0	1	0	1	0101	0	1
0	1	1	0	0110	0	1
0	1	1	1	0111	1	0
1	0	0	0	1000	1	0
1	0	0	1	1001	0	1
1	0	1	0	1010	0	1
1	0	1	1	1011	1	0
1	1	0	0	1100	0	1
1	1	0	1	1101	1	0
1	1	1	0	1110	1	0
1	1	1	1	1111	0	1

b. Hamming code

Hamming code is given by RW hamming (scientist) for error detection and correction.

7 bit hamming code is used commonly. In 7 bit hamming code, we are having 4 data bits and 3 parity bits. Then, we have to decide the position of data bits and parity bits in seven bit hamming code.

Following the hamming rules for the parity bits, we can get the position of parity bits as shown below.

2^n

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$

} parity position

$2^3 = 8$ → Discarded "We need 7 bits only"

D_7	D_6	D_5	P_4	D_3	P_2	P_1

Where, P_1 is related with D_3, D_5, D_7
P_2 is related with D_3, D_6, D_7
P_4 is related with D_5, D_6, D_7

Example:-

i. If the 7-bit hamming codeword received by a receiver is 1011011. Assuming the even parity checker method state whether the related codeword is correct or wrong. If wrong, locate the bit having an error and correct it.

1	0	1	1	0	1	1
D_7	D_6	D_5	P_4	D_3	P_2	P_1

First check for P_4

$P_4 \Rightarrow D_5\ D_6\ D_7\ P_4$ ∴ The total number of 1's is odd since error
1 0 1 1 occurs because of even parity method.

∴ So, $P_4 = 1$

Second check of P_2

$P_2 \Rightarrow D_3\ D_6\ D_7\ P_2$ ∴ The total number of 1's is even ∴ Since no error
0 0 1 1 because of even parity method.

∴ So, $P_2 = 0$

Third check of P_1

$P_1 \Rightarrow D_3 \cdot D_5\ D_7\ P_1$ ∴ The total number of 1's is odd. Since error occurs
0 1 1 1 because of even parity method.

∴ So, $P_1 = 1$

Again

$P_4 P_2 P_1 \Rightarrow (101)_2 = 5_{10}$

∴ This means that 5th position of bit have an error.

∴ Since the original message is 1001011

1	0	0	1	0	1	1

DATE _____ PAGE _____

9. If received 7-bit message is 1111011. Assuming even parity, state the data is correct or not. If yes correct the error.

1	1	1	1	0	1	1
D_7	D_6	D_5	P_4	D_3	P_2	P_1

First check for P_4

$P_4 \Rightarrow D_5\ D_6\ D_7\ P_4$

1 1 1 1 $\therefore$ No error $\therefore P_4 = 0$

$P_2 \Rightarrow D_3\ D_5\ D_6\ P_2$

0 1 1 1 $\therefore$ Error occurs $\therefore P_2 = 1$

$P_1 \Rightarrow D_3\ D_5\ D_7\ P_1$

0 1 1 1 $\therefore$ Error occurs $\therefore P_1 = 1$

Again,

$P_4 P_2 P_1 \Rightarrow (011)_2 \Rightarrow 3_{10}$

$\therefore$ This means that 3rd position of bit have an error.

$\therefore$ So, the original message is 1111111

1	1	1	1	1	1	1

Unit-3 Combinational Logic Design DATE _____ PAGE _____

1. Logic Gates

Logic gates are the basic building blocks of any digital system. It is an electron circuit having one or more than one input and only one output.

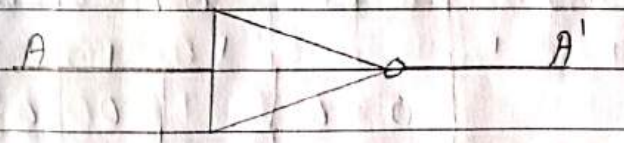
Types of gates

a. NOT Gate
b. AND Gate } Basic Gates
c. OR Gate
d. NAND Gate } Universal Gates
e. NOR Gate
f. XOR (ExOR) Gate } Extended/Hybrid Gates
g. X-NOR (Ex-OR) Gate

a. NOT Gate

When the input is LOW, the output is HIGH; When the input is HIGH, the output is LOW. This is a NOT Gate. It produce an inverted output pulse, so it is also called inverter.

Diagram:-



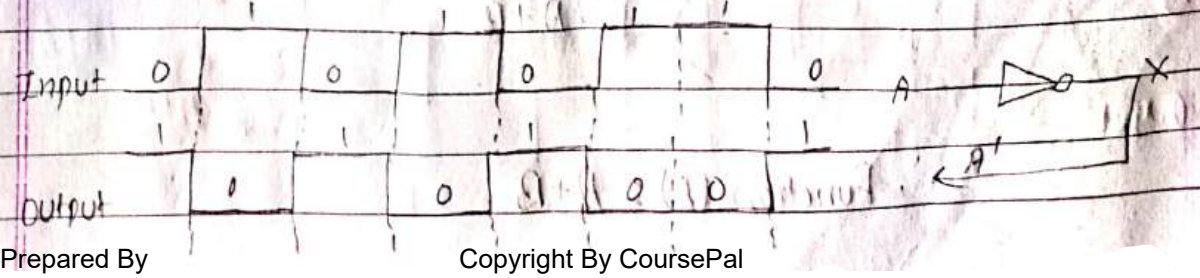
∴ NOT Gate

∴ Function (F) = $\bar{A}$ / A'

Truth Table

Input(A)	Output(A')
0	1
1	0

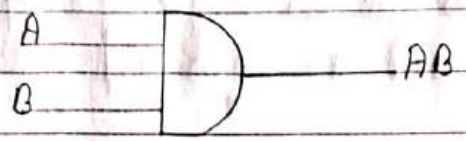
Wave form of NOT Gate Operation



b. AND Gate

When all the input is HIGH, the Output is HIGH; When one of the input is LOW, the output is LOW. This kind of gate is called AND gate.

Diagram :-

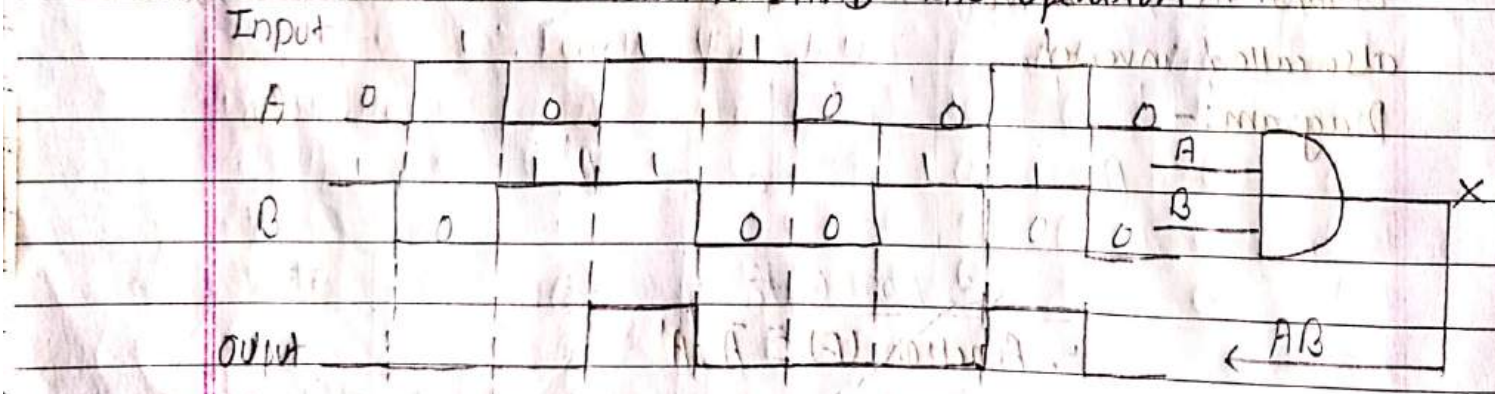


∴ Function (F) = AB

Truth Table

Input A	Input B	Output (AB)
0	0	0
0	1	0
1	0	0
1	1	1

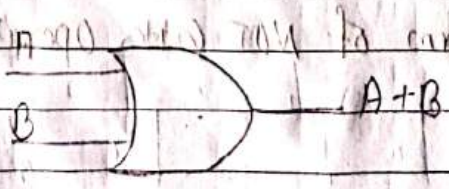
Wave form for AND Gate Operation



c. OR Gate

When one of the input is HIGH, the output is HIGH; When all the input is LOW, the Output is LOW. This kind of logic gate is known as OR Gate

Diagram :-

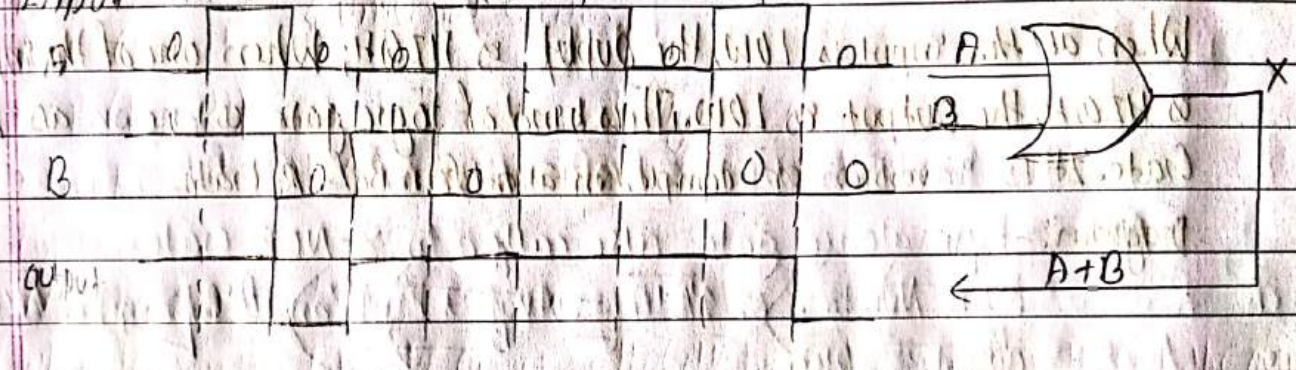


∴ Function (F) = A+B

Truth Table

Input A	Input B	Output (A+B)
0	0	0
0	1	1
1	0	1
1	1	1

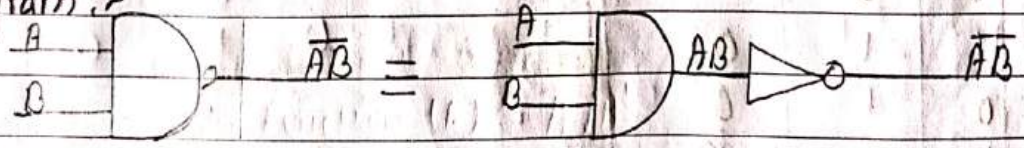
Waveform for OR Gate Operation



d. NAND Gate

When one of the input is LOW, the output is HIGH; When all the input is HIGH, the output is Low. It is known as NAND Gate. It is also known as inversion of AND Gate.

Diagram :-



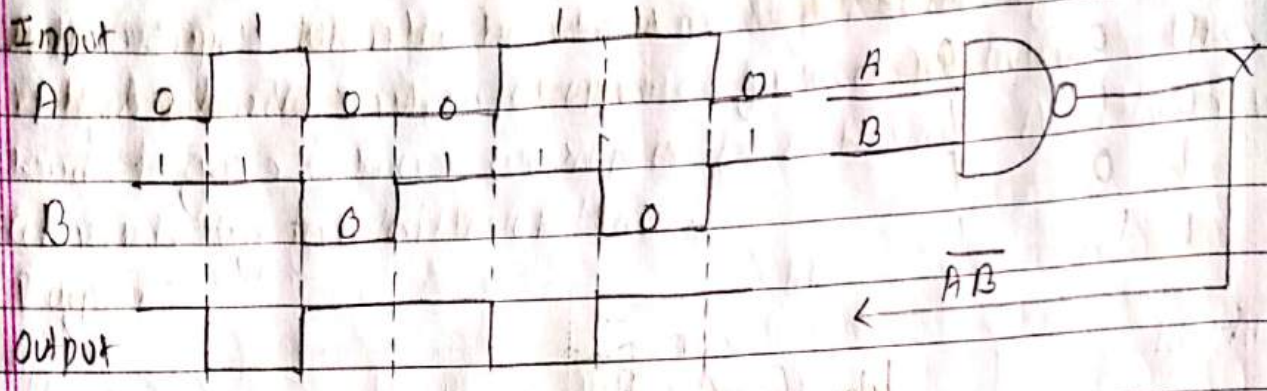
$\therefore$ Function (F) = $\overline{AB}$

Truth Table

Input A	Input B	AB	OUTPUT $\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

DATE

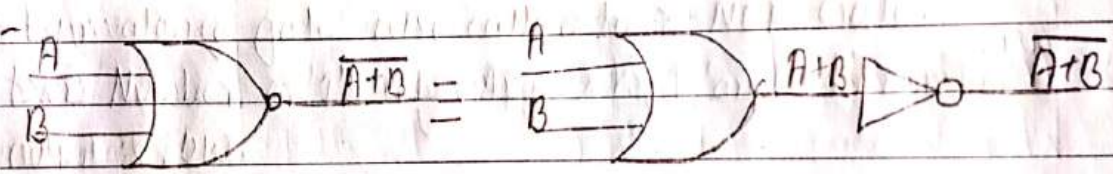
Waveform for NAND Gate Operation



e. NOR Gate

When all the input is LOW, the output is HIGH; When one of the input is is HIGH, the output is LOW. This kind of logic gate is known as NOR Gate. It is inversion or complementation form of OR Gate.

Diagram:-

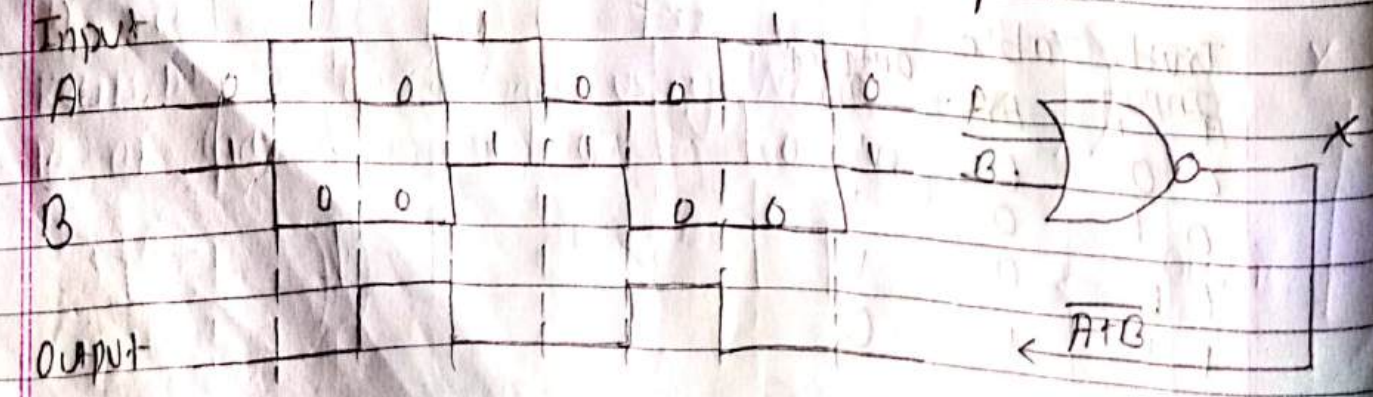


Function (F) = $\overline{A+B}$

Truth Table

Input A	Input B	A+B	Output $\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Waveform for NOR Gate Operation



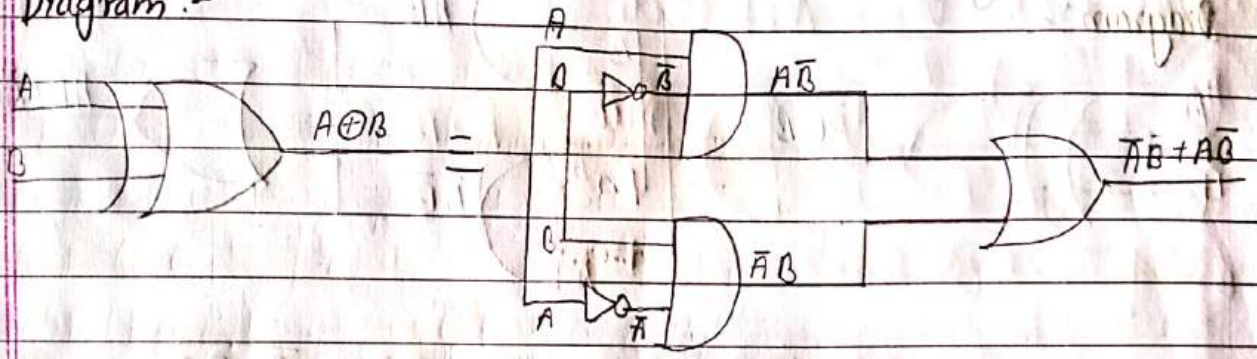
DATE _____ PAGE _____

f. XOR (Ex-OR) Gates

When In two inputs, when the inputs are at opposite logic levels, the output is HIGH; when the both inputs are same, the output is LOW. This kind of gates is known as X-OR / Ex-OR / Exclusive gate.

Note:- If I/s is odd output is 1, if I/s is even output is 0).

Diagram :-



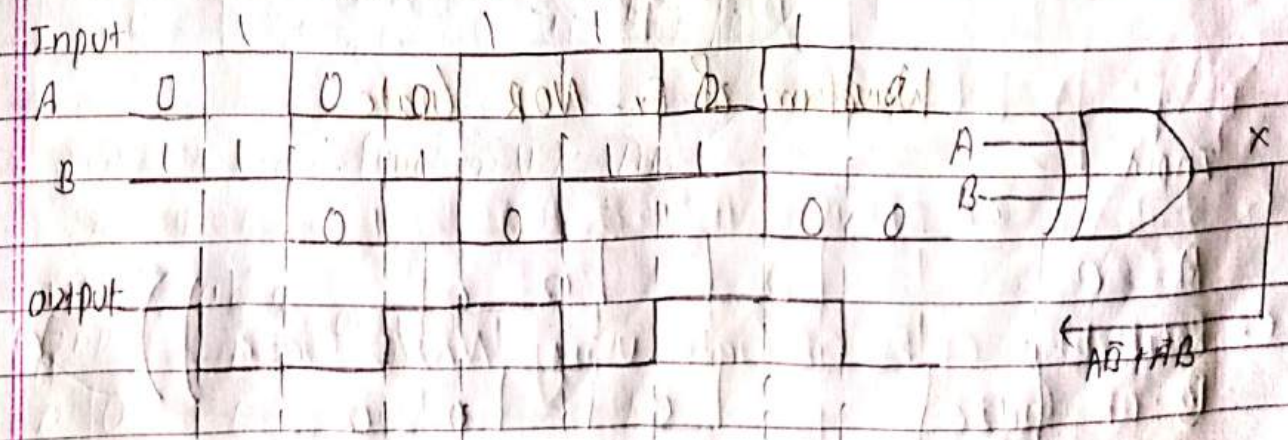
∴ Function (F) = $A \oplus B = A\bar{B} + \bar{A}B$

Truth Table

Input A	Input B	Output $A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

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

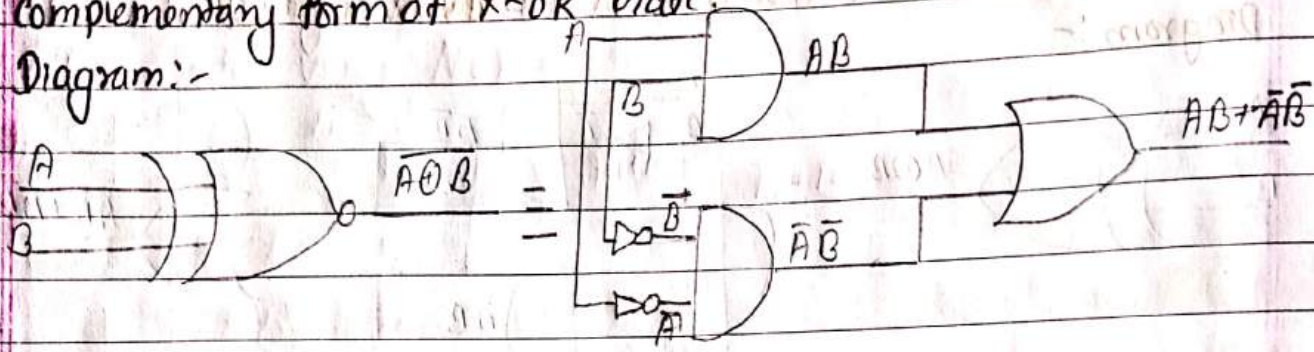
Waveform for X-OR Gate



Q. X-NOR (Ex-NOR) Gate

In two inputs, when the both input are High, same logic level, the output is High. When the input are at opposite logic level, the output is LOW. This kind of gates is known as XNOR or Ex-NOR Gate. It is the reverse or complementary form of x-OR Gate.

Diagram:-



$\therefore$ Function (F) = $\overline{A \oplus B} = A \odot B = AB + \bar{A}\bar{B} = \overline{A\bar{B} + \bar{A}B}$

$\therefore$ Note:- Equivalance gate also called to x-NOR Gate.

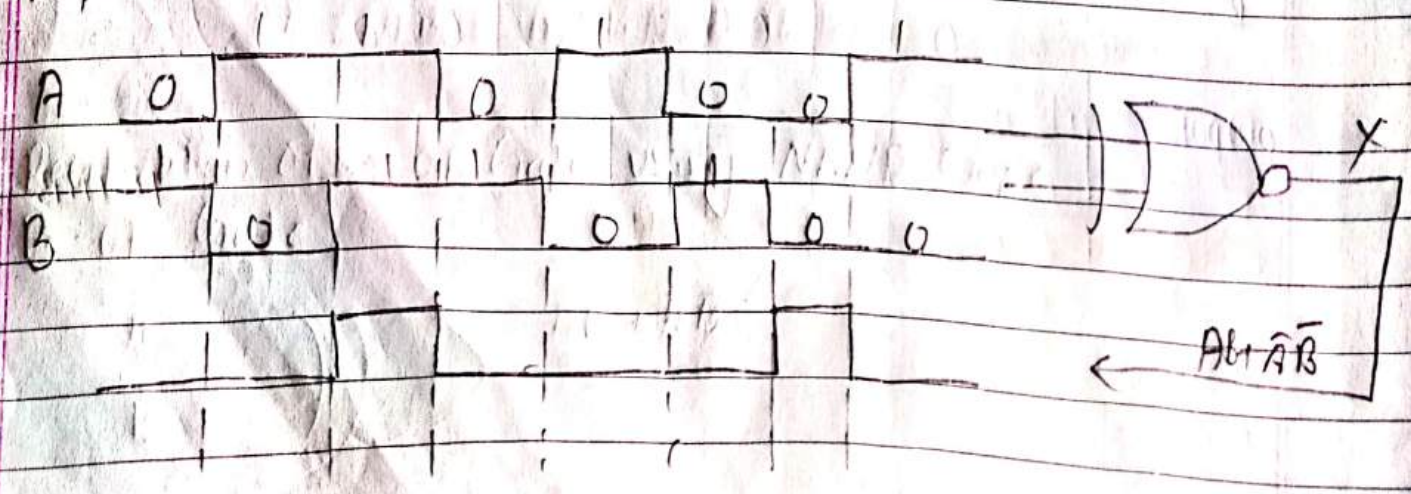
Even Number of 1's output is high and odd number of 1's, output is LOW.

Truth Table

Input A	Input B	Output $A \oplus B$
0	0	1
0	1	0
1	0	0
1	1	1

Input A	Input B	$\bar{A}$	$\bar{B}$	$A\bar{B}$	$\bar{A}\bar{B}$	Output $AB+\bar{A}\bar{B}$
0	0	1	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	1

Waveform of Ex-NOR Gate



DATE _____ PAGE _____

2. Universal Gates

A universal gates is a gates which can be implement any other Boolean function without need to use any other gate type. NAND and NOR Gates are the universal gates. NAND and NOR are called universal Gates because all other type of gates can be realized using these gates.

Somes rules:-

i. $\overline{\overline{A}} = A$

ii. $A+A = A$ and $A \cdot A = A$

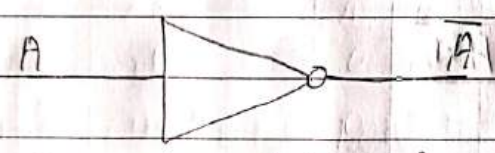
iii. $\overline{A+B} = \overline{A} \cdot \overline{B}$

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

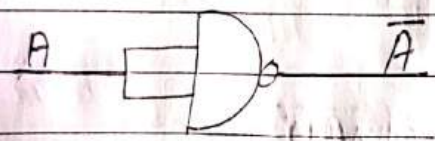
Realization of other Gates using NAND Gate

a. Realization of NOT Gate using NAND Gate.

NOT Gate



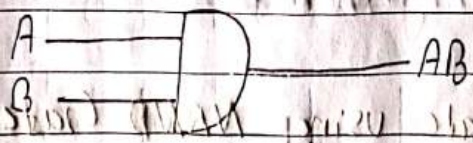
NOT Gate using NAND Gate



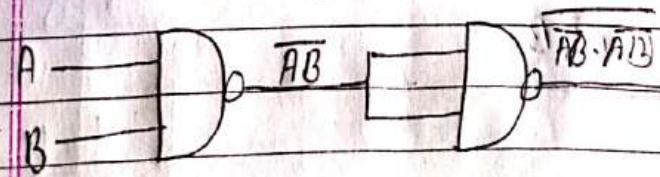
$\therefore A \cdot A = A$

b. Realization of AND Gate using NAND Gate

AND Gate



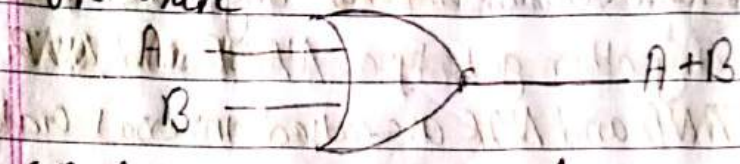
AND Gate using NAND Gate



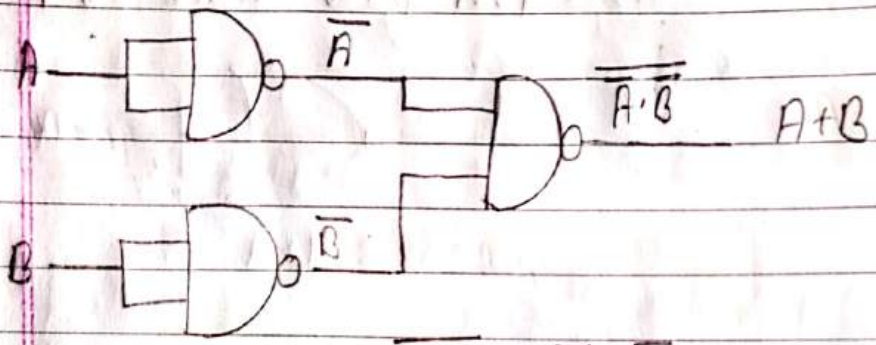
$= \overline{\overline{A}} = A$

c. Realization of OR Gate using NAND Gate

OR Gate



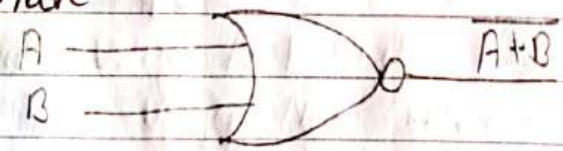
OR Gate using NAND Gate



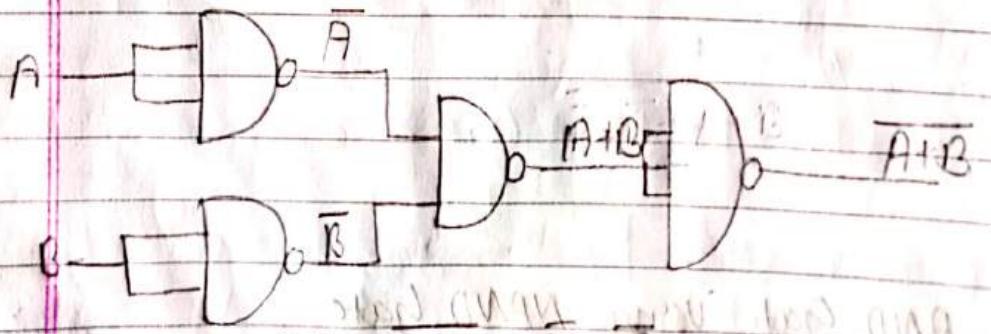
$\therefore \overline{A}\,\overline{B} = \overline{\overline{A}\,\overline{B}} = \overline{\overline{A} + \overline{B}} = A \cdot B$

d. Realization of NOR Gate using NAND Gate

NOR Gate



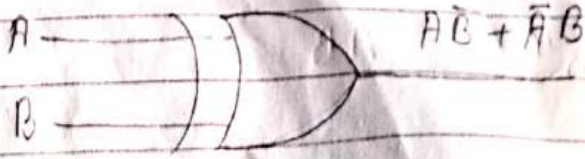
NOR Gate using NAND Gate



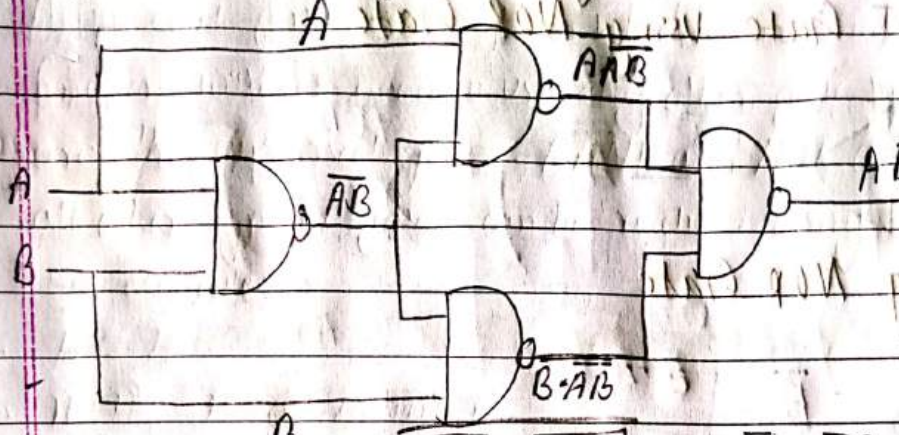
$\therefore \overline{A}\,\overline{B} = \overline{\overline{A}\,\overline{B}} = \overline{\overline{A}} + \overline{\overline{B}} = A + B$

e. Realization of Ex-OR Gate using NAND Gate

Ex-OR Gate



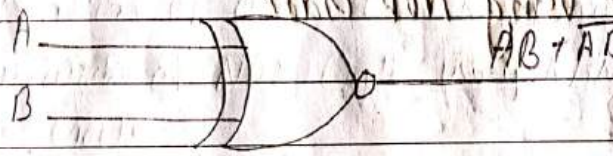
Ex-OR Gate using NAND Gate



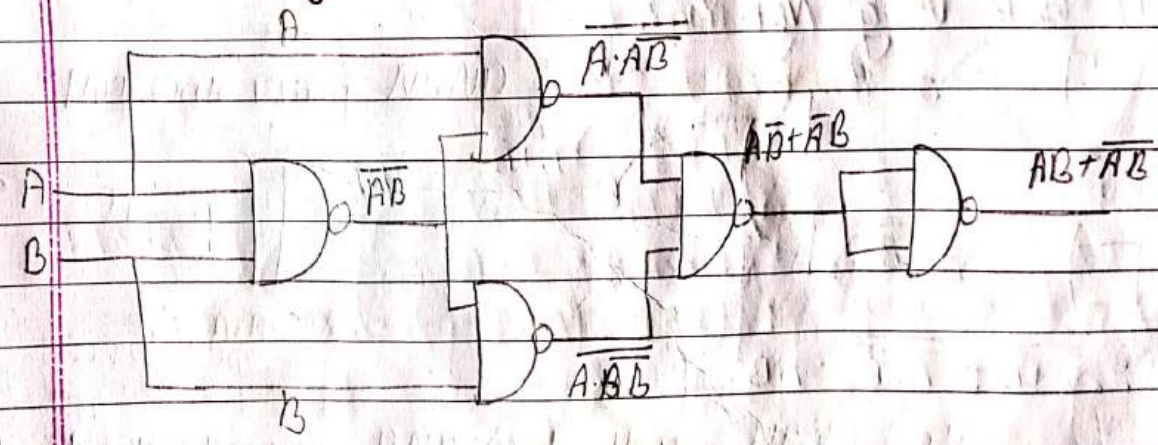
$$\therefore \overline{A\cdot\overline{AB}}\ \overline{B\cdot\overline{AB}} = A\overline{B}+\overline{A}B$$

5. Realization of Ex-NOR Gate using NAND Gate

Ex-NOR Gate



Ex-NOR using NAND Gate



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

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

$$\overline{A}+\overline{\overline{B}} \quad \overline{\overline{A}}+\overline{B}$$

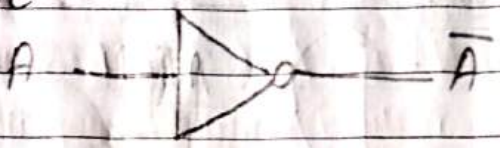
$$\overline{A\overline{B}+\overline{A}B} \qquad A\overline{B}+\overline{A}B$$

DATE

Realization of Other Gates using NOR Gate

a. Realization of NOT Gate using NOR Gate

NOT Gate

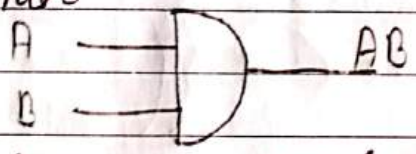


NOT Gate using NOR Gate

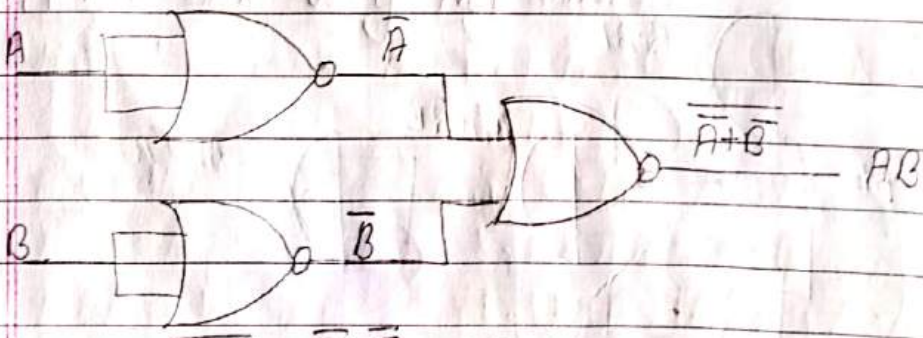
$\therefore A+A = A$

b. Realization of AND Gate using NOR Gate

AND Gate



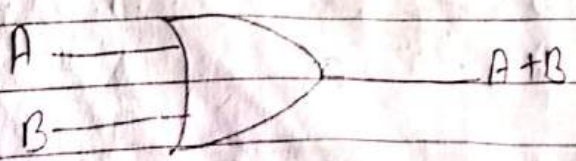
AND Gate using NOR Gate



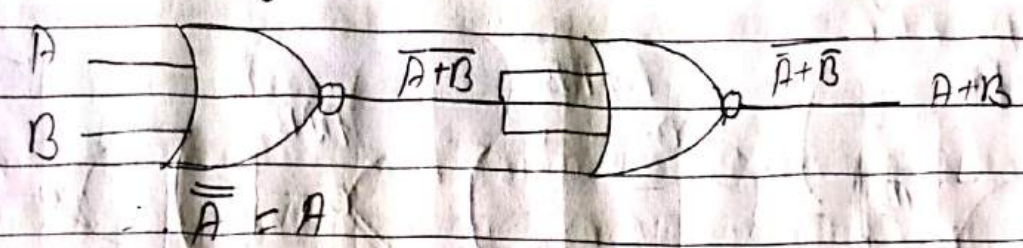
$\therefore \overline{\bar{A}+\bar{B}} = \bar{\bar{A}}.\bar{\bar{B}} = AB$

c. Realization of OR Gate using NOR Gate

OR Gate



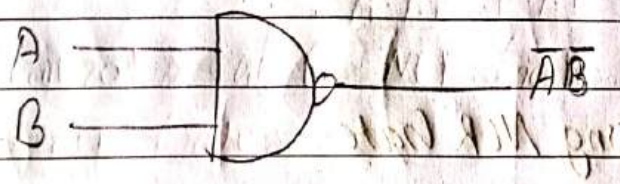
OR Gate using NOR Gate



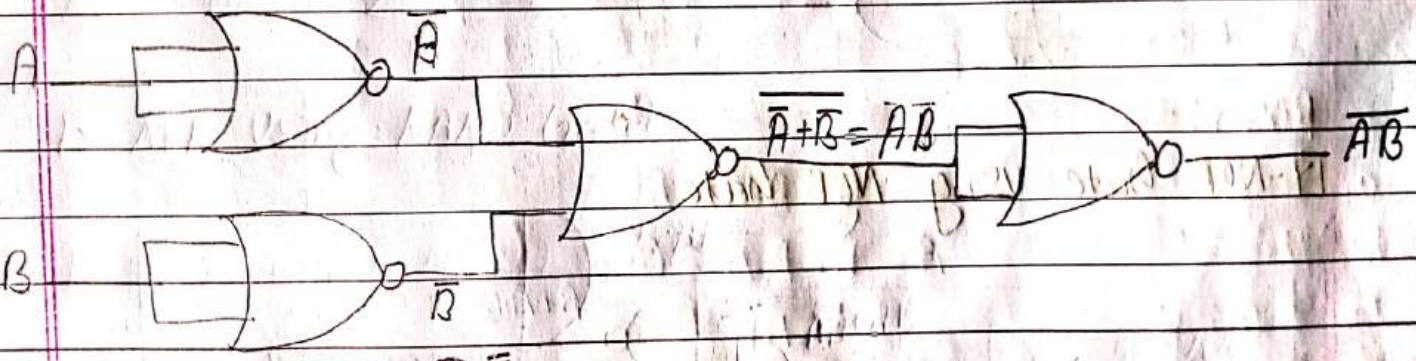
$\therefore \overline{\overline{A}} = A$

d. Realization of NAND Gate using NOR Gate

NAND Gate



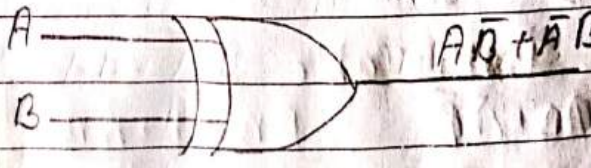
NAND Gate using NOR Gate



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

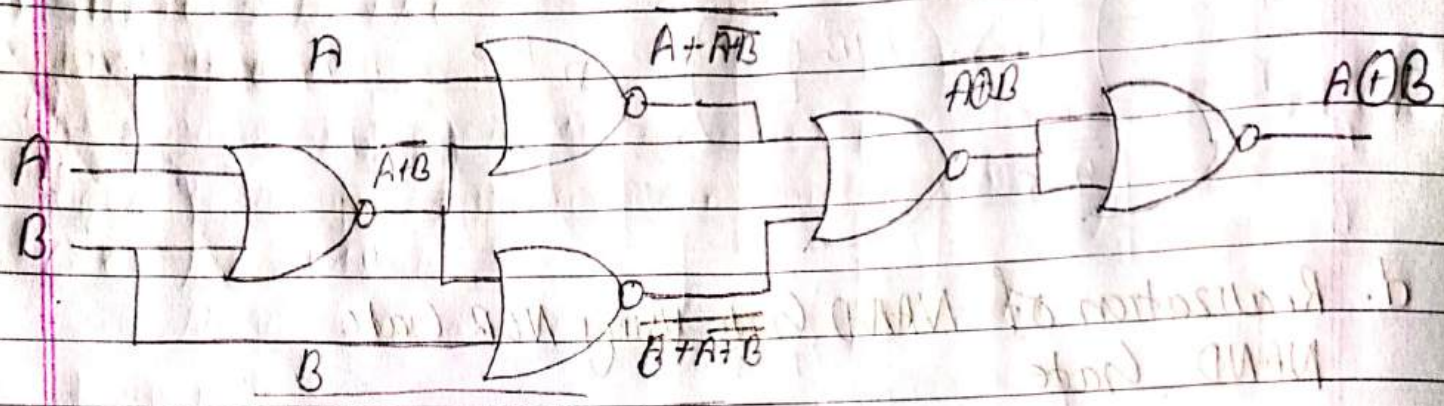
e. Realization of Ex-OR Gate using NOR Gate

Ex-OR Gate



DATE

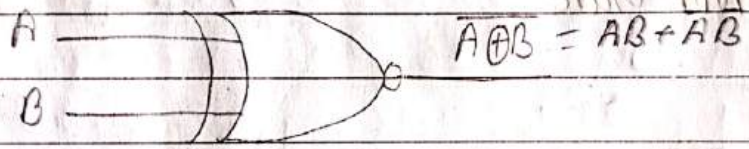
Ex-OR Gate using NOR Gate



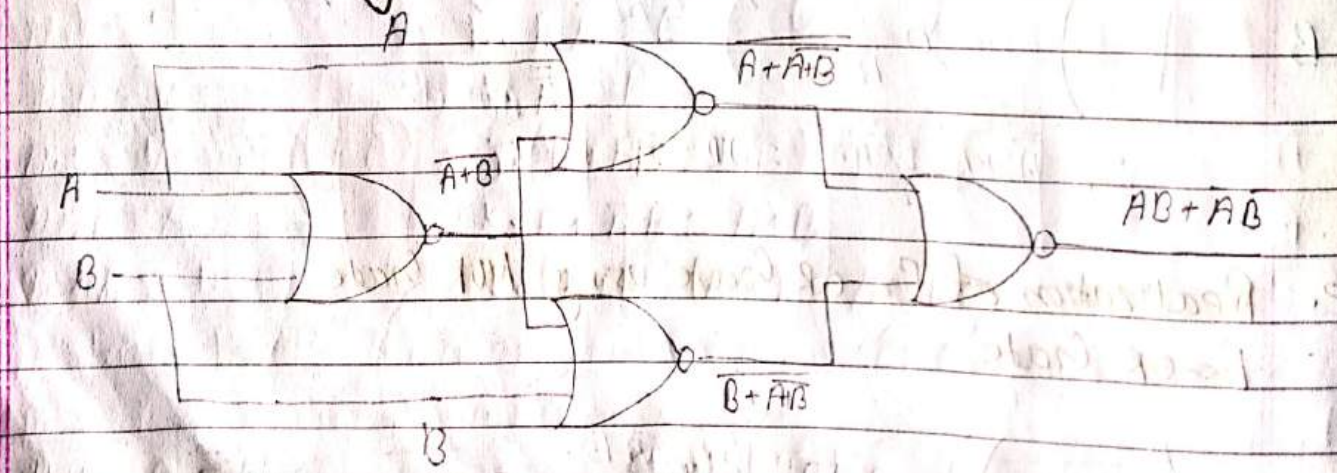
$\therefore \overline{A+\overline{A}+B} + \overline{B+A+B} = A \oplus B$

f. Realization of Ex-NOR Gate using NOR Gate

Ex-NOR Gate



Ex-NOR Gate using NOR Gate



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

3. Boolean Algebra and Logic Simplification

Boolean Algebra is the mathematics of digital logic.
It is a set of rules used to simplify the given logic expression without changing its functionality.
It helps to reduce a number of gates in the circuit design.

Examples:- $A'B + BC + ABC$

$= A'B + BC(1+A)$ $(\because 1+A=1)$

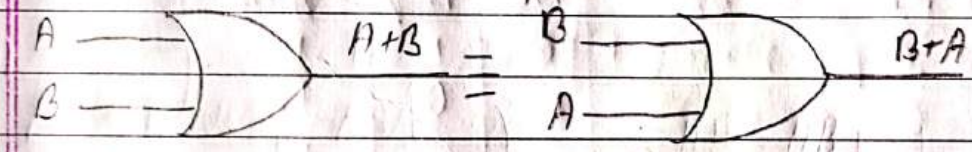
$= A'B + BC$

Laws and rules of Boolean Algebra

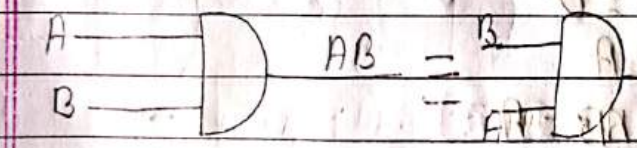
Laws:-

a. Commutative Laws

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

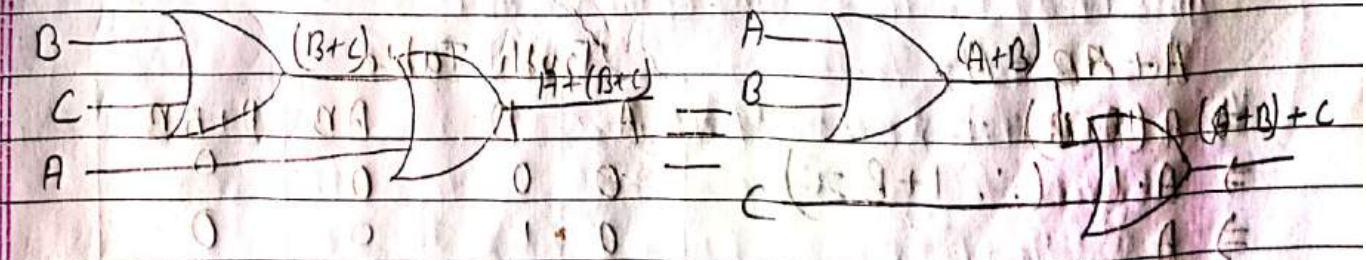


(ii) $AB = BA$

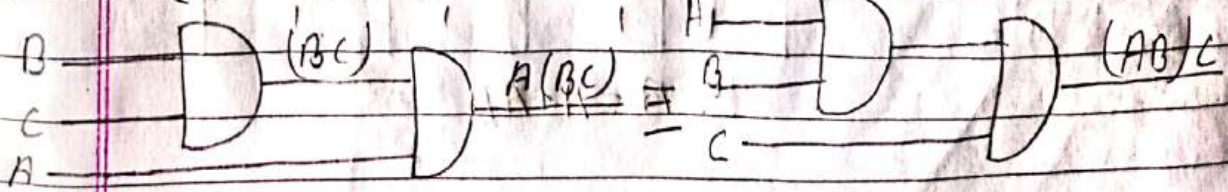


b. Associative Laws

(i) $A+(B+C) = (A+B)+C$

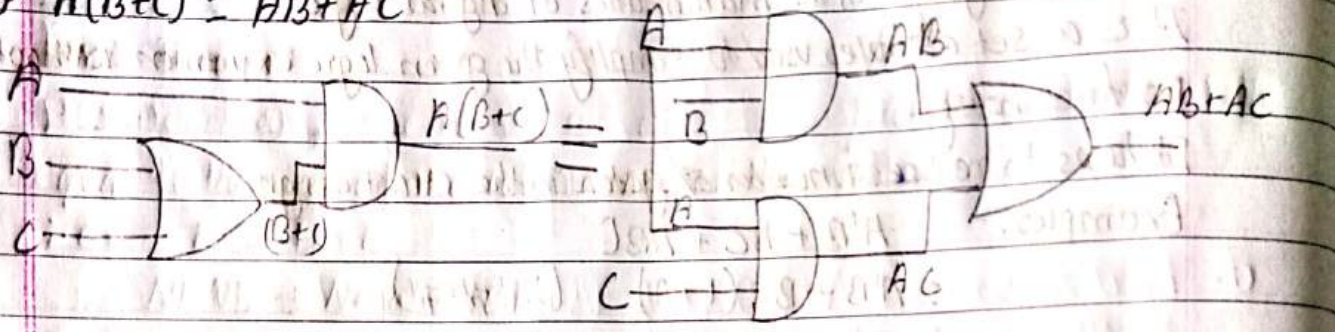


(ii) $A(BC) = (AB)C$

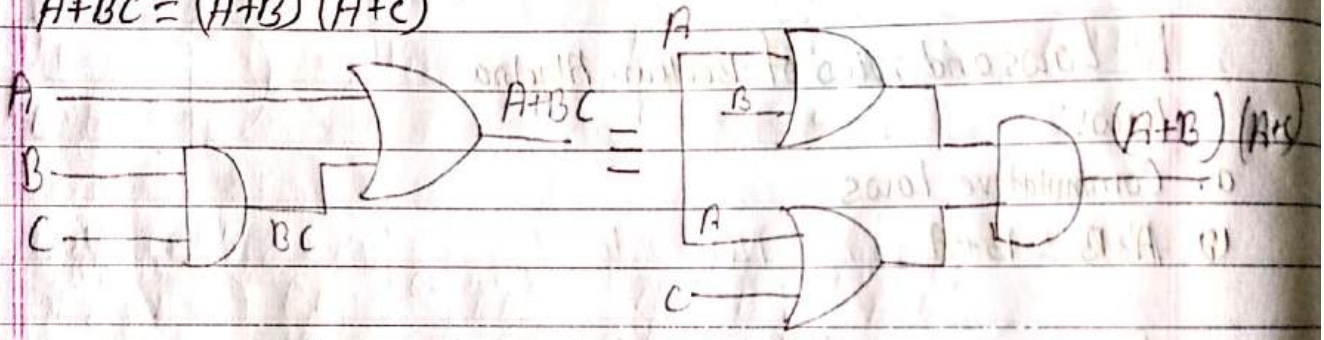


c. Distributive Laws

(i) $A(B+C) = AB+AC$



(ii) $A+BC = (A+B)(A+C)$



Rules :-

a. $A+0 = A$
b. $A+1 = 1$
c. $A \cdot 0 = 0$
d. $A \cdot 1 = A$
e. $A+A = A$
f. $A+\bar{A} = 1$
g. $A \cdot A = A$
h. $A \cdot \bar{A} = 0$
i. $\bar{\bar{A}} = A$
j. $A+AB = A$
k. $A+\bar{A}B = A+B$
l. $(A+B)(A+C) = A+BC$

$A+AB = A$

$\Rightarrow A(1+B)$

$\Rightarrow A \cdot 1 \quad (\because 1+B=1)$

$\Rightarrow A$

Truth Table

A	B	AB	A+AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

$\therefore A = A+AB$

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

$\Rightarrow (A+\bar{A})(A+B) \quad [\because A+BC = (A+B)(A+C)]$

$\Rightarrow 1 \cdot (A+B) \quad [\because A+\bar{A} = 1]$

$\Rightarrow A+B$

Truth Table

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

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

$(A+B)(A+C) = A+BC$

$\Rightarrow AA+AC+AB+BC$

$\Rightarrow A+AC+AB+BC \quad [\because A \cdot A = A]$

$\Rightarrow A(1+C)+AB+BC$

$= A+AB+BC \quad [\because 1+A = 1]$

$= A(1+B)+BC$

$= A+BC \quad [\because 1+A = 1]$

Truth Table

A	B	C	A+B	A+C	(A+B)(B+C)	BC	A+BC
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

$\therefore (A+B)(B+C) = A+BC$

De Morgan's Theorem

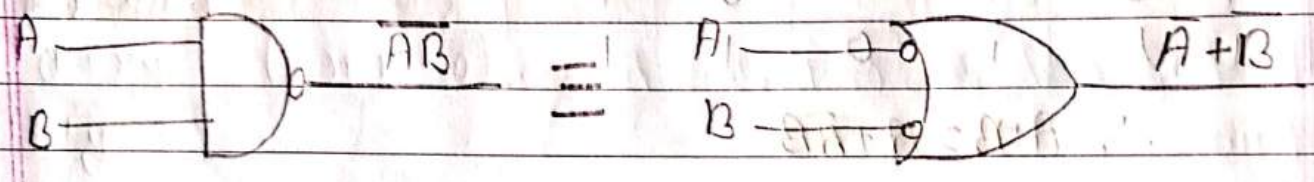
First Theorem

It states that, "The complement of a sum equal product of variables is equal to the sum of the complements of the variables."

i.e. $\overline{AB} = \overline{A} + \overline{B}$

NAND = Negative-OR

Diagram:-



Truth Table

Inputs A	Inputs B	AB	Output 1 $\overline{AB}$	$\overline{A}$	$\overline{B}$	Output 2 $\overline{A}+\overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

∴ Output 1 $(\overline{AB})$ = Output 2 $(\overline{A}+\overline{B})$

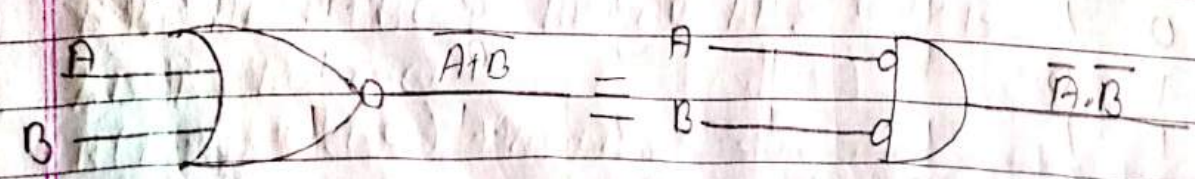
Second Theorem

It states that, "The complement of a sum of a variables is equal to the product of the complements of the variables."

i.e. $\overline{A+B} = \overline{A} \cdot \overline{B}$

NOR = Negative AND

Diagram:-



Truth Table

Inputs A	B	A+B	Output 1 $\overline{A+B}$	$\overline{A}$	$\overline{B}$	Output 2 $\overline{A}\cdot\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

$\therefore$ Output 1 $(\overline{A+B})$ = Output 2 $(\overline{A}\cdot\overline{B})$

Simplify the equation using De Morgan's Theorem to the equations:

i. $\overline{ABC}$

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

ii. $\overline{A+B+C+D}$

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

iii. $\overline{x+y+z}$

$= \overline{x}\cdot\overline{y}\cdot\overline{z}$

$= xyz$

iv. $\overline{(AB+C)}\cdot(A+BC)$

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

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

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

v. $\overline{\overline{A+B\overline{C}}+D(\overline{E+F})}$

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

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

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

Simplify the Equation using boolean Algebra.

i. $x + x'y$

$= (x+x')(x+y)$ $[\because A+BC = (A+B)(A+C)]$

$= 1 \cdot (x+y)$ $[\because x+x'=1]$

$= (x+y)$

ii. $x(x'+y)$

$= x \cdot x' + xy$

$= 0 + xy$ $(\because x \cdot x' = 0)$

$= xy$

iii. $x'y'z + x'yz + xy'$

$= x'z(y'+y) + xy'$

$= x'z \cdot 1 + xy'$ $[\because x'+x = 1]$

$= x'z + xy'$

iv. $xy + x'z + yz$

$= xy + x'z + yz \cdot (x+x')$

$= xy + x'z + xyz + x'yz$

$= xy + xyz + x'z + x'yz$

$= xy(1+z) + x'z(1+y)$ $[\because 1+z = 1]$

$= xy \cdot 1 + x'z \cdot 1$

$= xy + x'z$

v. $AB + A(B+C) + B(B+C)$

$= AB + AB + AC + B \cdot B + BC$

$= AB + AC + B + BC$ $[\because AB + AB = AB \text{ & } B \cdot B = B]$

$= AB + B + AC + BC$

$= B(A+1) + AC + BC$

$= B + AC + BC$

$= B(1+C) + AC$

$= B + AC$

vi. $[AB(C+\bar{B}\bar{D})+A\bar{B}]CD$

$= [ABC + AB\cdot(\bar{B}\bar{D}) + A\bar{B}]CD$

$= [ABC + AB\cdot(\bar{B}+\bar{D}) + A\bar{B}]CD$

$= [ABC + AB\cdot\bar{B} + AB\bar{D} + A\bar{B}]CD$

$= [ABC + 0 + AB\bar{D} + A\bar{B}]CD \quad [\because B\cdot\bar{B}=0]$

$= [ABCD\cdot C + ABCD\cdot\bar{D} + A\bar{B}\cdot CD]$

$= [ABCD + 0 + A\bar{B}\cdot CD] \quad [\because C\cdot C = C \ \& \ D\cdot\bar{D}=0]$

$= CD[AB + A\bar{B}]$

$= CD\cdot 1 \quad [\because AB + A\bar{B} = 1]$

$= CD$

vii. $[A\bar{B}(C+BD) + \bar{A}\bar{B}]C$

$= [A\bar{B}C + A\bar{B}\cdot BD + \bar{A}\bar{B}]C$

$= [A\bar{B}C + 0 + \bar{A}\bar{B}]C \quad [\because B\cdot\bar{B}=0]$

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

$= A\bar{B}C + \bar{A}\bar{B}C \quad [\because C\cdot C = C]$

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

$= \bar{B}C\cdot 1 = \bar{B}C \quad [\because A+\bar{A}=1]$

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

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

$= \bar{A}BC + AB\bar{C} + \bar{A}\bar{B}\bar{C} + AC\cdot 1 \quad [\because \bar{B}+B=1]$

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

$= \bar{A}BC + B\bar{C}\cdot 1 + AC \quad [\because A+\bar{A}=1]$

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

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

$= C(B+A) + B\bar{C} \quad [\because A+\bar{A}B = A+B]$

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

DATE _____ PAGE _____

4. Canonical and standard form
- Product term
- Sum term
- SOP
- POS
- SSOP
- SPOS
- Minterm
- Maxterm

Product term
The AND function is referred to as a product. The variables in the product term can be appeared either in complemented or un-complemented form.
Example:- ABC, $\bar{A}BC$, $ABCD$, $\bar{A}B\bar{C}\bar{D}$, $\bar{A}B\bar{C}D$,

Sum term
The OR function is referred to as a sum. The variables in the sum term can appeared either in complemented or un-complemented form. Example: $A+B+C$, $\bar{A}+\bar{B}+\bar{C}+D$, $A+B+C+\bar{D}$, $\bar{A}+\bar{B}+C+D$

SOP (Sum of product)
The logical sum of two or more product term is called SOP expression.
Example:- $AB+BCD$, $B\bar{C}D+\bar{A}\bar{B}$, $AB\bar{C}+\bar{A}C\bar{D}$

POS (Product of Sum)
The logical product of two or more sum term is called POS expression.
Example:- $(A+B+C).(\bar{A}+\bar{B}+\bar{C}),(A+B).(A+\bar{C}).(A+B+C)$

SSOP (Standard Sum of product)
In SSOP, the product variable should be same and equal but may contain complement.
Example:- $ABCD+A\bar{B}CD$, $\bar{A}BCD+AB\bar{C}D$

DATE _____ PAGE _____

Spos (Standard product of Sum)

In spos, the product sum variable should be same but the variable may be complement.

Example:- $(A+B+C+D)\cdot(A+B+\bar{C}+\bar{D})$, $(A+B+C)\cdot(\bar{A}+B+\bar{C})$

Minterm

Each individual term in standard sum of product is called minterm.

Example:- $\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D}$

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

0 1 2

$m_0 + m_1 + m_2$

Maxterm

Each individual term in standard product of sum is called maxterm.

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

0 0 0 0 0 1 0 1 0

0 1 2

$M_0 \cdot M_1 \cdot M_2$

Here:-

In SOP $\begin{bmatrix} 0 & - & \bar{A} \\ 1 & - & A \end{bmatrix}$ - product

In POS $\begin{bmatrix} 0 & - & A \\ 1 & - & \bar{A} \end{bmatrix}$ - sum

Boolean expression as a sum of minterm or product of maxterm is called are said to be in canonical form.

Minterm and Maxterm

x	y	z	product term		Sum Term	
			Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x+y+z$	M_0
0	0	1	$x'y'z$	m_1	$x+y+z'$	M_1
0	1	0	$x'yz'$	m_2	$x+y'+z$	M_2
0	1	1	$x'yz$	m_3	$x+y'+z'$	M_3
1	0	0	$xy'z'$	m_4	$x'+y+z$	M_4
1	0	1	$xy'z$	m_5	$x'+y+z'$	M_5
1	1	0	xyz'	m_6	$x'+y'+z$	M_6
1	1	1	xyz	m_7	$x'+y'+z'$	M_7

Canonical Representation

$F = AB+C'$

Truth Table

A	B	C	AB+C'
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

SOP

Rows with output 1

Each row product

0 is primed - $\bar{A}$

1 is unprimed - A

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

000 010 100 101 111

0 2 4 6 7

m_0 m_2 m_4 m_6 m_7

$m_0+m_2+m_4+m_6+m_7$

$\therefore F(A,B,C) = \Sigma m(0,2,4,6,7)$

POS

Row with output 0

Each row sum

1 is primed $- \bar{A}$

0 is unprimed $- A$

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

001 011 101

1 3 5

$M_1 \cdot M_3 \cdot M_5$

$\therefore F(A,B,C) = \Pi M(1,3,5)$

Standard Sum of product (SSOP)

A standard SOP expression is one in which all the variables in the domain appear in each product term in the expression.

Each term that doesn't contain other variables in the domain can be expanded into standard form to include all variables in the domain. Therefore any Non-standard expression can be converted into equivalent standard expression using boolean algebra.

Example of SSOP is $\bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D + AB\bar{C}\bar{D} + ABCD$, $AB + \bar{A}B + \bar{A}\bar{B}$, $xyz + \bar{x}yz + x\bar{y}z + xy\bar{z} + \bar{x}\bar{y}\bar{z}$,

Step for converting non-standard SOP to standard SOP

Step 1 :- Multiply each non-standard form by a term made up of the sum of missing variable and its complement $[A+\bar{A}]$.

Step 2 :- Repeat step 1 if the expression is not standard

Conversion of product term of a SOP to standard SOP.

a. $A\bar{B}C + \bar{A}\bar{B} + AB\bar{C}D$

The domain of this SOP expression is A, B, C, D.

Take one term at a time.

The first term is missing variable D or $\bar{D}$. So, multiply it by $D+\bar{D}$. We get

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

Likewise, the second term is missing variable C or $\bar{C}$ and D or $\bar{D}$. So, multiply it by $C+\bar{C}$ and $D+\bar{D}$. We get.

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

The third term is in standard form. So, no need of change.

$$\boxed{AB\bar{C}D}$$

The complete standard form of the original expression is

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

b. $W\bar{x}y + \bar{x}y\bar{z} + Wx\bar{y}$

The domain of this SOP expression is W, x, y and z.

Take one term at a time.

The first term is missing variable z or $\bar{z}$. So multiply it by $z+\bar{z}$. We get

$$W\bar{x}y = W\bar{x}y(z+\bar{z}) = \boxed{W\bar{x}yz + W\bar{x}y\bar{z}}$$

Likewise, the second term is missing variable W or $\bar{W}$. So multiply it by $W+\bar{W}$.

$$\bar{x}y\bar{z} = \bar{x}y\bar{z}(W+\bar{W}) = \boxed{W\bar{x}y\bar{z} + \bar{W}\bar{x}yz}$$

The third term is missing z or $\bar{z}$. So, multiply it by $z+\bar{z}$. We get

$$Wx\bar{y} = Wx\bar{y}(z+\bar{z}) = \boxed{Wx\bar{y}z + Wx\bar{y}\bar{z}}$$

The complete standard SOP form of the expression is

$$W\bar{x}yz + W\bar{x}y\bar{z} + W\bar{x}y\bar{z} + \bar{W}\bar{x}y\bar{z} + Wx\bar{y}z + Wx\bar{y}\bar{z}$$
$$\bar{W}\bar{x}y\bar{z} + W\bar{x}yz + W\bar{x}y\bar{z} + Wx\bar{y}\bar{z} + Wx\bar{y}z$$

Express the boolean function $F = A + \bar{B}C$ as a sum of minterms.

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

The function is not in standard SOP form!

The domain of this SOP function is A, B and C.

The first term is missing B or $\bar{B}$ and C or $\bar{C}$. So multiply it by $B+\bar{B}$ and $C+\bar{C}$.

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

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

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

The second term is missing A or $\bar{A}$. So, multiply it by $A+\bar{A}$.

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

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

The complete standard SOP form of the original function is

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

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

001 100 101 110 111

1 4 5 6 7

$m_1 + m_4 + m_5 + m_6 + m_7$

$\therefore F(A,B,C) = \Sigma m(1, 4, 5, 6, 7)$

Standard product of Sum (SPOS)

A standard POS expression is one in which all the variable in the domain appear in each sum term in the expression.

Example of SPOS $(\bar{A}+\bar{B}+\bar{C}+\bar{D}) \cdot (A+\bar{B}+\bar{C}+\bar{D}) \cdot (\bar{A}+B+\bar{C}+\bar{D})$, $(\bar{x}+y+\bar{z}) \cdot (x+\bar{y}+\bar{z}) \cdot (\bar{x}+\bar{y}+z)$

Step for converting non-standard POS to standard POS.

Step 1 :- Add each non-standard term by a term made up of the multiplication of missing variable and its complement $[A \cdot \bar{A}]$

Step 2 :- Use formulae $A+BC = (A+B)(A+C)$

Step 3 :- Repeat step 1 and step 2 if the equation is not standard.

DATE _____ PAGE _____

Conversion of the following boolean expression in standard pos form.

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

The domain of this pos expression is A, B, C, D.

Take one term at a time.

The first term is missing variable D or $\bar{D}$. So, add it by $D\cdot\bar{D}$. We get.

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

The second term is missing variable A or $\bar{A}$. So add it by $A\cdot\bar{A}$. We get.

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

The third term is in standard form. So, no need of change

$$\boxed{A+\bar{B}+\bar{C}+D}$$

The complete standard form of the original expression is

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

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

The domain of this pos expression is A, B, C, D.

Take one term at a time.

The first term is in standard form so, no need of change.

$$\boxed{\bar{A}+\bar{B}+\bar{C}+\bar{D}}$$

The second term is missing variable A or $\bar{A}$ and B or $\bar{C}$. So add it $A\cdot\bar{A}$ and $C\cdot\bar{C}$

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

The third term is missing variable C or $\bar{C}$. So, add it $C\cdot\bar{C}$.

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

The complete standard form of the original expression is

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

Express the boolean function $F = xy + \bar{x}z$ in a product of maxterm.

$F = xy + \bar{x}z$

The function $F = xy + \bar{x}z$ is not in pos form.

So, converting the function into OR term using distributive law.

$F = xy + \bar{x}z$

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

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

$\therefore F = (\bar{x} + y)(x + z)(y + z)$ $[\because x + \bar{x} = 1]$

The function is not in standard pos form.

The domain of a function is x, y, z

The first term is missing z or $\bar{z}$. So Add it by $z \cdot \bar{z}$. we get

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

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

The second term is missing y or $\bar{y}$. So, Add it by $y \cdot \bar{y}$. We get

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

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

The third term is missing x or $\bar{x}$. So, Add it by $x \cdot \bar{x}$. We get

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

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

The complete standard pos form of the original function is

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

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

$0\ 0\ 0 \quad 0\ 1\ 0 \quad 1\ 0\ 0 \quad 1\ 0\ 1$

$0 \cdot 2 \cdot 4 \cdot 5$

$(M_0 \cdot M_2 \cdot M_4 \cdot M_5)$

$\therefore F(x, y, z) = \Pi M(0, 2, 4, 5)$

Conversion between Canonical form

Example:-

$F(A,B,C) = \Sigma m(1,4,5,6,7)$

Since we know that total possible outputs are $2^3 = 8$.

Again we know that pos is complement of sop

$F(A,B,C) = \Sigma m(1,4,5,6,7)$

$\overline{F}(A,B,C) = \Sigma(0,2,3) \Rightarrow M_0 + M_2 + M_3$

If we take the complement of $\overline{F}$ by Demorgan's theorm. We obtain Fa

$F = \overline{M_0 + M_2 + M_3}$

$= \overline{M}_0 \cdot \overline{M}_2 \cdot \overline{M}_3 \Rightarrow M_0 \cdot M_2 \cdot M_3$

$= \Pi m(0,2,3)$

Converting Standard Sop to Standard POS

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

The evaluation are as follows:-

$\bar{A}\bar{B}\bar{C} = 000$, $\bar{A}B\bar{C} = 010$, $\bar{A}BC = 011$, $A\bar{B}C = 101$, $ABC = 111$

Since there are 3 variable in the given SOP expression. There are a total of 8 (2^3) positive combination.

The SOP expression contains 5 of these combination. So, the POS must contain other three. They are

001, 100, 110

The equivalent POS expression is

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

Converting Standard pos to Standard SOP

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

The evaluation are as follows:-

$A+B+\bar{C} = 001$, $A+\bar{B}+C = 010$, $A+\bar{B}+\bar{C} = 011$, $\bar{A}+B+\bar{C} = 101$

Since there are 3 variables in the given SOP expression. There are a total of 8 (2^3) positive combination.

The POS expression contains 4 of these combination. So the SOP must contain other four. They are

DATE _____ PAGE _____

000, 100, 110, 111

The equivalent sop expression is
$\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + AB\bar{C} + ABC$

Converting standard sop to standard pos
$\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + AB\bar{C} + ABC$
000 100 110 111
0 4 6 7 → equivalent of binary
1 2 3 5 → Remaining numbers
001 010 011 101 — Converting to binary
$(A+B+\bar{C}) \cdot (A+\bar{B}+C) \cdot (A+\bar{B}+\bar{C}) \cdot (\bar{A}+B+\bar{C})$

Sop to pos conversion steps
Step 1:- Evaluate each product term in the standard sop expression that is represented as the binary form.
Step 2:- Determine all of the remaining binary numbers not included in step 1.
Step 3:- Write the equivalent sum term of each binary number from step 2.
Step 4:- Compile equation from step 3 into pos form.

Similarly we can get pos to sop too.

Truth Table format of ssop expression
$\bar{A}\bar{B}C + A\bar{B}C + AB\bar{C}$

Inputs A	B	C	output (Y)	product term
0	0	0	0	
0	0	1	1	$\bar{A}\bar{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	0	
1	0	1	1	$A\bar{B}C$
1	1	0	1	$AB\bar{C}$
1	1	1	0	

$\bar{A}\bar{B}C + A\bar{B}C + AB\bar{C}$
001 101 110
Except these all are zero.
Otherwise All 1 in the output of above numbers.

Truth Table format of SPOS Expression

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

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

Inputs A	B	C	Outputs	Sum term
0	0	0	0	$A+B+C$
0	0	1	1	
0	1	0	0	$A+\bar{B}+C$
0	1	1	0	$A+\bar{B}+\bar{C}$
1	0	0	1	
1	0	1	0	$\bar{A}+B+\bar{C}$
1	1	0	1	
1	1	1	0	$\bar{A}+\bar{B}+\bar{C}$

000, 010, 011, 101, 100 now fill '0' as output of function

Determining Standard SOP / POS expression from truth table.

Input x	y	z	Output(y)
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

For SSOP expression

There are 4 output 's' in output column and their corresponding value are 000, 010, 011 and 110

000 = $\bar{x}, \bar{y}, \bar{z}$

010 = $\bar{x}y\bar{z}$

011 = $\bar{x}yz$

110 = $xy\bar{z}$

$\bar{x}\bar{y}\bar{z} + x\bar{y}\bar{z} + \bar{x}yz + xy\bar{z}$

∴ Hence, the resulting expression for the output y is

$\bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}yz + xyz$

For SPOS expression

There are 4 output 0's in output column and their corresponding value are

001, 100, 101 and 111

$001 = x + y + \bar{z}$

$100 = \bar{x} + y + z$

$101 = \bar{x} + y + \bar{z}$

$111 = \bar{x} + \bar{y} + \bar{z}$

$(x+y+\bar{z}).(\bar{x}+y+z).(\bar{x}+y+\bar{z}).(\bar{x}+\bar{y}+\bar{z})$

∴ Hence, the resulting expression for the output y is

$(x+y+\bar{z}).(\bar{x}+y+z).(\bar{x}+y+\bar{z}).(\bar{x}+\bar{y}+\bar{z})$

DATE _____ PAGE _____

5. The Karnaugh Map (K-Map)

The K-map provides the systematic method for simplifying boolean expression and thus provides the simplest SOP or POS expression.

It is similar to the truth table because it represent all of the positive values of input variables and resulting outputs for each value. It is organized in an array of cells in which each cell represents binary value of the input variables.

K-map can be used for expressions with number of variables which is greater than 2.

Number of cells in the maps is given by $2^{\text{no. of variables}}$.

For the K-Map the expression should be in standard form i.e.

SOP → SSOP → K-Map

POS → SPOS → K-Map

K-Map is organized in an array of cells in which each cell represents the binary value of the input variable.

Two variable K-Map

Variable A, B

A \ B	$\bar{B}$ (0)	B (1)
$\bar{A}$ (0)	00 $\bar{A}\bar{B}$ m_0	01 $\bar{A}B$ m_1
A (1)	10 $A\bar{B}$ m_2	11 AB m_3

Three variable K-Map

Variable A, B, C

A \ BC	$B\bar{C}$ (00)	$\bar{B}\bar{C}$ (01)	$\bar{B}C$ (11)	$B\bar{C}$ (10)
$\bar{A}$ (0)	000 $\bar{A}\bar{B}\bar{C}$ m_0	001 $\bar{A}\bar{B}C$ m_1	011 $\bar{A}BC$ m_3	010 $\bar{A}B\bar{C}$ m_2
A (1)	100 $A\bar{B}\bar{C}$ m_4	101 $A\bar{B}C$ m_5	111 ABC m_7	110 $AB\bar{C}$ m_6

AB \ C	$\bar{C}$ C = 0	C = 1
	000 $A\bar{B}\bar{C}$	001 $A\bar{B}C$
$\bar{A}\bar{B}$ 00	M_0	M_1
	010 $\bar{A}B\bar{C}$	011 $\bar{A}BC$
$\bar{A}B$ 01	M_2	M_3
	110 $AB\bar{C}$	111 ABC
AB 11	M_6	M_7
	100 $A\bar{B}\bar{C}$	101 $A\bar{B}C$
$A\bar{B}$ 10	M_4	M_5

Four variable k-Map

Variable A, B, C, D

AB \ CD	$\bar{C}\bar{D}$ C=0 D=0	$\bar{C}D$ C=0 D=1	CD C=1 D=1	$C\bar{D}$ C=1 D=0
$\bar{A}\bar{B}$ 00	0000 $\bar{A}\bar{B}\bar{C}\bar{D}$ M_0	0001 $\bar{A}\bar{B}\bar{C}D$ M_1	0011 $\bar{A}\bar{B}CD$ M_3	0010 $\bar{A}\bar{B}C\bar{D}$ M_2
$\bar{A}B$ 01	0100 $\bar{A}B\bar{C}\bar{D}$ M_4	0101 $\bar{A}B\bar{C}D$ M_5	0111 $\bar{A}BCD$ M_7	0110 $\bar{A}BC\bar{D}$ M_6
AB 11	1100 $AB\bar{C}\bar{D}$ M_{12}	1101 $AB\bar{C}D$ M_{13}	1111 $ABCD$ M_{15}	1110 $ABC\bar{D}$ M_{14}
$A\bar{B}$ 10	1000 $A\bar{B}\bar{C}\bar{D}$ M_8	1001 $A\bar{B}\bar{C}D$ M_9	1011 $A\bar{B}CD$ M_{11}	1010 $A\bar{B}C\bar{D}$ M_{10}

Map the following SOP expression in k-Map

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

001 010 110 111

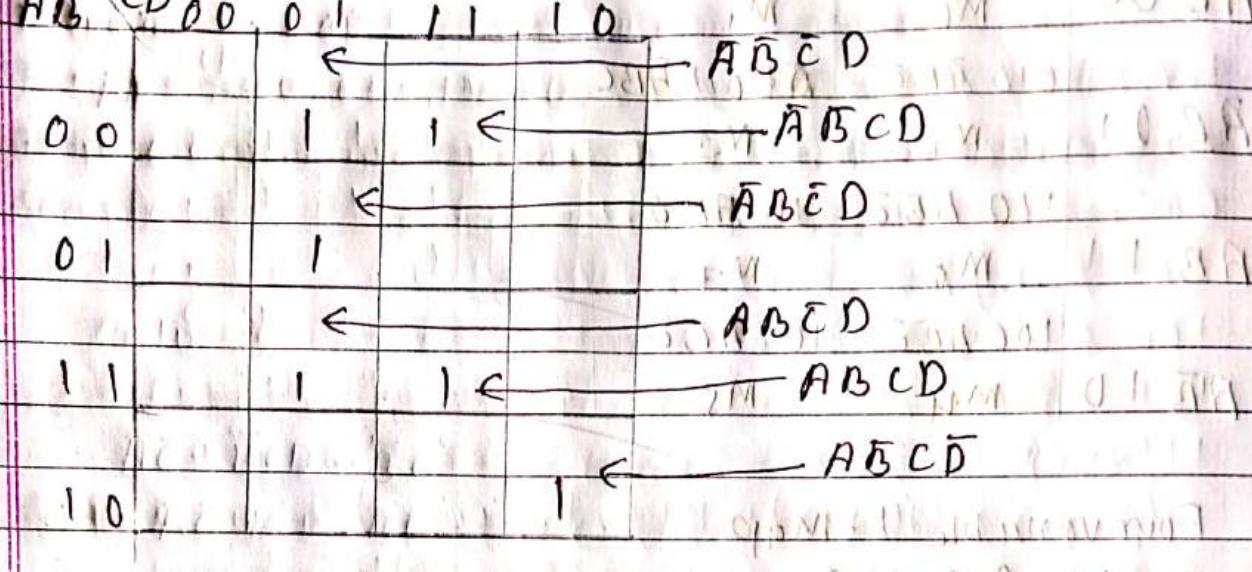
A \ BC	00	01	11	10
0		1	1	
1			1	1

$A\bar{B}C$, $\bar{A}B\bar{C}$, ABC, AB

DATE _____ PAGE _____

$\bar{A}\bar{B}CD + \bar{A}B\bar{C}D + AB\bar{C}D + ABCD + \bar{A}\bar{B}\bar{C}D + A\bar{B}C\bar{D}$

0011 0101 1101 1111 0001 1010



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

The expression is not in standard form. So, converting it into standard form.

The domain of this sop expression is A, B, C, D

The first term is missing B or $\bar{B}$ and C or $\bar{C}$. So, multiply first term by $(B+\bar{B})$ and $(C+\bar{C})$. The second term is missing C or $\bar{C}$. So, multiply second term by $(C+\bar{C})$. The third contain all the variable. So, do nothing to third term.

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

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

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

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

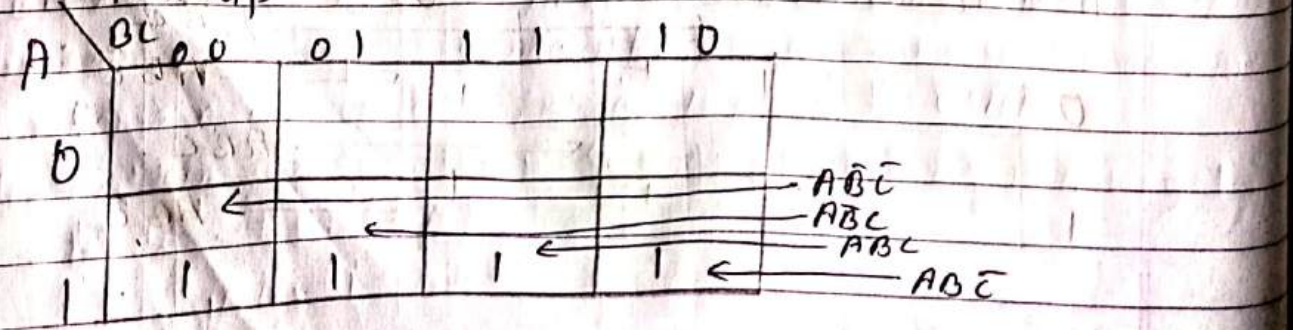
The standard sop form of original expression is

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

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

100 101 110 111

K-Map



DATE _____ PAGE _____

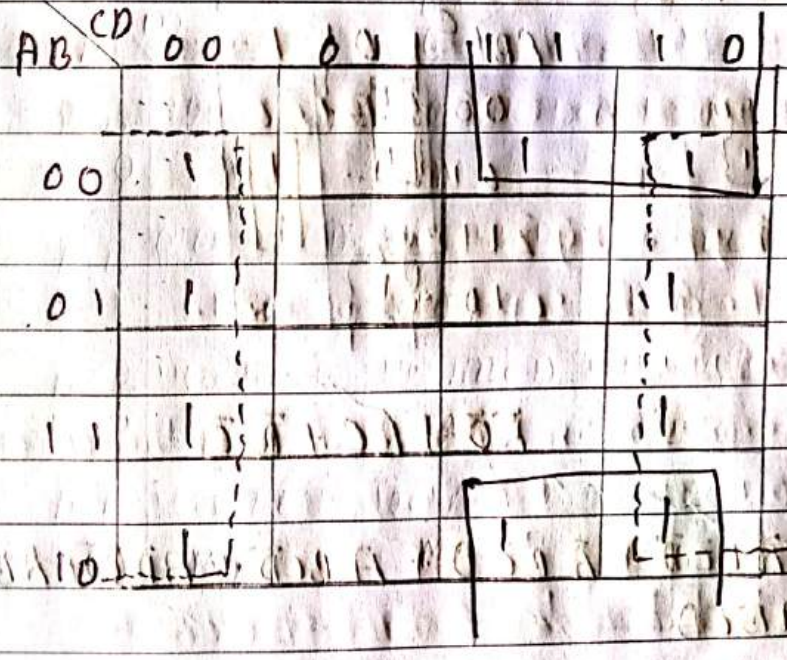
Karnaugh Map Simplification (Grouping 1's)

A group is formed by the outputs giving similar functionality or value.
A group can contain $2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$, $2^4=16$ ... cells (2^n)
Each cell in a group must be adjacent (wrap-around available)
Always try to make maximum numbers of cells as possible in a group.
Each one must be included in at least one group.

Example

$A\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}\bar{B}CD + A\bar{B}CD + \bar{A}\bar{B}C\bar{D} + \bar{A}BC\bar{D} + AB\bar{C}D + A\bar{B}C\bar{D}$

Filling in K-Map



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

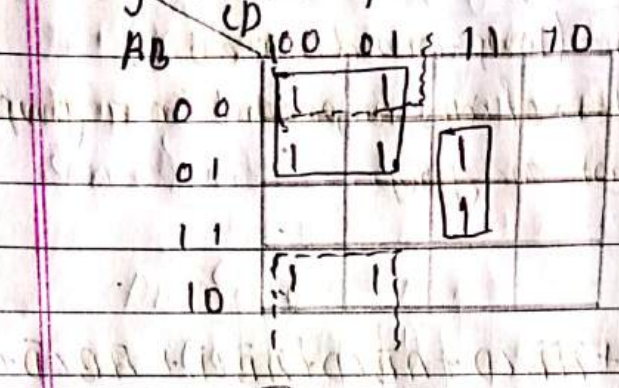
$\therefore F = \bar{D} + \bar{B}C$

$\therefore$ Note: Both the expression will give same value. Thus choose.

DATE _____ PAGE _____

$\bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D +$
$ABCD$

Filling in k-Map

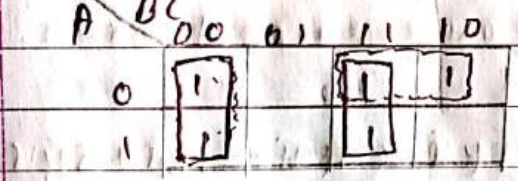


$\bar{A}\bar{C} + \bar{B}CD + \bar{B}\bar{C}$

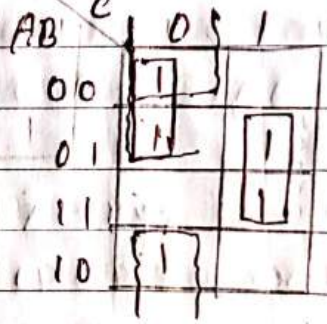
$\therefore F = \bar{A}\bar{C} + BCD + \bar{B}\bar{C}$

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

Filling in k-Map



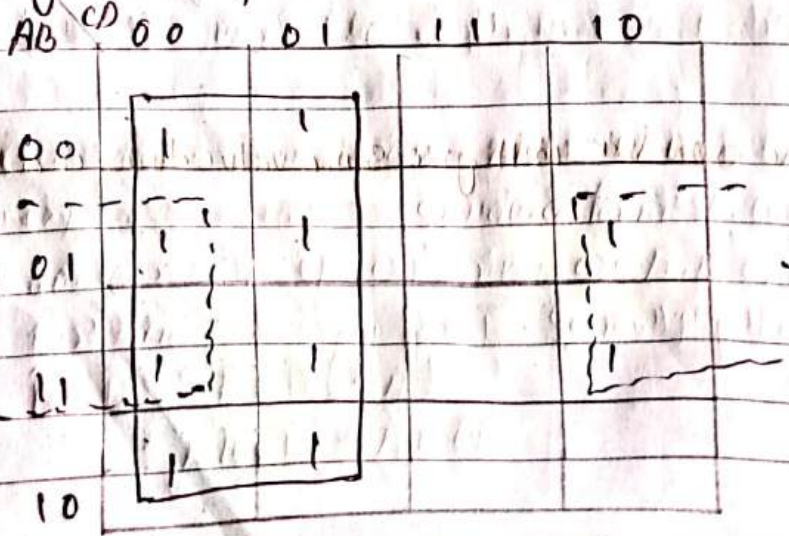
$\bar{B}\bar{C} + BC + \bar{A}B$



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

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

Filling in k-Map



$B + D$

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

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

DATE _____ PAGE _____

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

Using Boolean Algebra

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

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

$\bar{A}B\bar{C} + A\bar{B} + AB \quad [\because \bar{C} + C = 1]$

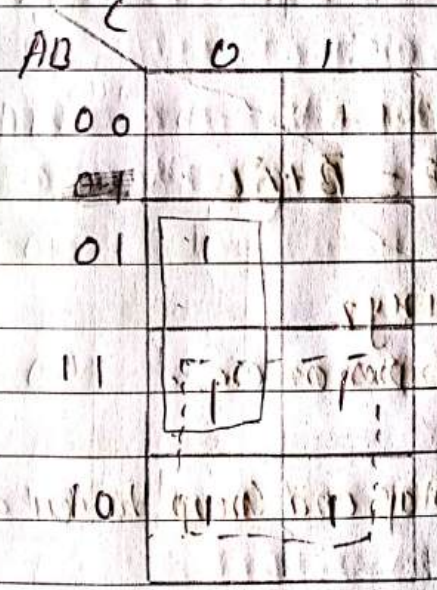
$\bar{A}B\bar{C} + A(\bar{B} + B)$

$\bar{A}B\bar{C} + A \quad [\because B + \bar{B} = 1]$

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

Using K-Map

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



$A + B\bar{C}$

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

DATE _____ PAGE _____

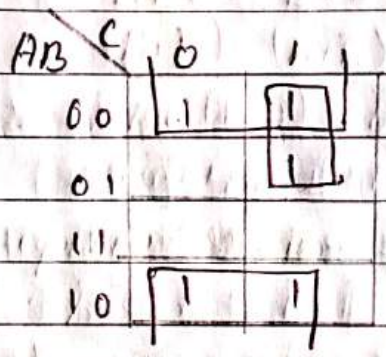
Use K-Map to minimize the following SOP expression.

a. $A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + AB\bar{C}$

The binary values of the given SOP expression are

101, 011, 001, 000, 100

Mapping the standard SOP expression in k-Map and Group the cell as shown below:-



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

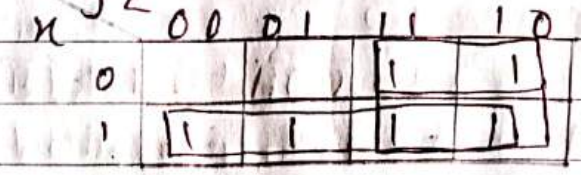
∴ Hence the minimize expression is $F = \bar{B} + \bar{A}C$

b. $x\bar{y}z + xy\bar{z} + \bar{x}yz + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$

The binary values of the given SOP expression are:-

101, 110, 011, 010, 100, 111

Map the standard SOP expression in k-Map and Group the cell as shown below



$x + y$

∴ Hence the minimize expression is $F = x + y$.

c. $\bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} + AB\bar{C}D + A\bar{B}CD + \bar{A}\bar{B}CD + \bar{A}BC\bar{D} + AB\bar{C}D + A\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D}$

The expression is not in standard form. So, converting in standard form

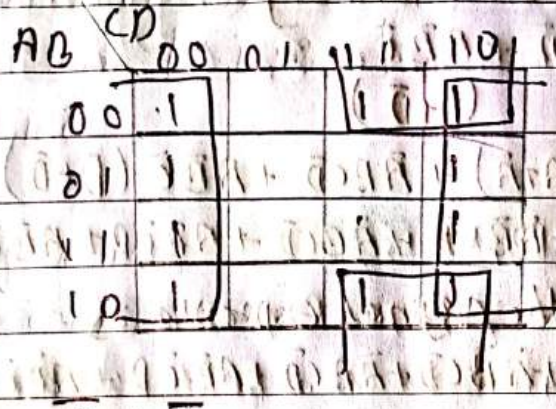
The domain of SOP expression are A, B, C, D

The first term is missing D or $\bar{A}$. So, multiply the term by $A + \bar{A}$, we get

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

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

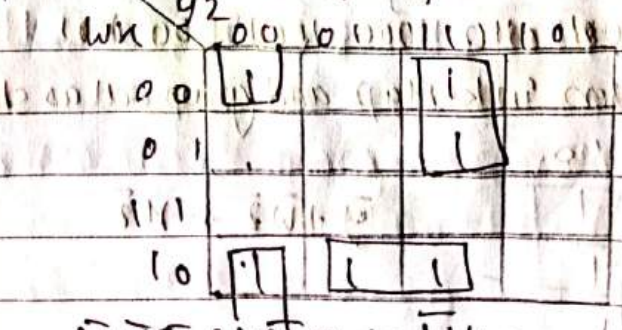
∴ The equivalent standard sop expression of original sop expression is
$A\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + AB\bar{C}D + \bar{A}\bar{B}CD + \bar{A}BC\bar{D} + ABC\bar{D} + AB\bar{C}\bar{D} + A\bar{B}CD$
The binary values of the given sop expression are :-
(1000, 0000, 0100, 1100, 0011, 0010, 0110, 1110, 1010, 1011)
Map the standard sop expression in k-Map and group the cell as shown below:-



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

∴ Hence, the minimize expression is $F = \bar{D} + \bar{B}C$

d. $\bar{W}\bar{X}\bar{Y}\bar{Z} + W\bar{X}YZ + W\bar{X}\bar{Y}Z + \bar{W}YZ + W\bar{X}\bar{Y}\bar{Z}$
The expression is not in standard form. So, converting it in standard form.
The domain of sop expression are W, X, Y, Z.
The fourth term is missing X or $\bar{X}$. So, multiply it with $X + \bar{X}$. We get
$\bar{W}YZ = \bar{W}YZ(X + \bar{X})$
$= \bar{W}XYZ + \bar{W}\bar{X}YZ$
∴ The equivalent standard sop expression of original sop expression is :-
$\bar{W}\bar{X}\bar{Y}\bar{Z} + W\bar{X}YZ + W\bar{X}\bar{Y}Z + \bar{W}\bar{X}YZ + \bar{W}XYZ + WX\bar{Y}\bar{Z}$
The binary values of given sop expression are :-
0000, 1011, 1001, 0011, 0111, 1000
Map the standard sop expression in k-Map and Group the cells as shown below:-



$\bar{X}\bar{Y}\bar{Z} + W\bar{X}Z + \bar{W}YZ$
∴ Hence the minimize expression is $F = \bar{X}\bar{Y}\bar{Z} + W\bar{X}Z + \bar{W}YZ$

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

The expression is not in standard form. So convert it in standard form.

The domain of SOP expression are A, B, C, D

The first term is missing D or $\bar{D}$, second term is missing A or $\bar{A}$ and fourth term is missing D or $\bar{D}$. So multiply the first, second and fourth term by $D+\bar{D}$, $A+\bar{A}$ and $D+\bar{D}$ respectively

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

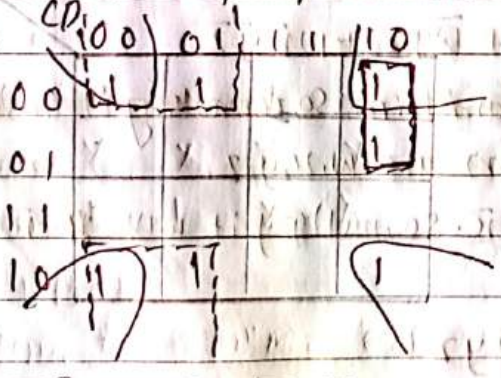
The equivalent Standard SOP of the original expression is

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

The binary value of given SOP expression are :-

0001 0000 1010 0010 1011 1001 1000

Map the standard SOP expression in k-Map and group the cell as shown below.



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

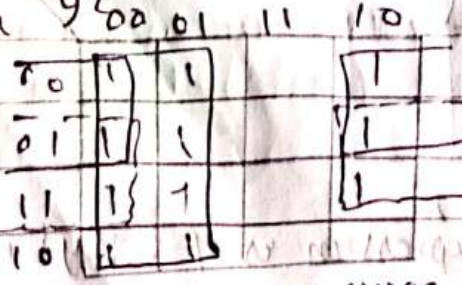
∴ Hence the minimize expression is $F = \bar{B}\bar{D} + B\bar{C} + \bar{A}C\bar{D}$

f. $F(W,X,Y,Z) = \Sigma(0,1,2,4,5,6,8,9,12,13,14)$

Converting the given minterm in binary number:

0000 0001 0010 0100 0101 0110 1000 1001 1100 1101 1110

Map the standard SOP expression in k-Map and group cell as shown below.



$$\bar{Y} + \bar{W}\bar{Z} + X\bar{Z}$$

∴ Hence, the minimize expression is $F = \bar{Y} + \bar{W}\bar{Z} + X\bar{Z}$

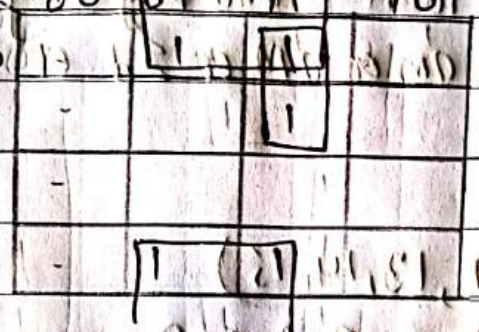
g. $F = \Sigma(1,3,7,9,11)$

Let Declare that the variable are A, B, C, D.

The binary value of given minterms are

0001 0011 0111 1001 1011

Map the expression in k-Map and group cell as shown below:-



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

∴ Hence, the minimize expression of $= \bar{B}D + \bar{A}CD$

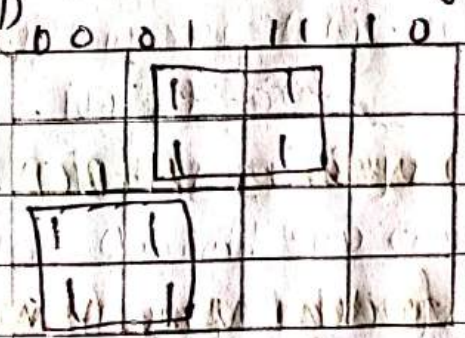
h. $y = m_1 + m_3 + m_5 + m_7 + m_8 + m_9 + m_{12} + m_{13}$

Let declare that the variable are A, B, C, D.

The binary value of given minterm are

0001 0011 0101 0111 1000 1001 1100 1101

Map the expression in k-Map and group the cell as shown below:-



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

∴ Hence, the minimize expression of $y = A\bar{C} + \bar{A}D$

"Don't Care" Conditions

Don't care conditions are situations which are not allowed to occur with input variable combinations.

It is Denoted by 'x' sign.

Since it has no fixed value so can be used as '1' while grouping with '1' and '0' while grouping with '0' and anything what we are grouping.

It Sometimes denoted by dm or dM or Σ_d or Π_d etc.

Example:-

$\Sigma(7,8,9), d(10,11,12,13,14,15)$

Let declear the variable are A, B, C, D.

The binary value of given minterms and don't care are:-

0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

Map the expression in k-map and group the cell as shown below

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

A + BCD

∴ Hence, the minimize expression is A+BCD.

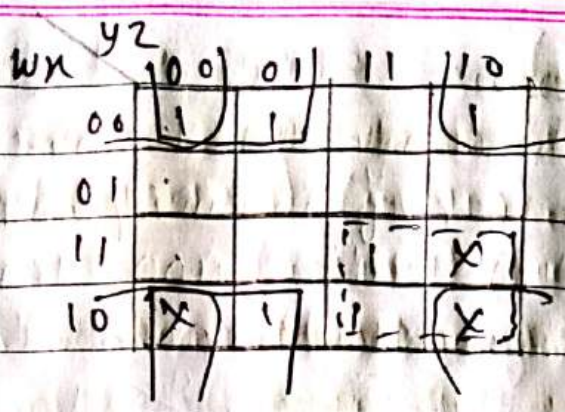
Minimize the given boolean function using K-map.

$F(w,x,y,z) = \Sigma m(0,1,2,9,11,15)$ and $\Sigma_d(8,10,14)$ and implement the obtained expression using NAND Gate only.

The binary value of the given minterms and don't care term are:-

{0000, 0001, 0010, 1001, 1011, 1111} {1000, 1010, 1110}

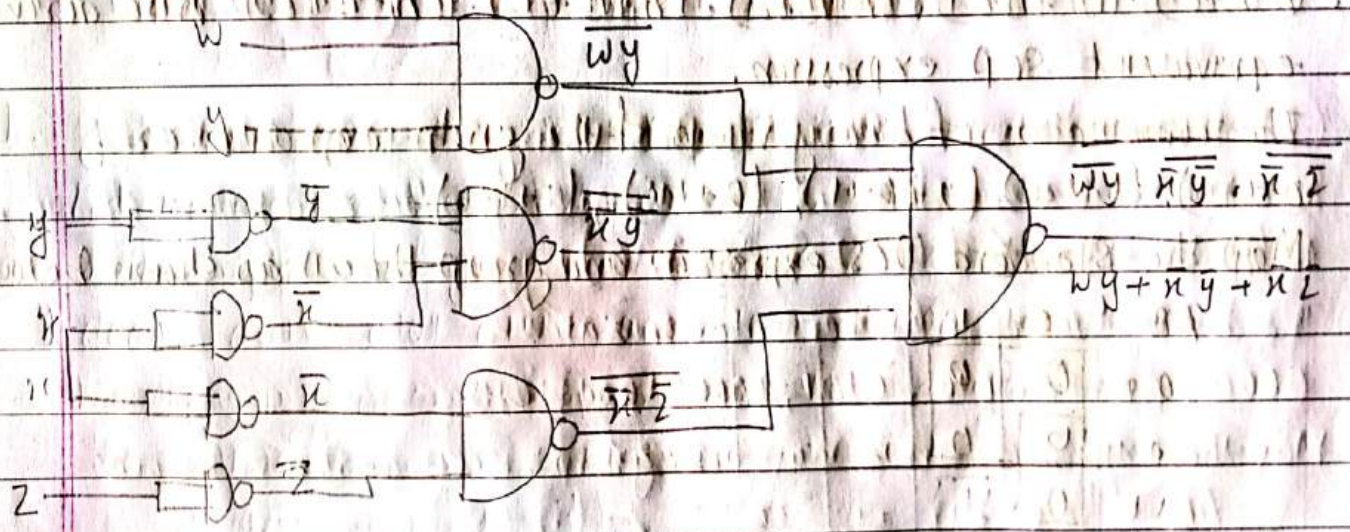
Map the expression in k-map and group the cell as shown below:-



$\bar{x}\bar{y}+\bar{y}y.\ \bar{x}\bar{z}+\bar{x}\bar{y}+wy$

∴ Hence, the minree expression is $f=\bar{x}\bar{z}+\bar{x}\bar{y}+wy$

Implementing the function $f=\bar{x}\bar{z}+\bar{x}\bar{y}+wy$ using NAND Gate:-

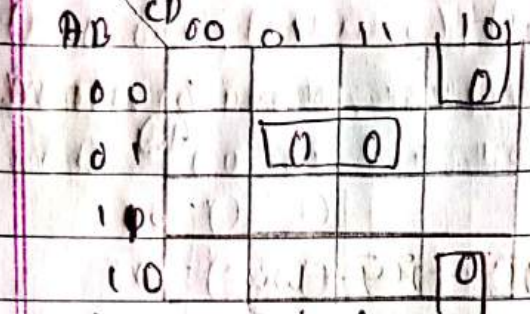


$\therefore \overline{wy}\cdot\overline{\bar{x}\bar{y}}\cdot\overline{\bar{x}\bar{z}} = \overline{\overline{wy}}+\overline{\overline{\bar{x}\bar{y}}}+\overline{\overline{\bar{x}\bar{z}}} = wy+\bar{x}\bar{y}+\bar{x}\bar{z}$

Mapping a standard pos expression

pos ⇒ spos ⇒ K-Map

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



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

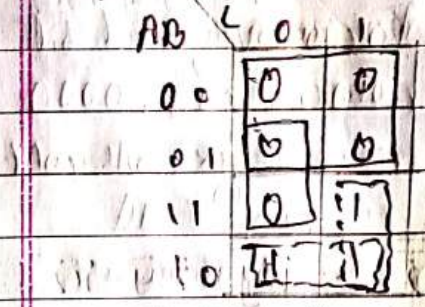
Use the K-map to minimize the following standard pos expression

a. $(A+B+C).(A+\bar{B}+C).(A+B+\bar{C}).(A+\bar{B}+\bar{C}).(\bar{A}+\bar{B}+C)$. Also derive equivalent sop expression.

The combination of binary values of the given expression are

(0+0+0), (0+1+0) (0+0+1), (0+1+1), (1+1+0)

Map the standard pos expression and group the cell as shown below:



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

∴ Hence, the minimize pos expression is $A(\bar{B}+C)$.

For equivalent sop expression

$AC+A\bar{B}$

∴ The equivalent sop expression is $AC+A\bar{B}$.

b. $(x+\bar{y}+z).(x+\bar{y}+\bar{z}).(\bar{x}+\bar{y}+z).(\bar{x}+y+z)$

The combination of binary values of given expression are:-

(0+1+0).(0+1+1).(1+1+0).(1+0+0)

Map the standard pos expression and group the cell as shown below.

xy \ z	0	1
00		
01	0	0
11	0	
10	0	

$(\bar{x}+z).(x+\bar{y})$

∴ Hence, the minimise pos expression is $(\bar{x}+z).(x+\bar{y})$

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

The expression is not in standard form. So, convert it in standard form.

The domain of a function is A, B, C, D.

The first term is missing A or $\bar{A}$. So, Add it by $A\cdot\bar{A}$. we get :

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

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

The equivalent standard pos form of the original function is

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

The combination of binary value of the given expression are

$(0+0+0+0).(1+0+0+0).(0+0+1+0).(1+0+0+1).(0+1+0+0)(1+1+0+0)$

Map the standard pos expression and group the cell as shown below:

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

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

∴ Hence, the minimise pos expression is $(C+D).(\bar{A}+B+C).(A+B+D)$

DATE _____ PAGE _____

d. $y = \pi(0,1,4,5,6,8,9,12,13,14)$

The binary value of given maxterm are:-

$(0+0+0+0)$, $(0+0+0+1)$, $(0+1+0+0)$, $(0+1+0+1)$, $(0+1+1+0)$, $(1+0+0+0)$, $(1+0+0+1)$, $(1+1+0+0)$, $(1+1+0+1)$, $(1+1+1+0)$

Map the expression in k-map and group the cell as shown below.

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

$C.(\bar{B}+D)$

∴ Hence, the minimize pos expression is $y = C.(\bar{B}+D)$

Converting between Sop and pos using k-map method.

Simplify the boolean function in Sum of product and product of sum.

$F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$.

The binary values of given maxterm are:-

0000, 0001, 0010, 0101, 1000, 1001, 1010.

Map the expression in k-map and group the cell as shown below:-

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

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

∴ Hence, the minimize sop expression is $F = \bar{B}\bar{D} + \bar{B}\bar{C} + \bar{A}\bar{C}D$

If the cell marked with zero's are combined, we get the simplified complemented function $\bar{F} = CD + AB + B\bar{D}$

Using D-Morgan's Theorem:- $\bar{\bar{F}} = \overline{CD + AB + B\bar{D}}$

$= \overline{CD} \cdot \overline{AB} \cdot \overline{B\bar{D}}$

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

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

DATE _____ PAGE _____

Using a k-Map method, Convert the following standard pos expression into minimum pos expression, A standard sop expression and a minimum sop expression.

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

The binary values of the given expression are

(1+1+0+0), (0+1+0+0), (0+0+0+1), (0+0+1+1), (1+0+0+0), (0+0+1+0)

Map the standard pos expression in k-Map and group the cell as shown below:-

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

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

∴ Hence, the minimize pos expression is $F = (\bar{A}+C+D).(B+C+\bar{D})(A+B+\bar{C})$

Now, 1's are added to the cell that don't contain zero's.

From each cell containing 1 standard product term obtained as indicated:-

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

Standard SOP

$\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + \bar{A}BC\bar{D} + AB\bar{C}D + ABCD + ABC\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D}$

For the minimum SOP expression:-

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

$BD + BC + AC + \bar{B}\bar{C}\bar{D}$

∴ Hence, the minimize SOP expression is $F = BD + BC + AC + \bar{B}\bar{C}\bar{D}$

DATE _____ PAGE _____

Simplify the boolean function $F(w,x,y,z) = \Sigma(1,3,7,11,15)$ and don't care conditions $d(w,x,y,z) = \Sigma(0,2,5)$ to get the minimize sop and pos expression.

The binary value of minterm and don't care term are :-
{0001, 0011, 0111, 1011, 1111}, {0000, 0010, 0101}

Map the expression in k-map and group the cell as shown below :-

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

$y2 + \bar{w}\bar{x}$

∴ Hence, the minimize expression is $F = y2 + \bar{w}\bar{x}$

If cell marked zeros are combined, we obtained the simplified complemented function $\bar{F} = \bar{2} + w\bar{y}$

Using D-Morgans Theorem $\bar{\bar{F}} = \overline{\bar{2} + w\bar{y}}$

$F = \bar{2} \cdot \overline{w\bar{y}}$

$= 2 \cdot (\bar{w} + \bar{\bar{y}})$

$= 2 \cdot (\bar{w} + y)$

∴ Hence, the minimize pos expression is $F = 2(\bar{w} + y)$

6. Arithmetic Circuit
Half-Adders
Full-Adders
Half-Subtractor
Full-Subtractor

Half-Adders

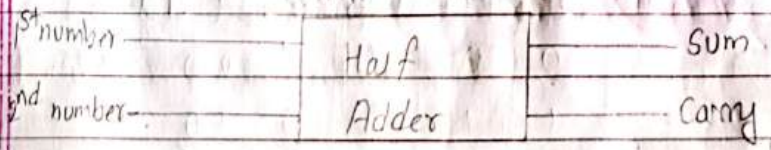
The simplest combinational circuit which performs the arithmetic operation (addition) of two bits is called a half adder.

It has two inputs and two outputs where inputs are two 0, 1 bit numbers and the outputs are sum of A and B and the carry bit.

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$

The operations are performed by a logic circuit called a half-adder.

Logical / Graphical Symbol



Truth Table

Inputs		Outputs	
A	B	Sum(S)	Carry (Cout)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

To draw logical circuit of Half-Adder, product term of each output is determined and it is expressed in SOP form:-

K-Map for simplified SOP:-

Sum:-

A \ B	0	1
0		1
1	1	

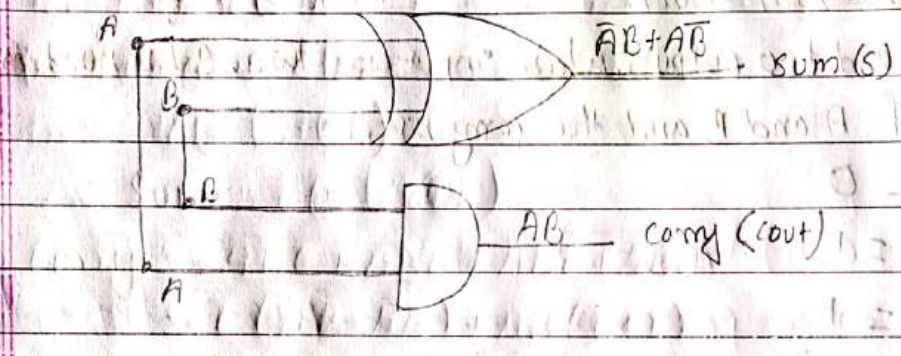
Carry:-

A \ B	0	1
0		
1		1

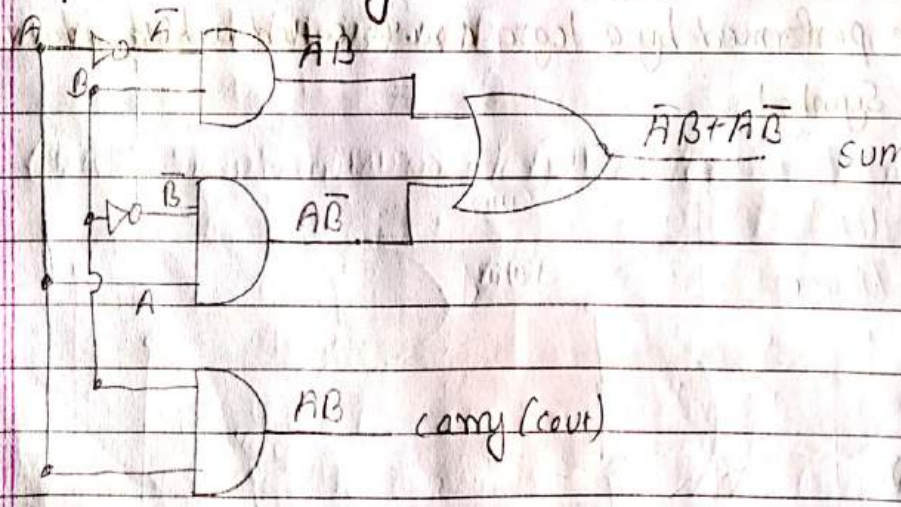
Sum = $A\bar{B} + \bar{A}B$

Carry = AB

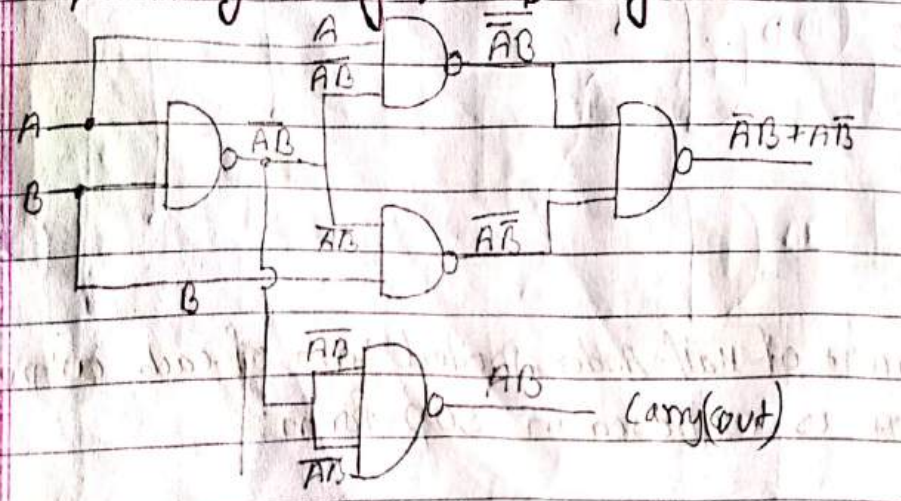
Circuit implementation of Half Adder



implementation using basic Gates



Implementing using NAND only



Full - Adder

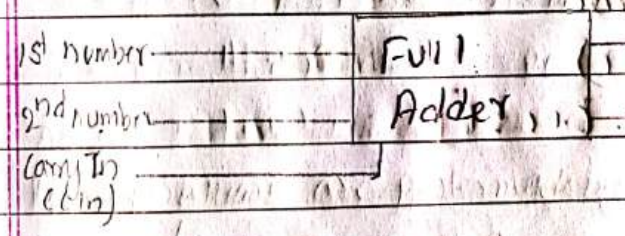
A full adder is a combinational circuit that forms the sum of 3 input bits. It consists of 3 inputs and two outputs.

Two of the input variables Denoted by A and B represented the two significant bits to be added and the third input variable represents the carry from the previous lower significant position.

Two outputs from full adder are sum and carry out for inputing in another full adder or acts as left most bit.

The full-adder accepts two input bits and an input carry and generates a sum output and an output carry.

Graphical Symbol



Truth Table

Inputs			Outputs	
A	B	C	Sum(S)	Carry(Cout)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

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

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

Sop form

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

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

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

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

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

Let $A \oplus B = x$

Then, $= \bar{C}x + C\bar{x}$

$= C \oplus x$

Put the value of x

$C \oplus (A \oplus B)$

$\therefore Sum = C \oplus (A \oplus B)$

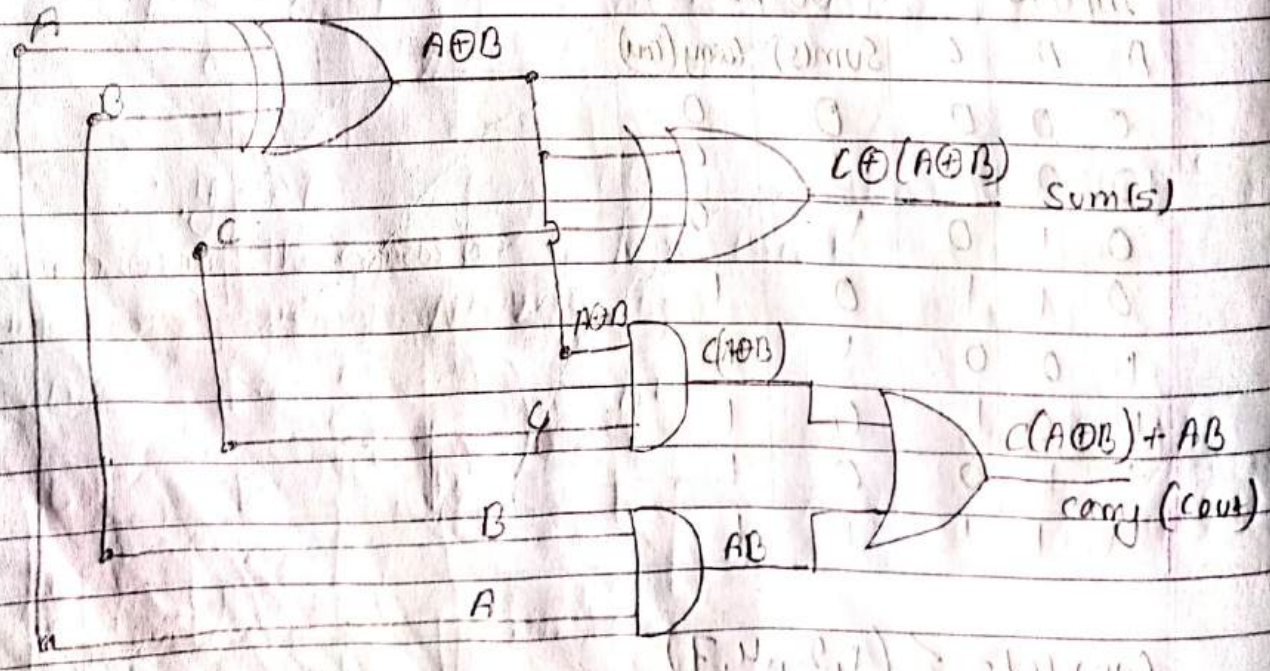
$Carry = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

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

$= C(A \oplus B) + AB \cdot 1 \quad [\because \bar{C} + C = 1]$

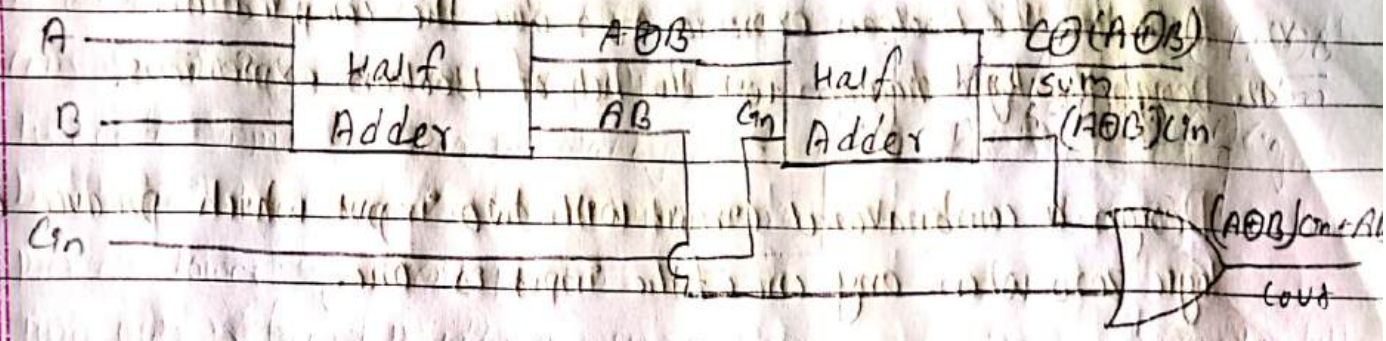
$\therefore Carry = C(A \oplus B) + AB$

Circuit implementation of Full Adder



DATE _____ PAGE _____

Arrangement of two Half-Adder to form a full-Adder



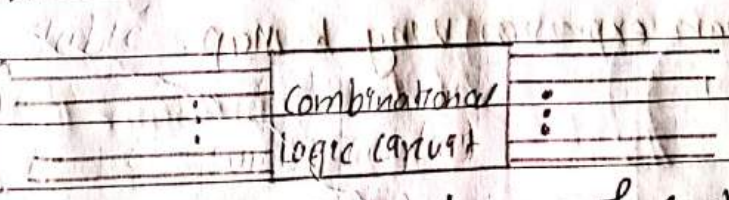
∴ Fig:- Full-Adder using Half-Adder

Combinational logic Circuit

Combinational logic circuit is the combination of various kinds of logic gates. At any instant of time, the output is based only on the level of current input terminals and not on the past state inputs.

These circuits do not need any kind of memory states or clock, so the past inputs show no influence on the current states of the circuit.

A combinational circuit can intake 'n' numbers of outputs and delivers only one output.



∴ Fig:- Block diagram of combinational logic Circuit

Design proc The examples of combinational logic Circuit are Half Adder, Full Adder, Half Subtractor, Full Subtractor, Multiplexer, Demultiplexer

Combinational logic Circuit Design procedure

The below steps explains the design procedure of how a combinational logic system is developed:-

i. Identifying the required number of input and output variables
ii. Symbolizing all the identified input and output variables.
iii. Express the relationship between these variables.
iv. With the relationship, construct truth table.

v. Derive the Boolean expression for all the outputs.
vi. Minimize the Boolean expression to reduce the complication.
vii. Design the logic diagram with the help of Boolean expressions.

Design a combinational circuit with two input which produce output as logic zero when any one of the input is one.

The inputs are 2 and let it decleares A and B variable.
The output is 1 and let it decleares y.
The relationship between variables are shown below with the help of Truth table:-

Truth Table

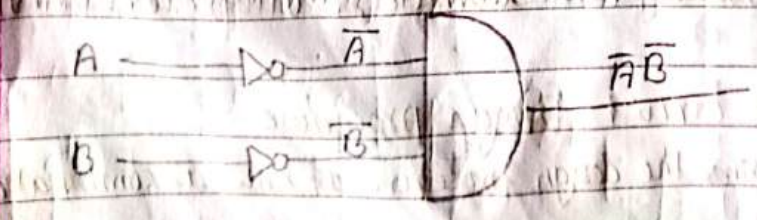
Inputs A	Inputs B	Outputs (y)
0	0	1
0	1	0
1	0	0
1	1	0

Simplifying the boolean expression using K-Map.

A \ B	0	1
0	(1)	
1		

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

Implementing the expression in logic circuit

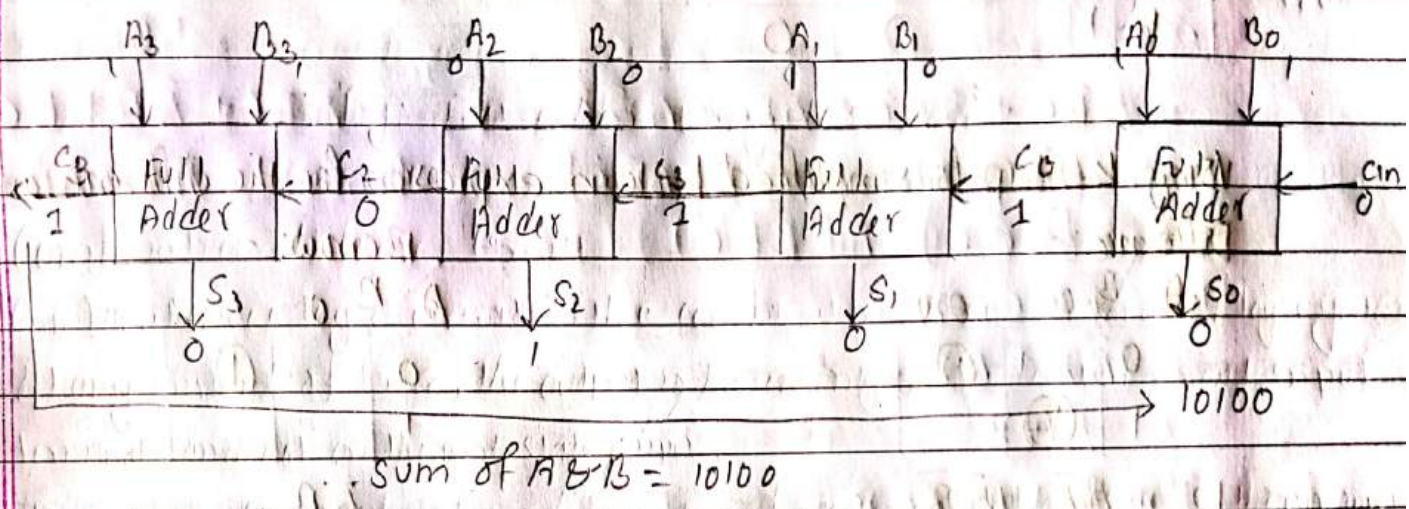


Parallel Binary Adder

Parallel Binary Adder is a cascade of several full adder to add multiple bit number.

Two or more full adders are connected to form parallel binary adders.

Let A = 1011 and B = 1001



∴ Sum of A & B = 10100

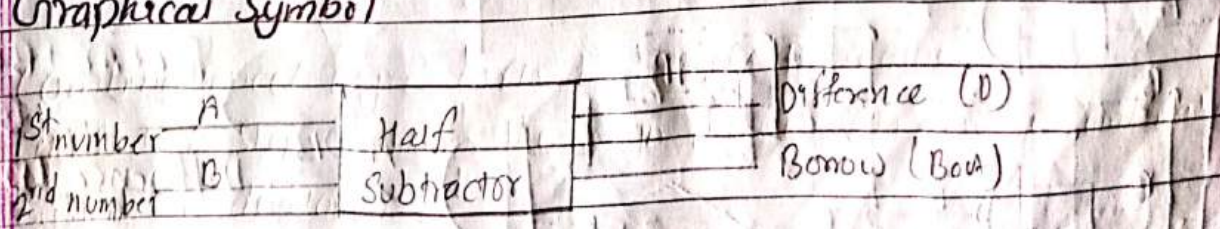
Half-Subtractor

The simplest combinational circuit that performs subtraction of two bit number is called Half-Subtractor.

It has two binary numbers 'A' and 'B' as inputs and derives the outputs borrow and difference.

Difference is the difference between two input numbers and while Borrow is the one that carries borrow if any. In both the inputs A is termed as Minuend and B is termed as Subtrahend.

Graphical Symbol



The truth table of Half-Subtractor can be derived as below

Truth Table

Inputs		Outputs	
A	B	Difference (D)	Borrow (Bout)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Using K-Map for simplified boolean expression for the difference and borrow.

Difference :-

A \ B	0	1
0		1
1	1	

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

$\therefore$ Difference $= A \oplus B$

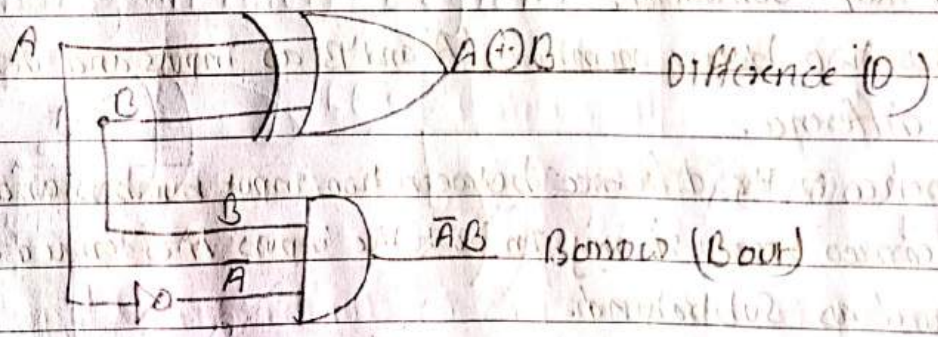
Borrow :-

A \ B	0	1
0		1
1		

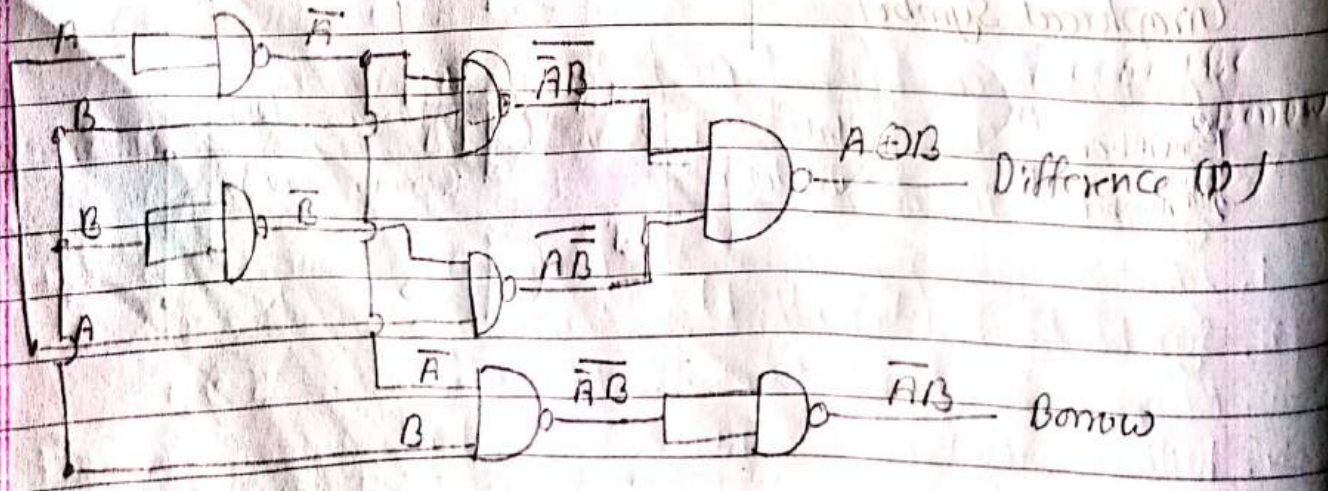
$\bar{A}B$

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

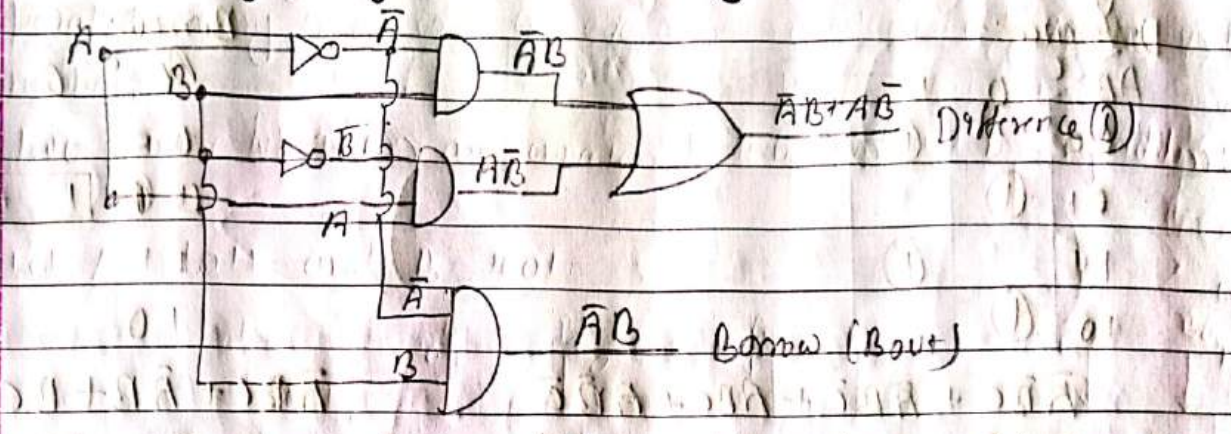
Implementing the boolean expression in circuit diagram



Implementing using NAND only



Implementing using Basic Gates only:



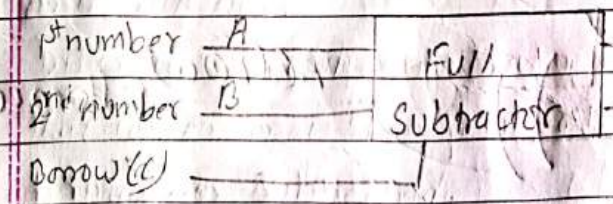
Full-Subtractor

The full subtractor is a combinational circuit which performs subtractions of three bits or input numbers.

Two of the inputs variables A and B represents two significant bits to be subtracted and C bit represents the borrow already taken by previous operation.

Two outputs from full subtractor are the difference and the Borrow to be taken from next operation.

Graphical Symbol



The truthtable of full-subtractor can be shown below

Truthtable

Inputs			Outputs	
A	B	C	Difference (D)	Borrow (Bout)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

DATE _____ PAGE _____

Using the k-Map Simplified boolean expression for the difference and Borrow

Difference :-

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

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

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

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

let $B \oplus C = x$

$\bar{A}x + A\bar{x}$

$A \oplus x$

putting the value of x

$A \oplus (B \oplus C)$

$\therefore$ Difference $= A \oplus (B \oplus C)$

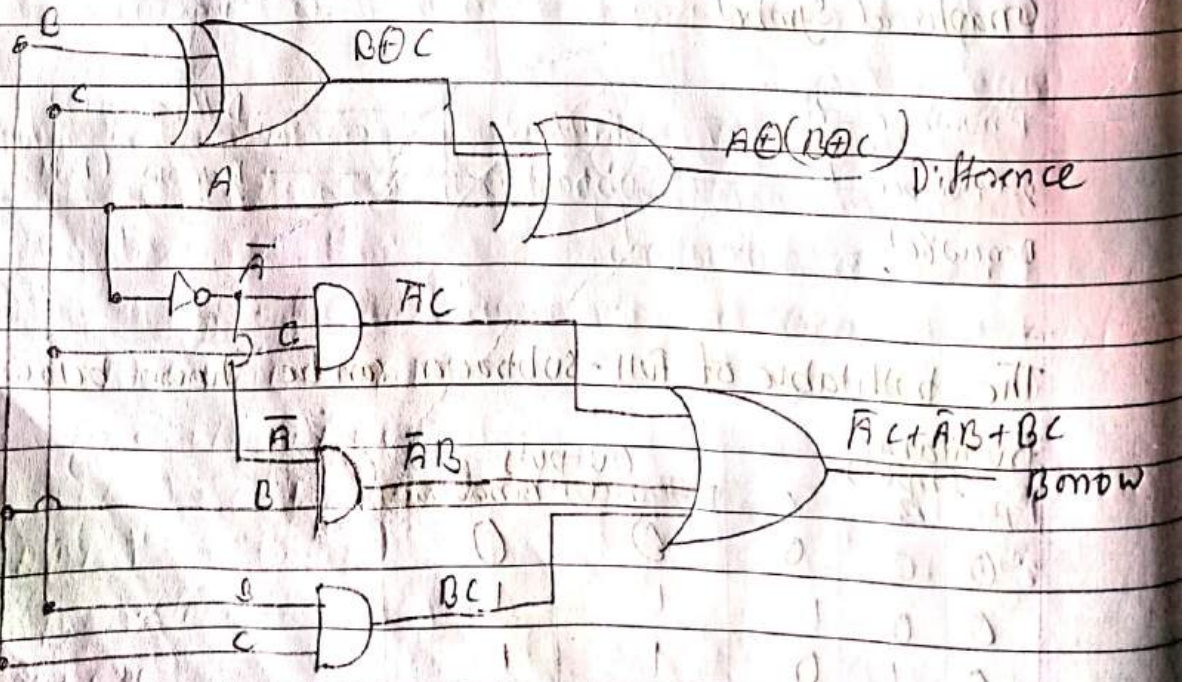
Borrow :-

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

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

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

Implementing in circuit Diagram



DATE _____ PAGE _____

Parity Generator and checkers

Truth

Parity Generator

Even parity Generator

Truth Table

Inputs			Outputs
A	B	C	Even parity
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

Simplifying the boolean expression using K-Map.

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

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

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

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

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

Let $A \oplus B = X$. Then

$\bar{C}X + C\bar{X}$

$C \oplus X$

Putting the value of X

$C \oplus (A \oplus B)$

$P_{even} = C \oplus (A \oplus B)$

implementing in circuit diagram

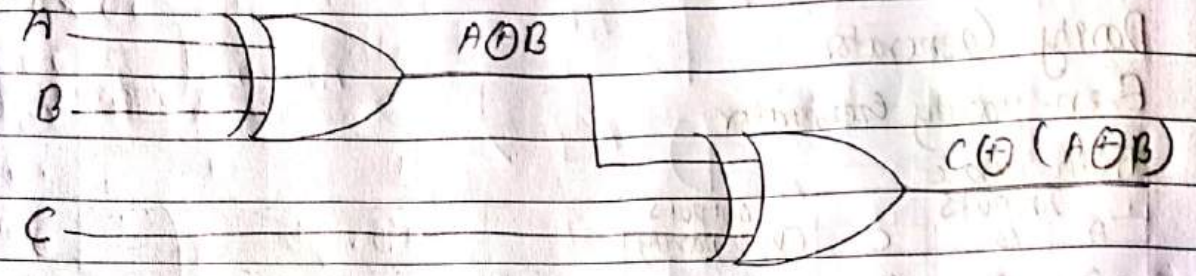


Fig:- Circuit for even parity generator

Odd Parity Generator

Truth Table:

Inputs			output
A	B	C	odd parity
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Simplifying the boolean expression using K-Map

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

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

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

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

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

Let $A \oplus B = x$, Then

$\bar{C}\bar{x} + Cx$

$C \oplus x$

putting the value of x

$C \oplus (A \oplus B)$ $\therefore P_{odd} = C \oplus (A \oplus B)$

Implementing the circuit diagram

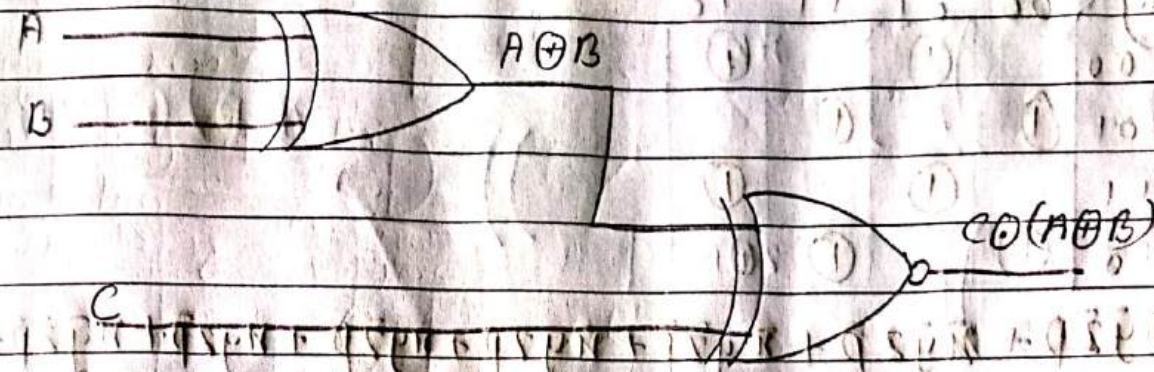


Fig:- Circuit for 3 bit parity generator

Parity checker

Even parity checker [1=error, 0=no error]

Truth Table

Received bits				checker (P)
x	y	z	p	
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

DATE _____ PAGE _____

Simplifying the boolean expression using K-Map.

xy \ zp	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

$\bar{x}\bar{y}\bar{z}p + \bar{x}\bar{y}z\bar{p} + \bar{x}y\bar{z}\bar{p} + \bar{x}yzp + xy\bar{z}p + xyz\bar{p} + x\bar{y}\bar{z}\bar{p} + x\bar{y}zp$

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

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

$(z \oplus p)(\overline{x \oplus y}) + (\overline{z \oplus p})(x \oplus y)$

Let $z \oplus p = A$ and, $x \oplus y = B$, Then

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

$A \oplus B$

putting the value of A and B

$(z \oplus p) \oplus (x \oplus y)$

Implementing the boolean expression in circuit diagram

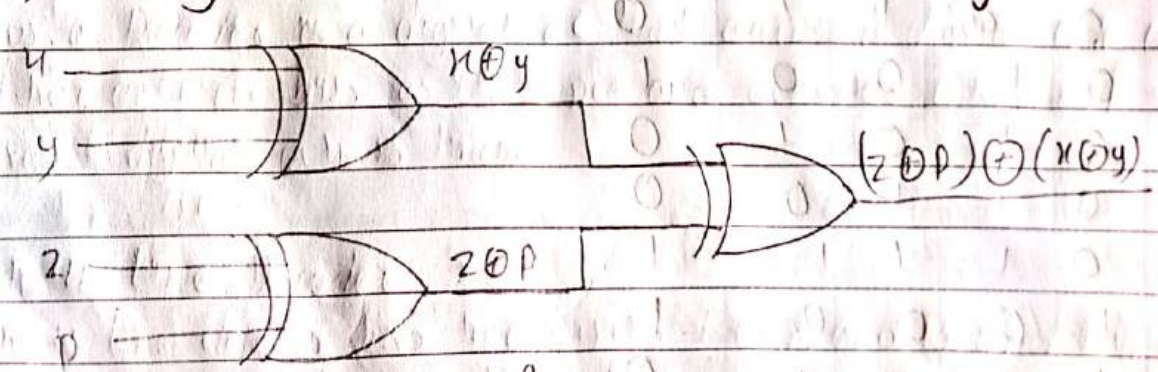


Fig:- Circuit for even parity checker

DATE _____ PAGE _____

Odd parity checker

Truth Table

Received bits				Checker p
x	y	z	p	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Simplifying the boolean expression using K-map

xy \ zp	00	01	11	10
00	1		1	
01		1		1
11	1		1	
10		1		1

$\bar{x}\bar{y}\bar{z}\bar{p} + \bar{x}\bar{y}z p + \bar{x}y\bar{z}p + \bar{x}yz\bar{p} + x\bar{y}\bar{z}\bar{p} + x\bar{y}zp + xy\bar{z}p + xyz\bar{p}$

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

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

$= (\overline{z \oplus p})(\overline{x \oplus y}) + (z \oplus p)(x \oplus y)$

Let $z \oplus p = A$ and $x \oplus y = B$

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

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

Putting the value of A and B

$(z \odot p) \odot (x \oplus y)$

Implementing in circuit diagrams:-

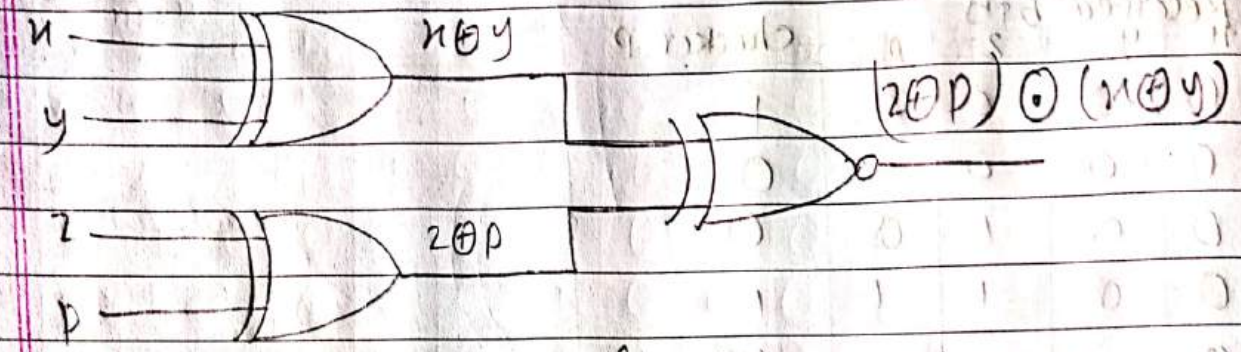


fig:- Circuit for odd parity checker

Decoder

A Decoder is a digital circuit that detects the presence of a specified combination of bits (code) on its input and indicates the presence of that port by specified output level.

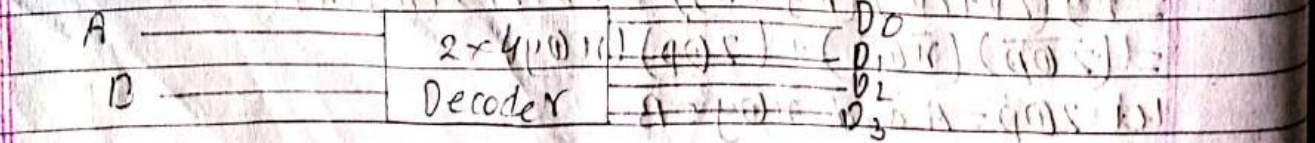
In other words, A Decoder is a combinational circuit that converts n bit binary input code into 2^n output such that each output line will be activated for only one of the possible combinations of the inputs.

Thus, a Decoder is a digital IC that accepts a digital code consisting of two or more bits as its input and activates only one of it's several digital outputs lines.

2 to 4 Line Decoder

2 to 4 line decoder has two inputs i.e. A and B and four outputs i.e. D_0 to D_3. Based on these two inputs combination, one of the four output line is activated at a time.

Graphical Symbol



The truth table of 2 to 4 line decoder

Truth Table

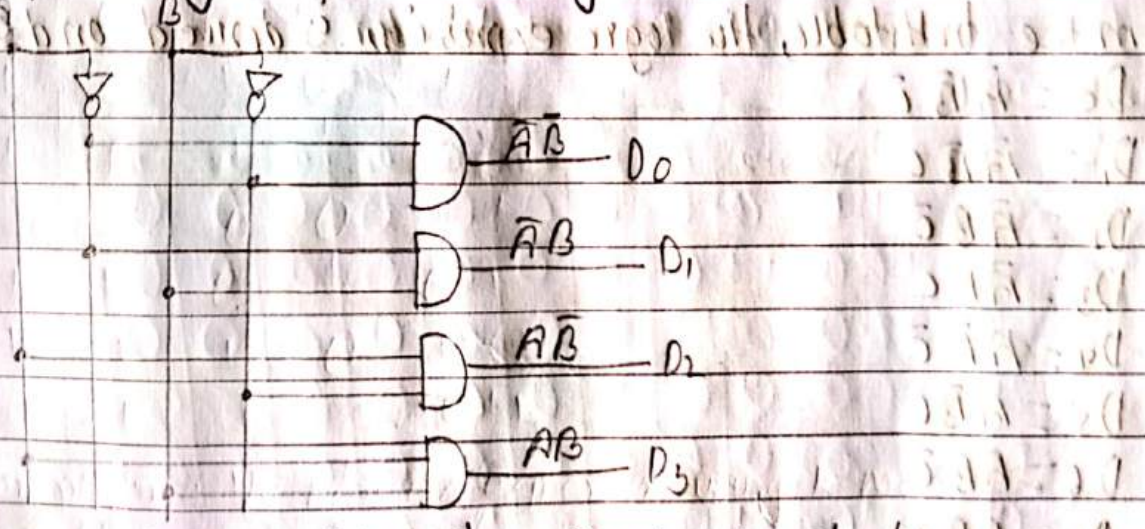
Inputs		Outputs			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Logical expression from the truth table is shown Below:-

$D_0 = \bar{A}\bar{B}$ $D_2 = A\bar{B}$

$D_1 = \bar{A}B$ $D_3 = AB$

Implementing in Circuit Diagram

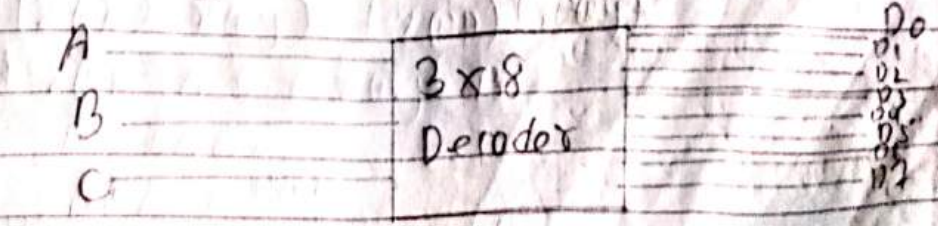


∴ Fig:- 2 to 4 line Decoder circuit diagram

3 to 8 line Decoder

3 to 8 line decoder has three inputs i.e. A, B and C and eight outputs i.e. D_0 to D_7. Based on these three inputs combination, one of the eight output line is activated at a time.

Graphical symbol



Truth Table of 3 to 8 line decoder:

Truth Table

Inputs			Outputs							
A	B	C	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

From the truth table, the logic expression is derived and shown below:

$D_0 = \bar{A}\bar{B}\bar{C}$

$D_1 = \bar{A}\bar{B}C$

$D_2 = \bar{A}B\bar{C}$

$D_3 = \bar{A}BC$

$D_4 = A\bar{B}\bar{C}$

$D_5 = A\bar{B}C$

$D_6 = AB\bar{C}$

$D_7 = ABC$

DATE _____ PAGE _____

Implementing in Circuit Diagram:-

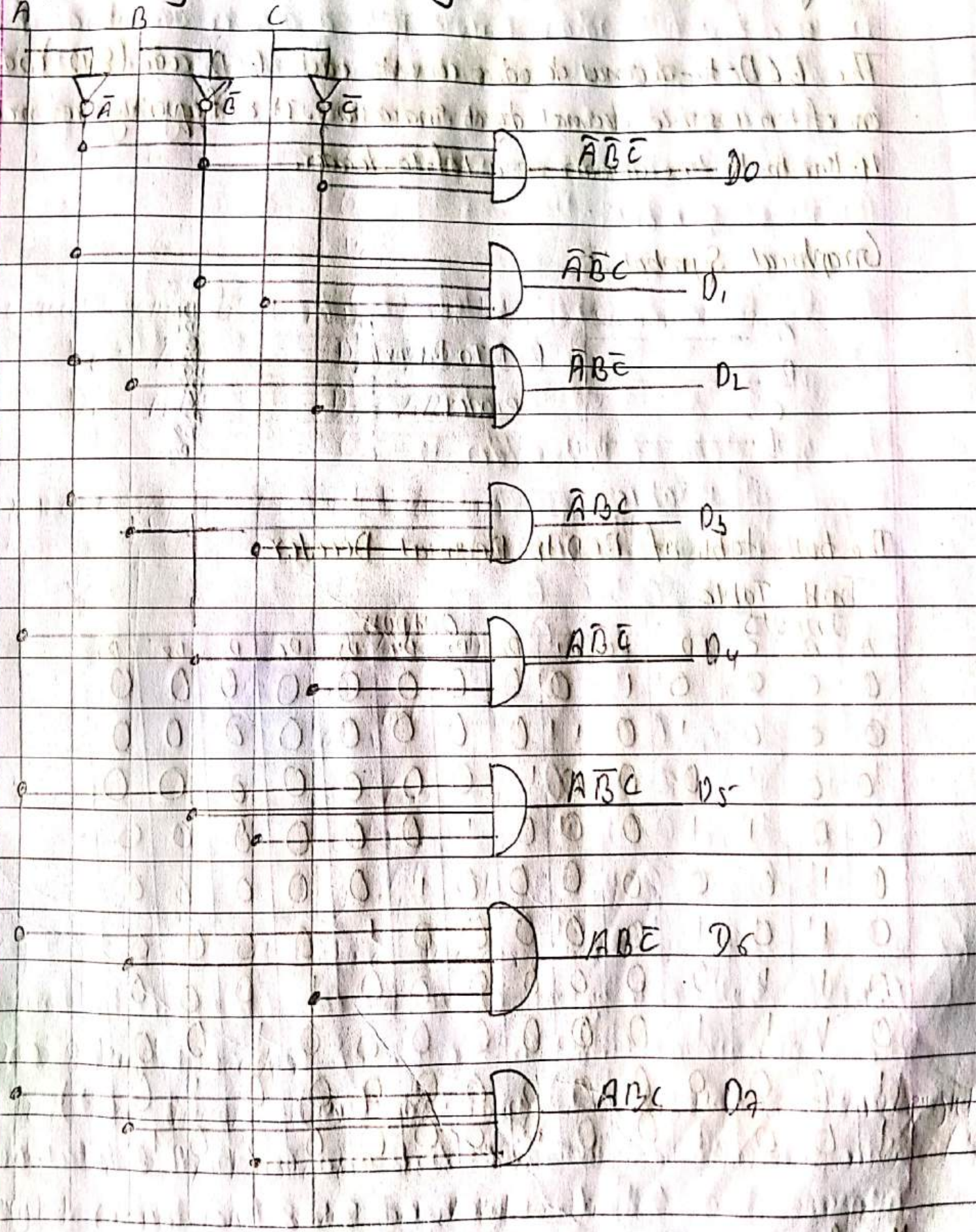
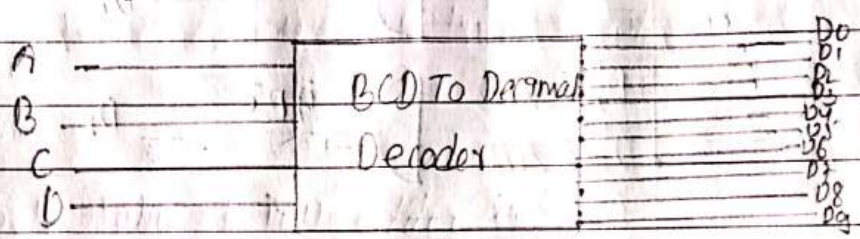


Fig: 3 to 8 Line Decoder Circuit Diagram

Design of BCD to Decimal Decoder

The BCD-to-decimal decoder converts each BCD code (8421 code) into one of ten possible decimal digit indications. It is frequently referred as a 4-line-to-10-line decoder or a 1-of-10 decoder.

Graphical Symbol :-



The truth table of BCD to Decimal Decoder

Truth Table

Inputs				Outputs									
A	B	C	D	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

Map for simplifying BCD to Decimal decoder expression :-

AB \ CD	00	01	11	10
00	D_0	D_1	D_3	D_2
01	D_4	D_5	D_7	D_6
11	X	X	X	X
10	D_8	D_9	X	X

$D_0 = \bar{A}\bar{B}\bar{C}\bar{D}$
$D_1 = \bar{A}\bar{B}\bar{C}D$
$D_2 = \bar{A}\bar{B}C\bar{D}$
$D_3 = \bar{A}\bar{B}CD$
$D_4 = \bar{A}B\bar{C}\bar{D}$
$D_5 = \bar{A}B\bar{C}D$
$D_6 = \bar{A}BC\bar{D}$
$D_7 = \bar{A}BCD$
$D_8 = A\bar{D}$
$D_9 = AD$

implementing the expression in circuit Diagram:-

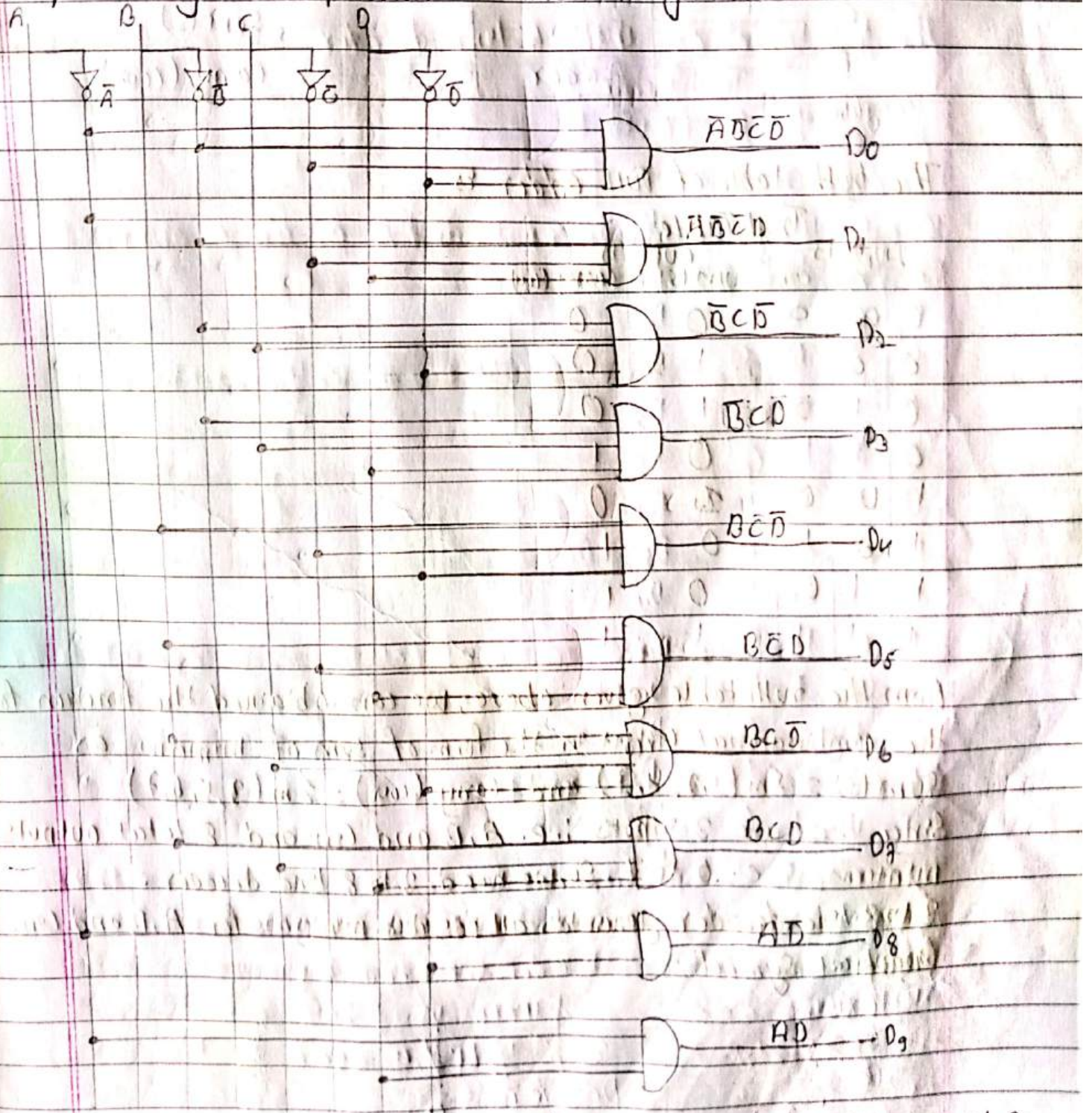


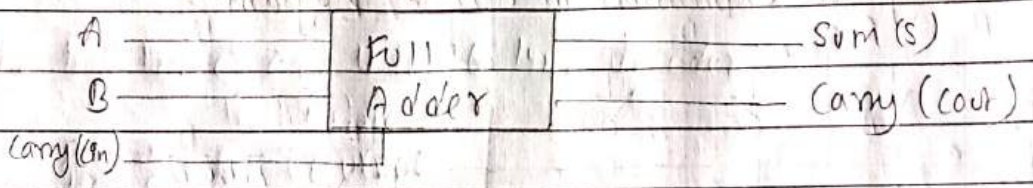
Fig:- Implementation of BCD to Decimal Decoder circuit Diagram

DATE _____ PAGE _____

Decoder with Full Adder

Implement a full adder circuit with a decoder and two OR Gates
A full adder is a combinational circuit that forms the sum of 3 input bits. It accepts two input bits and an input carry and generates a Sum output and an output carry.

Graphical symbol:-



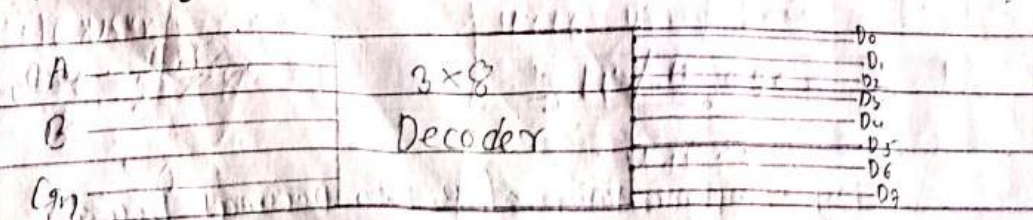
The truth table of Full adder is

Truth Table

Inputs			Outputs	
A	B	Cin	Sum (S)	Carry (Cout)
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

From the truth table given above, we can obtained the function for the combinational circuit in the form of sum of minterms as
Sum (S) = $\Sigma m(1,2,4,7)$ and Carry (Cout) = $\Sigma m(3,5,6,7)$
Since, these are 3 inputs i.e. A, B and Cin and 8 total outputs minterm i.e. 0 to 7. So, we need 3 to 8 line decoder.
3 to 8 line decoder generates all possible minterms for A, B and Cin.

Graphical Symbol:-



The OR Gates outputs Sum forms the sum of minterms (1,2,4,7) and the OR Gates output Carry forms the sum of minterms (3,5,6,7).

Implementing the circuit Diagram

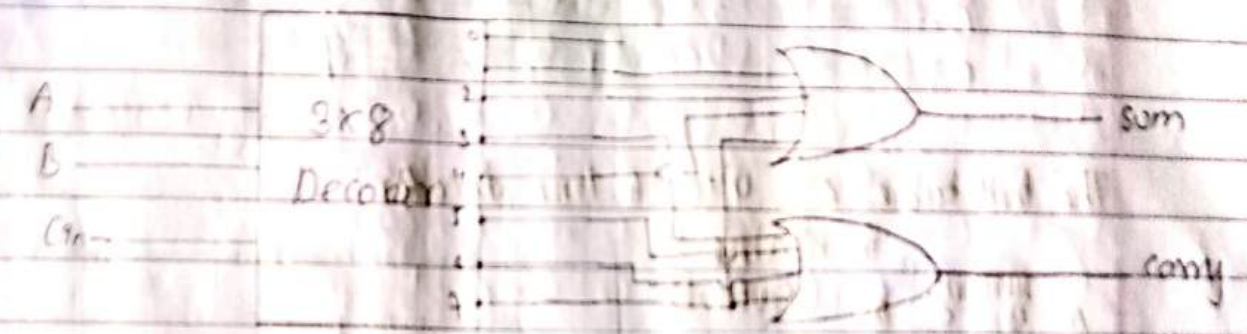


Fig:- Implementation of full adder with 3 Decoder

Enabler

An enabler is an input line to decide that the corresponding circuit will work or not.

The corresponding circuit will work only if the enabler is high/positive 1/True.

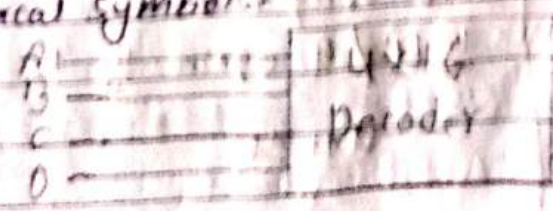
It is similar to Adding each output with the value that of enabler.

Higher Order Decoder Using lower Order Decoder

4×16 Decoder Using 3×8 Decoder

The 4 to 16 line decoder has four inputs i.e. A, B, C and D and Sixteen outputs i.e. D0 to D15 Based on the four inputs combination, One of the sixteen output line is activated at a time.

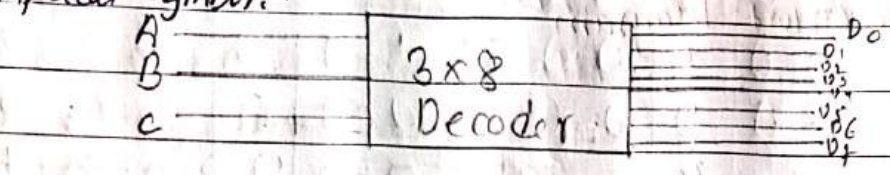
Graphical Symbol:



DATE _____ PAGE _____

The 3 to 8 line decoder has three inputs i.e. A, B and C and eight outputs i.e. D_0 to D_7. Based on this three inputs combination, one of the eight output line is activated at a time.

Graphical Symbol:-



The truth table of 4 to 16 line decoder:-

Truth Table

Inputs				Outputs
A	B	C	D	
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
0	0	1	1	D_3
0	1	0	0	D_4
0	1	0	1	D_5
0	1	1	0	D_6
0	1	1	1	D_7
1	0	0	0	D_8
1	0	0	1	D_9
1	0	1	0	D_{10}
1	0	1	1	D_{11}
1	1	0	0	D_{12}
1	1	0	1	D_{13}
1	1	1	0	D_{14}
1	1	1	1	D_{15}

* We can find number of decoder we need by

4x16 using 3x8

$\frac{16}{8} = 2 < 8$

$\therefore 2$

4x16 using 2x4

$\frac{16}{4} = 4 = 4$ again $\frac{4}{4} = 1$

$\therefore 4+1 = 5$

Using B, C, D as an input and A as an Enable. Then, implementing in circuit Diagram.

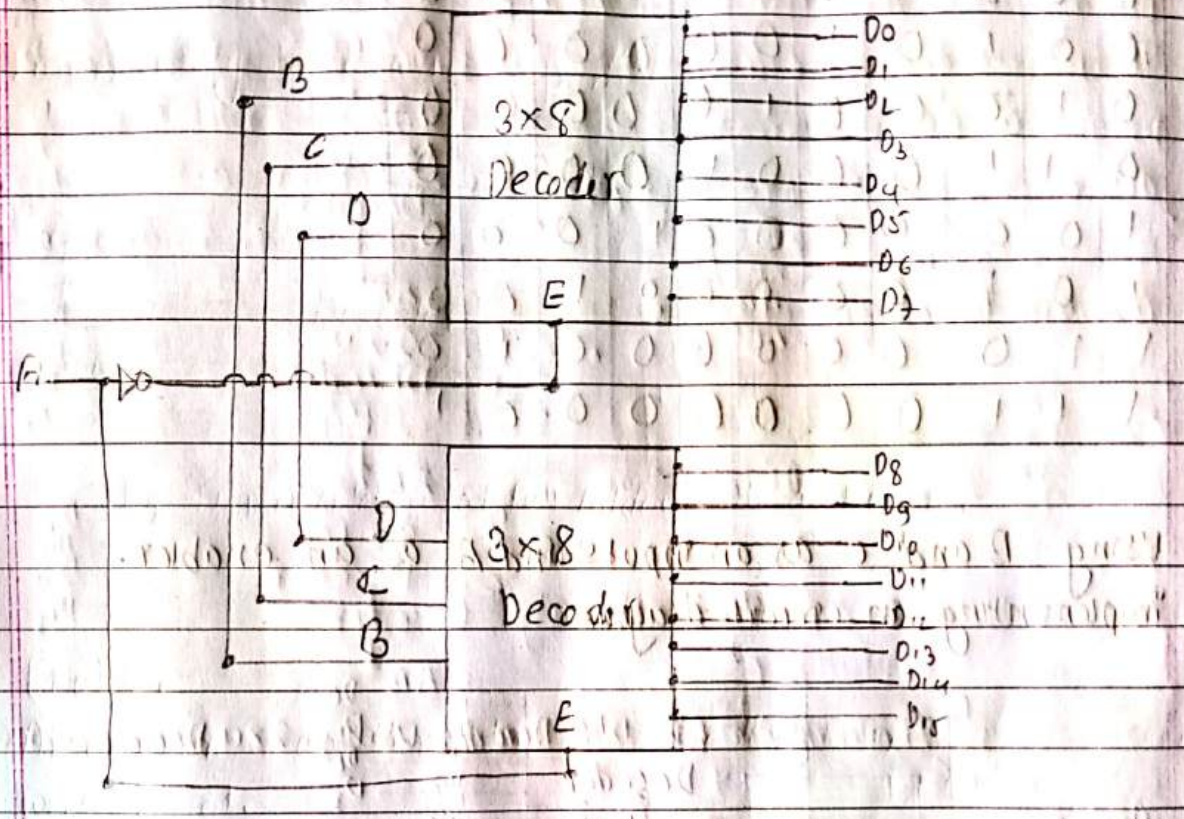
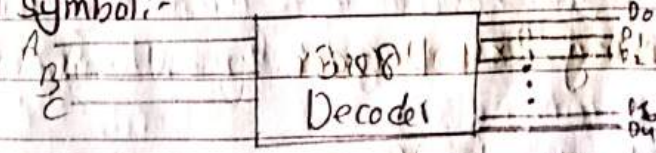


Fig:- Implementation of 4x16 Decoder using 3x8 Decoder circuit diagram

3x8 Decoder using 2x4 Decoder

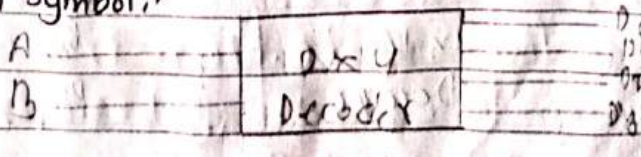
The 3 to 8 line decoder has three inputs i.e. A, B and C and eight outputs i-e. D_0 to D_7.

Graphical symbol:-



The 2 to 4 line decoder has two inputs, i.e. A and B and four outputs i.e. D_0 to D_3. Based on this two inputs combination one of the four output line is activated at a time.

Graphical symbol:-



Truthtable of 3 to 8 line Decoder:-

Inputs			Outputs							
A	B	C	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

Using B and C as an inputs and A as an enabler. implementing in circuit diagram

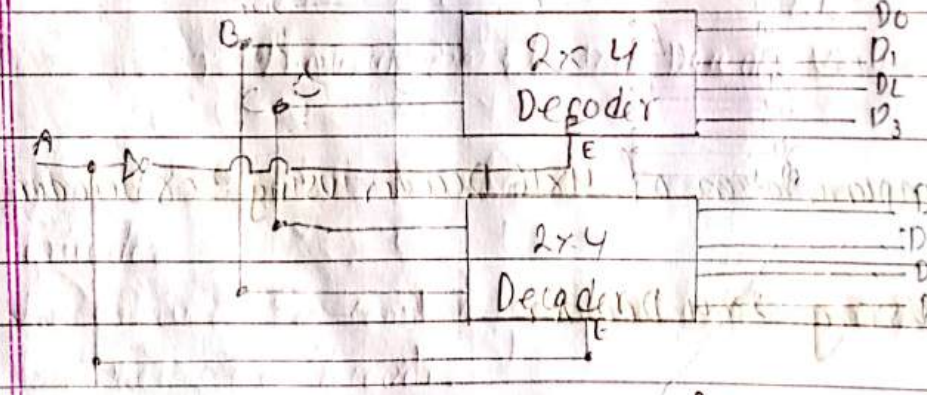


Fig:- Implementation of 3×8 Decoder using 2×4 Decoder Circuit Diagram

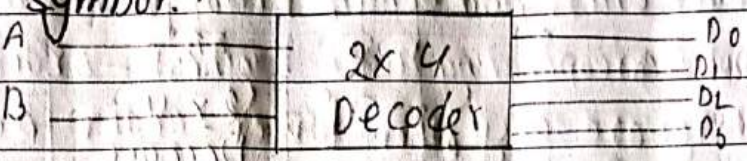
4×16 Decoder using 2×4 Decoder

The 4 to 16 line decoder has four inputs i.e. A, B, C and D and sixteen outputs i.e. D_0 to D_{15}. Based on these four inputs combination, one eight output line is activated at a time.

Graphical Symbol:-

The 2 to 4 line decoder has two inputs i.e. A and B and four outputs i.e. D_0 to D_3. Based on this two inputs combination, one of the four outputs line is activated at a time.

Graphical symbol:-



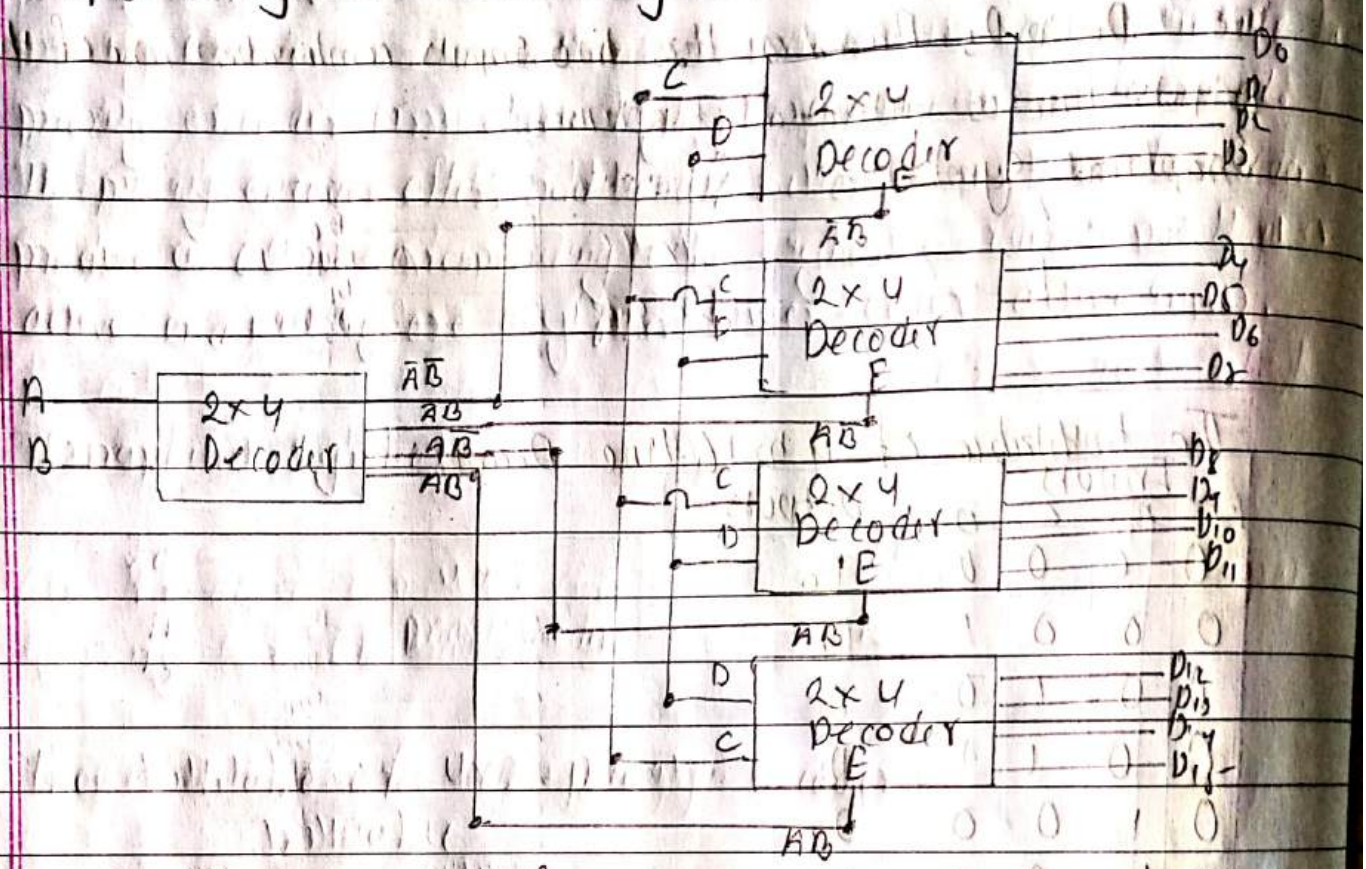
The truth table of 4 to 16 line Decoder:

Inputs				Outputs
A	B	C	D	
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
0	0	1	1	D_3
0	1	0	0	D_4
0	1	0	1	D_5
0	1	1	0	D_6
0	1	1	1	D_7
1	0	0	0	D_8
1	0	0	1	D_9
1	0	1	0	D_{10}
1	0	1	1	D_{11}
1	1	0	0	D_{12}
1	1	0	1	D_{13}
1	1	1	0	D_{14}
1	1	1	1	D_{15}

To use 2x4 Decoder in the form of 4x16 Decoder, This total five 2x4 decoder is required when 4 decoder have C and D as a inputs and A, B as a enabler. The one decoder has A and B inputs and the output is used as a enabler to other 4 decoders

DATE _____ PAGE _____

Implementing in Circuit Diagram



∴ Fig:- Implementation of 4x16 Decoder using 2x4 Decoder circuit Diagram

Encoder

An encoder is a combinational logic circuit that produces a reverse operation from that of a decoder.

An encoder has 2^n input lines and n output lines.

The output lines generates the binary code for the 2^n input variables.

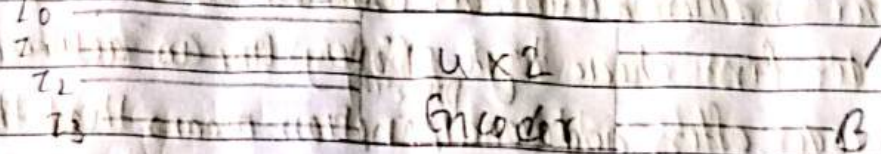
Thus, The encoder is a combinational logic circuit that converts an active input signal into a coded output signal. Hence, the opposite of Decoding process is called Encoding.

An encoder has a number of input lines where only one of the activated lines At the output, it display the value corresponding to a activated input.

4 to 2 Encoder

4 to 2 Encoder has four input i.e. I_0, I_1, I_2 and I_3 and two output i.e. A and B. Based on these input signal, an activated signal is converted into a coded output signal.

Graphical symbol:



The truth table of 4x2 Encoder,

Truth Table

Inputs				Outputs	
I_0	I_1	I_2	I_3	A	B
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

For the boolean expression

$$A = \bar{I}_0 \bar{I}_1 I_2 \bar{I}_3 + \bar{I}_0 I_2 \bar{I}_2 I_3$$

$$B = \bar{I}_0 I_1 \bar{I}_2 \bar{I}_3 + \bar{I}_0 \bar{I}_2 \bar{I}_2 I_3$$

Implementing in Circuit Diagram

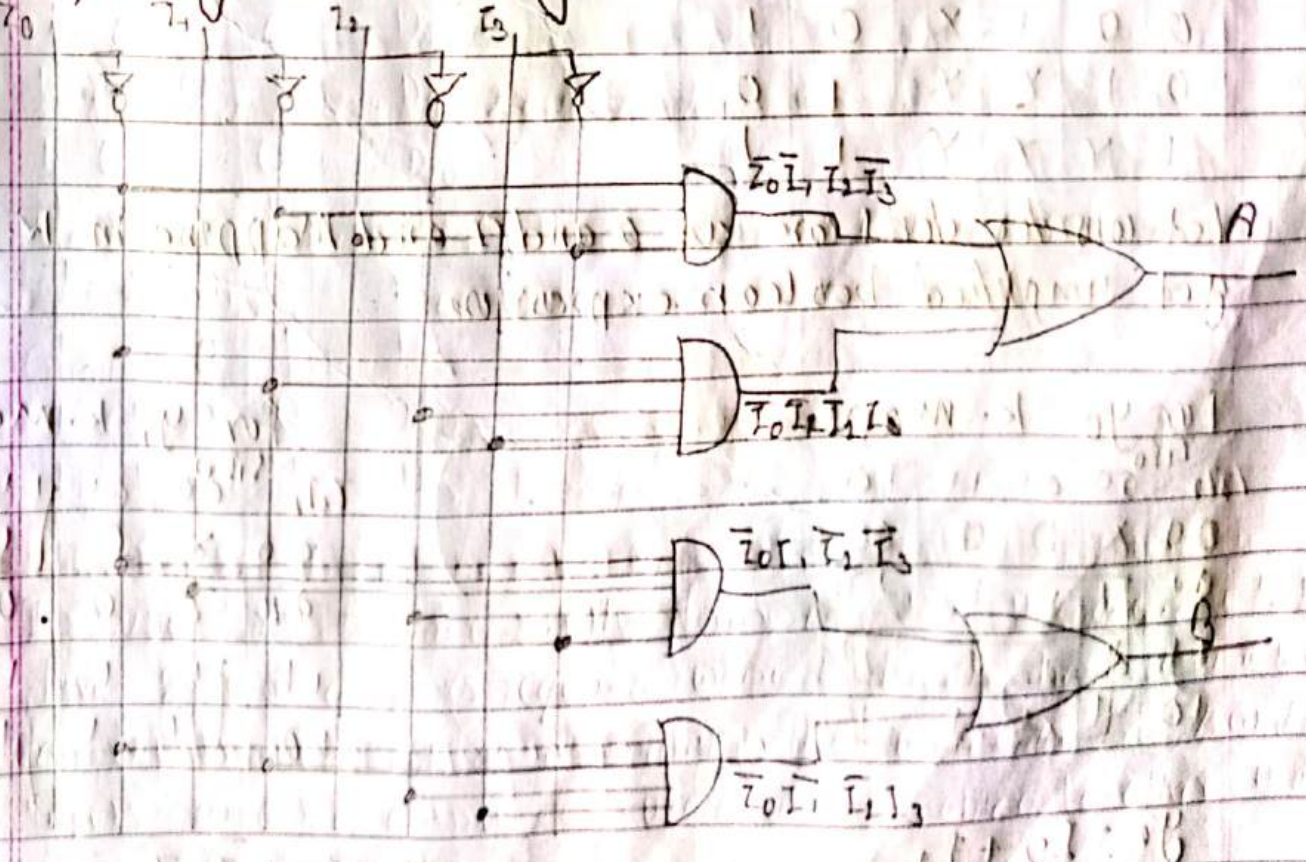
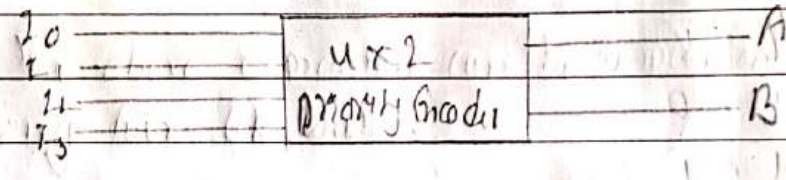


Fig:- Implementation of 4x2 Encoder

Priority Encoder

Priority Encoder and normal encoder works with same principle but priority encoder first check high priority line is active or not. The priority encoder offers additional function means that the encoder will produce a output corresponding to the highest - order input that is active and will ignore any other lower - order active inputs.

Graphical Symbol



The truth table of 4x2 priority Encoder

Truth Table

Inputs				Outputs	
I_3	I_2	I_1	I_0	y_0	y_1
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

Let consider don't cox as 0 and 1 and mapping in K-Map to get simplified boolean expression

For y_0 K-Map

I_3I_2 \ I_1I_0	00	01	11	10
00	x	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$y_0 = I_2 + I_3$

For y_1 K-Map

I_3I_2 \ I_1I_0	00	01	11	10
00	x	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$y_1 = I_3 + \bar{I}_2 I_1$

Implementing in circuit Diagram

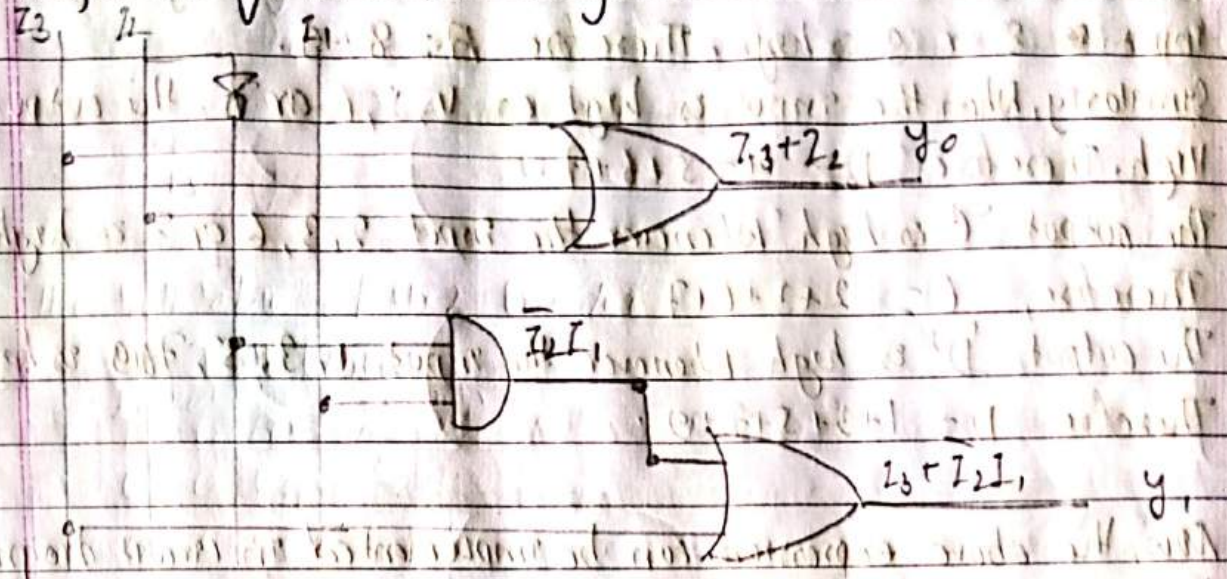
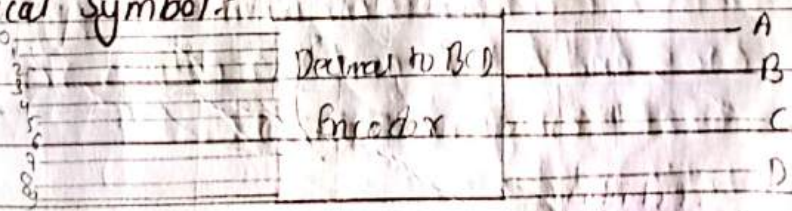


Fig:- Implementation of 4x2 priority Encoder

Decimal-to-BCD Encoder

Decimal-to-BCD Encoder is one which have ten inputs corresponding to Decimal 0 to 9 and four outputs representing the binary coded value of these given inputs.

Graphical Symbol:



The truth table of Decimal to BCD Encoder is shown Below

Truth Table

Decimal Inputs										BCD outputs			
0	1	2	3	4	5	6	7	8	9	A	B	C	D
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

In the truth table, It is seen that the output 'A' is high whenever the input 8 or 9 is high. Therefore A = 8+9.
Similarly, when the input is high on 4, 5, 6 or 7, the output 'B' is High. Therefore B = 4+5+6+7
The output 'C' is high whenever the input 2, 3, 6 or 7 is high.
Therefore, C = 2+3+6+7
The output 'D' is high whenever the input 1, 3, 5, 7 or 9 is high.
Therefore D = 1+3+5+7+9.

Now, the above expression can be implemented in circuit diagram by using four OR Gates.

Implementing in a Circuit Diagram

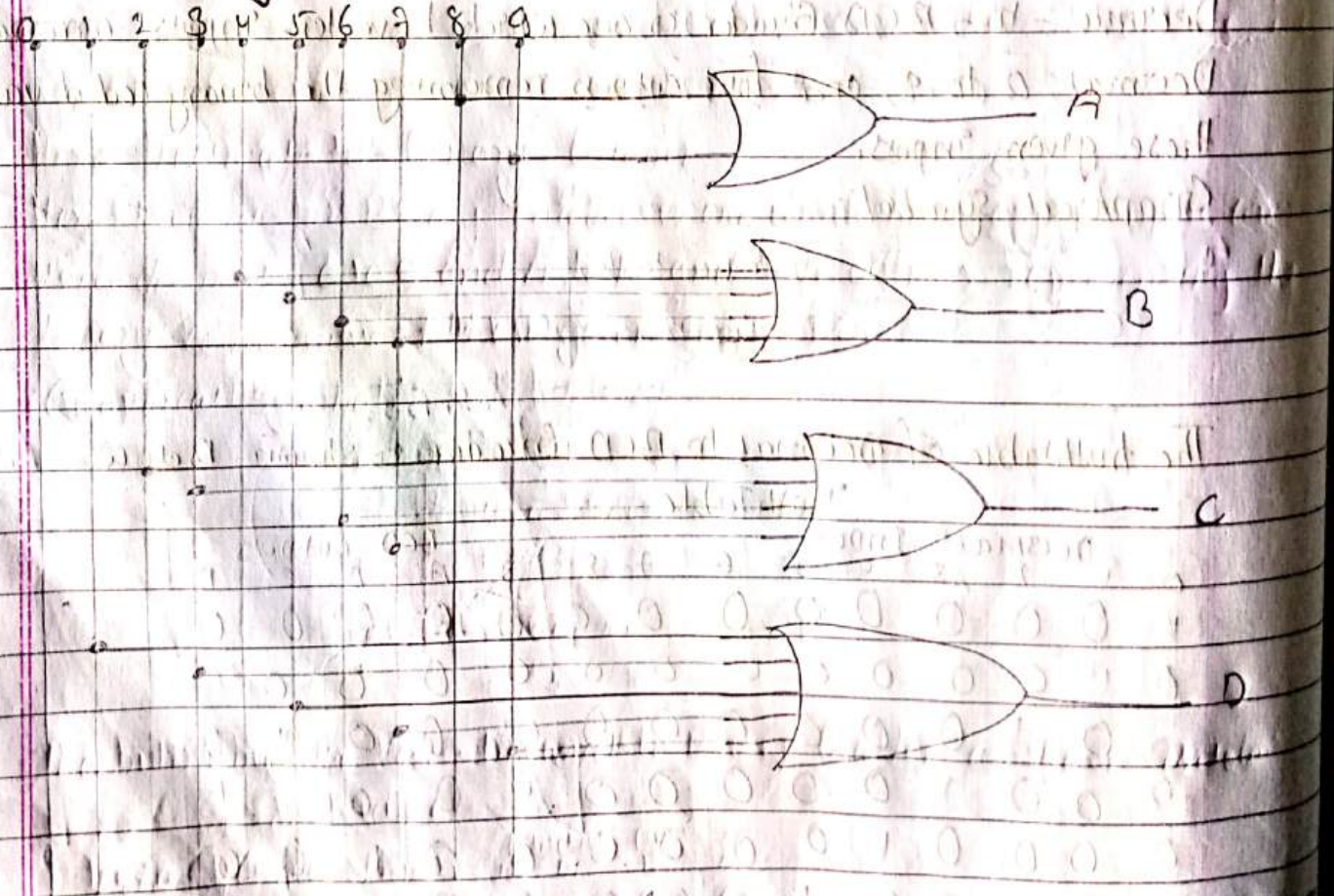


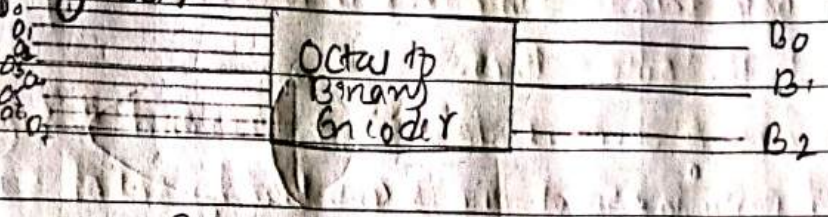
Fig. Implementation of Decimal to BCD Encoder circuit Diagram

DATE _____ PAGE _____

Octal-to-Binary Encoder

Octal-to-Binary Encoder is one which have eight inputs corresponding to octal 0 to 7 and three outputs representing the binary coded value of these given inputs.

Graphical Symbol:-



The truthtable of Octal-to-Binary Encoder is shown Below:-

Truth Table

Octal Inputs								Outputs		
O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	B_2	B_1	B_0
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

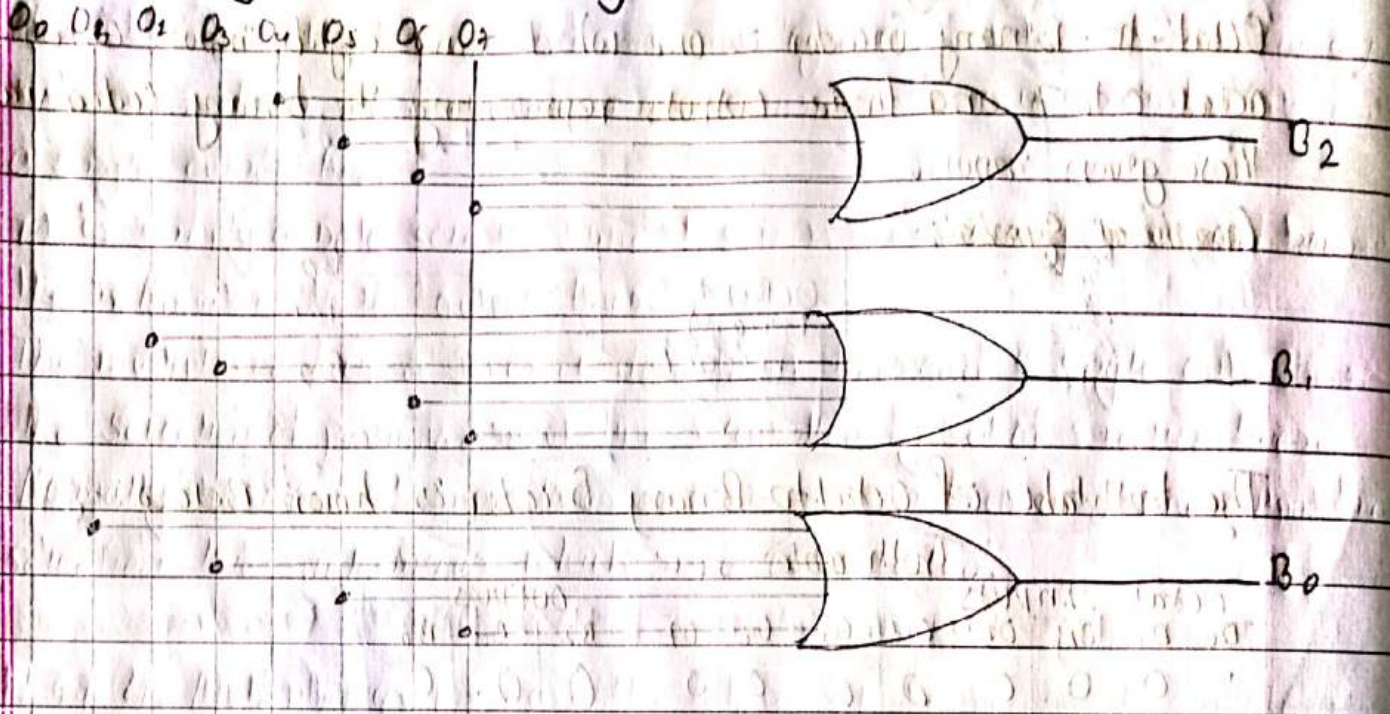
In the above truthtable we can observe that the output of 'B_2' is high whenever the input O_4, O_5, O_6 or O_7 is high. Therefore, $B_2 = O_4 + O_5 + O_6 + O_7$

The output of 'B_1' is high whenever the input of O_2, O_3, O_6 or O_7 is high. Therefore $B_1 = O_2 + O_3 + O_6 + O_7$

The output of 'B_0' is high whenever the input of O_1, O_3, O_5 or O_7 is high. Therefore $B_0 = O_1 + O_3 + O_5 + O_7$

By using the above expression, the octal-to-Binary encoder can be implemented by using four input 3 OR Gate.

Implementing in Circuit Diagram

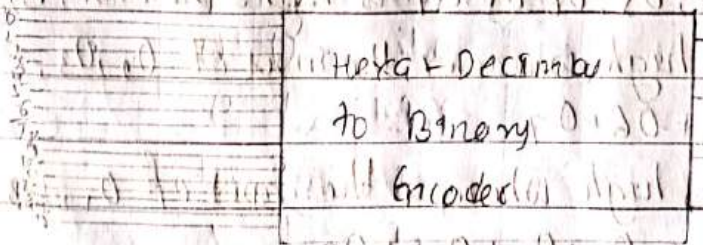


∴ Fig:- Implementation of Octal-to-Binary Encoder Circuit Diagram.

Hexa-Decimal-to-Binary Encoder

Hexa-Decimal-to-Binary Encoder is one which have ~~eight~~ sixteen inputs corresponding Hexa-decimal 0 to 9 and A to F and four outputs representing the binary coded value of those given inputs.

Graphical Symbol:-



The truth table for Hexa-Decimal to Binary Encoder as shown Below:-

Truth Table

Hexa-Decimal Inputs	Binary Outputs B_3	B_2	B_1	B_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0

6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

In the truth table, It can observe that the output of 'B_3' is high whenever the input are 8, 9, A, B, C, D, E, or F. Therefore, $B_3 = 8+9+A+B+C+D+E+F$

The output of 'B_2' is high whenever the input are 4, 5, 6, 7, C, D, E or F

Therefore $B_2 = 4+5+6+7+C+D+E+F$

The output of B_1 is high whenever the input are 2, 3, 6, 7, A, B, E and F.

Therefore $B_1 = 2+3+6+7+A+B+E+F$

The output of B_0 is high whenever the inputs are 1, 3, 5, 7, 9, B, D or F. Therefore

$B_0 = 1+3+5+7+9+B+D+F$

By using the above expression, the [illegible] decimal to Binary encoder can be implemented by using eight input 4 OR Gate.

Implementing in circuit Diagram

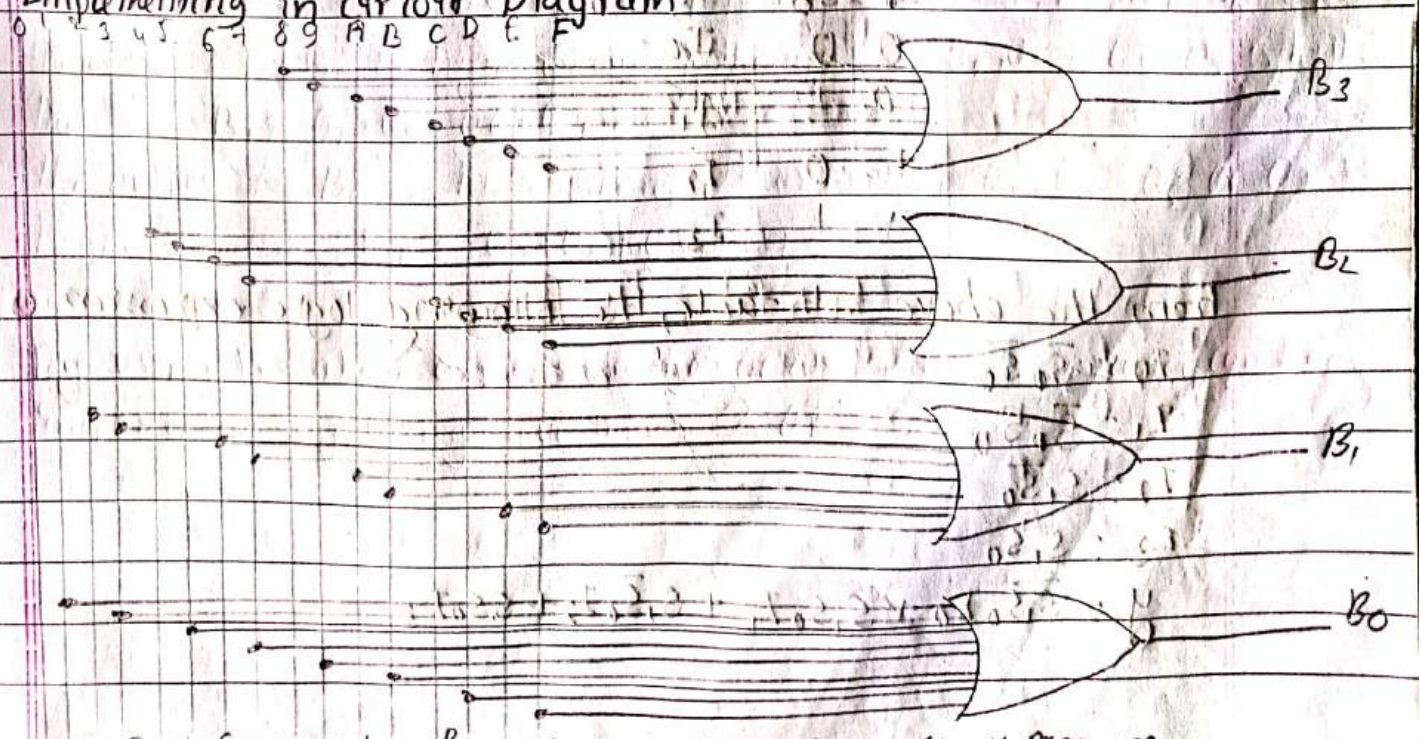


Fig:- Implementation of [illegible] Binary Encoder circuit Diagram

DATE

Multiplexers (Mux)

A multiplexers (Mux) is a combinational logic circuit that selects one digital information from several sources and transmits the selected information into a single output line.

It is simply a Data selector since it selects one of many inputs and transmits the information to the output lines.

The multiplexer has a several data input lines and a single output lines.

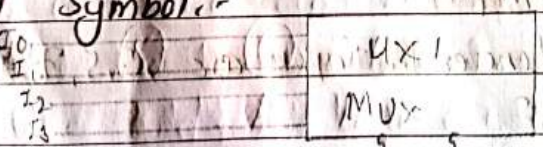
The selection of particular input line is controlled by set of selection lines.

Normally, there are 2^n input lines and n selection lines and one output line. The selection lines determines which input is selected.

4:1 of 4 Multiplexer

4:1 of 4 multiplexer is one of the combinational logic circuit which have four input line i.e. I_0, I_1, I_2 and I_3 and only one outline i.e. y and two selection line i.e. S_0 and S_1.

Graphical Symbol:-



The truthtable with (4x1) Mux is shown below:-

Mux Truthtable

Inputs S_1	Inputs S_0	output y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

From the above truth table, the following logic expression is derived

$I_0 = \bar{S}_1 \bar{S}_0$

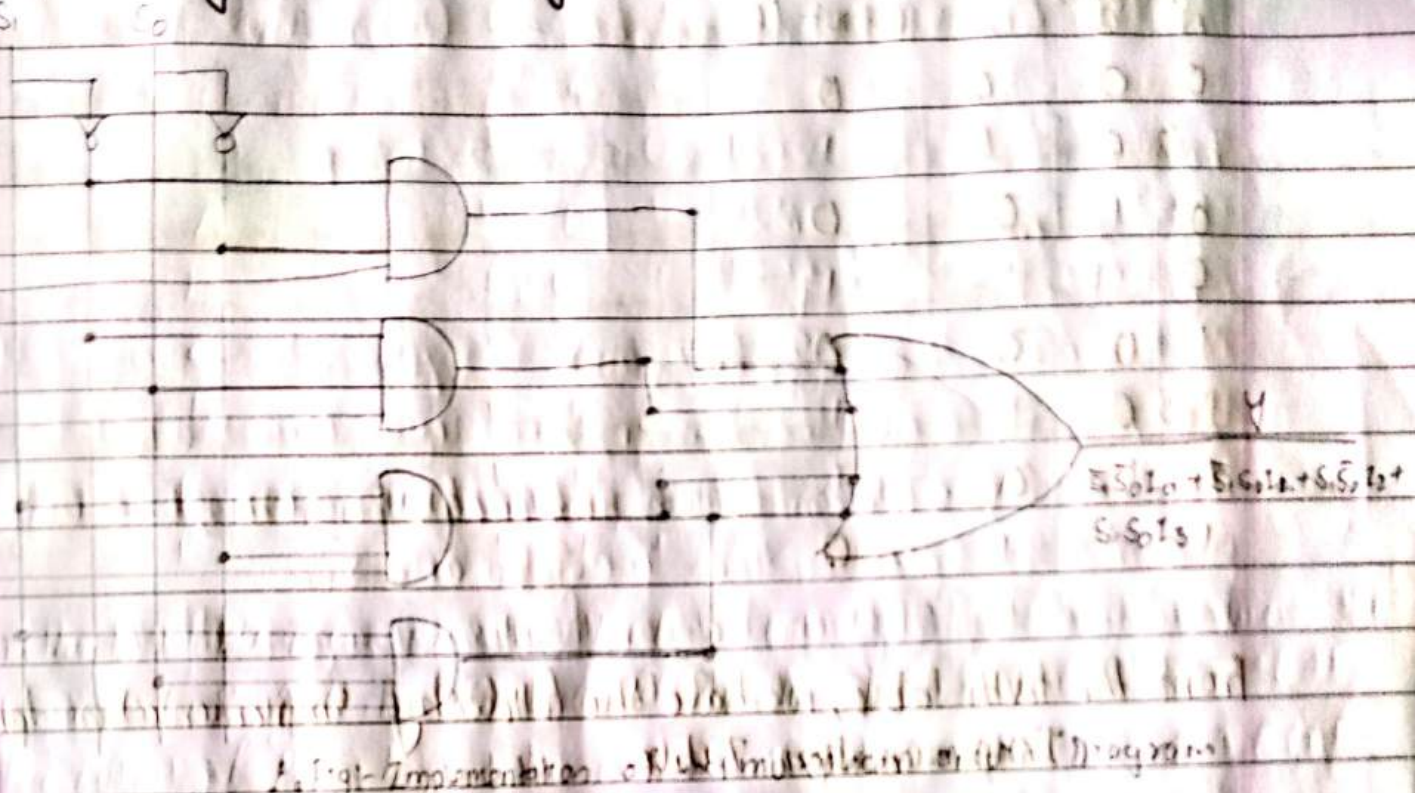
$I_1 = \bar{S}_1 S_0$

$I_2 = S_1 \bar{S}_0$

$I_3 = S_1 S_0$

$y = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$

implementing in circuit Diagram



Boolean function implementation using Multiplexers

Implement the given boolean function using 4x1 Multiplexers.

$F(A,B,C) = \Sigma(1,3,5,6)$

There is a three variable function and therefore we need a multiplexers with two selection lines and four inputs which is to apply variable B and C to the selection lines and A as input variable. The input variable are chosen to be A or $\bar{A}$ or 0 or 1.

Thus, function table and Truth table are shown below:-

Function Table

Inputs B	C	output (F)
0	0	I_0
0	1	I_1
1	1	I_2
1	0	I_3

Truth Table

Inputs A	B	C	output (F)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

From the truth table, we have the following implementation table:-

Implementation Table

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

From the implementation table, we have the following implementation circuit:-

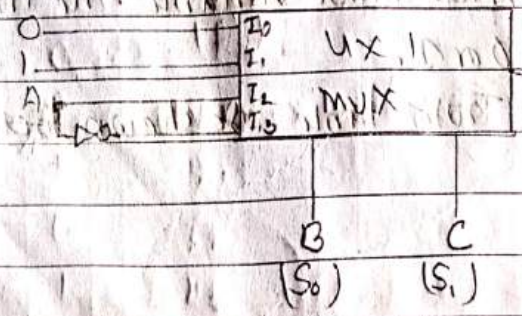


Fig:- Boolean Function implementation circuit.

Implementation of given boolean function using multiplexers and taking C as input.

$F(A,B,C) = \Sigma(1,3,5,6)$

This is a three variable function and therefore we need a multiplexers with two selection lines and four inputs which is to apply variable A and B to the selection lines and C as input variable. The input are chosen to be C or $\bar{C}$ or 0 or 1.

Thus, Function table and Truth Table are shown as given below:-

Function Table

Inputs A	Inputs B	Output (F)
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Truth Table

Inputs A	Inputs B	Inputs C	Output (F)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

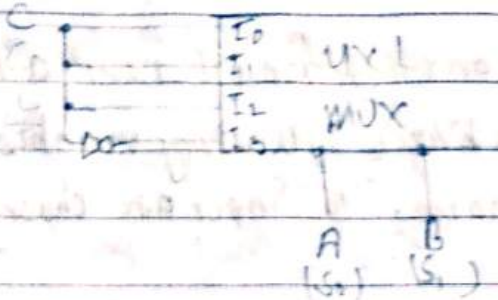
From the truth table, we have the following implementation table:-

Implementation table

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

From the implementation table, we have following implementation circuit:

MUX implementation



∴ Fig: Boolean function implementation (cont'd)

Implementation of given boolean function using multiplexers :-

$F(A,B,C,D) = \Sigma(0,1,3,4,8,9,15)$

This is a four variable function and therefore we need a multiplexer with three selection lines and eight inputs which is to apply variable A, B and C to the selection lines and D as input variable. They are chosen to be D or $\bar{D}$ or 0 or 1.

Thus, Function and Truth table are shown as given below :-

Function Table

Input			Output (F)
A	B	C	
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

DATE _____ PAGE _____

Truth Table

Input A	B	C	D	output (F)
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

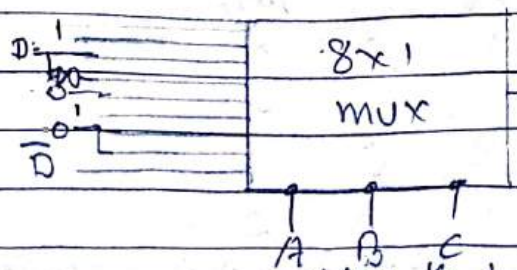
From the truth table, we have the following implementation table:-

Implementation table

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

From the implementation table, we have following implementation circuit

MUX Implementation



Prepared By
Santosh Bhandari

DATE _____ PAGE _____

Mux Tree
(Higher order Mux using Lower Order Mux)

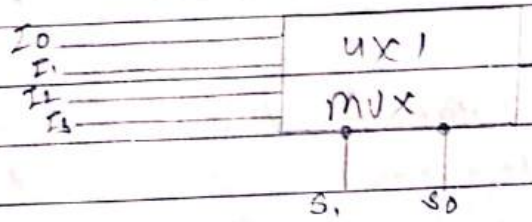
Implement that 4x1 mux using 2x1 MUX

4x1

$\frac{4}{2} = 2$, $\frac{2}{2} = 1$ $\quad 2+1=3$

∴ This mean that 3, 2x1 mux is required to implement 4x1 mux 4x1 mux is one of the combinational logic circuit which have eight four input lines i.e. I_0, I_1, I_2, and I_3 and only one output line i.e y and two selection line i.e. S_0 and S_1.

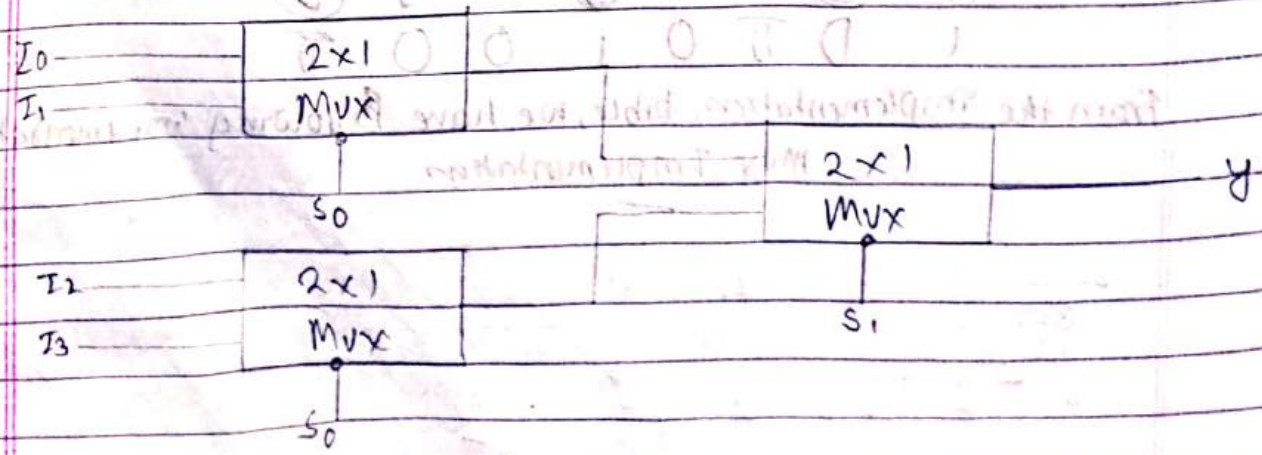
Graphical Symbol:-



Truth Table

Input S_1	Input S_0	Output (y)
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

To implement 4x1 mux using 2x1 mux, $2+1=3$, 2x1 mux is required



∴ Fig:- Implementation of 4x1 MUX using 2x1 MUX circuit

DATE _____ PAGE _____

Implement 8x1 Mux using 2x1 Mux

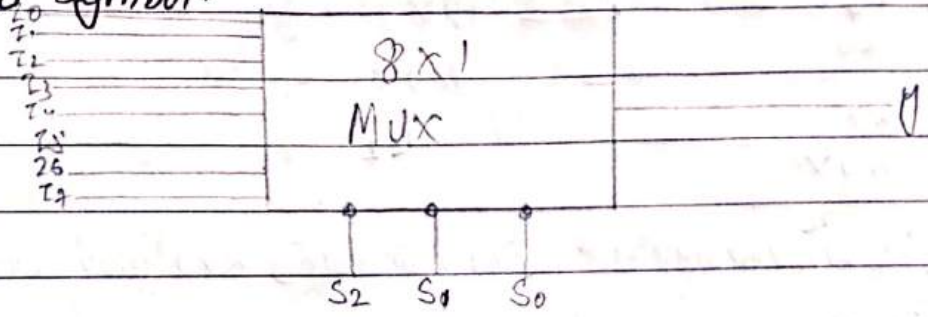
8x1

$\frac{8}{2}=4$, $\frac{4}{2}=2$, $\frac{2}{2}=1$, $4+2+1=7$

∴ This mean that 7, 2x1 mux is required to implement 8x1 mux

8x1 mux is one of the logic circuit which have eight input lines i.e. $I_0, I_1, I_2, I_3, I_4, I_5, I_6$ and I_7 and only one output line i.e. Y and three selection lines i.e. S_0 and S_1.

Graphical symbol:-



Truth Table

Input			Output (Y)
S_2	S_1	S_0	
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

To implement 8x1 mux using 2x1 mux. $4+2+1=7$, 2x1 mux is required where 4, 2x1 mux has S_0 selection line, 2, 2x1 mux has S_1 selection line and 1, 2x1 mux has S_2 selection.

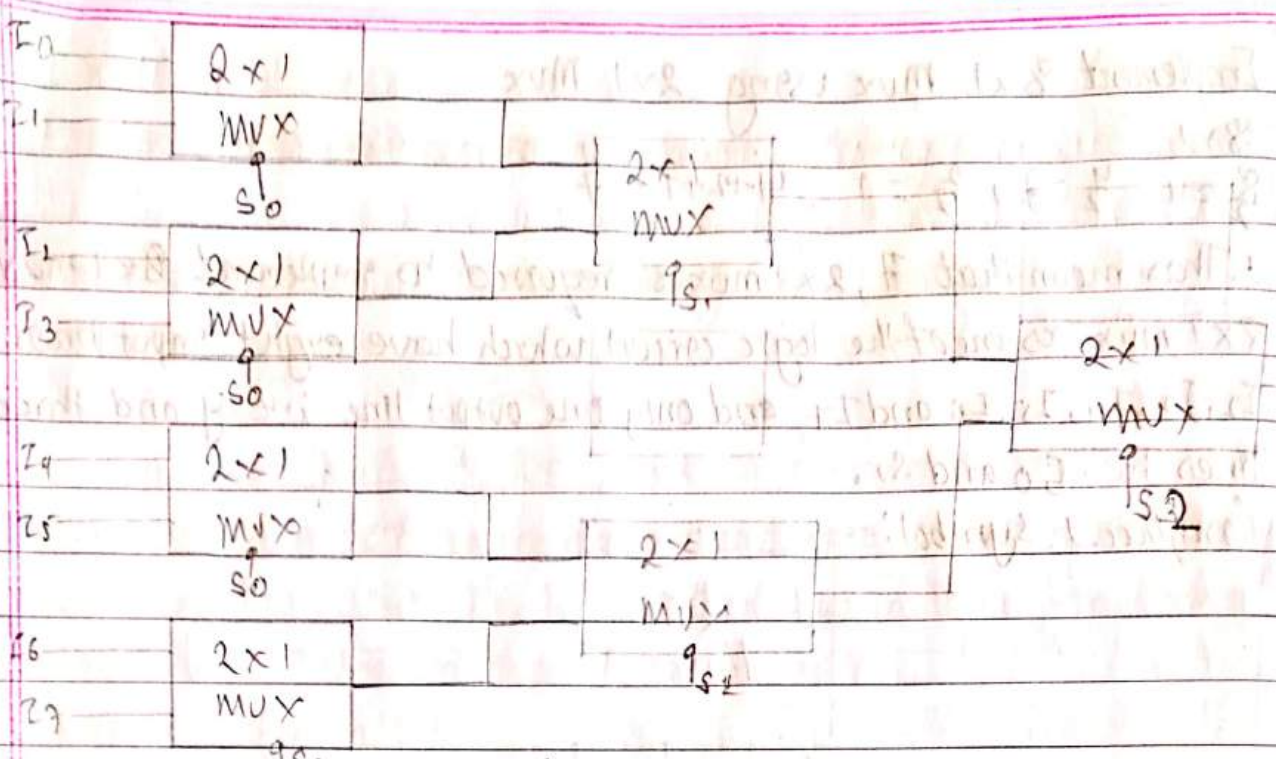
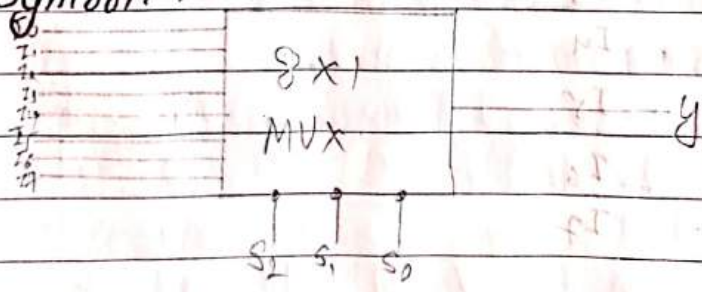


Fig:- Implementation of 8×1 mux using 2×1 Mux circuit

Implement 8×1 mux using 4×1 Mux

8×1 mux is one of the combinational logic circuit which have eight input lines i.e. $I_0, I_1, I_2, I_3, I_4, I_5, I_6$ and I_7 and only one output line i.e. y and three selection lines i.e. S_2, S_1 and S_0.

Graphical symbol:-



Function table:

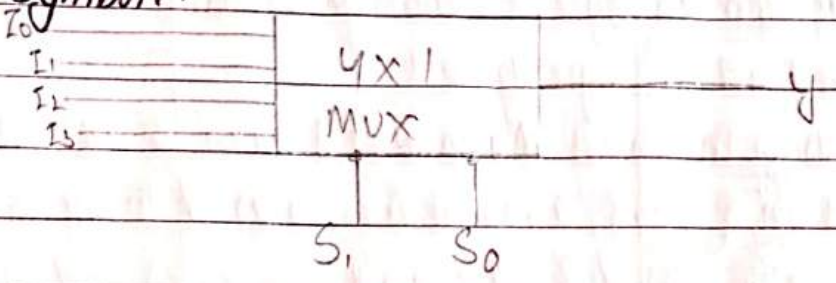
Inputs			Output(y)
S_2	S_1	S_0	
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

$\frac{8}{4} = 2$, $\frac{2}{4} = 0.5$

Using Enable, we can implement 8x1 mux using 4x1 mux

4x1 mux is also logic circuit which have four input lines i.e. I_0, I_1, I_2 and I_3 and only one output line i.e. y and two selection lines i.e. S_1 and S_0

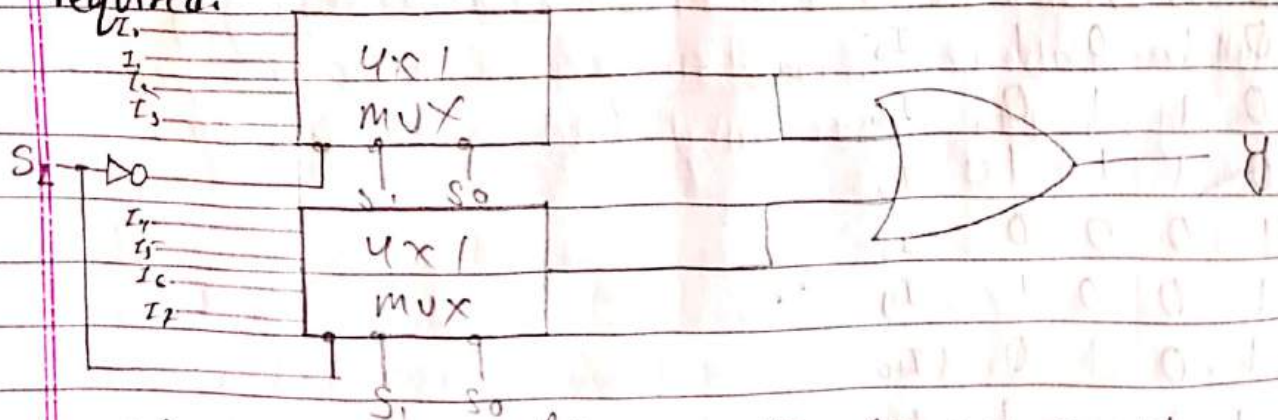
Graphical symbol:-



Function Table

Input		Output (y)
S_1	S_0	
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

To implement 8x1 mux using 4x1 mux. 2, 4x1 mux and 1 OR gate is required.



∴ Fig:- Implementation of 8x1 MUX using 2x1 mux circuit

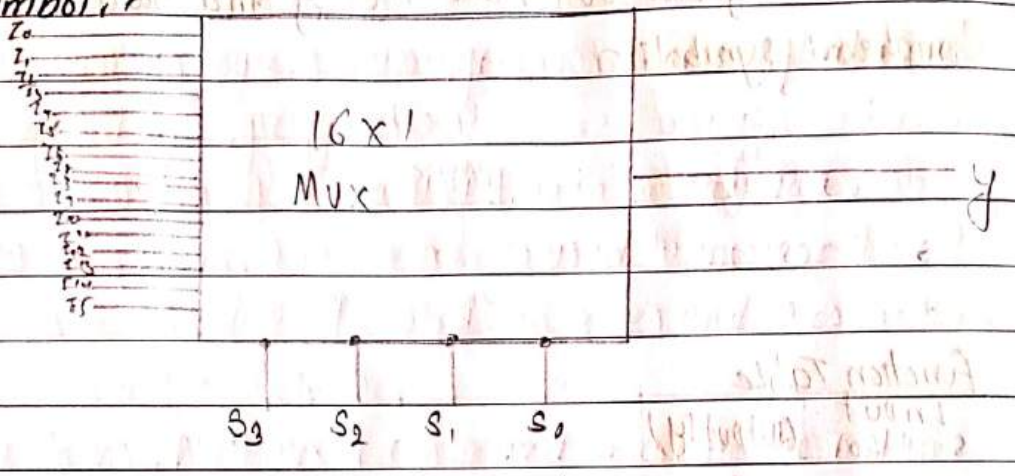
In the above circuit when S_2 is used as an Enable input when $S_2 = 0$, the first 4x1 mux becomes operational and second 4x1 mux is disabled and return 0 as output. Likewise when $S_2 = 1$, the second 4x1 mux becomes operational and first 4x1 mux is disabled and return 0 as output. Whenever 4x1 mux is operational it will select one of the many input sources as selected depending upon the bit combinations of select lines $S_1 S_0$.

DATE _____ PAGE _____

1 of 16 Multiplexer

1 of 16 multiplexer is one of the combinational logic circuit which have 16 input lines i.e. I_0 to I_{15} and only one output lines and four selection lines i.e. S_3, S_2, S_1 and S_0.

Graphical symbolic



Function Table

Inputs				Output (y)
S_3	S_2	S_1	S_0	
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5
0	1	1	0	I_6
0	1	1	1	I_7
1	0	0	0	I_8
1	0	0	1	I_9
1	0	1	0	I_{10}
1	0	1	1	I_{11}
1	1	0	0	I_{12}
1	1	0	1	I_{13}
1	1	1	0	I_{14}
1	1	1	1	I_{15}

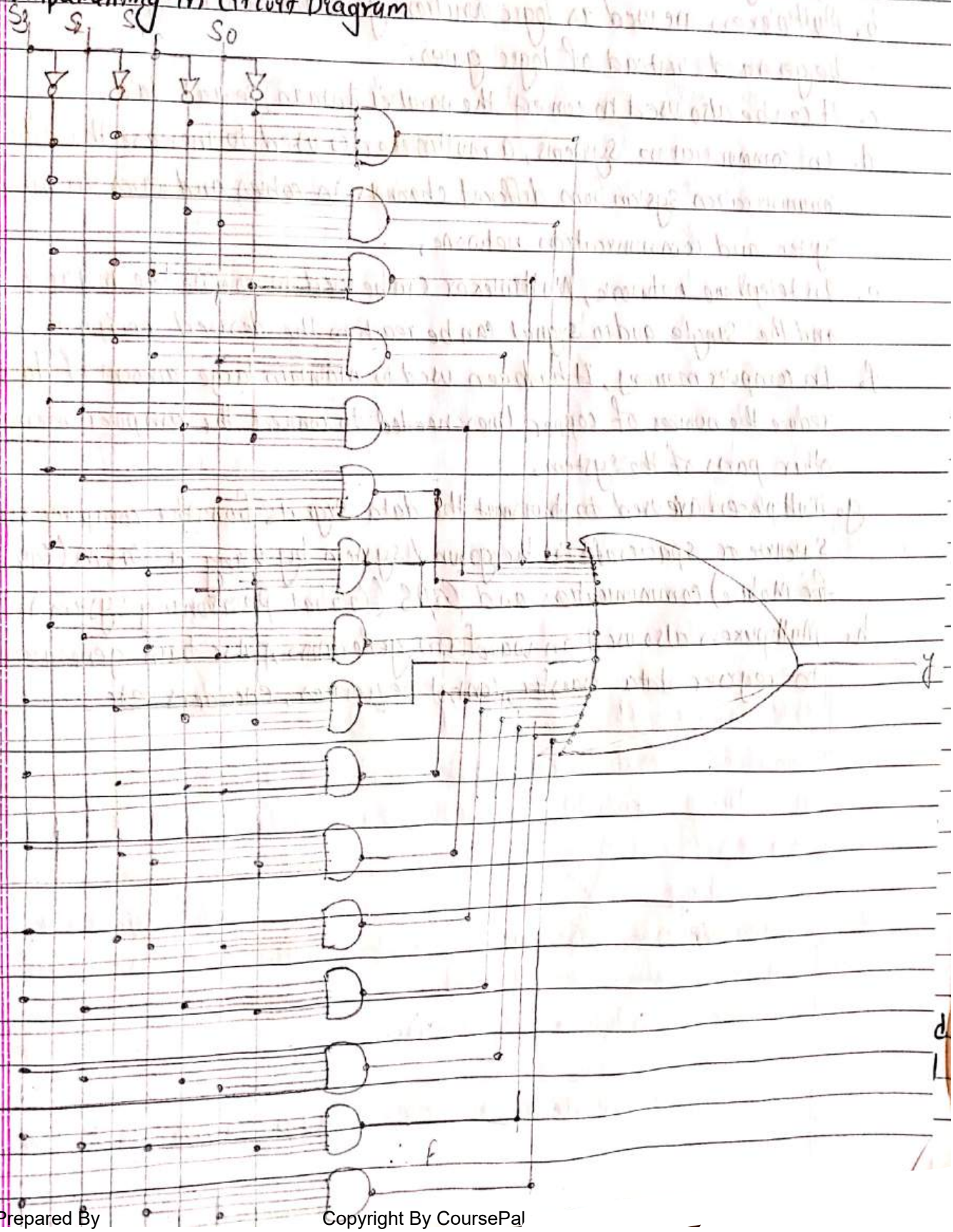
Prepared By
Santosh Bhandari

DATE _____ PAGE _____

From the truth table, the following logical expression is derived

$$y = \bar{S}_3\bar{S}_2\bar{S}_1\bar{S}_0 I_0 + \bar{S}_3\bar{S}_2\bar{S}_1 S_0 I_1 + \bar{S}_3\bar{S}_2 S_1\bar{S}_0 I_2 + \bar{S}_3\bar{S}_2 S_1 S_0 I_3 + \bar{S}_3 S_2\bar{S}_1\bar{S}_0 I_4 + \bar{S}_3 S_2\bar{S}_1 S_0 I_5$$
$$+ \bar{S}_3 S_2 S_1\bar{S}_0 I_6 + \bar{S}_3 S_2 S_1 S_0 I_7 + S_3\bar{S}_2\bar{S}_1\bar{S}_0 I_8 + S_3\bar{S}_2\bar{S}_1 S_0 I_9 + S_3\bar{S}_2 S_1\bar{S}_0 I_{10} + S_3\bar{S}_2 S_1 S_0 I_{11} +$$
$$S_3 S_2\bar{S}_1\bar{S}_0 I_{12} + S_3 S_2\bar{S}_1 S_0 I_{13} + S_3 S_2 S_1\bar{S}_0 I_{14} + S_3 S_2 S_1 S_0 I_{15}$$

Implementing in circuit Diagram



DATE _____ PAGE _____

Write the uses or applications of Multiplexers.

→ The application of mux are as follow:-

a. Multiplexers are used in data routing application to route the data in sequence and particular direction as well as destination as a single output from several input signals.

b. Multiplexers are used as logic function generator where logical expression can be generated instead of logic gates.

c. It can be also used to convert the parallel data in serial data.

d. In communication systems, a multiplexer is used to increase the efficiency of t communication system from different channels in cables and wires in transmission system and communication networks.

e. In telephone network, Multiplexer can be used to isolate the multiple audio signa and the single audio signal can be reach to the desired recipients.

f. In computer memory, It has been used to maintain large amount of data as well a reduce the number of copper lines needed to connect the computer memory to other parts of the system.

g. Multiplexers are used to transmit the data signals from the computer system of a satellite or spacecraft to the ground system by using a GSM (Global system for Mobile) communication and GPS (Global positioning system) satellite.

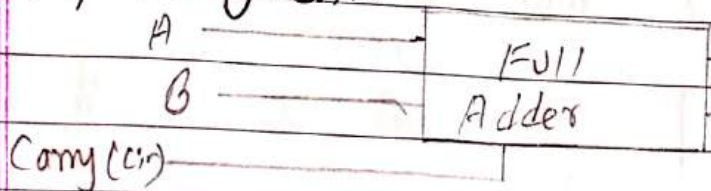
h. Multiplexers also used in waveform generators, pulse train generators, register to register data transfer, control sequences, encoders etc.

DATE _____ PAGE _____

Implementation of Full Adder Using 4x1 Mux

A full adder is a combinational circuit that forms the sum of 3 inputs bits. It accepts two inputs bits and an input carry and generates a sum output and and output carry.

Graphical symbol:-



The Truth Table of full Adder

Truth Table

Inputs			Outputs	
A	B	Cin	Sum(S)	Carry(Cout)
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

From the truth table given above, we can obtained the function for the combinational circuit in the form of sum of minterms as

$Sum(A, B, C_{in}) = \Sigma m(1,2,4,7)$ and $Carry(A, B, C_{in}) = \Sigma(3,5,6,7)$

Implementation of function $Sum(A, B, C_{in}) = \Sigma m(1,2,4,7)$ by using 4x1 multiplexers.

There is a three variable function and therefore we need a multiplexers with two selection lines and four input which is to apply variable B and C_{in} to the selection lines and A as input variable. The variable are chosen to be A or $\bar{A}$ or 0 or 1.

Thus, function and Truth Table are shown below

Function Table

Inputs B	Cin	output (S)
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Truth Table

Inputs A	B	Cin	Output (S)
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

From the truth table, we have the following implementation table

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

From the implementation table, we have the following implementation circuit:-

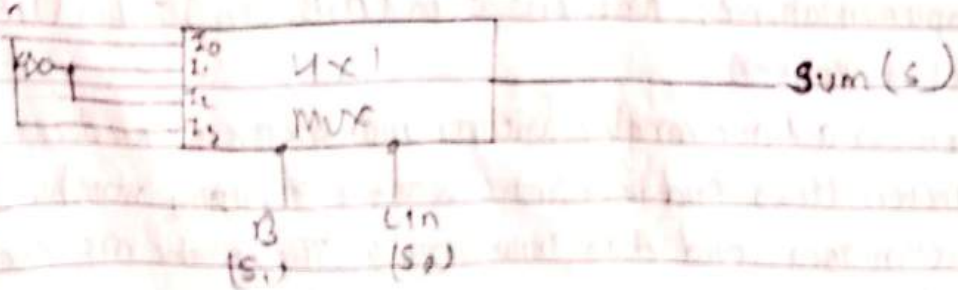


Fig:- Implementation of full adder sum using MUX

Carry (cout) $(A, B, C_{in}) = \Sigma m\,(3, 5, 6, 7)$ by using 4x1 mux

There is a three variable function and therefore we need a multiplexer with two selection lines and four input which is to apply variable B and C_{in} to the selection lines and A as input variable. The input variable are chosen to be A or $\bar{A}$ or 0 or 1.

Thus, Function and Truth Table are shown below:-

Function Table

Inputs		Output (C_{out})
B	C_{in}	
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Truth Table

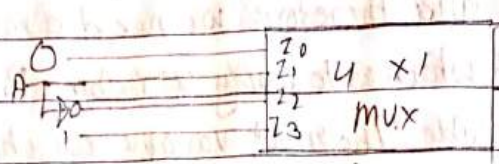
Inputs			Output (C_{out})
A	B	C_{in}	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

From the truth Table, we have the following implementation table

Implementation Table

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

From the implementation table, we have the following implementation circuit



∴ Fig:- Implementation of full adder carry using Mux

Demultiplexer (Demux)

Demultiplexing is the process of taking information from one input and transmitting the same over one of several outputs.

Demux is the combinational logic circuit that has only single input line, 2^n output lines and n selection lines.

Depending upon bit combination of selection lines, the input is going to be directed to one of the several output lines.

In other words, A Demux is a logic circuit that receives information on a single input and transmits the same information over one of the output line as per selection lines directs.

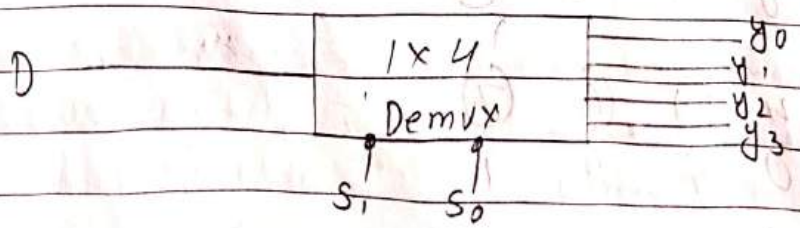
Demux does reverse operation that of mux thats why it is called data distributor.

In short, A mux selects an input lines. A Demux selects an output line.

1x4 Demultiplexer

1 to 4 Demultiplexer has single input i.e D, four outputs lines i.e. y_0, y_1, y_2 and y_3 and two select lines i.e. S_1 and S_0.

Graphical symbol:-



Truth Table

Data Input	Select lines		Outputs			
D	S_1	S_0	y_0	y_1	y_2	y_3
D	0	0	D	0	0	0
D	0	1	0	D	0	0
D	1	0	0	0	D	0
D	1	1	0	0	0	D

From the table, it is cleared that the data input is connected to output y_0 when $S_1=0$ and $S_0=0$, and so on.

From the truth table, the expressions for outputs can written as follows:-

$y_0 = \bar{S}_1 \bar{S}_0 D$

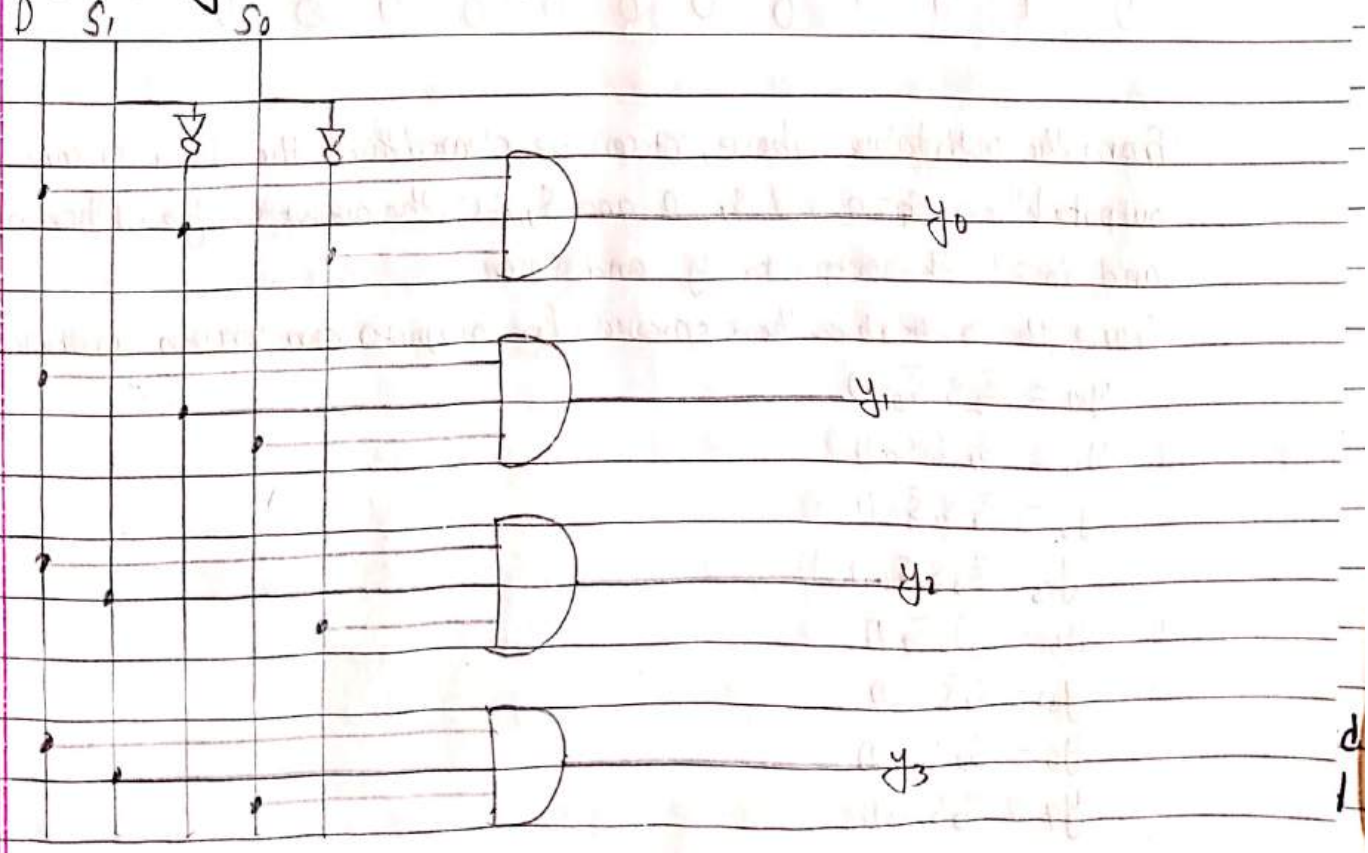
$y_1 = \bar{S}_1 S_0 D$

$y_2 = S_1 \bar{S}_0 D$

$y_3 = S_1 S_0 D$

Now, using this expression 1x4 demux can be implemented using four 3 input AND gates and two NOT Gates.

Implementing 1x4 Demux logic circuit

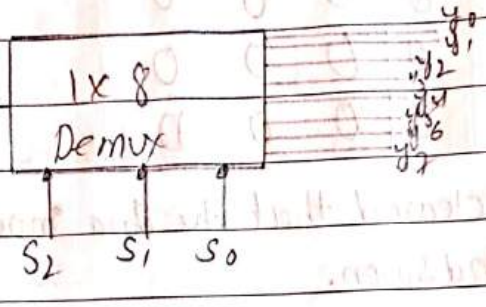


∴ Fig:- Implementation of 1x4 Demultiplexer circuit

1×8 Demultiplexer

1×8 demultiplexer has single input i.e. D, four eight output lines i.e. y_0 to y_7 and three selection lines i.e. S_2, S_1 and S_0.

Graphical symbol:-



Truth Table

Data input	Select lines S_2	S_1	S_0	Outputs y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
D	0	0	0	D	0	0	0	0	0	0	0
D	0	0	1	0	D	0	0	0	0	0	0
D	0	1	0	0	0	D	0	0	0	0	0
D	0	1	1	0	0	0	D	0	0	0	0
D	1	0	0	0	0	0	0	D	0	0	0
D	1	0	1	0	0	0	0	0	D	0	0
D	1	1	0	0	0	0	0	0	0	D	0
D	1	1	1	0	0	0	0	0	0	0	D

From the truth table above, It can be cleared that the data is connected to output. When $S_2=0$, and $S_1=0$ and $S_0=0$, the output is y_0. When $S_2=0$, $S_1=0$ and $S_0=1$, the output is y_1 and so on.

From the truth table, the expression for outputs can written as follows:-

$y_0 = \bar{S}_2 \bar{S}_1 \bar{S}_0 D$

$y_1 = \bar{S}_2 \bar{S}_1 S_0 D$

$y_2 = \bar{S}_2 S_1 \bar{S}_0 D$

$y_3 = \bar{S}_2 S_1 S_0 D$

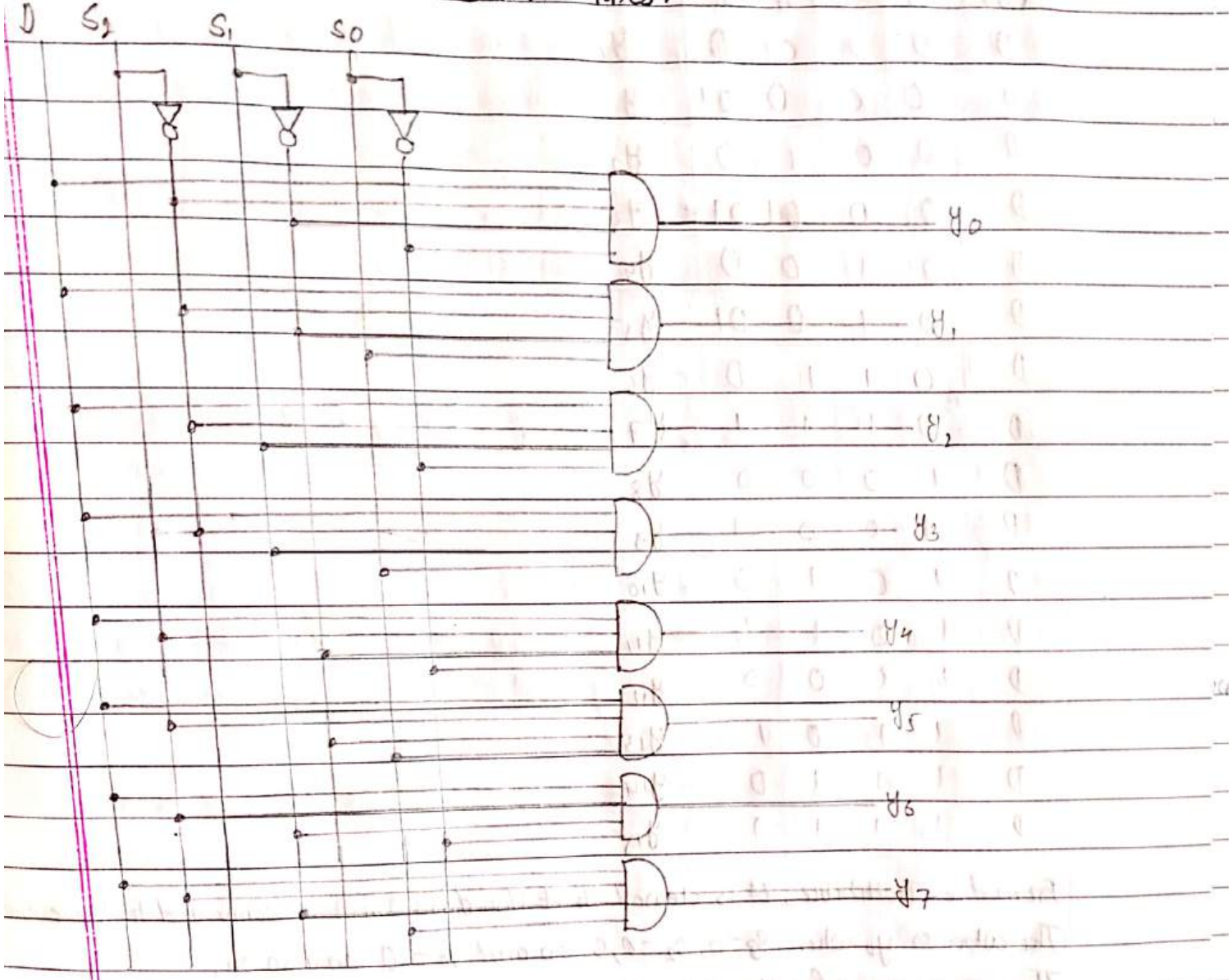
$y_4 = S_2 \bar{S}_1 \bar{S}_0 D$

$y_5 = S_2 \bar{S}_1 S_0 D$

$y_6 = S_2 S_1 \bar{S}_0 D$

$y_7 = S_2 S_1 S_0 D$

Now, Using the expression, 1x8 Demux can be implemented using eight, 4input AND Gates and 3 NOT Gates.



∴ Fig:- Implementation of 1x8 Demultiplexers circuit

1x16 Demultiplexer

1x16 demultiplexer has single input line i.e. D, Sixteen output lines i.e. Y0 to Y15 and four selection lines i.e. S3, S2, S1 and S0.

Graphical Symbol:-

Truth Table

Data Input (D)	Select lines S_3	S_2	S_1	S_0	Outputs
D	0	0	0	0	y_0
D	0	0	0	01	y_1
D	0	0	1	0	y_2
D	0	0	01	01	y_3
D	0	1	0	0	y_4
D	0	1	0	01	y_5
D	0	1	1	0	y_6
D	0	1	1	1	y_7
D	1	0	0	0	y_8
D	1	0	0	1	y_9
D	1	0	1	0	y_{10}
D	1	0	1	1	y_{11}
D	1	1	0	0	y_{12}
D	1	1	0	1	y_{13}
D	1	1	1	0	y_{14}
D	1	1	1	1	y_{15}

From the Truth table, it is cleared that the data input is connected to the output. The output is y_0 when $S_3=0, S_2=0, S_1=0$ and $S_0=0$ and so on.

The expression for the outputs can be derived from truth table :-

$y_0 = \bar{S}_3\bar{S}_2\bar{S}_1\bar{S}_0 D$

$y_1 = \bar{S}_3\bar{S}_2\bar{S}_1 S_0 D$

$y_2 = \bar{S}_3\bar{S}_2 S_1\bar{S}_0 D$

$y_3 = \bar{S}_3\bar{S}_2 S_1 S_0 D$

$y_4 = \bar{S}_3 S_2\bar{S}_1\bar{S}_0 D$

$y_5 = \bar{S}_3 S_2\bar{S}_1 S_0 D$

$y_6 = \bar{S}_3 S_2 S_1\bar{S}_0 D$

$y_7 = \bar{S}_3 S_2 S_1 S_0 D$

$y_8 = S_3\bar{S}_2\bar{S}_1\bar{S}_0 D$

$y_9 = S_3\bar{S}_2\bar{S}_1 S_0 D$

$y_{10} = S_3\bar{S}_2 S_1\bar{S}_0 D$

$y_{11} = S_3\bar{S}_2 S_1 S_0 D$

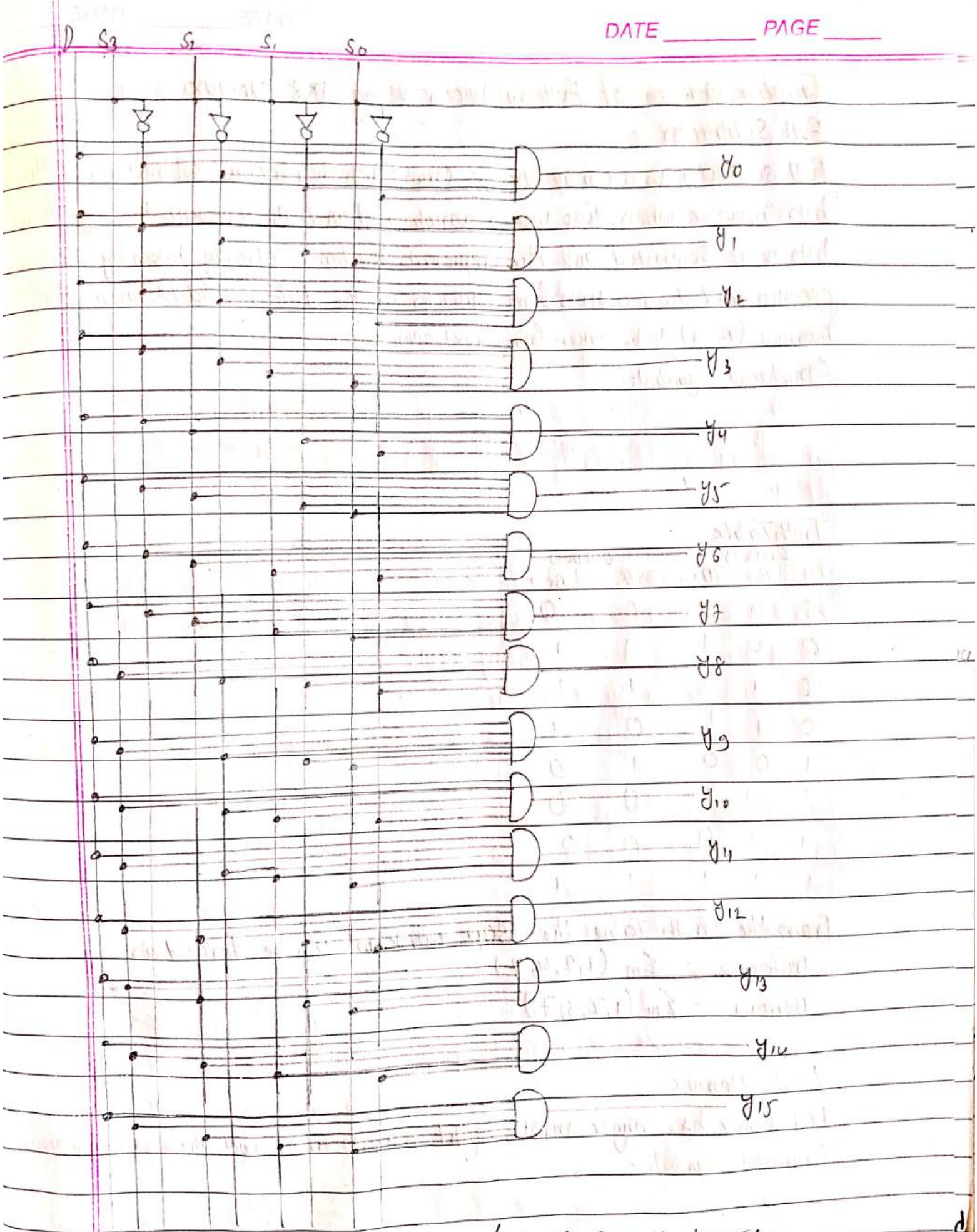
$y_{12} = S_3 S_2\bar{S}_1\bar{S}_0 D$

$y_{13} = S_3 S_2\bar{S}_1 S_0 D$

$y_{14} = S_3 S_2 S_1\bar{S}_0 D$

$y_{15} = S_3 S_2 S_1 S_0 D$

Now, using the following expression 1x16 Demux can be implemented using sixteen 5 input AND Gates and 4 NOT Gates



∴ Fig:- Implementation of 1x16 Demultiplexers

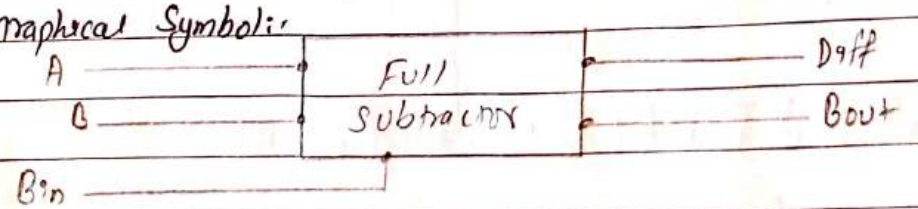
DATE _____ PAGE _____

Implementation of Full subtractor using 1×8 Demux

Full Subtractor

Full subtractor is a combinational circuit which performs subtractions of three bits input numbers. Two inputs variables A and B represents two significant bits to be subtracted and Bin represents the borrow already taken by previous operation and Two outputs from subtractor are the difference (Diff) and the Borrow (Bout) to be taken from next operation.

Graphical Symbol:



Truth Table

Inputs			Outputs	
A	B	Bin	Diff	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

From the Truth Table the SOP minterm can be derived as

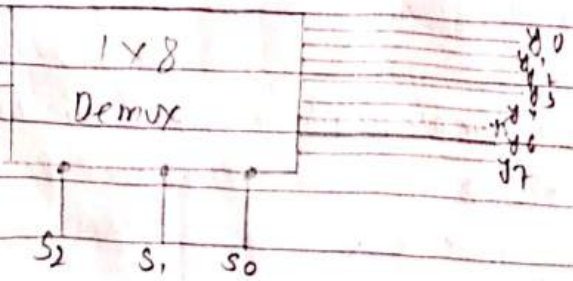
Difference = $\Sigma m\ (1,2,4,7)$

Borrow = $\Sigma m\ (1,2,3,7)$

1×8 Demux

1×8 Demux has single input, eight output lines and three selection lines.

Graphical Symbol:-



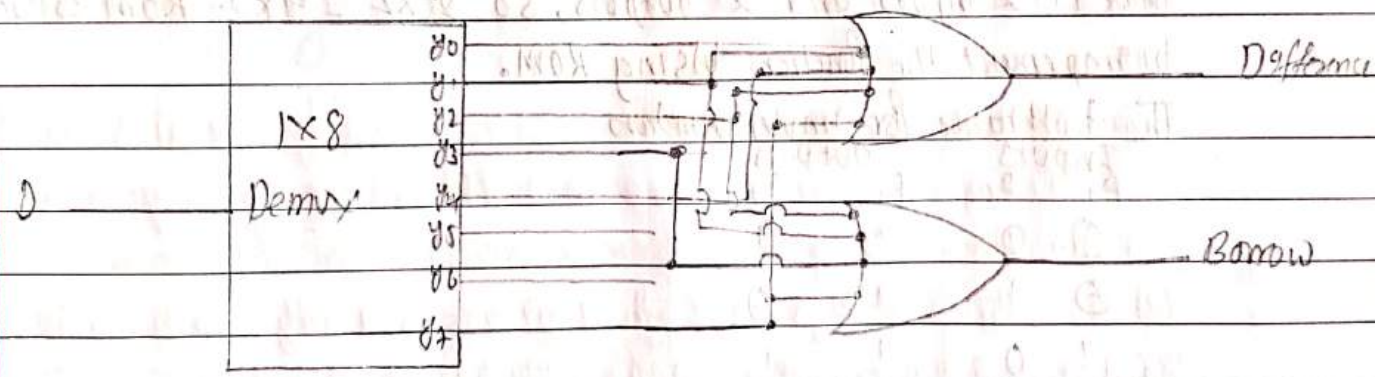
Truth Table

Inputs			Outputs							
A	B	Bin	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_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

Implementing the minterm in 1x8 Demux

Difference = Σm (1, 2, 4, 7)

Borrow = Σm (1, 2, 3, 7)



∴ Fig:- Implementation of full subtractor using 1x8 Demux circuit

7. Design of ROM

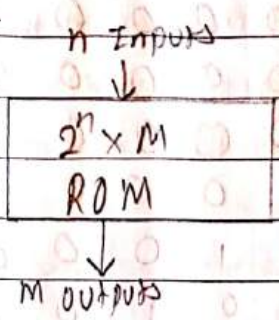
ROM is a memory device in which permanent binary memory is stored. ROM is a single device that consists both Decoder and OR Gates in a single IC package.

The connections between the outputs of the decoder and the inputs of the OR Gates can be specified for each particular configuration by programming the ROM. ROM comes with special internal links that can be fused or Broken and the desired interconnection for a particular application requires that links to be fused or broken to the required circuit path.

DATE _____ PAGE _____

Once the pattern is established for a ROM, it remains fixed even when the power is turn off.

Block Diagram:-



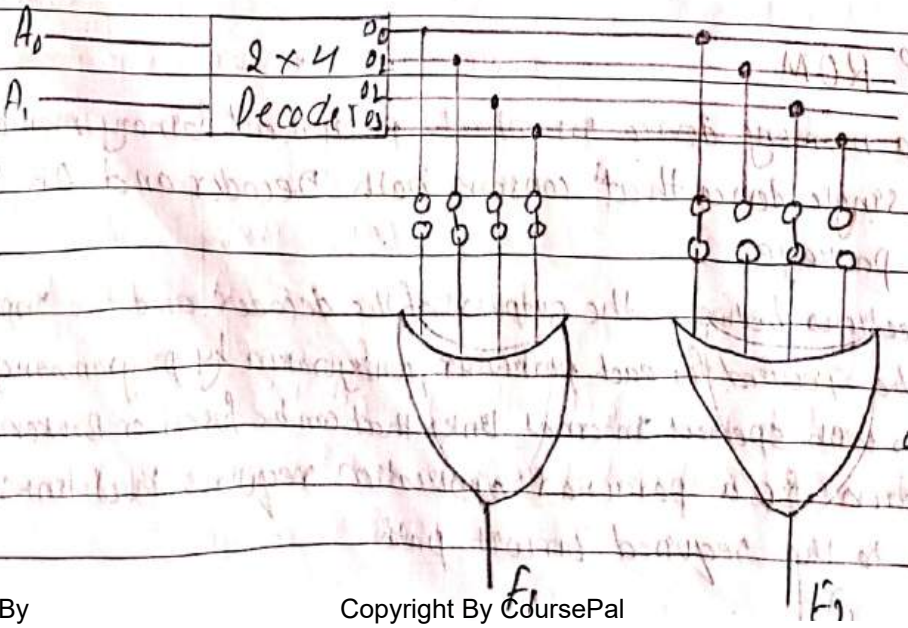
Implement $F_1(A_1, A_0) = \Sigma m(1, 2, 3)$ and $F_2(A_1, A_0) = \Sigma m(0, 2)$ using ROM.

There is 2 inputs, so 2x4 Decoder is required to implement. Similarly, There is 2 inputs and 2 outputs, so $2^2 \times 2 = 4 \times 2$ Rom is required to implement the function using ROM.

The truth table for implementation

Inputs		outputs	
A_1	A_0	F_1	F_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

Implementation in circuit with the help of truth Table:-



∴ Fig: Implementation of boolean expression in ROM

DATE _____ PAGE _____

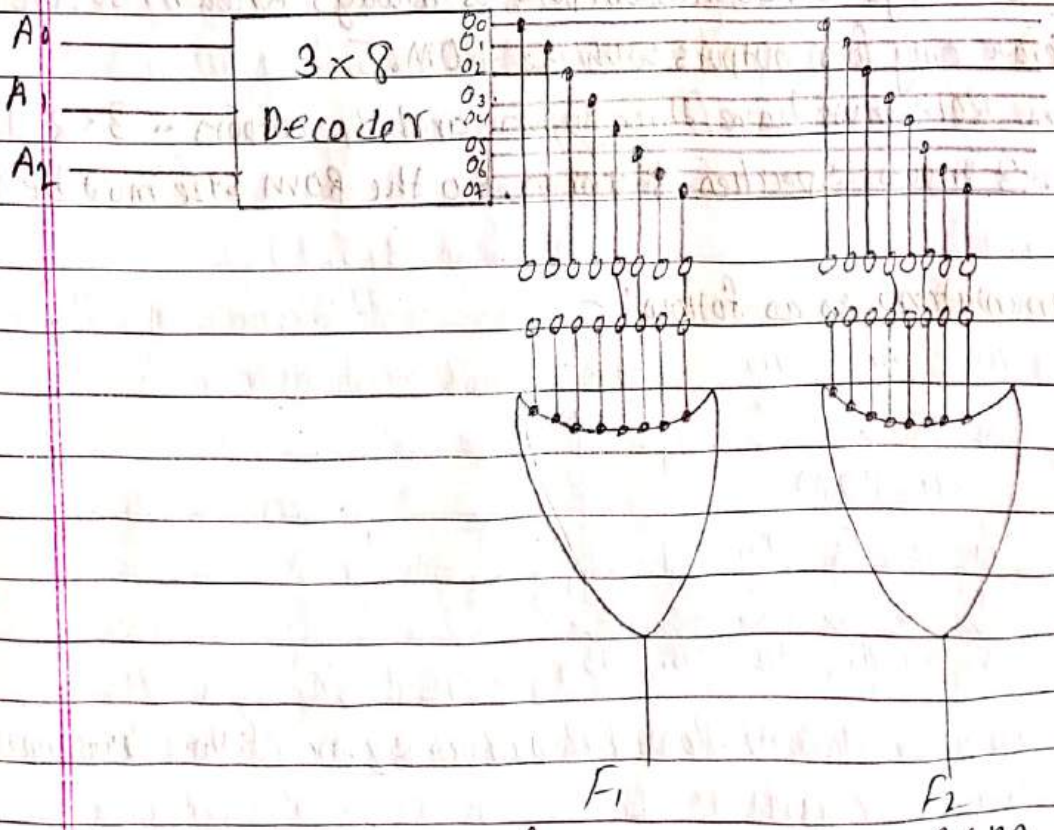
Implement $F_1 = \Sigma(4,5,7)$ and $F_2 = \Sigma(3,5,7)$ using ROM.

There are 3 inputs lines and let they be A_2, A_1 and A_0. So, 3×8 Decoder is required to implement the function. Similarly, There is 3 input and 2 outputs, so $2^3 \times 2 = 8 \times 2$ Rom is required to implement the given function.

The Truth Table for implementation

Inputs			Outputs	
A_2	A_1	A_0	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

Implementation in circuit with the help of truth table :-



Implementation of boolean expression in ROM

Design a circuit that provides the square of three bit number using ROM

The first step is to derived the Truth Table for the combinational circuit as given below:-

Inputs			Outputs					
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1

Three inputs and six outputs are required to accommodate all possible numbers.

From the truth table, We note that output B_0 is always equal to input A_0. So, there is no need to generate B_0 with a ROM, since it is equal to an input variable A_0.

Likewise, B_1 is always 0, so this output is always known. So, we actual need to generate only four outputs with the ROM.

The minimum ROM must have three inputs and 4 outputs. 3×8 Decoder is required. 3 inputs specifies 8 words, so the ROM size must be 8×4 size.

The implementation is as follow:-

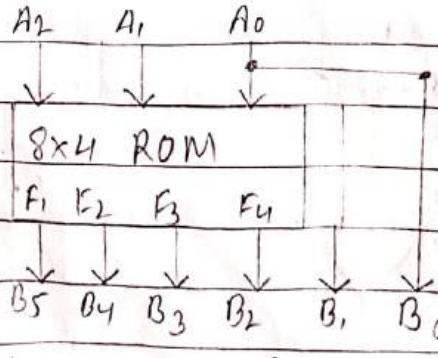


Fig:- Implementation of ROM that produces square of three bit input number circuit

The Truth Table of ROM

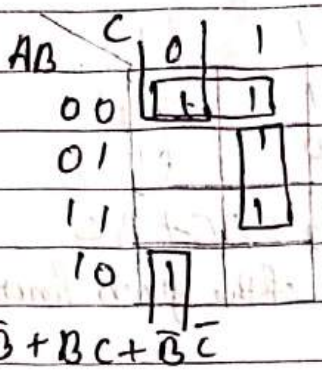
Inputs			Outputs			
A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

Programmable logic Device (PLD)
Programmable logic Array (PLA)
Programmable Array Logic (PAL)

Programmable logic Array (PLA)
Programmable logic Array is a programmable logic device having array of arrangement of programmable AND Gates and programmable OR Gates.
PLA = programmable AND Gates + Programmable OR Gates

Implement the function $F(A,B,C) = \Sigma m(0,1,3,4,7)$ using PLA.

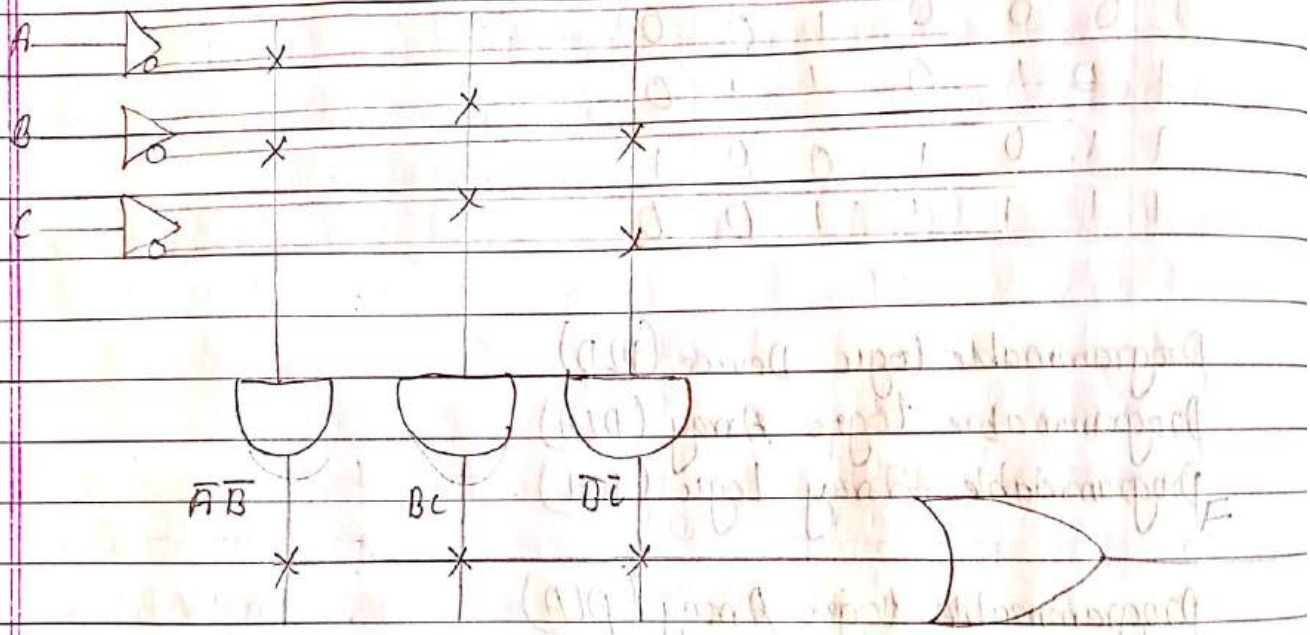
The minimized SOP expression for the given function can be obtained by using K-map as shown below:-



$A\bar{B} + Bc + \bar{B}\bar{c}$

Hence, the minimized sop expression of the given function is $F = A\bar{B} + BC +$

For implementing the function using PLA 3 Buffer input, 3 AND programmable AND Gates and 1 programmable OR Gates is required. The implementation is shown below:-



∴ Fig:- Implementation of given function using PLA.

Implement the function using PLA $F_1 = \Sigma m(4,5,7)$, $F_2 = \Sigma m(3,5,7)$
Let the input three variable be A, B and C. The minimized sop expression for the given function can be obtained by using K-Map as shown below:-

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

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

$F_1 = A\bar{B} + AC$

$F_2 = \Sigma m(3,5,7)$

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

$F_2 = AC + BC$

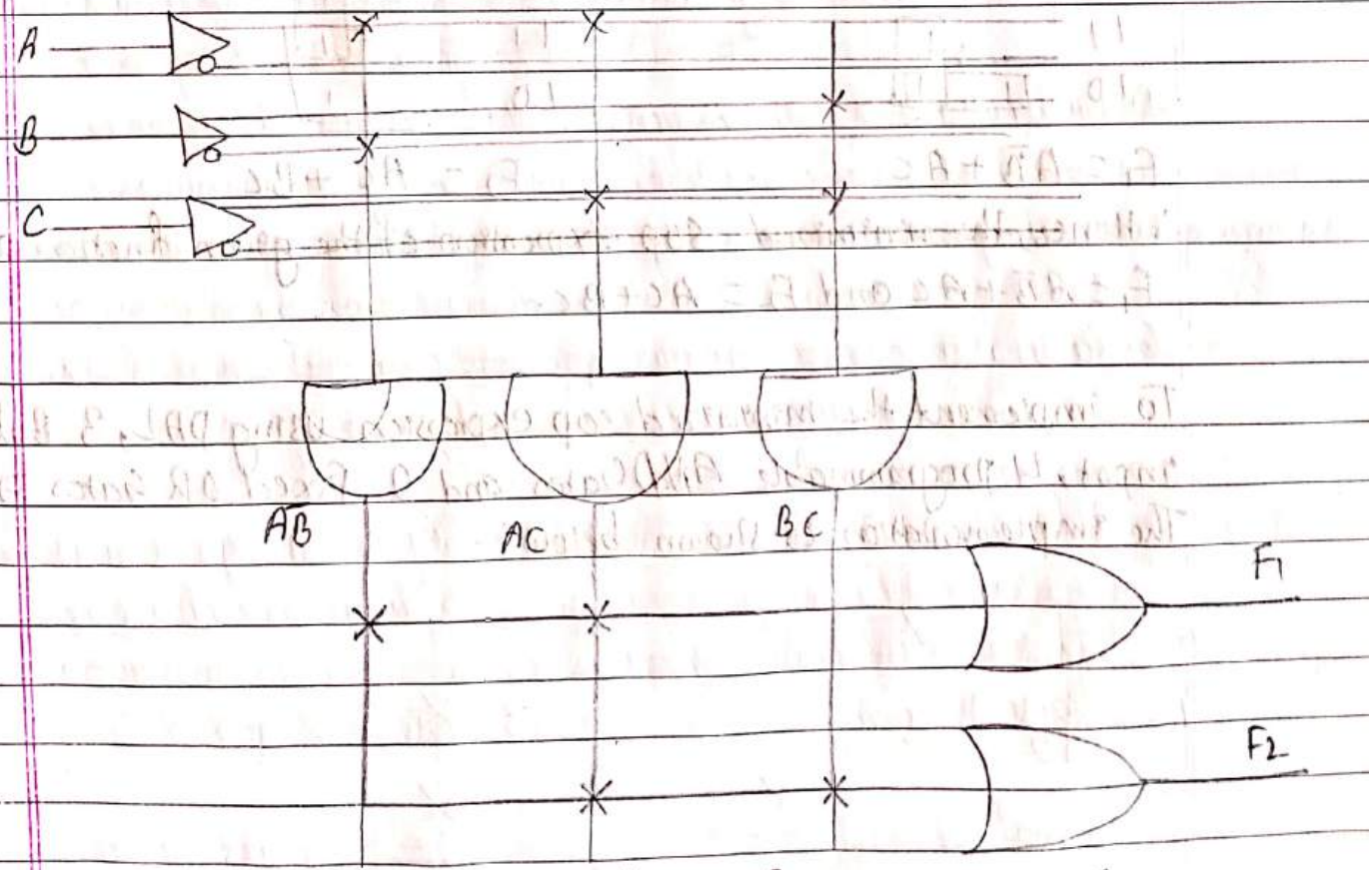
Hence, the minimized sop expression of the given function are $F_1 = A\bar{B} + AC$ and $F_2 = AC + BC$

To implement the minimized sop expression using PLA, 3 Buffer input i.e. A or $\bar{A}$, B or $\bar{B}$ and C or $\bar{C}$, 3 programmable AND Gates because in both function there is common sharable gates and 2 programmable OR Gates is required. The implementation is shown below with program Table of PLA.

PLA Program Table

	Product term	Inputs A	B	C	outputs F_1	F_2
$A\bar{B}$	1	1	0	–	1	–
AC	2	1	–	1	1	1
BC	3	–	1	1	–	1

PLA Implementation



∴ Fig:- Implementation of given function using PLA.

DATE _____ PAGE _____

Programmable Array logic (PAL)

Programmable Array logic is a programmable logic device having array of programmable AND Gates and fixed array of OR Gates.

PAL = Programmable AND Gates + Fixed OR Gates

Implement the given function using PAL

$F_1 = \Sigma(4,5,7)$, $F_2 = \Sigma(3,5,7)$

The minimized SOP expression for the given function can be obtained by using K-Map are shown below:-

$F_1 = \Sigma(4,5,7)$

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

$F_1 = A\bar{B} + AC$

$F_2 = \Sigma(3,5,7)$

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

$F_2 = AC + BC$

∴ Hence, the minimized SOP expression of the given function are $F_1 = A\bar{B} + AC$ and $F_2 = AC + BC$.

To implement the minimized SOP expression using PAL, 3 Buffer input, 4 programmable AND Gates and 2 Fixed OR Gates is required. The implementation is shown below:-

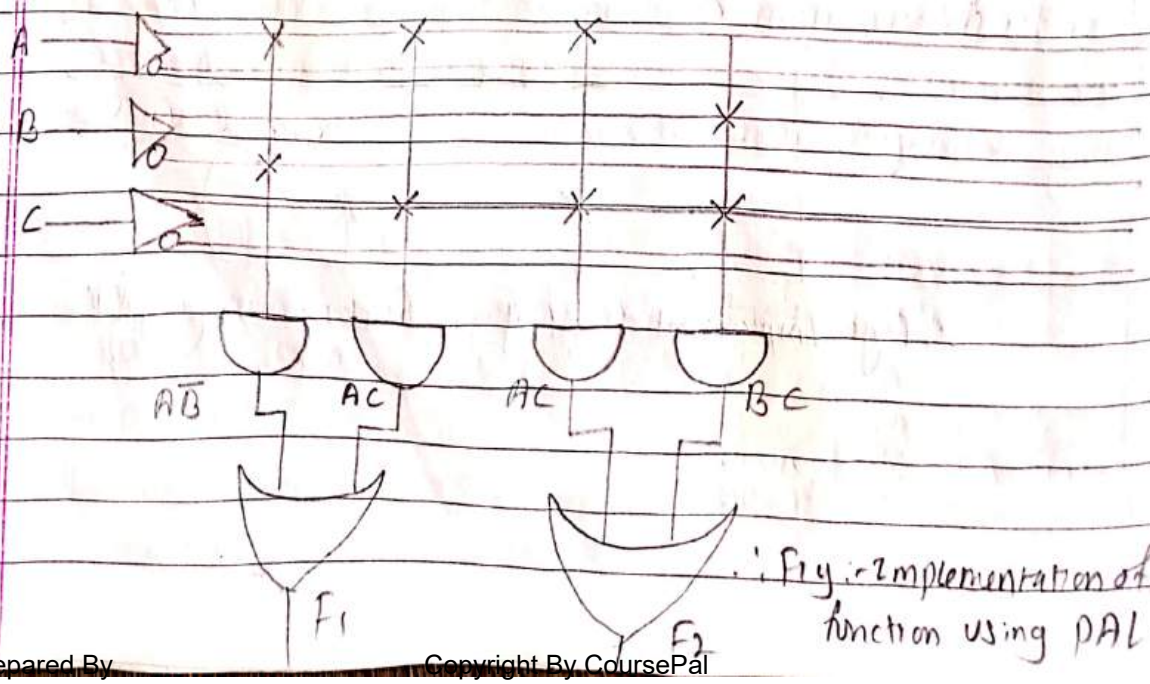


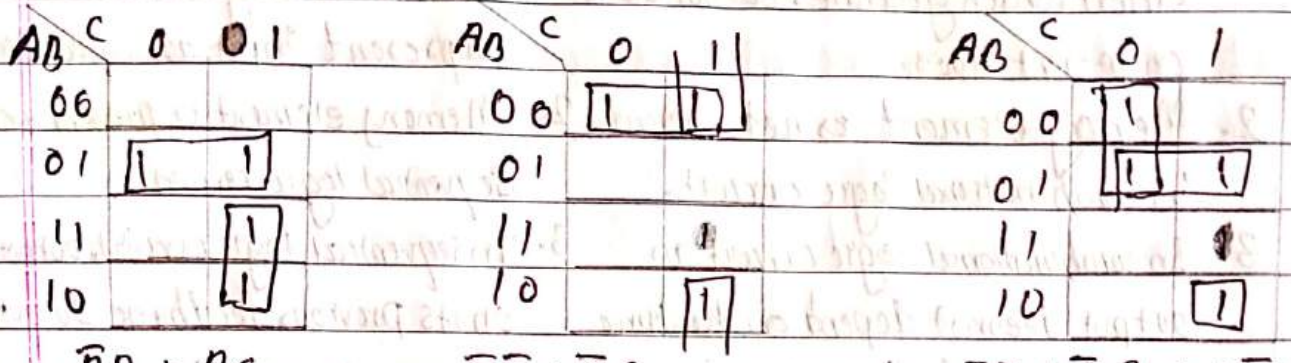
Fig:- Implementation of given function using PAL.

Implement the given function using PAL

$F_1(A,B,C) = \Sigma(2,3,5,7)$; $F_2(A,B,C) = \Sigma(0,1,5)$ and $F_3(A,B,C) = \Sigma(0,2,3,5)$

The minimized sop expression for the given function can be obtained by using k-map are shown below

$F_1 = \Sigma(2,3,5,7)$, $F_2 = \Sigma(0,1,5)$ $F_3 = \Sigma(0,2,3,5)$

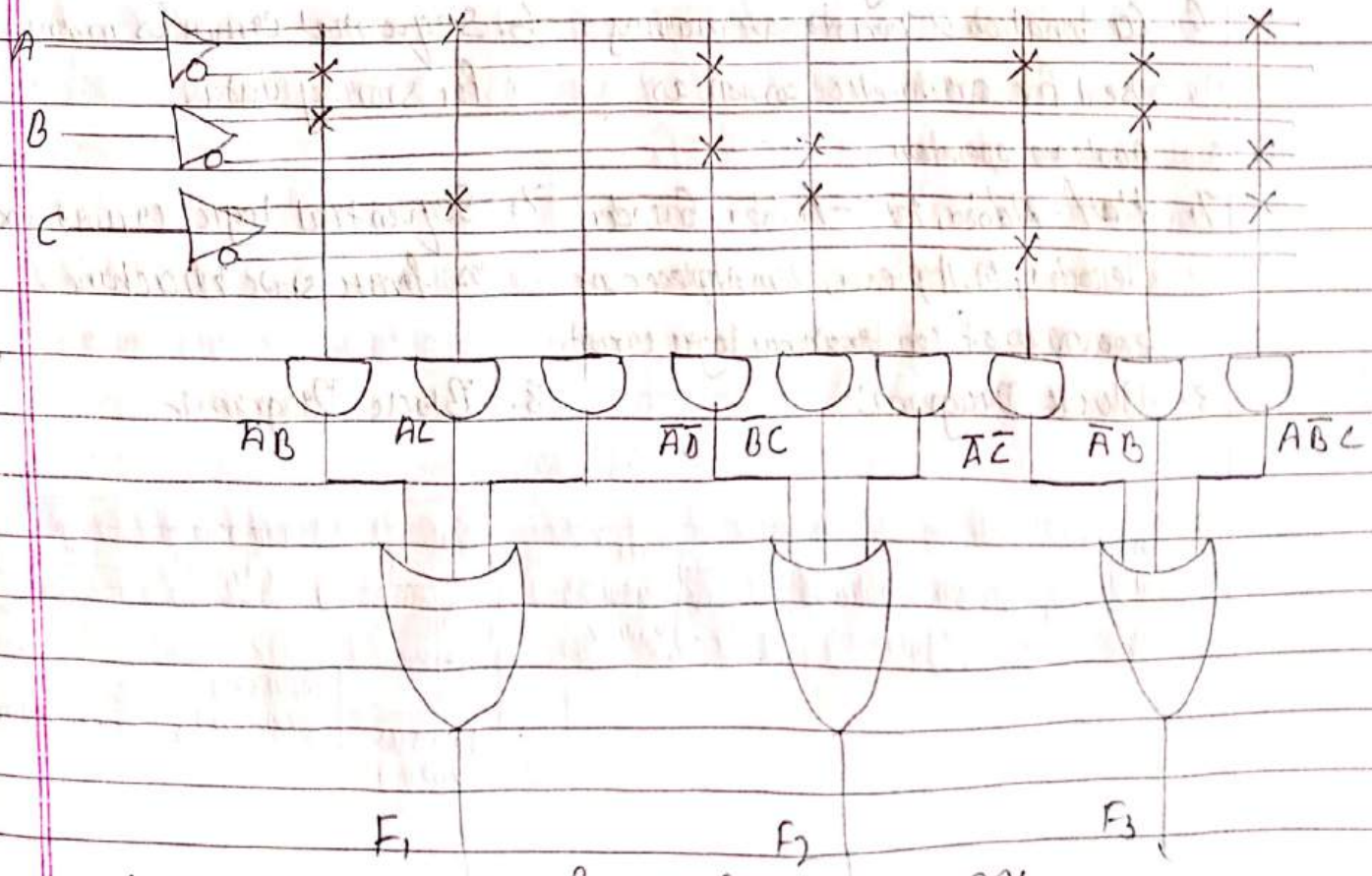


$\bar{A}B + AC$ $\bar{A}\bar{B} + \bar{B}C$ $A\bar{C} + \bar{A}B + A\bar{B}C$

∴ Hence, the minimized sop expression of the given functions are:-

$F_1 = \bar{A}B + AC$, $F_2 = \bar{A}\bar{B} + \bar{B}\bar{C}$, $F_3 = A\bar{C} + \bar{A}B + A\bar{B}C$

To implement the minimize sop expression using PAL, 3 input Buffer, 9 programmable AND Gates because we have to take the highest AND Gate function and implement same way to all and 3 Fixed OR Gate is required. The implementation is shown below:-

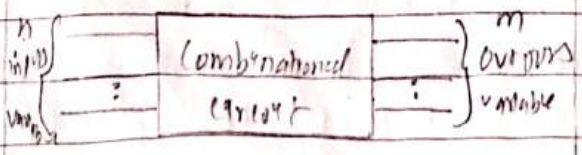


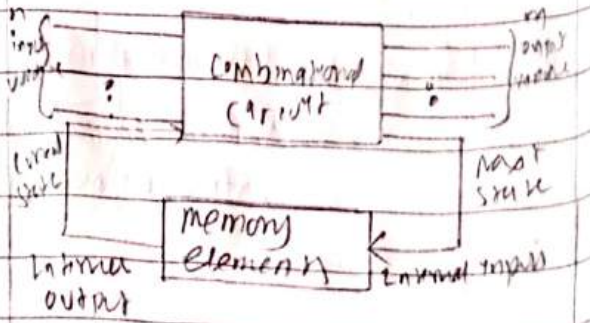
∴ Fig:- Implementation of given function using PAL

1. Differentiate between combinational logic circuit and sequential logic circuit.

The difference between combinational logic circuit and sequential logic circuit are:-

S.N	Combinational Logic Circuit	S.N	Sequential Logic Circuit
1.	Combinational logic circuit is a logic circuit which generates output based on present inputs.	1.	Sequential logic circuit is a logic circuit which generates output based on present input as well as past outputs.
2.	Memory element is not present in combinational logic circuit.	2.	Memory element is present in sequential logic circuit.
3.	In combinational logic circuit as output doesn't depend on the time instant, no feedback loop is required for its next output generation.	3.	In sequential logic circuit, output relies on its previous feedback so output of previous input is being transmitted as feedback, so feedback loop is used for next output generation.
4.	Combination of operands are not necessary in same sequence.	4.	Sequence of operands are necessary in sequential circuit.
5.	Building Block for combinational circuit are logic gates.	5.	Building Block for sequential circuit are flipflops.
6.	Combinational circuit are mainly used for arithmetic as well as boolean operation.	6.	Sequential circuit is mainly used for storing data.
7.	Half-Adder, Full-Adder, Encoder, Decoder, Multiplexer, Demultiplexer are examples of combinational logic circuit.	7.	Sequential logic circuit example is finite state machine.
8.	Block Diagram:-	8.	Block Diagram:-





2. Sequential logic Design

In sequential circuit, the present output depend upon the present input aswell as the past output.

The sequential logic includes memory elements along with the logic gates. Thus, the output generated by sequential circuit depends not only on the present stage of input but also on the previous output.

The sequential circuit involves usage of feedback loops, with the help of this loop, the state of previous output is recorded. Thus, the next output is controlled by the state of previous output.

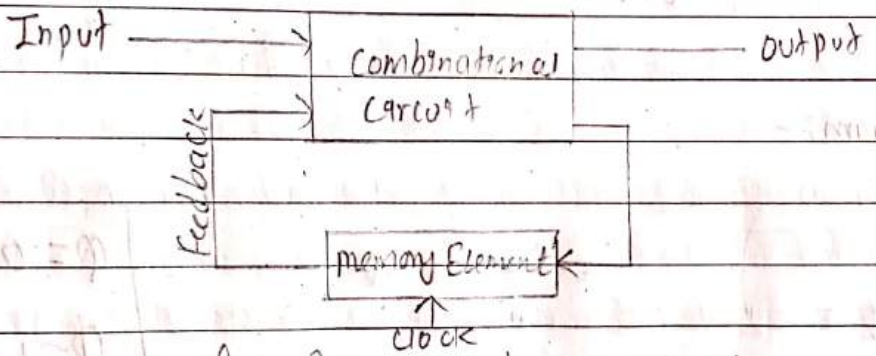


fig:- Sequential logic circuit

3. Basic Flip Flop Circuit (Basic storage element) (Latch)

The basic storage emement is called latch.

A basic flip flop circuit (latch) can be constructed from two NAND Gates or two NOR Gates.

Latch is a crossed coupled connection of two NAND gates or two NOR gates.

Latch is a type of temporary storage device that has two stables states which is also called bistable.

Bistable (two state) are outcome of using a feedback arrangment in which the outputs are connected back to the opposite input.

latch is a type of temporary storage device that has two stables states.

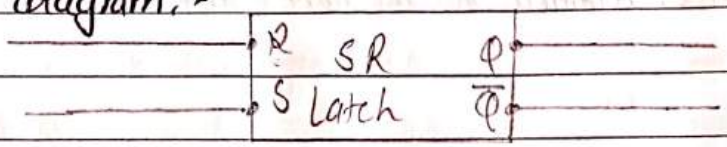
DATE _____ PAGE _____

SR Latch (SET RESET Latch)

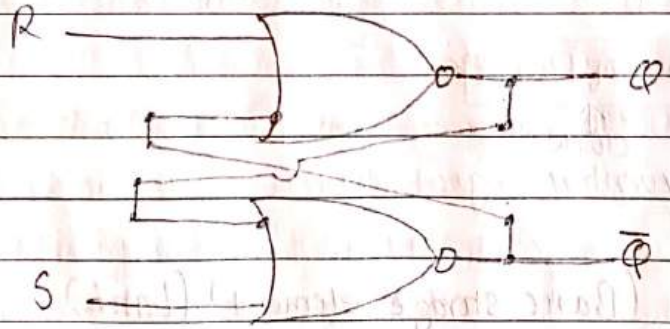
The SR latch has two inputs namely SET (S) and Reset (R) and two outputs Q and $\bar{Q}$. The two outputs are complements to each other. The SR latch can be constructed by using two cross coupled NAND gates or NOR gates.

SR Latch using two cross-coupled NOR Gates

Block diagram:-



Circuit Diagram:-



$Q = 0$ – RESET

$Q = 1$ – SET

Case I:- When $S=0$ and $R=1$, The output of $Q=0$ and $\bar{Q}=1$ which are complement to each other. And if we input $S=0$ and $R=0$, we get the previous output i.e. $Q=0$, $\bar{Q}=1$.

Case II:- When $S=1$ and $R=0$, then the output will be $Q=1$ and $\bar{Q}=0$. If the input is changed to $S=0$ and $R=0$, the output remains as previous i.e. $Q=1$ and $\bar{Q}=0$.

Case III:- When $S=1$ and $R=1$, then the output will be $Q=1$, $\bar{Q}=1$ which violates the requirements that the outputs must be complement of each other. This is contradiction, so it is not use state or invalid state.

The Truth Table of SR NOR Gates latch:-

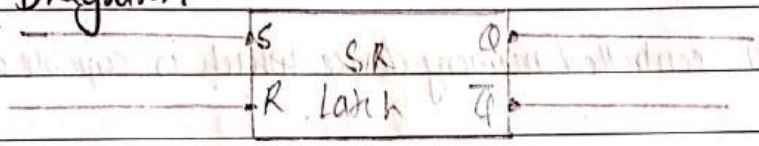
Inputs S	Inputs R	Outputs Q	Outputs $\bar{Q}$
0	0	As Before (No change)	
0	1	0	1
1	0	1	0
1	1	Not use (Invalid state)	

or

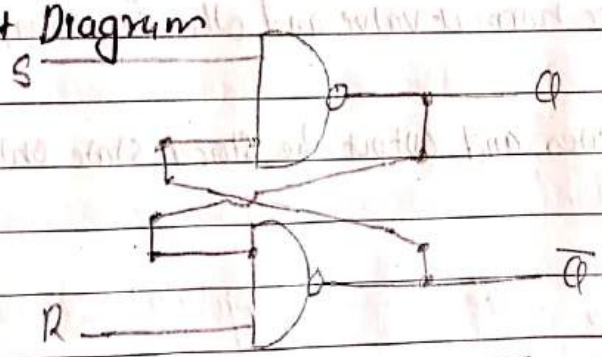
Inputs S	Inputs R	Outputs Q	Outputs $\bar{Q}$	
1	0	1	0	(SET)
0	0	1	0	(No change)
0	1	0	1	(RESET)
0	0	0	1	(No change)
1	1	0	0	(Invalid state)

SR latch using two cross-coupled NAND Gates

Block Diagram:-



Circuit Diagram



Case I:- When S=0 and R=1, The output of $Q=1$ and $\bar{Q}=0$ which are complement to each other. Whe the input is change to S=0 1 and R=1, we get the past output i.e. $Q=1$, $\bar{Q}=0$.

Case II:- When S=1 and R=0, then the output will be $Q=0$ and $\bar{Q}=0$ 1. If the input is changed to S=1 and R=1, the output remains as previous i.e. $Q=0$ and $\bar{Q}=1$.

Case III:- When S=0 and R=0, then the output will be $Q=1$ and $\bar{Q}=1$ which is contradiction, so it is not use or invalid state.

The Truth Table of SR NAND Gates Latch

Inputs S	Inputs R	outputs Q	outputs $\bar{Q}$
0	0	Not use (Invalid condition)	
0	1	1	0
1	0	0	1
1	1	As Before (No change)	

Inputs S	Inputs R	outputs Q	outputs $\bar{Q}$	
0	1	1	0	(SET)
1	1	1	0	(No change)
1	0	0	1	(RESET)
1	1	0	1	(No change)
0	0	1	1	(Invalid state)

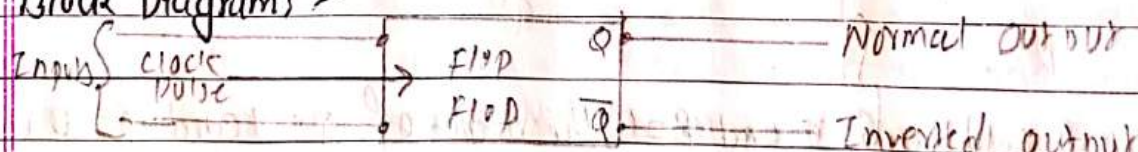
4. Flip Flop

A flip flop is a clock controlled memory device which is capable of storing one bit of data.

A flip flop has two outputs, one for normal value and other for complement value of the bit stored in it.

Flip flop stores the input states and output the stored state only in response to the clock signal.

Block Diagram:-



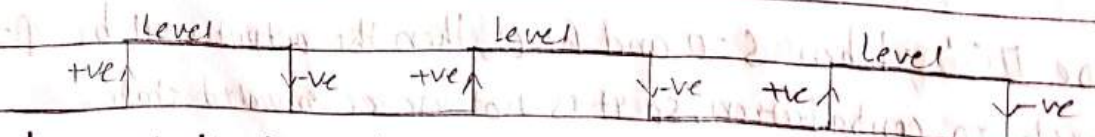
Types of Triggering Method

a. Level Triggering

b. Edge Triggering

   i. Positive Edge Triggering

   ii. Negative Edge Triggering



When all High → Level Triggering

When Low to High – Positive Edge Triggering

When High to Low – Negative Edge Triggering

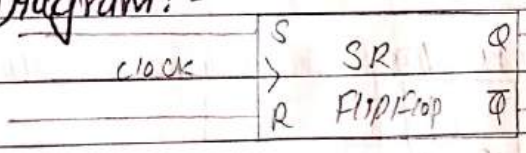
Different Types of Flip Flop

a. SR Flip Flop
b. D Flip Flop
c. JK Flip Flop
d. Master-Slave Flip Flop
e. T Flip Flop

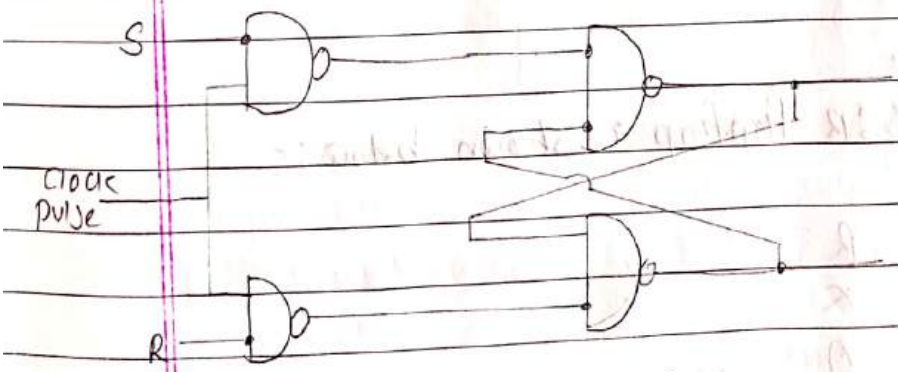
a. SR Flip Flop

The SR flip flop consists of the basic NAND latch and two other NAND gates to provide clock pulse.
Clock pulse is used for synchronization and acts as an additional control input.

Block Diagram:-



Circuit Diagram:-



The Truth Table of the SR flip flop is shown below:-

Inputs			Outputs		
Clock	S	R	Q	$\bar{Q}$	
0	X	X	Memory state		No change
1	0	0	Memory state		
1	0	1	0	1	
1	1	0	1	0	
1	1	1	Not use state		

Clock	S	R	Q_{n+1}
0	X	X	Q_n (memory state)
1	0	0	Q_n (memory state
1	0	1	0
1	1	0	1
1	1	1	Invalid state

DATE _____ PAGE _____

The characteristics table of SR flipflop is shown below:-

Inputs			Output
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	X
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	X

From the characteristics table given above, we have the characteristic equation using K-map.

Q_n \ SR	00	01	11	10
0	0	0	X	1
1	1	0	X	1

$\therefore Q_{n+1} = S + Q_n \bar{R}$

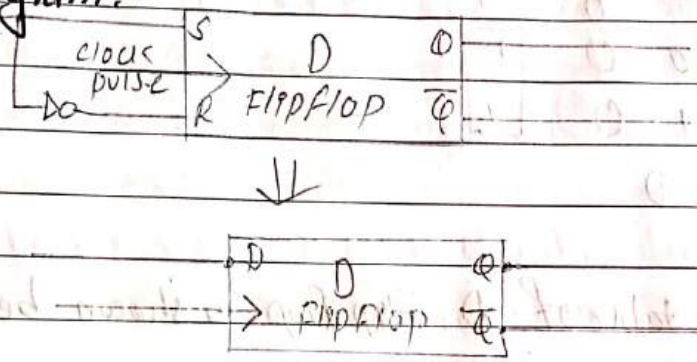
The excitation table of SR flipflop is shown below:-

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

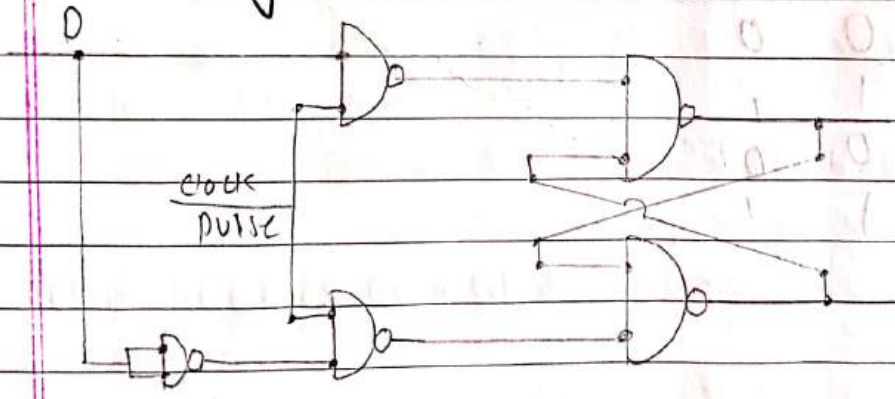
b. D Flip Flop

D Flip Flop is the modification of SR Flip Flop.
D Flip Flop is used to eliminate the possibility of race condition that the SR flipflop has created. So, we need to design a new circuit to eliminate the race condition that the SR flipflop has created. Thus. The concept of this new circuit was built. In D flipflop, there we can just give one input to this flipflop and complement the other as

Block Diagram:-



Circuit Diagram



The TruthTable of the D flipflop is shown below:-

Inputs		outputs
clock	D	Q_{n+1}
0	x	Q_n
1	0	0
1	1	1

Input(D)	output (Q_{n+1})
0	0
1	1

The characteristics Table of D Flip-flop is shown below:-

Inputs		Output (Q_{n+1})
Q_n	D	
0	0	0
0	1	1
1	0	0
1	1	1

From the characteristics table given above, we have characteristics equation using K-map as shown below:-

Q_n \ D	0	1
0	0	1
1	0	1

$\therefore Q_{n+1} = D$

The excitation table of D Flip flop is shown below:-

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

c. JK Flip Flop

The reason we need JK flip-flop is to overcome the disadvantages of SR flip flop and D flip flop.

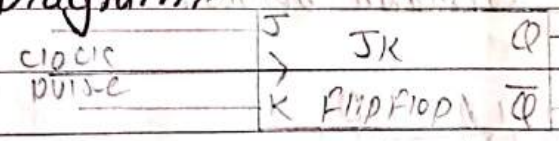
Thus, JK flip-flop is the refinement of SR flip flop to solve the problem of indeterminant (Invalid) state when both inputs are 1.

In JK flip flop inputs J and K behave like inputs S and R to SET and RESET the flip flop.

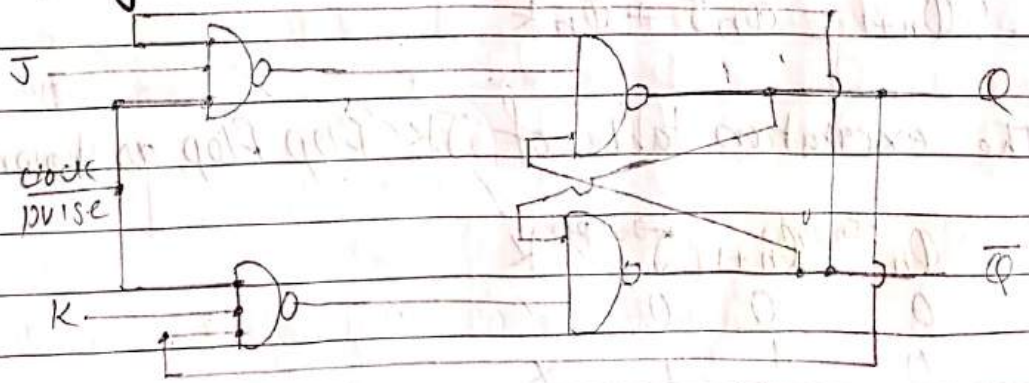
Simply JK flipflop is used to overcome race or not used condition and used it as toggle switch.

For JK flip flop if J=K=1 and if clock is too long, then the function time of circuit the flipflop keep on Toggling (Racing) which leads to uncertaining/indetermining output state of flip-flop, this problem is called Race around.

Block Diagram:-



Circuit Diagram:-



The Truth Table of JK Flip Flop is shown below:-

Inputs clocks	J	K	Outputs (Q_{n+1})	
0	x	x	Q_n	
1	0	0	Q_n	
1	0	1	0	
1	1	0	1	
1	1	1	$\overline{Q}_n$	(Toggle)

DATE _____ PAGE _____

The Characteristics Table of JK Flip Flop as shown below:-

Inputs Q_n	J	K	Output (Q_{n+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

From the characteristics table given above, we have characteristics equation using K-Map as shown below:-

Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$\therefore Q_{n+1} = \overline{Q}_n J + Q_n \overline{K}$

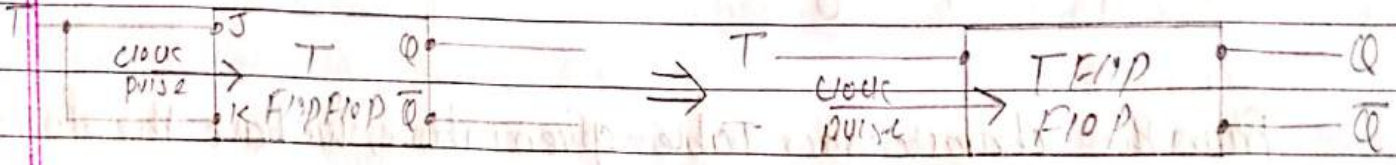
The excitation table of JK Flip Flop as shown below:-

Q_n	Inputs Q_{n+1}	J	Outputs K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

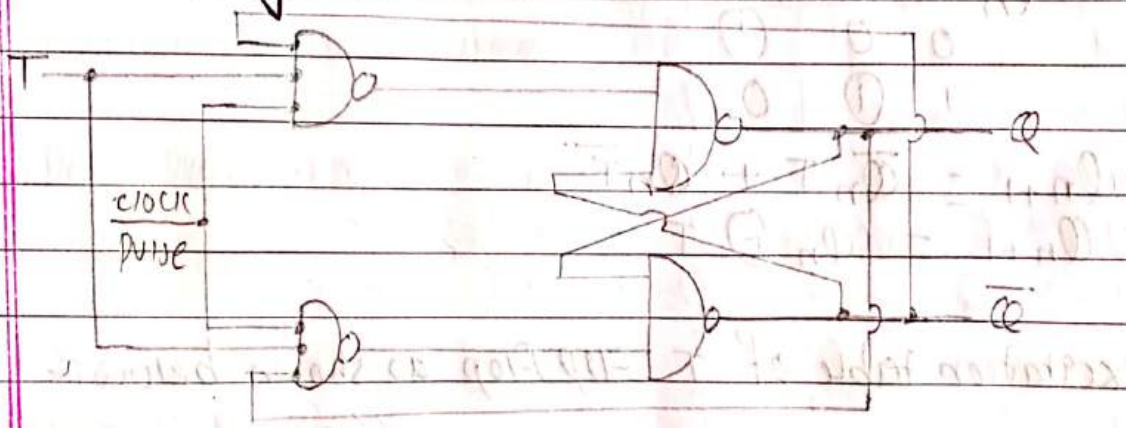
d. T Flip Flop

T flip Flop is a single input version of JK Flip Flop which was created specially for the cases where SET and RESET was not required but toggeling or storing was required.

Block Diagram:-



Circuit Diagram:-



The TruthTable of T Flip Flop as shown below:-

Inputs		Output (Q_{n+1})
Clock	T	
0	X	Q_n
1	0	Q_n
1	1	$\overline{Q_n}$

DATE _____ PAGE _____

The characteristics Table of T Flip Flop as shown below:-

Inputs		output
Q_n	T	(Q_{n+1})
0	0	0
0	1	1
1	0	1
1	1	0

From the characteristics Table given above, we have the following characteristics equation using Karnugh map:-

Q_n \ T	0	1
0	0	(1)
1	(1)	0

$$Q_{n+1} = \bar{Q}_n T + Q_n \bar{T}$$
$$\therefore Q_{n+1} = Q_n \oplus T$$

The excitation table of T Flip Flop as shown below:-

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

c: Jk Master-Slave Flip Flop

JK Master slave flip flop is constructed by using two saperate JK Flip flop. One circuit served as master and another as slave and overall circuits are refers to JK Master slave flip flop.

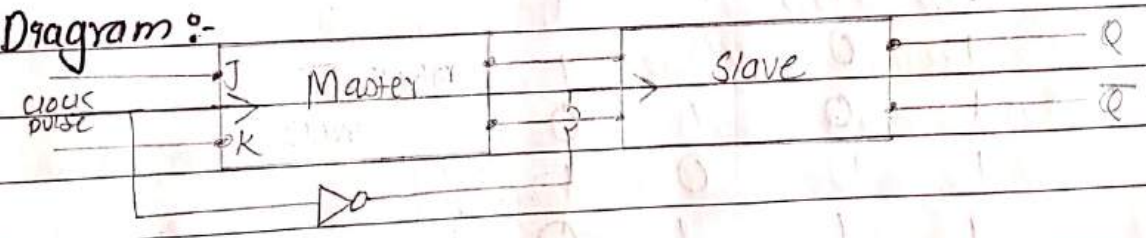
The first flip flop known as the master is driven by positive clock pulse and second known as slave is driven by the negative clock pulse.

During positive clock, Slave remains un-operational and master gives the intermediate output.

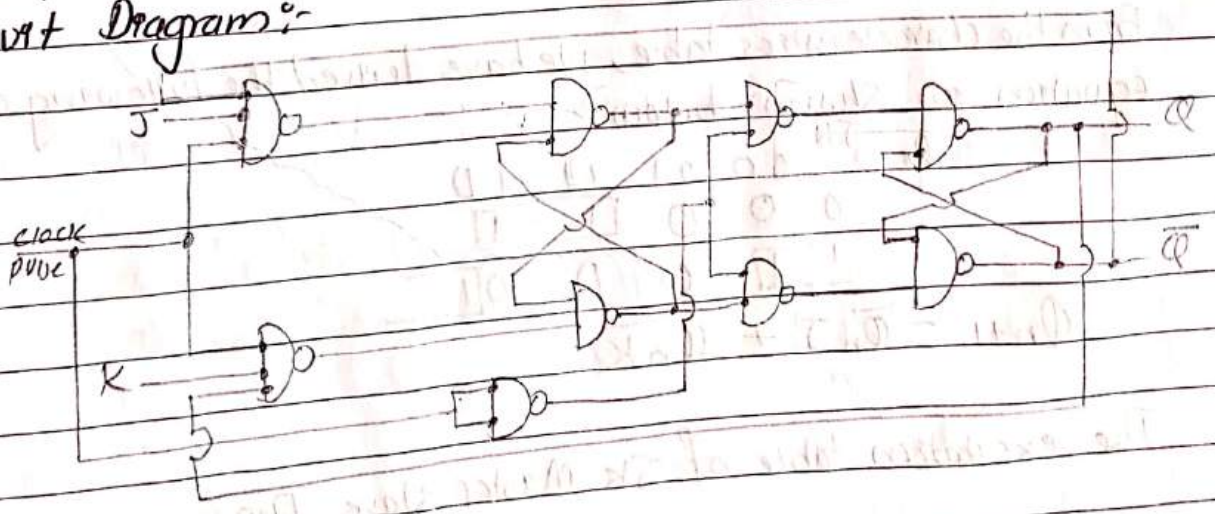
During Negative clock, slave becomes operational and it receives the previous output of master and produces the final output.

In order to overcome the race around condition of JK flipflop, the JK Master slave flip flop is introduced.

Block Diagram:-



Circuit Diagram:-



DATE _____ PAGE _____

The Truth Table of JK Master slave flip flop as shown below:-

Inputs			output
Clock	J	K	(Q_{n+1})
0	x	x	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	$\overline{Q_n}$

The characteristics Table of JK Master slave flip flop as shown below:-

Q_n	Inputs J	K	output (Q_{n+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

From the characteristics table, we have derived the following characteristics equation as shown below:-

Q_n \ JK	00	01	10	11
0	0	0	1	1
1	1	0	0	1

$$Q_{n+1} = \overline{Q_n} J + Q_n \overline{K}$$

The excitation table of JK Master slave flipflop as shown below:-

Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

State Diagram

State diagram is a compose of finite number of state that is used to describe the behaviour of the circuit.
In state Diagram, each state is represented by a circle and the transition between the states is specified (represented) by directed lines (arrow).
The binary numbers in each circle identifies the state.

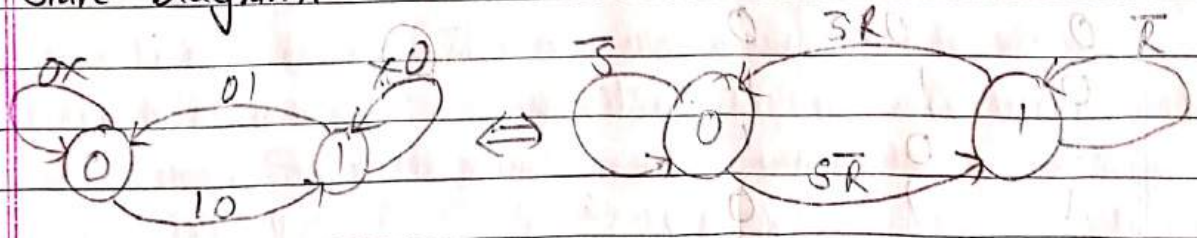
State Diagram of Various Flip Flop

a. SR Flip Flop

Excitation Table

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

State Diagram:-

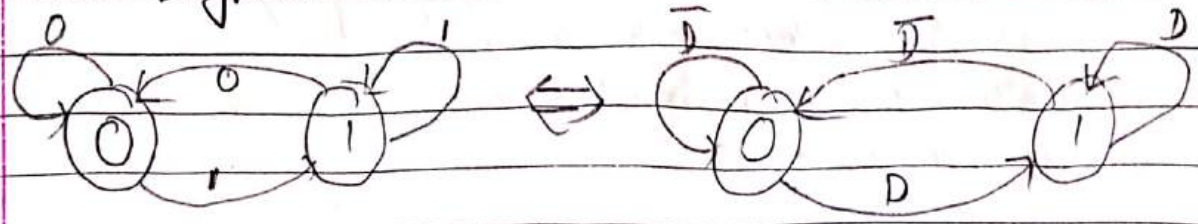


b. D Flip Flop

Excitation Table

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

State Diagram:-

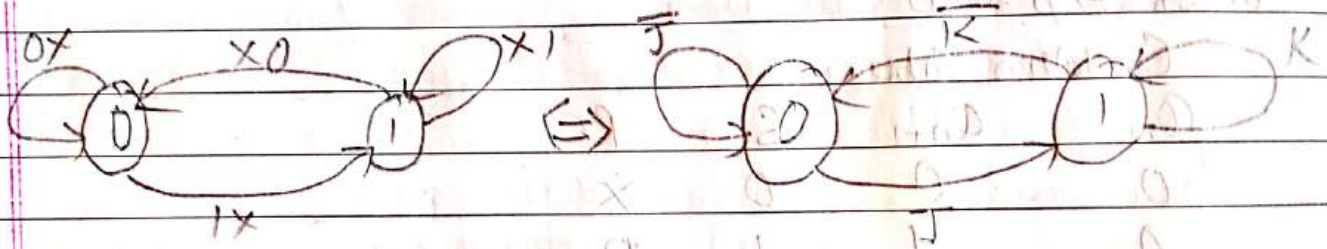


c. JK Flip Flop

Excitation Table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

State Diagram:-

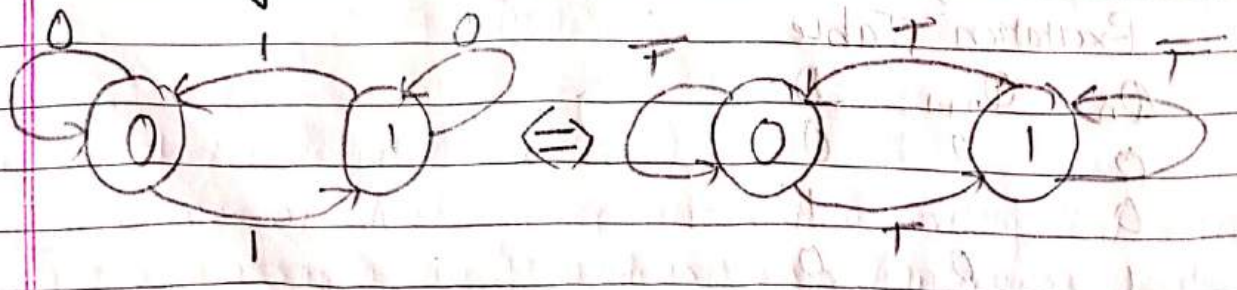


d. T Flip Flop

Excitation Table

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

State Diagram:-

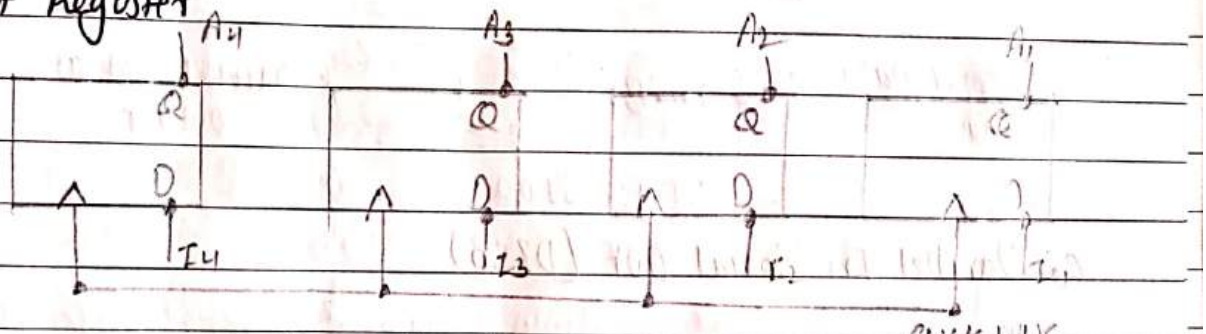


DATE _____ PAGE _____

5. Register

Register is a group of flipflops used for storing binary information. Each flipflop of a binary cell is capable of storing one bit of data.
An n-bit register has a group of n flip-flops and is capable of storing any binary information containing n bits.
The register is mainly used for storing and shifting binary data entered into it from an external source.

4-Bit Register



∴ Fig: 4-Bit Register using D Flip Flop

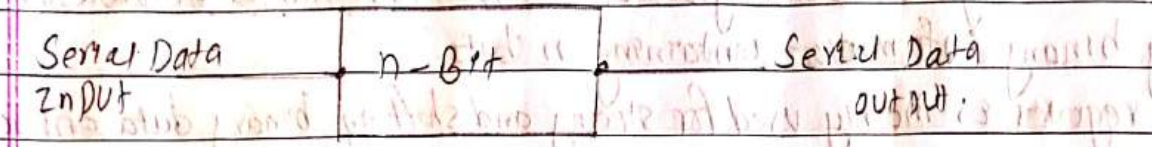
The given 4-bit register is constructed by using 4 D Flip Flop and the common clock pluse input. This clock pluse enables all other flipflops so that information available at 4-inputs (I_1, I_2, I_3, I_4) can be transfered to the four bit register. Finally, the preset stored information in the register can be obtained as we need.

Shift Register

A register that is used to store binary information is known as a memory register.
A register capable of shifting binary information either to the right or to the left is called Shift register.
Shift register permits the stored data to move from a particular location to some other location within the register.
In a shift register the flipflops are connected in such away that the bits of binary number are entered into the shift register shifted from one position to another and finally shifted output. There are two methods of shifting the data that is Serial shifting and parallel shifting.

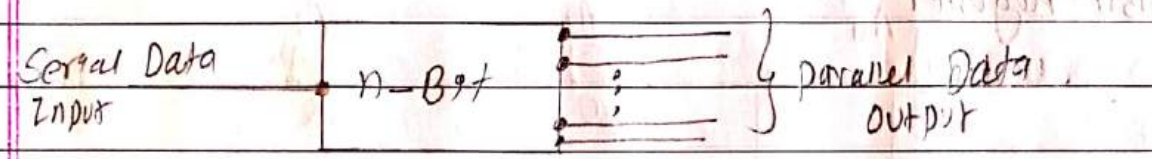
The shift register are classified into the following types based how binary information is entered and shift out.

a. Serial In Serial Out (SISO)



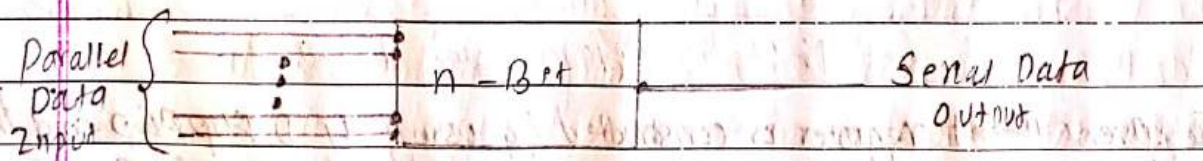
∴ Fig:- SISO

b. Serial In Parallel out (SIPO)



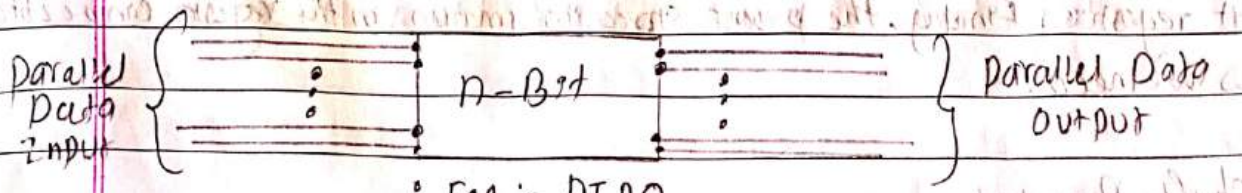
∴ Fig:- SIPO

c. Parallel In Serial Out (PISO)



∴ Fig:- PISO

d. Parallel In Parallel Out (PIPO)



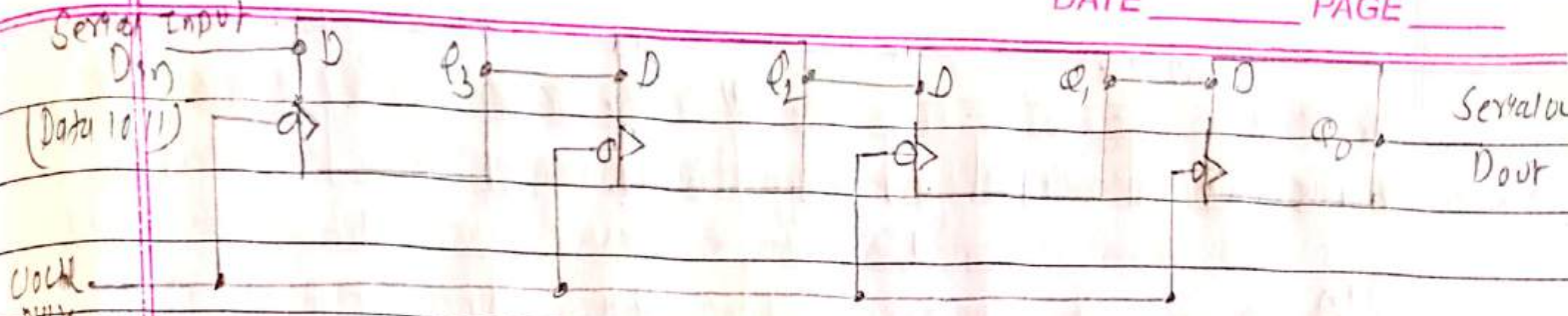
∴ Fig:- PIPO

a. Serial In Serial Out (SISO)

This types of shift register accepts data serially that is one bit at a time on a single line. It produces the stored information on its single output line also in a serial form.

Data may be shifted left to right using right shift register or right to left by using left shift register.

Let us state 4-bit right shift Serial In Serial out register where input data is 1011.



∴ Fig:- 4-Bit Right shift Serial In Serial out register

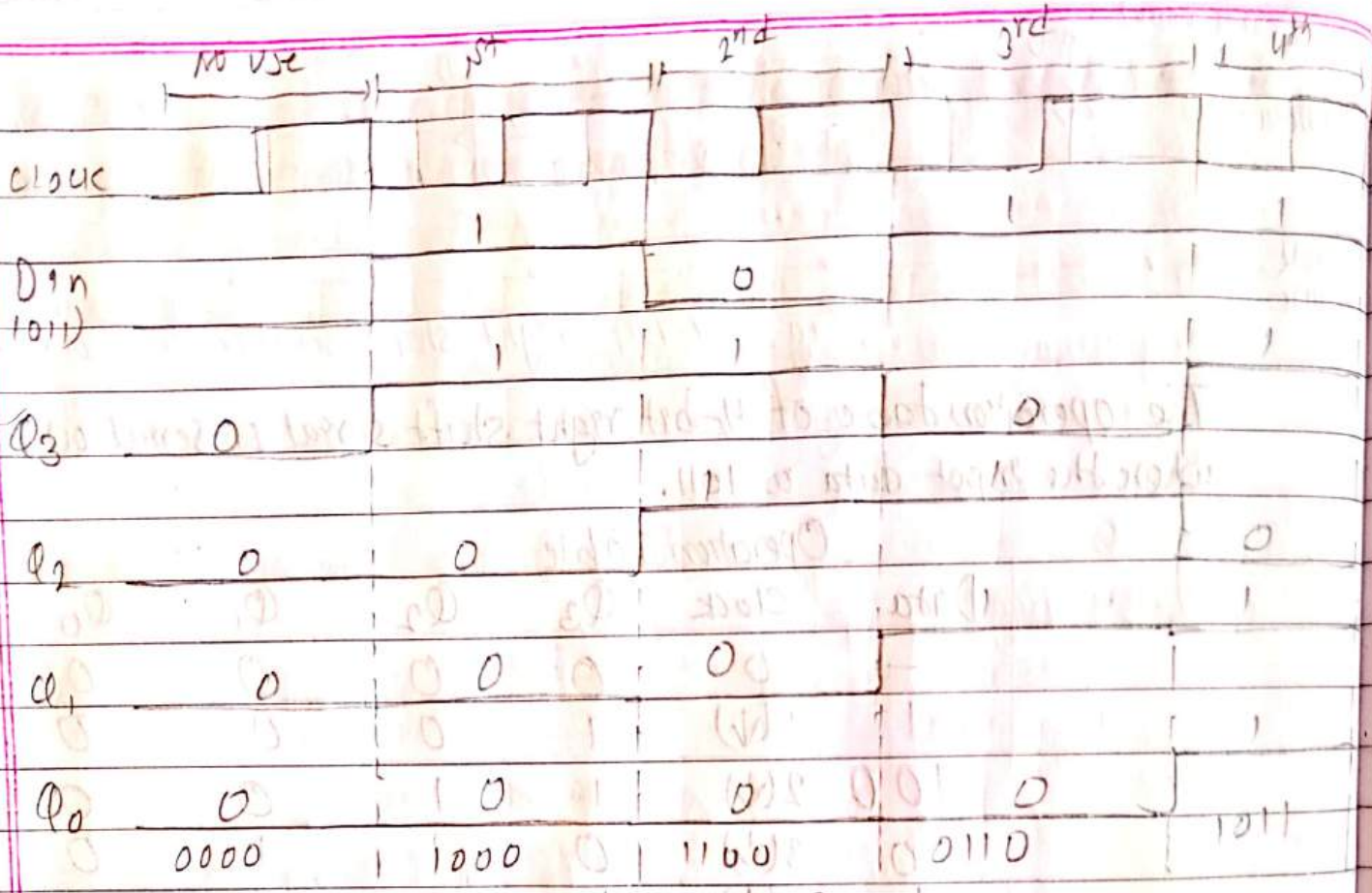
The operation table of 4-bit right shift serial In serial out register where the input data is 1011.

Operation Table

Data	clock	Q_3	Q_2	Q_1	Q_0
–	0	0	0	0	0
1	1 (↓)	1	0	0	0
1 0	2 (↓)	1	1	0	0
0	3 (↓)	0	1	1	0
1	4 (↓)	1	0	1	1

Illustrating the state with entry of 4-bit binary number 1011 in the register. A first a bit as 1 is applied at serial input at first flipflop. When the first clock pulse is provided, first flipflop stored 1 and put it at input point of second flipflop. On second clock pulse we put 1 at input point of first flipflop so that first flipflop store 1 and second flipflop will take previous output of first flipflop which is 1 and store it. Accordingly on final clock pulse, all four bits are stored in four flipflops.

The timing diagram or waveform of the above operation table is shown below:-



∴ Fig :- Timing Diagram of Serial In serial out register

On the above timing diagram or wave form, we have illustrated the operation of 4-bit right shift serial In serial out register. We can confirm that the four flipflop will store all four inputed binary value within four clock pulse but to output it we will need three more clock pulse that is also one of the backdoor of serial In serial out shift register.

b. Serial In Parallel Out (SIPO)

In serial In parallel out shift register, the data is stored into the register serially where it is received or outputed from it parallely.

It consists of one serial input and outputs are taken from all flip flops in parallel in order to shift the data out in parallel, it is necessary to have all the data available at output lines at the same time.

Once the data is stored, each bit appears on it's respective output lines and all bits are available simultaneously rather than on a bit by basis as with the serial output.

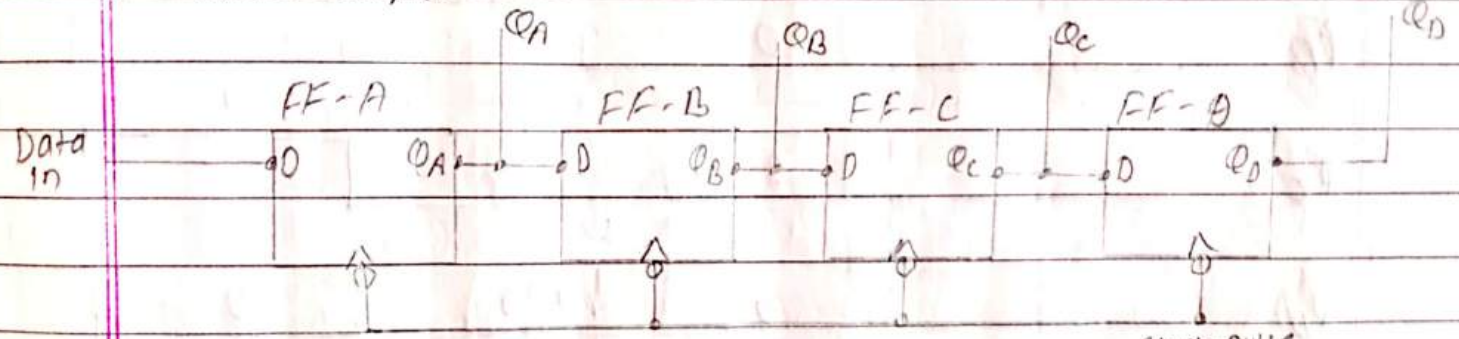


Fig:- 4-Bit right Shift Serial in parallel out register

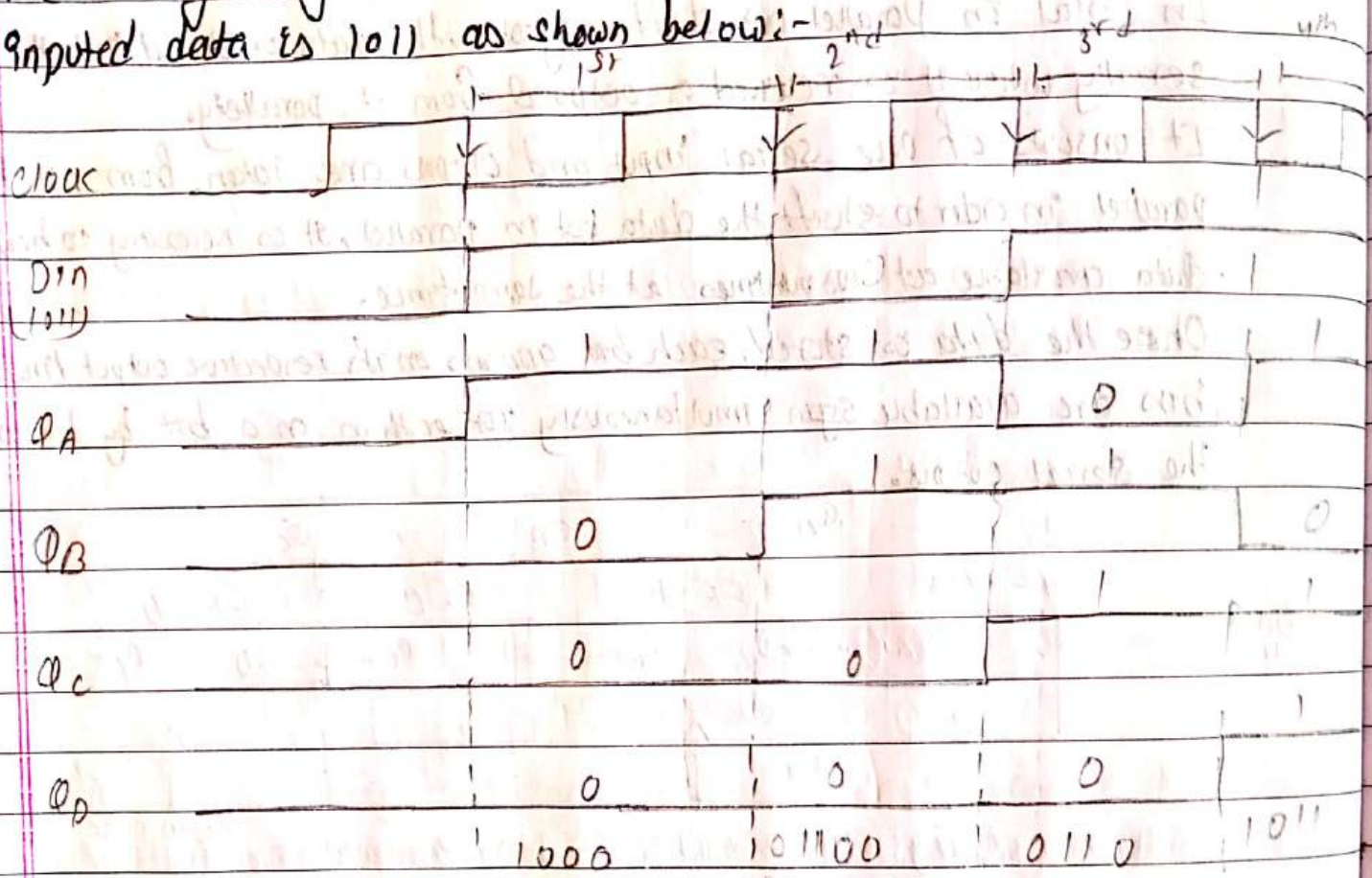
The operational table of 4-bit right shift serial in parallel out register where we input 1011 as data in.

Data	clock	Q_A	Q_B	Q_C	Q_D
—	0	0	0	0	0
1	1 (↓)	1	0	0	0
1	2 (↓)	1	1	0	0
0	3 (↓)	0	1	1	0
1	4 (↓)	1	0	1	1

From the above operation table we confirm that Serial in parallel out register need 4 clock pulse to input the data and make available the data to all the flipflop. After that the output is done at once in parallel form when all flipflop have data or the data is stored.

DATE _____ PAGE _____

The timing diagram or waveforms of the operational table where the inputed data is 1011 as shown below:-



On the above clock diagram, we have loaded 4 clock pulse to pass the ready data into the flipflop which is in serially inputed and the da When the data is loaded in all the flipflop, the output is provided at once in parallel mode.

c. Parallel In Serial out (PISO)

In parallel In Serial out (PISO) shift register. The data is loaded into the register in parallel format while it is received from it serially.

Let A, B, C and D be the four parallel data input lines and SHIFT/$\overline{LOAD}$ is a control input that allows the four bits of data at A, B, C and D inputs to enter into the register in parallel or shift the data in serial.

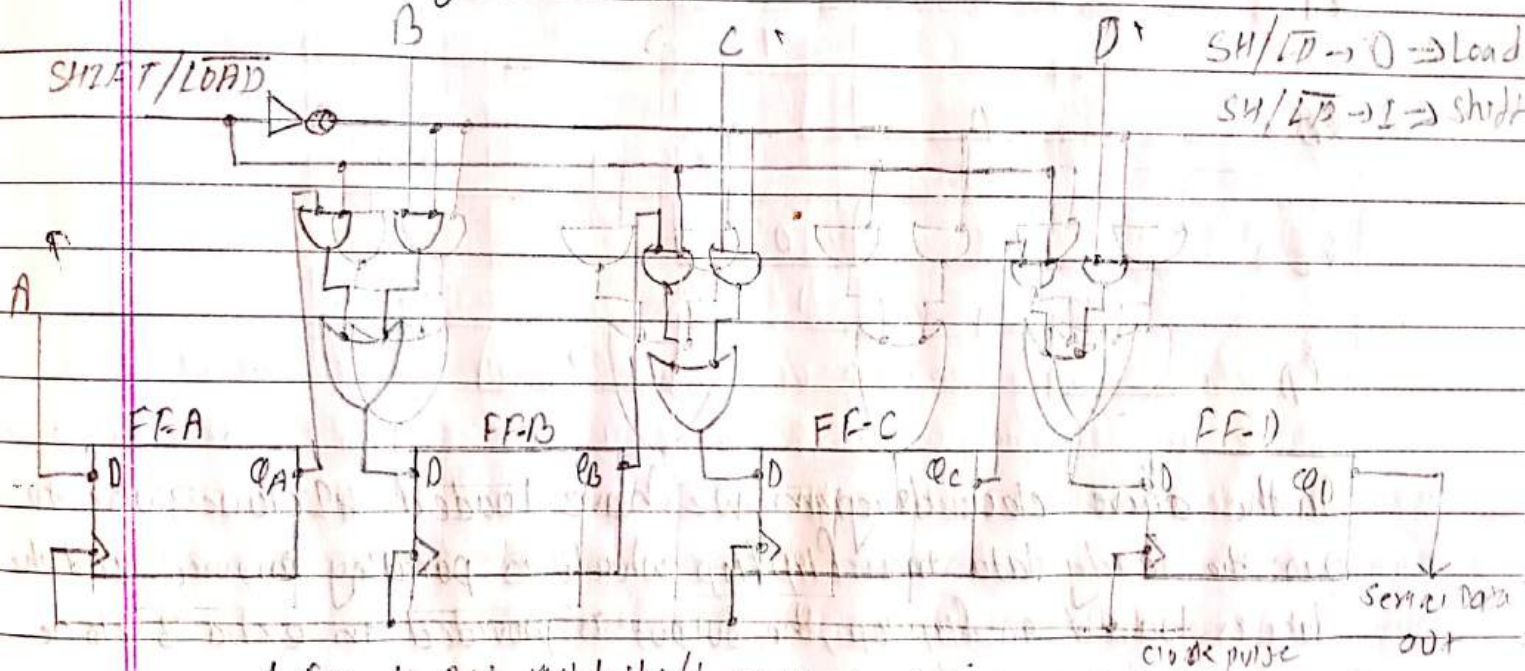


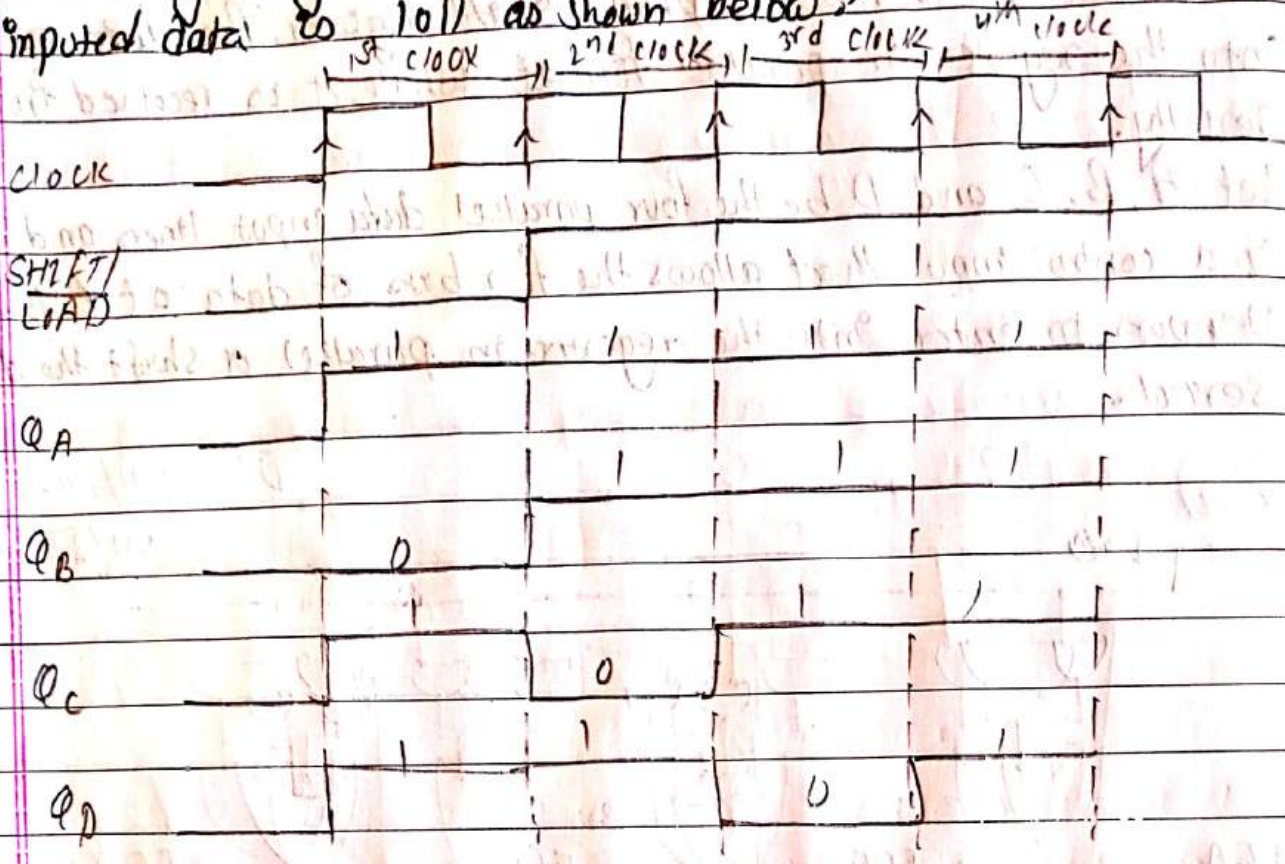
Fig:- 4-Bit right shift parallel in serial out register

The operational table of 4-bit right shift parallel in serial out register where we input 1011 as data in.

Clock	SHIFT/$\overline{LOAD}$	Parallel inputs A	B	C	D	Q_A	Q_B	Q_C	Q_D	Serial output
0	—	—	—	—	—	0	0	0	0	—
1 (↑)	0	1	0	1	1	1	0	1	1	1
2 (↑)	1	1	0	1	1	1	1	0	1	1
3 (↑)	1	1	0	1	1	1	1	1	0	0
4 (↑)	1	1	0	1	1	1	1	1	1	1

From the above operation table, we can confirm that parallel in serial out register need four clock pulse to output where only a single clock pulse to store the data in the flip flop.

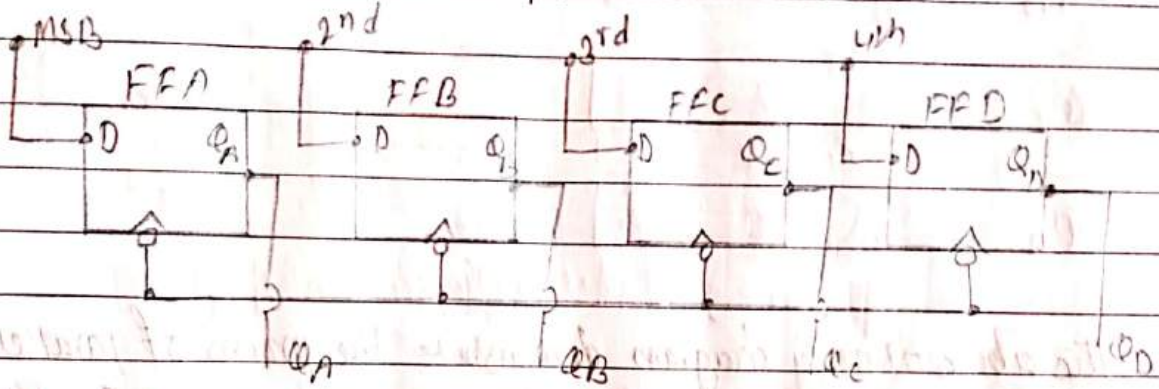
The timing diagram or wave form of the operational table where the inputed data is 1011 as shown below:-



On the above clock diagram, we have loaded 4 clock pulse to pass the ready data into flipflop which is parallely inputed and the data is loaded on flipflop, the output is provided in extra 3 clock pulse in serial mode.

d. Parallel In Parallel out (PIPO)

Parallel In parallel Out shift register are the type of storage devices in which both data loading as well as data retrival process occur in parallel mode that is data inputs can be shifted either in or out of register in parallel.
In this register, there is no interconnection between successive flipflops since, no serial shifting is required.
Therefore, the moment the parallel entry of data is accomplished, the respective bits will appear as parallel outputs.



i. Fig.: 4-bit right shift parallel in parallel out register

The operational example given above demonstrate that the parallel inputs to be entered should be applied at MSB, 2nd, 3rd and 4th input which are directly connected to D inputs of the flipflops A, B, C, and D. When first clock pulse is applied, the inputs are entered to the register and they are immediately available at outputs Q_A, Q_B, Q_C and Q_D.

The operational table of 4-bit right shift parallel in parallel out register where the entered data is 1011.

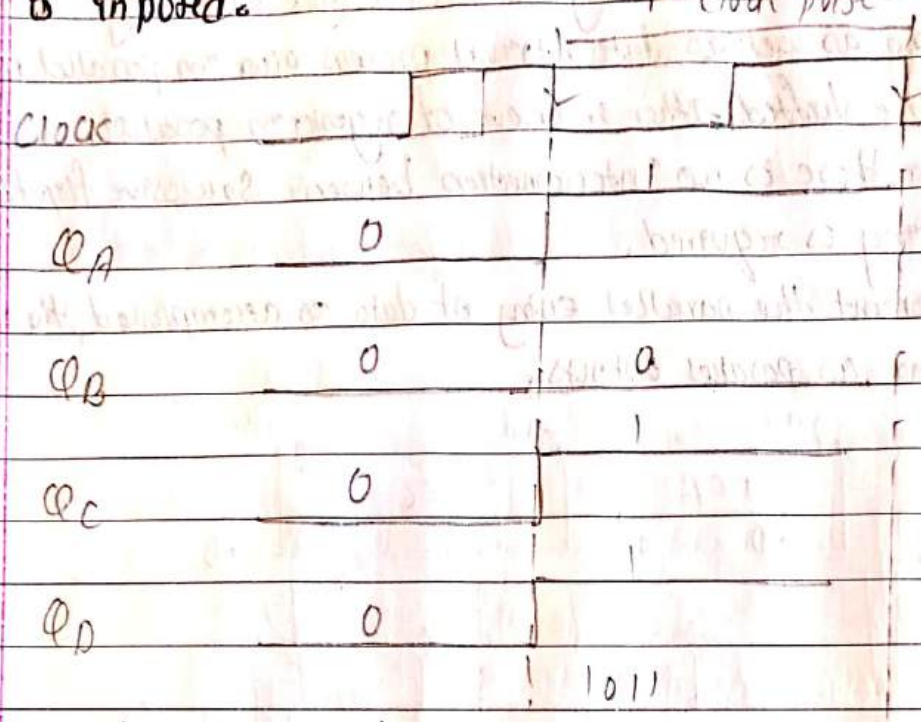
Operational Table

Data	clock	Q_A	Q_B	Q_C	Q_D
—	0	0	0	0	0
1011	1 (↓)	1	0	1	1

From the operational table, it is clearly seems that parallel in parallel out register need one clock pulse to input the data as well as for output also. When the input is completed, it is immediately available to the output.

DATE _____ PAGE _____

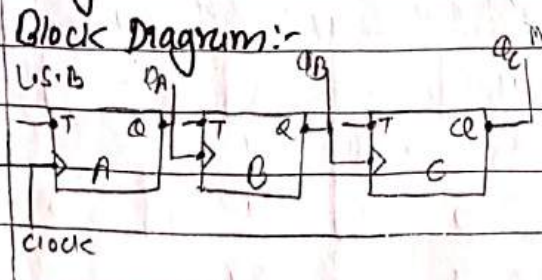
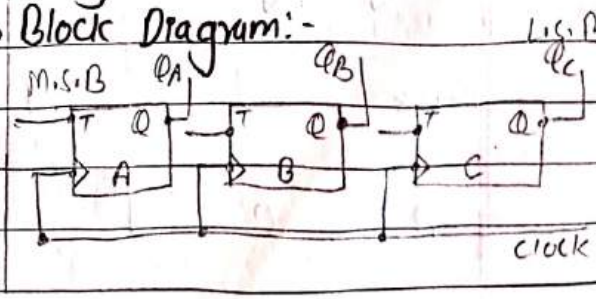
The timing diagram or wave form of the operational table accord as 1011 is inputed.



The above timing diagrams demonstrate the process of parallel in and parallel out register where single clock pulse is used for data input at once as well as data output.

6. Counters

Counters are sequential circuits capable of counting clock pulses.
They provides accurate timing and control signal to other devices.
Counting can be ascending, descending or non-linear.
A counter having 'n' flipflops can count 2^n clock pulses.
There are two types of counter i.e. (i) Asychronous counter and (ii) Synchronous counter.

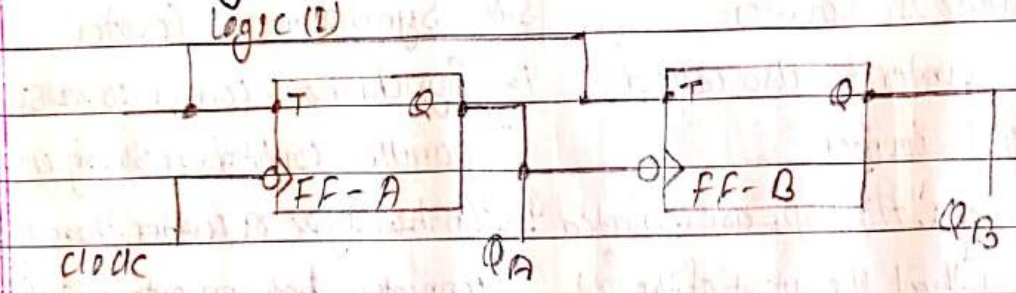
S.N	Asychronous Counter	S.N	Synchronous Counter
i.	Asychronous counter is also called Serial or ripple counters.	i.	Synchronous counter is also called parallel counters or Binary counter.
ii.	In this counters, flipflops are connected in such a way that the output of the first flipflop drives the clock for the next flip flop.	ii.	In this type of counter, there is no connection between output of first flipflop and clock of next flipflop.
iii.	All flip flops are not clocked simultaneously.	iii.	All flipflops are clocked simultaneously.
iv.	Asych Asynchronous counter will operate only in fixed count sequence (up/Down).	iv.	Synchronous counter will operate in any desired count sequence.
v.	A logic circuit is very simple even for more number of states.	v.	A logic circuit is a complex as the number of states increases.
vi.	Less speed because the propagation delay is more.	vi.	High speed because the propogation delay is less.
vii.	Block Diagram:- 	vii.	Block Diagram:- 

Asychronous / Ripple counter

Asynchronous counter is one in which the first flipflop is given an external clo
and each of other flipflop is triggered by the output of the pervious flip
flop which limits the speed of operation.

2-bit Ripple up counter

Block Diagram:-

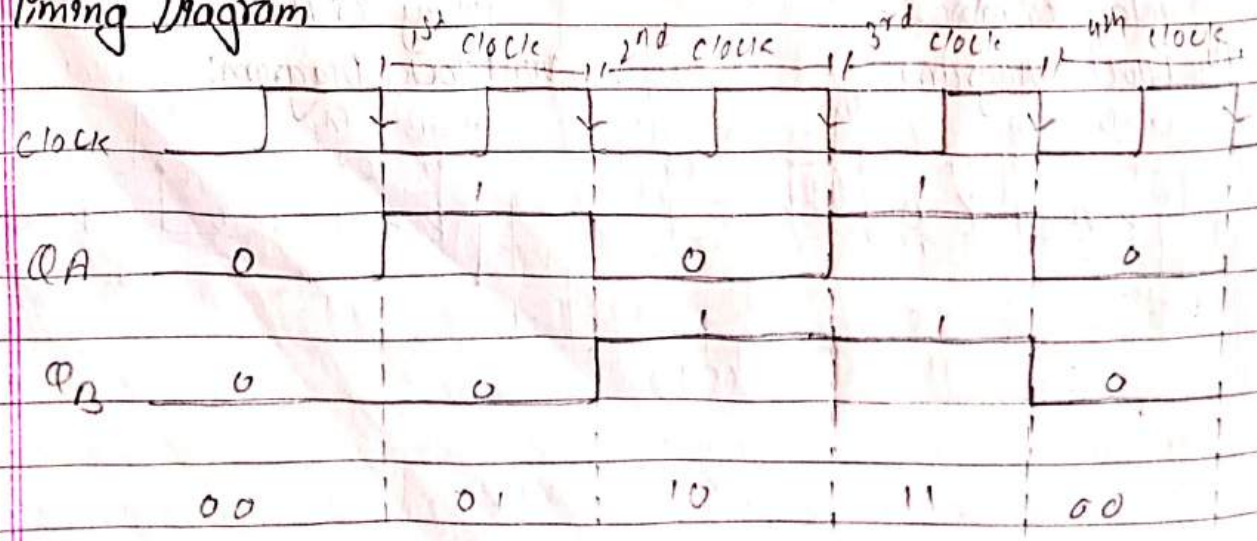


∴ Fig:- 2-bit ripple UD counter

Truth Table

Clock	Counter Output Q_A	Q_B	State Number	Decimal count output
Initially	0	0	—	0
1(↓)	0	1	1	1
2(↓)	1	0	2	2
3(↓)	1	1	3	3
4(↓)	0	0	4	0

Timing Diagram



∴ Fig:- Timing Diagram

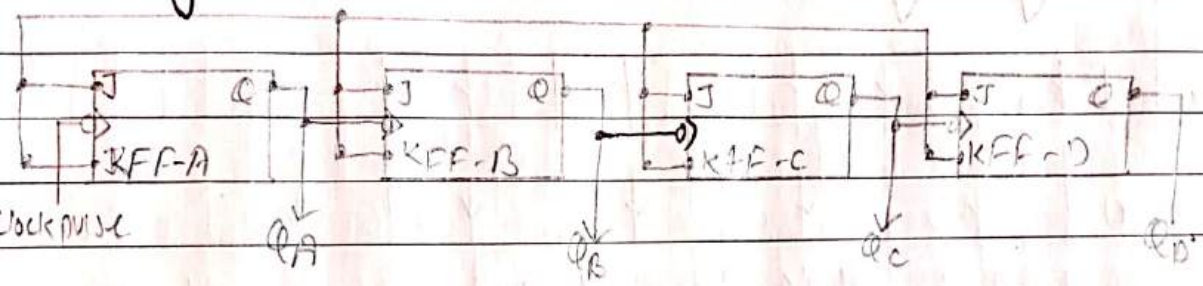
DATE _____ PAGE _____

4-Bit Asynchronous counter

4-bit Asynchronous counter or binary ripple counter can be constructed using either 'T' flipflop or 'JK' flip flop.

The system clock drives the flipflop A. The output of flipflop A drives flip flops B, the output of flipflop B drives flipflop C and the output of flipflop C drives flip flop D.

Block Diagram:-



∴ Fig:- 4-Bit Asynchronous counter

Q_A is always complemented.

Q_B is complemented whenever Q_A goes from HIGH to LOW.

Q_C is complemented whenever Q_B goes from HIGH to LOW.

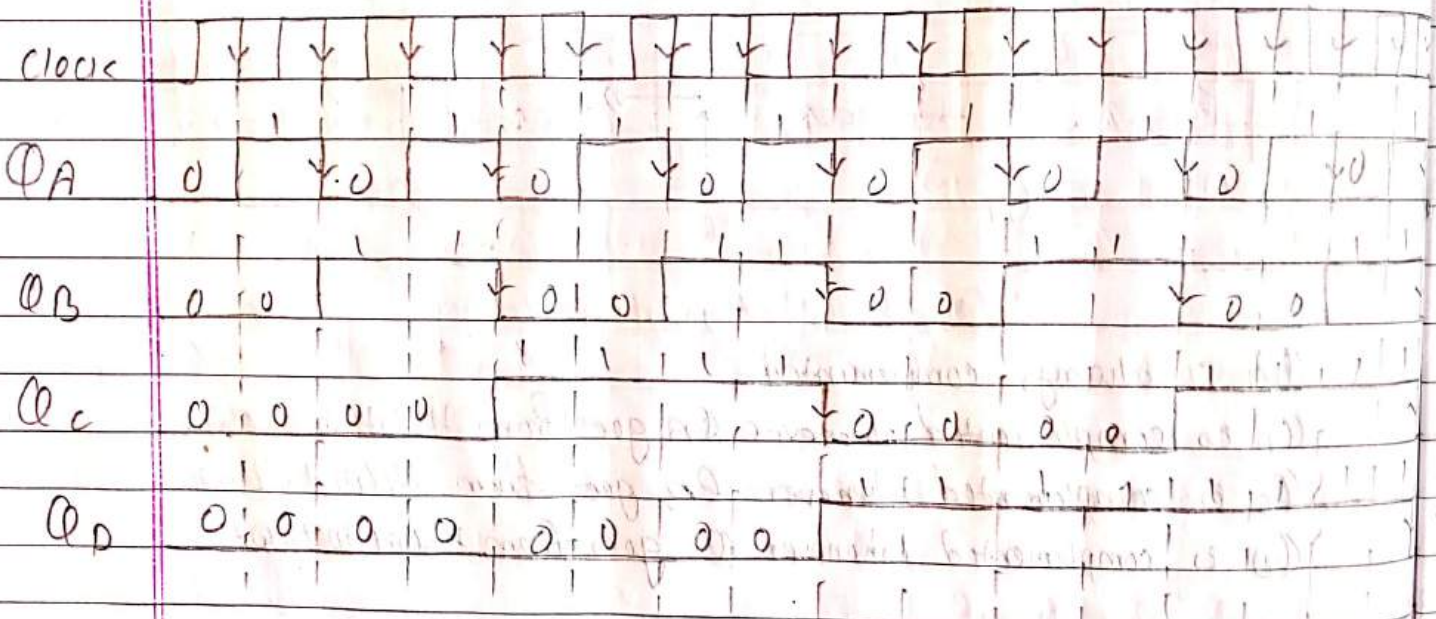
Q_D is complemented whenever Q_C goes from HIGH to LOW.

TruthTable

Clock	Counter output Q_D	Q_C	Q_B	Q_A	State Number	Decimal Counter output
Initially	0	0	0	0	—	0
1 (↓)	0	0	0	1	1	1
2 (↓)	0	0	1	0	2	2
3 (↓)	0	0	1	1	3	3
4 (↓)	0	1	0	0	4	4
5 (↓)	0	1	0	1	5	5
6 (↓)	0	1	1	0	6	6
7 (↓)	0	1	1	1	7	7
8 (↓)	1	0	0	0	8	8
9 (↓)	1	0	0	1	9	9
10 (↓)	1	0	1	0	10	10

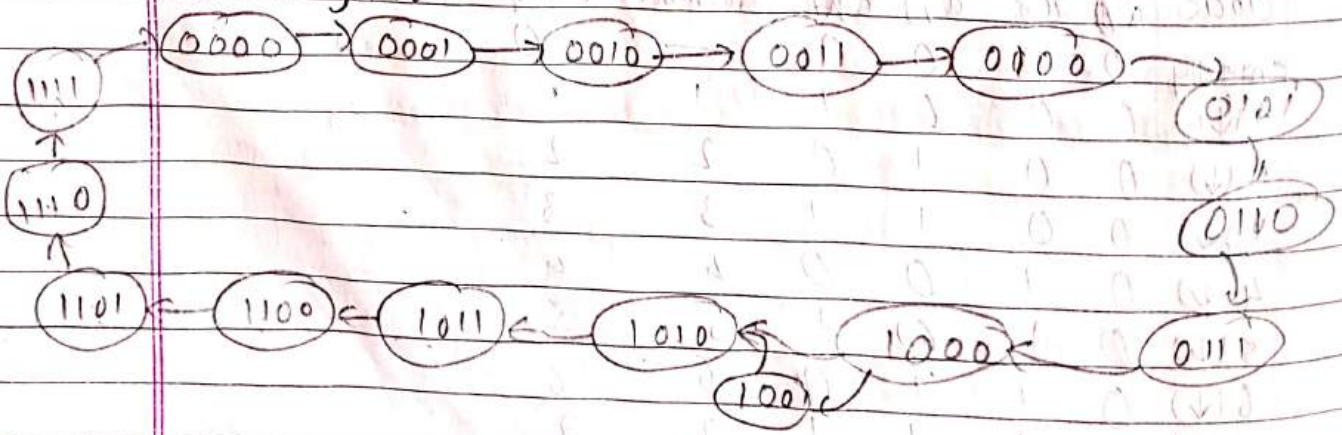
11 (↓)	1	0	1	1	11	11
12 (↓)	1	1	0	0	12	12
13 (↓)	1	1	0	1	13	13
14 (↓)	1	1	1	0	14	14
15 (↓)	1	1	1	1	15	15
16 (↓)	0	0	0	0	16	0

Timing Diagram:-



∴ Fig:- Timing Diagram

State Diagram:-



∴ Fig:- State diagram

DATE _____ PAGE _____

Asynchronous (Ripple) Down Counter

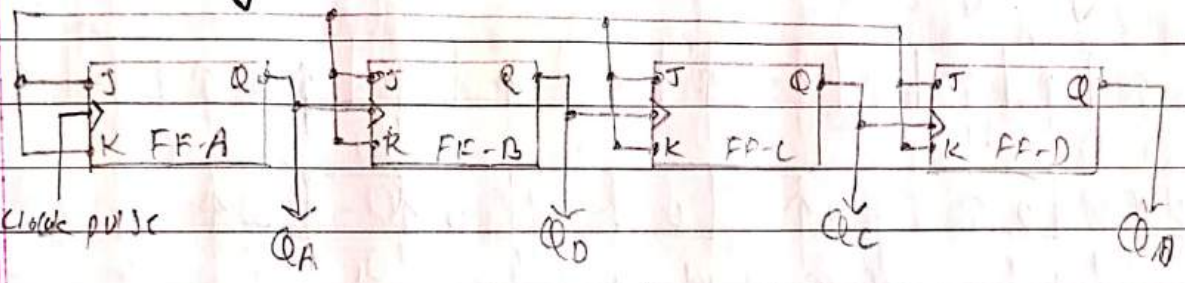
A asynchronous down counter using 'n' flipflop counts downwards from a maximum $2^n - 1$ to 0.

A Binary counter with reverse count is called ripple down counter.

In ripple down counter, the binary count is decremented by '1' with every clock input pulses.

4-Bit Binary ripple Down Counter counts from 1111 to downwards 0000.

Block Diagram:-



∴ Fig:- 4-Bit Ripple Down Counter

Q_A is always complemented.

Q_B is complemented whenever Q_A goes from Low to HIGH.

Q_C is complemented whenever Q_B goes from LOW to HIGH.

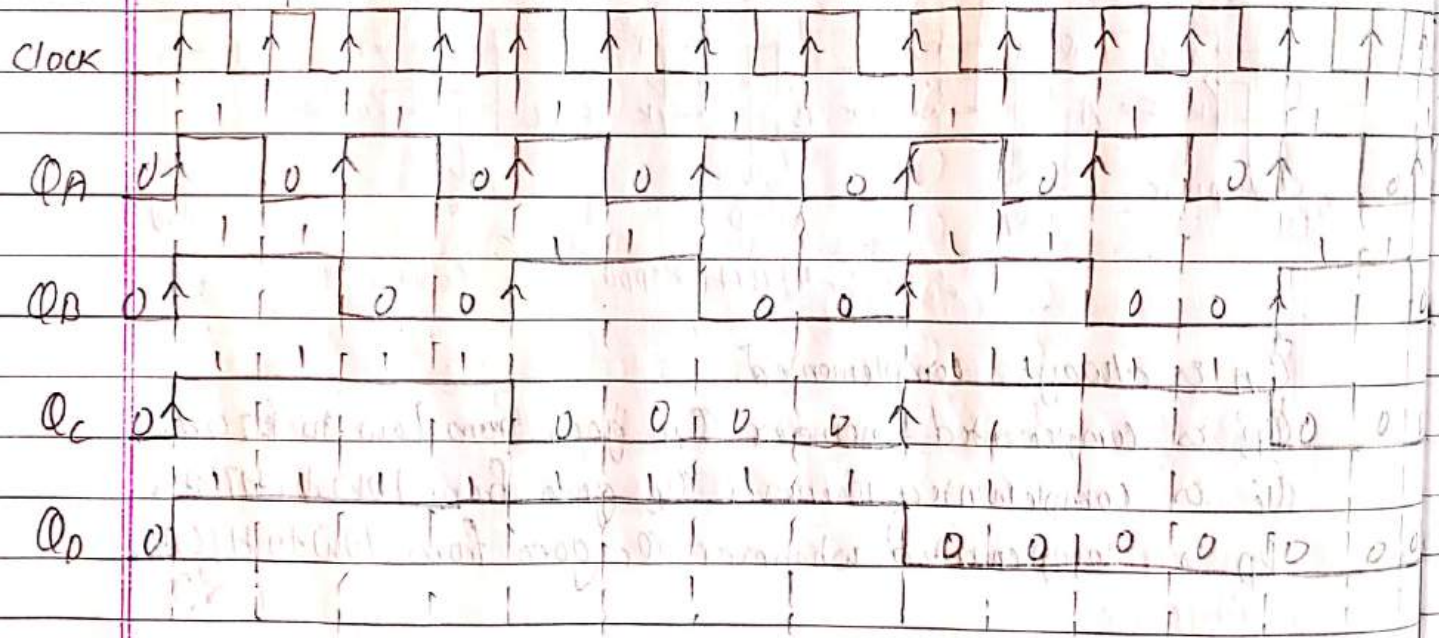
Q_D is complemented whenever Q_C goes from LOW to HIGH.

TruthTable

Clock	Counter Output Q_D	Q_C	Q_B	Q_A	State Number	Decimal counter output
Initially	0	0	0	0	—	0
1(↑)	1	1	1	1	1	15
2(↑)	1	1	1	0	2	14
3(↑)	1	1	0	1	3	13
4(↑)	1	1	0	0	4	12
5(↑)	1	0	1	1	5	11
6(↑)	1	0	1	0	6	10
7(↑)	1	0	0	1	7	9
8(↑)	1	0	0	0	8	8
9(↑)	0	1	1	1	9	7

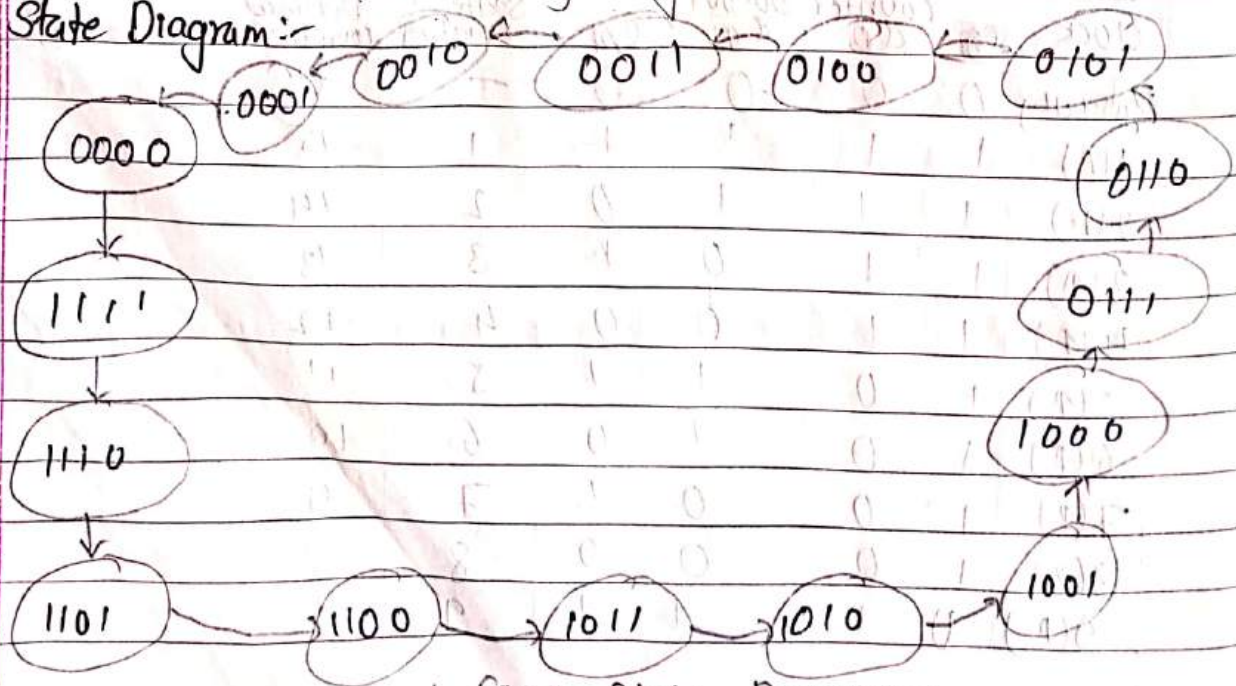
10 (↑)	0	0	0	0	10	6
11 (↑)	0	1	0	1	11	5
12 (↑)	0	1	0	0	12	4
13 (↑)	0	0	1	1	13	3
14 (↑)	0	0	1	0	14	2
15 (↑)	0	0	0	1	15	1
16 (↑)	0	0	0	0	16	0

Timing Diagram:-



∴ Fig:- Timing Diagram

State Diagram:-



∴ Fig:- State Diagram

DATE _____ PAGE _____

Asynchronous (Ripple) UP/DOWN Counter

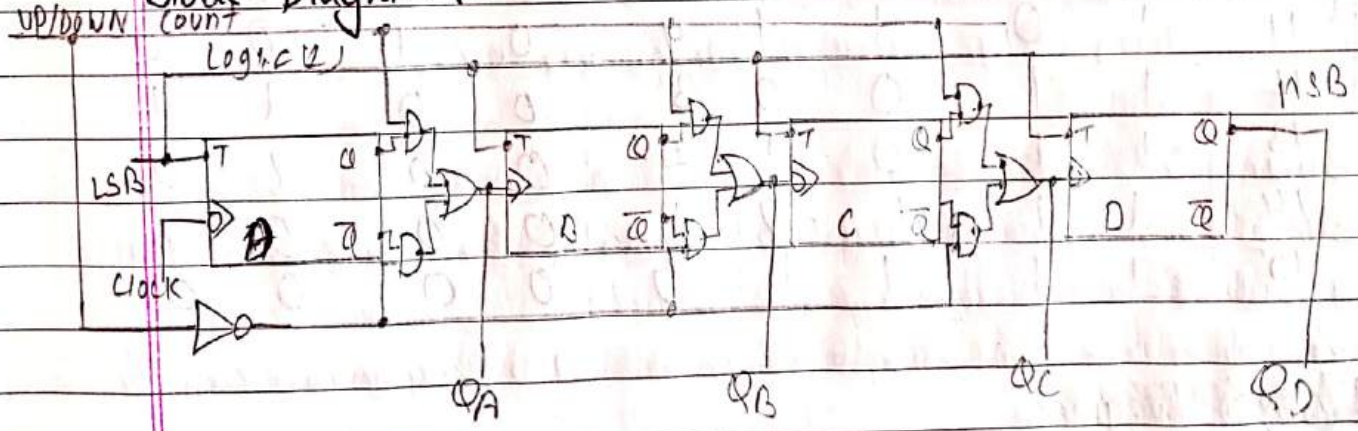
The UP/DOWN counter is a combination of the UP counter and the DOWN counter. As the UP/DOWN counter has the capability of counting upward as well as downwards, it is also called as multimode counters.

In an UP counter, Each flip flops is triggered by the normal output (from output Q of preceding flipflop to the clock input of the next flip flop) of the preceding flip flop.

In DOWN counter, Each flip flop is triggered by the complement output of the preceding flip flop (from output $\overline{Q}$ of the preceding flipflop to the clock input of the next flip flop).

In both the counters, The first flip flop is triggered by the external clock.

Block Diagram :-



i. Fig :- 4-Bit asynchronous UP/DOWN Counter

In the figure given above, UP/DOWN count line is held HIGH i.e. 1, the counter counts upwards from 0000 to 1111. When UP/DOWN count line is held LOW i.e. 0, the counter counts downwards from 1111 to 0000.

Truth Table

State	UP Count Mode Q_D	Q_C	Q_B	Q_A	State	DOWN Count Mode Q_D	Q_C	Q_B	Q_A
0	0	0	0	0	15	1	1	1	1
1	0	0	0	1	14	1	1	1	0
2	0	0	1	0	13	1	1	0	1
3	0	0	1	1	12	1	1	0	0
4	0	1	0	0	11	1	0	1	1
5	0	1	0	1	10	1	0	1	0
6	0	1	1	0	9	1	0	0	1
7	0	1	1	1	8	1	0	0	0
8	1	0	0	0	7	0	1	1	1
9	1	0	0	1	6	0	1	1	0
10	1	0	1	0	5	0	1	0	1
11	1	0	1	1	4	0	1	0	0
12	1	1	0	0	3	0	0	1	1
13	1	1	0	1	2	0	0	1	0
14	1	1	1	0	1	0	0	0	1
15	1	1	1	1	0	0	0	0	0

DATE _____ PAGE _____

Synchronous Counter

Synchronous counters can be distinguished from asynchronous counter, in that, a common clock pulse is provided to the clock pulse input of all flipflop simultaneously. Simply, a common clock pulse triggers all the flip flops at a time rather than one at a time in succession as in a ripple counter.

Synchronous UP counter (Binary UP counter):-

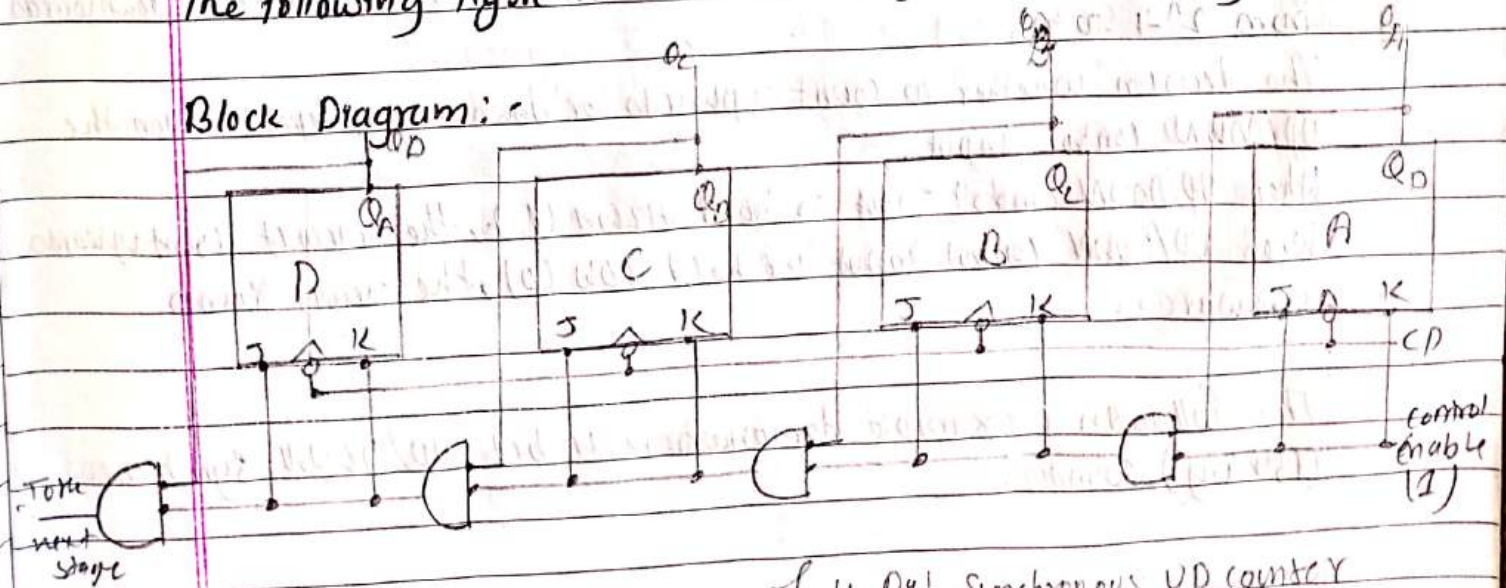
In a synchronous binary counter (UP), the flipflop in the lowest order position is complemented with every clock pulse. This means that its J and K inputs must be maintain at logic '1'.

A flipflop in any other position is complemented with a pulse provided that all the bits in the lower order position are equal to 1.

Simply, the lowest order flipflop, let say 'A', is always complemented. A_2 is complemented whenever the present state of A, is 1. A_3 is complemented when the present state of $A_2 A_1$ is 11. Similarly, A_3 is complemented whenever the present state of A_3, A_2, A_1 is 111.

The following figure illustrates 4-bit Synchronous or binary up counter:-

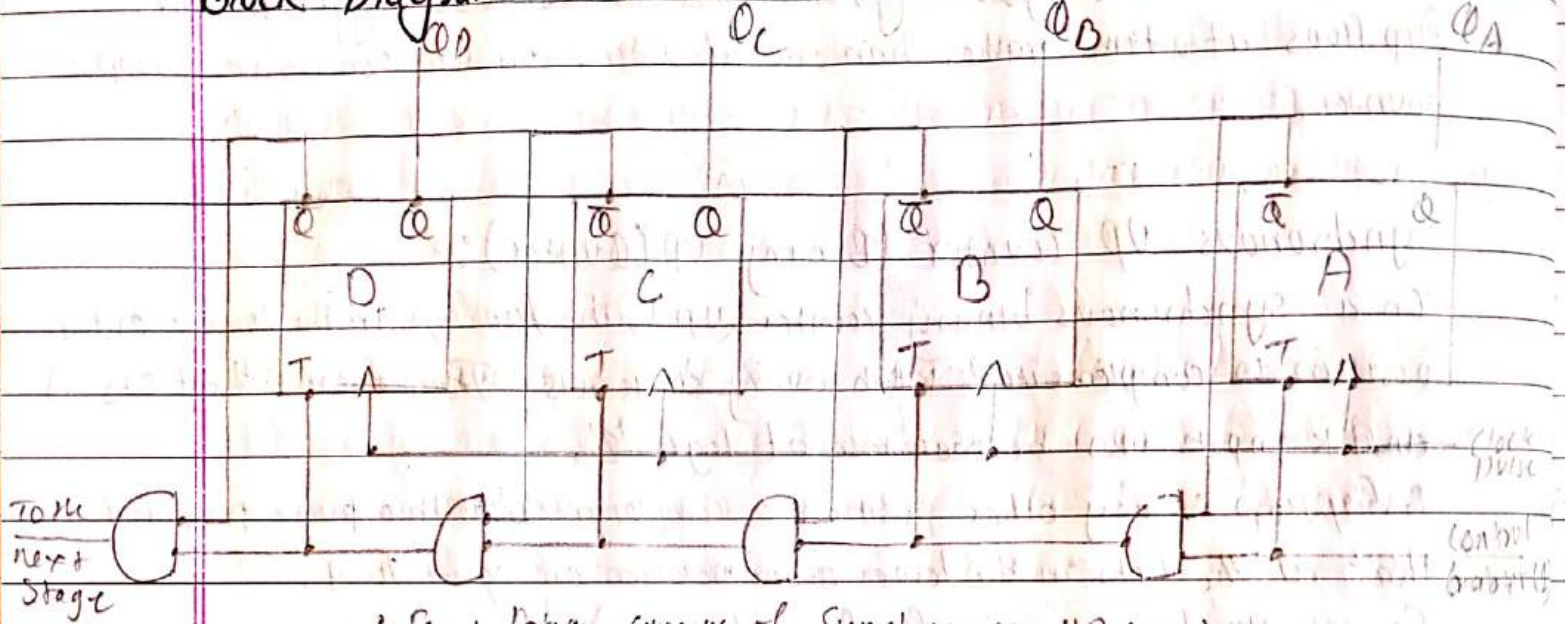
Block Diagram:-



∴ Fig:- Block Diagram of 4-Bit Synchronous UP counter

The following figure illustrates 4-bit Synchronous (Binary) Down Counters:- (output of previous flip must be all 0)

Block Diagram:-



∴ Fig:- Logic circuit of Synchronous UP counter.

Binary UP/DOWN Counter

Synchronous (Binary) UP/DOWN counter has the capability of counting upwards as well as downwards.

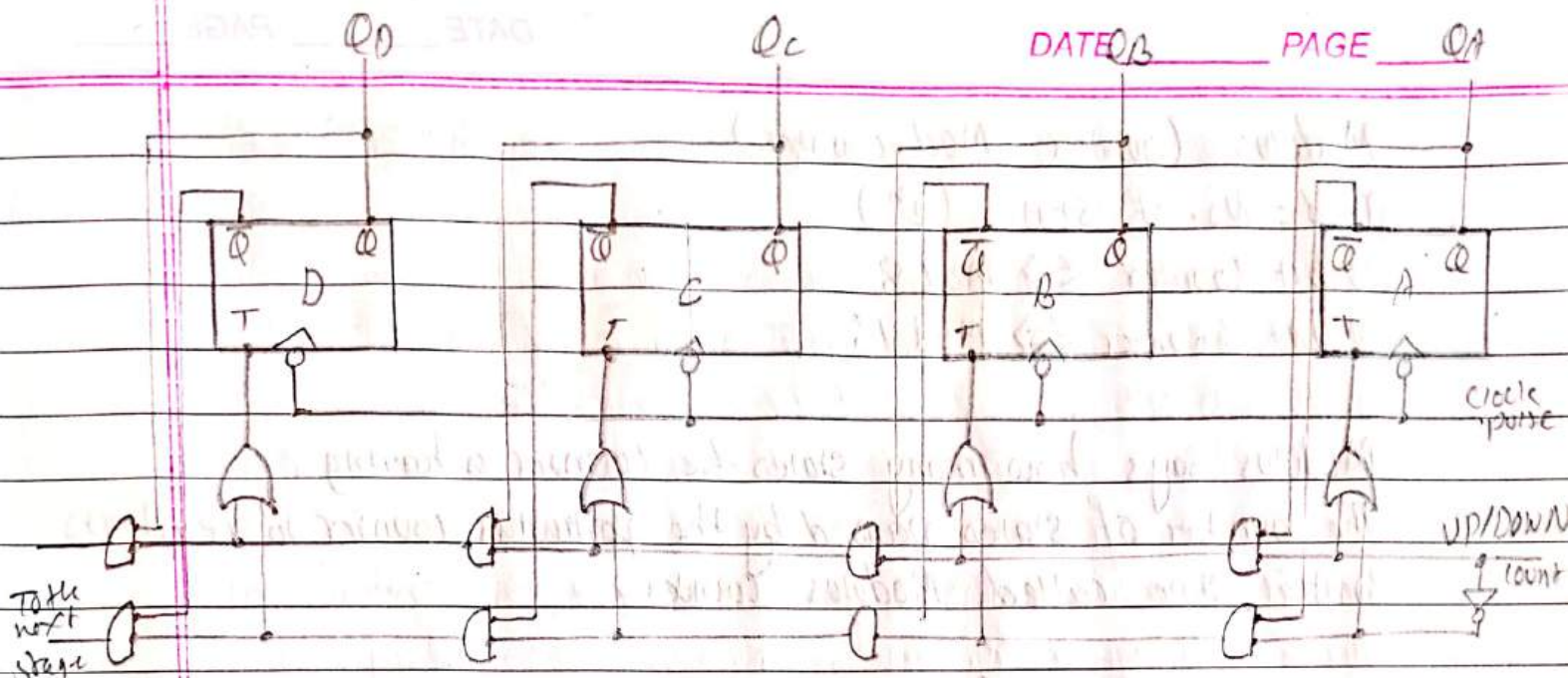
The counter can count either upwards from 0 to 2^n-1 or can count downwards from 2^n-1 to 0.

The decision whether to count upwards or downwards depends upon the UP/DOWN control input.

When UP/DOWN control input is held HIGH (1), the circuit count upwards.

When UP/DOWN control input is held LOW (0), the circuit counts downwards.

The following example demonstrate 4-bit UP/DOWN Synchronous (Binary) counters:-



∴ Fig: Block diagram of 4-Bit Synchronous UP/DOWN counter

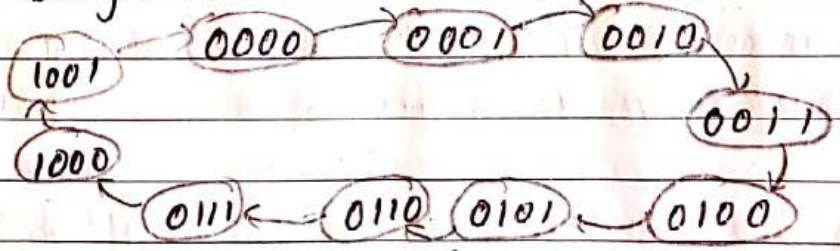
DATE _____ PAGE _____

Modulus Counter (Mod Counter)
Mod = No. of states (2^n)
3 bit counter ⇔ Mod 8
4 bit counter ⇔ Mod 16

Modulus says how many states the counter is having. The number of states passed by the particular counter to reach its initial state called Modulus counter.

MOD 10 Counter (BCD Ripple / Decader Counter)
BCD Ripple counter is having ten states. It counts from 0000 to 1001. As soon as the circuit gets the counter value 1010, the control is jumped to its initial position again i.e. 0000.

State Diagram:-



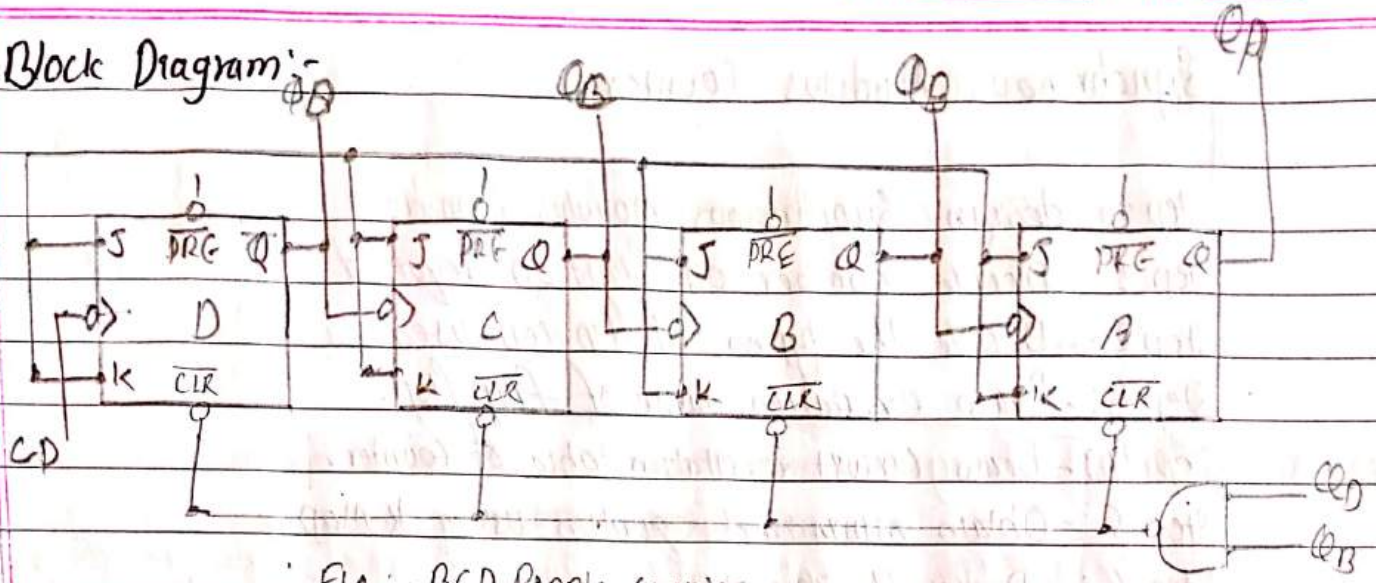
∴ State Diagram of MOD 10 counter

Truth Table

Clock	Q_D	Q_C	Q_B	Q_A	Decimal
0	0	0	0	0	0
1st	0	0	0	1	1
2nd	0	0	1	0	2
3rd	0	0	1	1	3
4th	0	1	0	0	4
5th	0	1	0	1	5
6th	0	1	1	0	6
7th	0	1	1	1	7
8th	1	0	0	0	8
9th	1	0	0	1	9
10th	1	0	1	0	

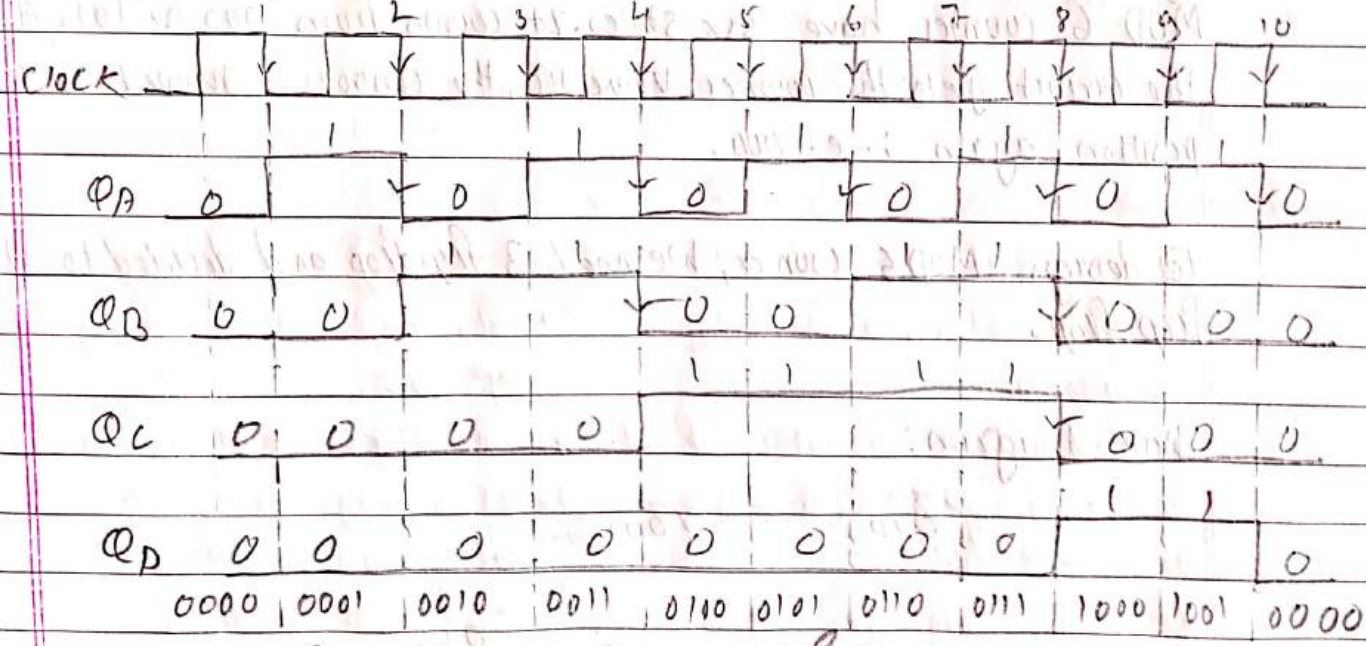
DATE _____ PAGE _____

Block Diagram:-



∴ Fig:- BCD Ripple counter

Timing Diagram:-



∴ fig:- Timing Diagram of BCD counter

Synchronous Modulus Counter

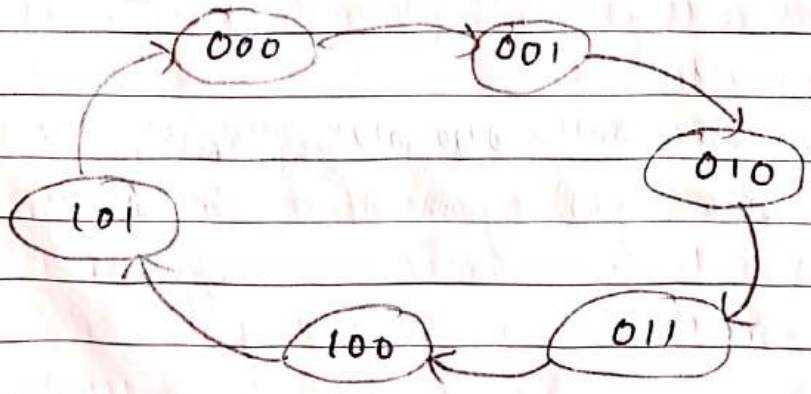
Step to design Synchronous Modulus Counter:-
Step 1:- Decide number of flipflop required.
Step 2:- Decide the types of flipflop used.
Step 3:- Draw excitation table of flip flop.
Step 4:- Draw circuit excitation table of counter.
Step 5:- Obtain minimized equation using K-Map.
Step 6:- Design the circuit.

Synchronous MOD 6 Counter

MOD 6 counter have six states. It counts from 000 to 101. As soon as the circuit gets the counter value 110, the control is jumped to initial position again i.e. 000.

To design MOD 6 counter, we need 3 flip flop and decided to use JK flip flop.

State Diagram:-



∴ Fig:- State Diagram of MOD 6 Counter.

Excitation Table of JK Flip Flop

Q_n	Q_{n+1}	S	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

By using above state diagram and excitation table of JK flip flop, we will have following circuit Excitation table of the counter:-

Present State			Next State			Inputs					
Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	0	0	0	x	1	0	x	x	1
1	1	0	x	x	x	x	x	x	x	x	x
1	1	1	x	x	x	x	x	x	x	x	x

From the circuit excitation table given above, we will obtain minimized expression as shown below using k-Map.

K-Map for J_A

Q_A \ Q_BQ_C	00	01	11	10
0	0	0	1	0
1	x	x	x	x

$\therefore J_A = Q_BQ_C$

K-Map for K_A

Q_A \ Q_BQ_C	00	01	11	10
0	x	x	x	x
1	0	1	x	x

$\therefore K_A = Q_C$

K-Map for J_B

Q_A \ Q_BQ_C	00	01	11	10
0	0	1	x	x
1	0	0	x	x

$\therefore J_B = \overline{Q_A}Q_C$

K-Map for K_B

Q_A \ Q_BQ_C	00	01	11	10
0	x	x	1	0
1	x	x	x	x

$\therefore K_B = Q_C$

K-Map for J_C

Q_A \ $Q_B Q_C$	00	01	11	10
0	1	X	X	1
1	1	X	X	X

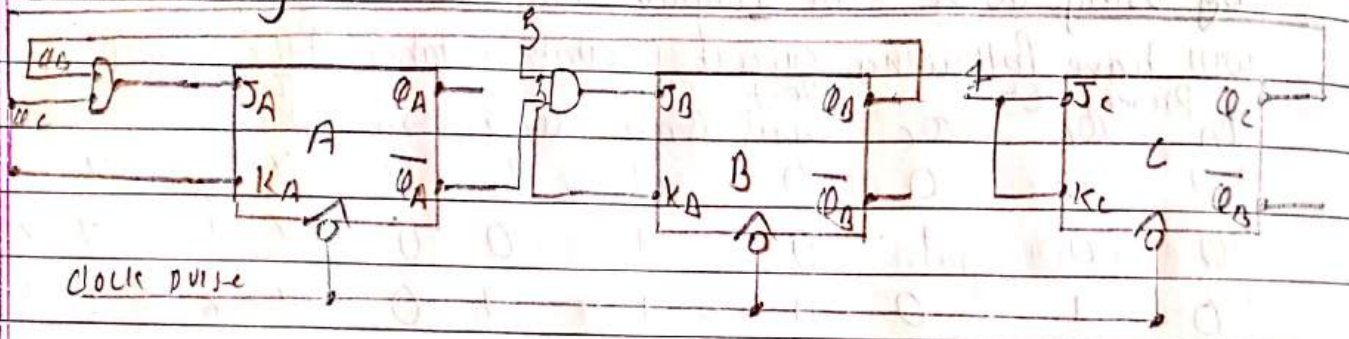
$\therefore J_C = 1$

K-Map for K_C

Q_A \ $Q_B Q_C$	00	01	11	10
0	X	1	1	X
1	X	1	X	X

$\therefore K_C = 1$.

Circuit Diagram:-



$\therefore$ Fig:- Circuit Diagram of MOD 6 Counter.

Difference Between Latch and Flip Flop.

S.N	Latch	S.N	Flip Flop
1.	Latch follows a level triggering approach.	1.	Flip-flop utilizes an edge triggering approach.
2.	The clock signal is absent	2.	The clock signal is present.
3.	It is design using logic gates.	3.	It is design using latch along with clock.
4.	It has fast operating speed.	4.	It has low operating speed.
5.	It requires less power.	5.	It requires more power.
6.	It requires less Area.	6.	It requires more area.
7.	It works using the binary input only;	7.	It works using the binary input and the clock signal.
8.	J-K, S-R, are example of Latch	8.	JK, S-R, D, T are example of flip flop