

UNIT - 1

Number System:

Number system is a basis for counting various items. On hearing the word 'number', all of us immediately think of the familiar decimal number system with its 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.

Modern computers communicate and operate with binary numbers which use only the digits 0 and 1. For large decimal quantities are dealt with very large binary strings and therefore they do not like working with binary numbers. This gave rise to three new number systems.

- Octal
- Hexadecimal
- Binary Coded Decimal (BCD)

→ To define any number system, we have to specify following aspects:

- * Base of the number system, such as 2, 8, 10 or 16.
- * The base (or) radix decides the total number of digits available in that number system.
- * First digit in the number system is always zero (0) and last digit in the number system is always.

Base - 1.

- * In general a number in a system having base can be written as

$$a_n \times r^n + a_{n-1} \times r^{n-1} + \dots + a_0 \times r^0 + a_{-1} \times r^{-1} + a_{-2} \times r^{-2} + \dots + a_{-m} \times r^{-m}$$

Where a_n is the value of the n^{th} digit, $r = \text{radix}$.

Decimal Number System:

- The decimal number system contains ten unique symbols 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.
- In decimal system 10 symbols are involved, so the base or radix is 10.
- It is a weighted number system.
- The value attached to the symbol depends on its location with respect to the decimal point.

In general, $d_n d_{n-1} d_{n-2} \dots d_0 \cdot d_{-1} d_{-2} \dots d_{-m}$ is given by

$$(d_n \times 10^n) + (d_{n-1} \times 10^{n-1}) + \dots + (d_0 \times 10^0) + (d_{-1} \times 10^{-1}) + (d_{-2} \times 10^{-2}) + \dots + (d_{-m} \times 10^{-m}).$$

$$\text{ex: } 9256.26 = 9 \times 10^3 + 2 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 + 2 \times 10^{-1} + 6 \times 10^{-2} \\ = 9 \times 1000 + 2 \times 100 + 5 \times 10 + 6 + (2/10) + (6/100).$$

Binary Number System:-

- The binary number system is a weighted system.
- The base of this number system is 2.
- It has two independent symbols.
- The symbols used are 0 and 1.
- A binary digit is called a bit.

$$\text{ex: } 1101.101$$

$$\Rightarrow 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

$$\Rightarrow 8 + 4 + 0 + 1 + 0.5 + 0 + 0.125$$

$$\Rightarrow (13.625)_{10}$$

Octal Number System

- It is also a weighted system.
- Its base or radix is 8.
- It has 8 independent symbols 0, 1, 2, 3, 4, 5, 6 and 7.
- Its base $8 = 2^3$, every 3-bit group of binary can be represented by an octal digit.

Ex:- $(567)_8$

Hexa Decimal Number System:

- The hexadecimal number system is a weighted system.
- The base or radix of this number system is 16.
- The symbols used are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F.
- The base $16 = 2^4$, every 4-bit group of binary can be represented by an hexadecimal digit.

Ex:- $(3FD)_{16}$

Conversions:

• **Decimal to binary Conversion:-**

∴ Convert $(52)_{10}$ into binary

Sol:-

$$\begin{array}{r} 2 \overline{) 52} \\ \underline{26} \\ 2 \overline{) 13} \\ \underline{6} \\ 2 \overline{) 6} \\ \underline{3} \\ 2 \overline{) 3} \\ \underline{1} \\ 1 \end{array}$$

$$\begin{array}{r} 2 \overline{) 26} \\ \underline{0} \\ 2 \overline{) 13} \\ \underline{0} \\ 2 \overline{) 6} \\ \underline{1} \\ 2 \overline{) 3} \\ \underline{0} \\ 1 \end{array}$$

$$\begin{array}{r} 2 \overline{) 13} \\ \underline{0} \\ 2 \overline{) 6} \\ \underline{1} \\ 2 \overline{) 3} \\ \underline{0} \\ 1 \end{array}$$

$$\begin{array}{r} 2 \overline{) 6} \\ \underline{1} \\ 2 \overline{) 3} \\ \underline{0} \\ 1 \end{array}$$

$$\begin{array}{r} 2 \overline{) 3} \\ \underline{0} \\ 1 \end{array}$$

$$\begin{array}{r} 1 \end{array}$$

$$\begin{array}{r} 1 \end{array}$$

$$(52)_{10} = (110100)_2$$

ii) Convert $(105.15)_{10}$ into binary

Integer part

$$\begin{array}{r}
 2 \overline{)105} \\
 2 \overline{)52} - 1 \\
 2 \overline{)26} - 0 \\
 2 \overline{)13} - 0 \\
 2 \overline{)6} - 1 \\
 2 \overline{)3} - 0 \\
 1 - 1
 \end{array}$$

Fraction part

$$0.15 \times 2 = 0.30 \rightarrow 0$$

$$0.30 \times 2 = 0.60 \rightarrow 0$$

$$0.60 \times 2 = 1.20 \rightarrow 1$$

$$0.20 \times 2 = 0.40 \rightarrow 0$$

$$0.40 \times 2 = 0.80 \rightarrow 0$$

$$0.80 \times 2 = 1.60 \rightarrow 1$$

$$(105.15)_{10} = (1101001.001001)_2$$

• Decimal to octal Conversion:

i) Convert $(378.93)_{10}$ into octal.

$$\begin{array}{r}
 8 \overline{)378} \\
 8 \overline{)47} - 2 \\
 5 - 7
 \end{array}$$

$$0.93 \times 8 = 7.44 \rightarrow 7$$

$$0.44 \times 8 = 3.52 \rightarrow 3$$

$$0.52 \times 8 = 4.16 \rightarrow 4$$

$$0.16 \times 8 = 1.28 \rightarrow 1$$

$$(378.93)_{10} = (572.7341)_8$$

• Decimal to hexadecimal conversion:

• Convert $(2598.675)_{10}$ into hexadecimal.

$$\begin{array}{r}
 16 \overline{)2598} \\
 16 \overline{)162} - 6 \\
 10 - 2
 \end{array}$$

$$0.675 \times 16 = 10.8 \rightarrow A$$

$$0.8 \times 16 = 12.8 \rightarrow C$$

$$0.8 \times 16 = 12.8 \rightarrow C$$

$$0.8 \times 16 = 12.8 \rightarrow C$$

$$(2598.675)_{10} = (A26.ACCC)_{16}$$

Binary to Decimal Conversion

i) Convert $(10101)_2$ to decimal

$$\begin{aligned}(10101)_2 &= (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\ &= 16 + 0 + 4 + 0 + 1 \\ &= (21)_{10}\end{aligned}$$

ii) Convert $(111.101)_2$ to decimal

$$\begin{aligned}(111.101)_2 &= (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) \\ &= 4 + 2 + 1 + 0.5 + 0 + 0.125 \\ &= 7.625\end{aligned}$$

$$\Rightarrow (111.101)_2 = (7.625)_{10}$$

Octal to Decimal Conversion

Convert $(756.603)_8$ to decimal

$$\begin{aligned}(756.603)_8 &= (7 \times 8^2) + (5 \times 8^1) + (6 \times 8^0) + (6 \times 8^{-1}) + (0 \times 8^{-2}) + (3 \times 8^{-3}) \\ &= 448 + 40 + 6 + 0.75 + 0 + 0.005 \\ &= (494.755)_{10}\end{aligned}$$

Hexadecimal to decimal Conversion

Convert $(A0F9.0EB)_{16}$ to decimal

$$\begin{aligned}(A0F9.0EB)_{16} &= (10 \times 16^3) + (0 \times 16^2) + (15 \times 16^1) + (9 \times 16^0) \\ &\quad + (0 \times 16^{-1}) + (14 \times 16^{-2}) + (11 \times 16^{-3}) \\ &= 40960 + 0 + 240 + 9 + 0 + 0.0546 + 0.0026 \\ &= (41209.0572)_{10}\end{aligned}$$

Binary to Octal Conversion

for binary to octal conversion the binary numbers are divided into groups of 3 bits each,

octal	binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

i. Convert $(101111010110 \cdot 110110011)_2$ into octal.

$$\begin{array}{ccccccc} \underline{101} & \underline{111} & \underline{010} & \underline{110} & \cdot & \underline{110} & \underline{110} & \underline{011} \\ 5 & 7 & 2 & 6 & & 6 & 6 & 3 \end{array}$$

$$\Rightarrow (101111010110 \cdot 110110011)_2 = (5726 \cdot 663)_8$$

ii. Convert $(10101111001 \cdot 0111)_2$ into octal.

$$\begin{array}{ccccccc} \underline{010} & \underline{101} & \underline{111} & \underline{001} & \cdot & \underline{011} & \underline{100} \\ 2 & 5 & 7 & 1 & & 3 & 4 \end{array}$$

$$\Rightarrow (10101111001 \cdot 0111)_2 = (2571 \cdot 34)_8$$

Binary to hexadecimal conversion:

4

for binary to hexadecimal conversion, the binary numbers are divided into groups of 4 bits each.

Hexadecimal	Binary	Hexadecimal	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

i) Convert $(1011011011)_2$ into hexadecimal.

0010 1101 1011 01 0111 1001 0101 1100

2 D B

$$\Rightarrow (1011011011)_2 = (2DB)_{16}$$

ii) Convert $(01011111011 \cdot 011111)_2$ into hexadecimal.

0010 1111 1011 · 0111 1100

2 F B · 7 C

$$\Rightarrow (01011111011 \cdot 011111)_2 = (2FB.7C)_{16}$$

Octal to Binary Conversion:

To Convert octal number to binary, replace each octal digit by its 3-bit binary equivalent.

Convert $(367.52)_8$ into binary

3 6 7 . 5 2

011 110 111 . 101 010

$$\Rightarrow (367.52)_8 = (01110111.101010)_2$$

Hexadecimal to Binary Conversion:

To Convert Hexadecimal number to binary, replace each hexadecimal digit by its 4-bit binary group.

Convert $(3A9E.B0D)_{16}$ into binary

(3A9E.B0D)₁₆

0011 1010 1001 1110 . 1011 0000 1101

$$\Rightarrow (3A9E.B0D)_{16} = (0011101010011110.101100001101)_2$$

Octal to hexadecimal Conversion:

For octal to Hexadecimal Conversion, first convert the given octal numbers to binary and then binary numbers to hexadecimal.

Convert $(756.603)_8$ to hexadecimal

7 5 6 . 6 0 3

111 101 110 . 110 000 011

0001 1110 1110 . 1100 0001 1000
1 E E . C 1 8

$$\Rightarrow (756.603)_8 = (1EE.C18)_{16}$$

5

Hexadecimal to Octal Conversion

For hexadecimal to Octal Conversion, first convert the given hexadecimal number to binary and then binary number to octal.

Convert $(B9F.AE)_{16}$ to Octal.

$$\begin{array}{ccccccc} B & 9 & F & . & A & E & \\ 1011 & 1001 & 1111 & . & 1010 & 1110 & \end{array}$$

$$\underline{101} \underline{110} \underline{011} \underline{111} . \underline{101} \underline{011} \underline{1000}$$

$$5 \ 6 \ 3 \ 7 . 5 \ 3 \ 4$$

$$\Rightarrow (B9F.AE)_{16} = (5637.534)_8$$

Binary Arithmetic Operation:

1. Binary Addition:

$$0+0=0,$$

$$0+1=1,$$

$$1+0=1,$$

$$1+1=10, \text{ i.e. } 0 \text{ with carry } 1$$

* Add $(100101)_2$ and $(1101111)_2$

$$\begin{array}{r} 1001011 \\ + 1101111 \\ \hline 10010100 \end{array}$$

2. Binary Subtraction:

$$0 - 0 = 0$$

$$0 - 1 = 1, \text{ with a borrow of } 1$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

Subtract $(111.111)_2$ from $(1010.01)_2$

$$\begin{array}{r} 1010.010 \\ 111.111 \\ \hline \end{array}$$

$$\underline{0010.011}$$

3. Binary Multiplication:

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

Multiply $(1101)_2$ by $(110)_2$

$$\begin{array}{r} 1101 \\ \times 110 \\ \hline 0000 \\ 110100 \\ 1101110 \\ \hline 1001110 \end{array}$$

4. Binary Division:

$$0 \div 1 = 0$$

$$1 \div 1 = 1$$

Divide $(101101)_2$ by $(110)_2$

$$\begin{array}{r} 110 \overline{) 101101} \quad (111.1) \\ \underline{110} \\ 1010 \\ \underline{110} \\ 1001 \\ \underline{110} \\ 110 \\ \underline{110} \\ 000 \\ \underline{000} \end{array}$$

Representation of Signed no.s.

Two ways of representing signed no.s

- Sign Magnitude form
- Complemented form

→ In Sign magnitude form, an additional bit called the sign bit is placed in front of the no. If the sign bit is 0, the no. is +ve, if it is 1, the no. is -ve

$$\text{Ex: } - 0110100 = +52$$

↓
sign bit

$$1 \ 110100 = -52$$

↓
sign bit

* Complemented Form

- 1's Complement
- 2's Complement

→ 1's Complement Representation

The 1's Complement of a binary number is obtained by changing each 0 to 1 and each 1 to 0.

Ex:- Find $(1100)_2$ 1's Complement

$$\begin{array}{r} 1001 \\ 1100 \\ \hline \end{array}$$

$$0011 \rightarrow 1's \text{ Complement}$$

→ 2's Complement Representation

The 2's Complement of a binary number is a binary number which is obtained by adding 1 to the 1's Complement of a number i.e.

$$2's \text{ Complement} = 1's \text{ Complement} + 1$$

Ex:- Find $(1010)_2$ 2's Complement

$$1010$$

$$0101 \rightarrow 1's \text{ Complement}$$

$$\begin{array}{r} + \quad 1 \\ 0101 \\ \hline 0110 \rightarrow 2's \text{ Complement} \end{array}$$

• perform $(1011)_2 - (0100)_2$ using 1's complement method.

$$\begin{array}{r} 11 \\ - 4 \\ \hline 7 \end{array} \quad \begin{array}{l} \rightarrow 1011 \quad = A \\ \rightarrow 0100 \quad = B \end{array} \quad A > B$$

1's complement for 0100 is 1011

$$\begin{array}{r} 10111001 \\ + 1011101 \\ \hline 1001110 \\ \text{Carry } 1 \rightarrow \\ \hline 01111 \end{array}$$

• Subtract $(9)_{10}$ from $(4)_{10}$ using 1's complement method

$$A = (4)_{10} = (0100)_2 \quad A < B$$

$$B = (9)_{10} = (1001)_2$$

1's complement for 1001 is 0110

$$\begin{array}{r} 0100 \\ + 0110 \\ \hline 1010 \end{array}$$

Perform $(9)_{10} - (5)_{10}$ using 2's Complement method.

$$A = (9)_{10} = (1001)_2$$

$$B = (5)_{10} = (0101)_2$$

1's Complement for 0101 is 1010

$$2's \text{ Complement} \leftarrow \begin{array}{r} + 1 \\ \hline 1011 \end{array}$$

$$\begin{array}{r} 1001101 \\ 1011 \\ \hline 11100 \end{array}$$

Discard carry.

perform $(4)_{10} - (9)_{10}$ using the 2's Complement method.

$$A = (4)_{10} = (0100)_2$$

$$B = (9)_{10} = (1001)_2$$

2's complement for 1001 is 0111

$$\begin{array}{r} 0100 \\ 0010 \\ 0111 \\ \hline 1011 \end{array}$$

Add -45.75 to +87.5 using 12 bit arithmetic

$$+87.5 = 01010111.1000$$

$$-45.75 = 11010010.0100 \rightarrow 2's \text{ Complement}$$

$$\begin{array}{r} \text{Discard} \\ \text{carry} \end{array} \leftarrow \begin{array}{r} \hline 100101001.1100 \\ \hline \end{array}$$

9's Complement & 10's Complement

* 9's Complement of 3465 and 782.54

$$\begin{array}{r} 9999 \\ - 3465 \\ \hline 6534 \end{array}$$

$$\begin{array}{r} 999.99 \\ - 782.54 \\ \hline 217.45 \end{array}$$

* 10's Complement of 4069 is

$$\begin{array}{r} 9999 \\ - 4069 \\ \hline 5930 \end{array} \rightarrow 9's \text{ Complement}$$

$$\begin{array}{r} 5930 \\ + 1 \\ \hline 5931 \end{array} \rightarrow 10's \text{ Complement}$$

Subtract 745.81 - 436.62 using 9's complement

$$\begin{array}{r} 745.81 \\ - 436.62 \\ \hline 309.19 \end{array}$$

$$\begin{array}{r} 745.81 \\ + 563.37 \\ \hline 1309.18 \\ \rightarrow 1 \\ \hline 309.19 \end{array}$$

$$\begin{array}{r} 999.99 \\ - 436.62 \\ \hline 563.37 \end{array} \rightarrow 9's \text{ Complement}$$

Subtract using 9's Complement $436.62 - 745.82$

$$\begin{array}{r} 436.62 \\ - 745.82 \\ \hline -309.20 \end{array}$$

9's Complement for 745.82 is 999.99

$$\begin{array}{r} 999.99 \\ - 745.82 \\ \hline 254.17 \end{array}$$

$$\begin{array}{r} 436.62 \\ + 254.17 \\ \hline 690.79 \end{array}$$

If no carry result is -ve

If it has carry it is +ve

Result in 9's complement 690.79

→ 999.99

690.79

$$\begin{array}{r} 999.99 \\ + 690.79 \\ \hline -309.20 \end{array}$$

Subtract $2928.54 - 416.73$ using 10's Complement

$$\begin{array}{r} 2928.54 \\ - 0416.73 \\ \hline 2511.81 \end{array}$$

10's Complement for 416.73

999.99

- 0416.73

9583.26

+

9583.27

↓

10's Complement

2928.54

+ 9583.27

12511.81

↳ Ignore carry

Subtract $416.73 - 2928.54$ using 10's Complement

$$\begin{array}{r} 0416.73 \\ - 2928.54 \\ \hline - 2511.81 \end{array}$$

10's complement for 2928.54 is

$$\begin{array}{r} 9999.99 \\ - 2928.54 \\ \hline 7071.45 \rightarrow 9's \text{ comp} \\ + 1 \\ \hline 7071.46 \rightarrow 10's \text{ comp} \end{array}$$

$$\begin{array}{r} 0416.73 \\ + 7071.46 \\ \hline 7488.19 \end{array}$$

Binary Coded Decimal (BCD):

BCD is a weighted codes, each successive digit from right to left represents weights, equal to some specified value and to get the equivalent decimal number add the products of the weights by the corresponding binary digit. 8421 is the most common because 8421 BCD is the most natural amongst the other possible codes.

Decimal

BCD

8421

0

0000

1

0001

2

0010

3

0011

4

0100

5

0101

6

0110

7

0111

8

1000

9

1001

Decimal

BCD

10

8421

8421

0001

0000

11

0001

0001

12

0001

0010

13

0001

0011

* BCD Addition:-

Case 1: Sum is equal or less than 9 and carry is 0.

perform BCD addition of $(2)_{10}$ and $(6)_{10}$

$$(2)_{10} \rightarrow \text{BCD} \quad 0010$$

$$(6)_{10} \rightarrow 0110$$

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

$$\begin{array}{r} 1000 \end{array}$$

$\rightarrow$ Sum is a valid number.

Case 2: Sum greater than 9 but carry = 0

perform BCD addition of $(7)_{10}$ and $(6)_{10}$

$$7 \rightarrow \text{BCD} \quad 0111$$

$$\begin{array}{r} 6 \rightarrow + 0110 \\ \hline 13 \end{array}$$

$$\begin{array}{r} 0111 \\ + 0110 \\ \hline 1101 \end{array} \rightarrow \text{Invalid BCD}$$

$$\begin{array}{r} 1101 \\ + 0110 \\ \hline 10011 \end{array} \rightarrow \text{add 6 to invalid BCD}$$

$$\begin{array}{r} 0001 \quad 0011 \\ \hline 1 \quad 3 \end{array} \rightarrow \text{valid BCD}$$

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

perform BCD addition of $(9)_{10}$ and $(8)_{10}$

$$9 \rightarrow 1001$$

$$+ 8 \rightarrow + 1000$$

$$\begin{array}{r} 1001 \\ + 1000 \\ \hline 17 \end{array}$$

$$\begin{array}{r} 10001 \end{array} \rightarrow \text{Valid BCD sum and carry = 1}$$

$$\begin{array}{r} 0001 \quad 0001 \end{array} \rightarrow \text{Incorrect BCD results}$$

$$\begin{array}{r} 0000 \quad 0110 \end{array}$$

$$\begin{array}{r} 0001 \quad 0111 \\ \hline 1 \quad 7 \end{array} \rightarrow \text{Correct BCD}$$

• Add +6 to

Incorrect BCD
result

Add $(569)_{10}$ and $(687)_{10}$ in BCD

$$\begin{array}{r}
 569 \rightarrow 0101 \ 0110 \ 1001 \\
 + 687 \rightarrow + 0110 \ 1000 \ 0111 \\
 \hline
 1256 \rightarrow 1011 \ 1111 \ 0000
 \end{array}$$

$\downarrow$ Invalid BCD $\downarrow$ Invalid BCD $\rightarrow$ valid BCD with carry 1

Add $(0110)_2$ to Invalid BCD number to get correct BCD.

$$\begin{array}{r}
 1011 \ 1111 \ 0000 \\
 + 0110 \ 0110 \ 0110 \\
 \hline
 10010 \ 0101 \ 0110
 \end{array}$$

$\downarrow$

$$\begin{array}{cccc}
 0001 & 0010 & 0101 & 0110 \\
 \hline
 1 & 2 & 5 & 6
 \end{array}
 \rightarrow \text{Valid BCD}$$

* BCD Subtraction:

Subtract $(38)_{10}$ and $(15)_{10}$ in BCD

$$\begin{array}{r}
 38 \rightarrow 0011 \ 1000 \\
 - 15 \rightarrow - 0001 \ 0101 \\
 \hline
 23 \rightarrow 0010 \ 0011
 \end{array}$$

Valid BCD

• $(206.7)_{10} - (147.8)_{10}$ in BCD

$$\begin{array}{r}
 206.7 \rightarrow 0010 \ 0000 \ 0110 \cdot 0111 \\
 -147.8 \rightarrow 0001 \ 0100 \ 0111 \cdot 1000 \\
 \hline
 58.9 \rightarrow 0000 \ 1011 \ 1110 \cdot 1111 \\
 0110 \ 0110 \cdot 0110 \\
 \hline
 \text{Valid BCD} \rightarrow 0101 \ 1000 \cdot 1001 \\
 \phantom{\text{Valid BCD} \rightarrow} \downarrow \\
 \phantom{\text{Valid BCD} \rightarrow} 5
 \end{array}$$

borrow from the next group
Subtract with $(0110)_2$

• perform $(83)_{10} - (21)_{10}$ using 9's complement method

9's complement for 21 is 99

$$\begin{array}{r}
 99 \\
 -21 \\
 \hline
 78
 \end{array}$$

$$83 \rightarrow 1000 \ 0011$$

$$78 \rightarrow +0111 \ 1000$$

$$\begin{array}{r}
 1000 \ 0011 \\
 +0111 \ 1000 \\
 \hline
 1111 \ 1011
 \end{array}$$

Invalid BCD Invalid BCD

$$\begin{array}{r}
 1111 \ 1011 \\
 +0110 \ 0110 \\
 \hline
 0010 \ 0001
 \end{array}$$

Carry added to LSB

that is 0110 0001

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

Correct BCD

perform $(54)_{10} - (22)_{10}$ in BCD using 10's complement method.

10's Complement for 22 is 99

$$\begin{array}{r} 22 \\ \hline 77 \rightarrow 9's \text{ comp} \\ +1 \\ \hline 78 \rightarrow 10's \text{ comp} \end{array}$$

BCD

$$54 \rightarrow 0101 \ 0100$$

$$78 \rightarrow +0111 \ 1000$$

$$\begin{array}{r} 0111 \ 1000 \\ \hline 1100 \ 1100 \end{array} \rightarrow \text{Invalid BCD}$$

$$+0110 \ 0110$$

$$\begin{array}{r} 1100 \ 1100 \\ +0110 \ 0110 \\ \hline 10011 \ 0010 \end{array}$$

Ignore carry $\leftarrow$ $\textcircled{1} 0011 \ 0010 \rightarrow \text{Valid BCD}$

Excess-3 (XS-3) code:

It is a non-weighted BCD code. Each binary code word is the corresponding 8421 code word plus 0011 (3). It is a sequential code & therefore can be used for arithmetic operation.

Decimal digit Excess-3 Code

0 00 11

1 01 00

2 01 01

3 01 10

4 01 11

5 10 00

6 10 01

7 10 10

8 10 11

9 11 00

* Excess-3 Addition

To perform Excess-3 addition, we have to

- Add two excess-3 numbers.
- If carry = 1 $\rightarrow$ add 3 to the sum of two digits
 $= 0 \rightarrow$ subtract 3

Ex:-

4	$\rightarrow$	XS-3 code	
+ 3	$\rightarrow$	+ 0110	
<u>7</u>		<u>1101</u>	$\rightarrow$ carry is 0
		- 0011	
		<u>1010</u>	

* Excess-3 Subtraction

To perform Excess-3 subtraction, we have to

- Complement the subtrahend
- Add complemented subtrahend to minuend.
- If carry = 1, Result is +ve. Add 3 and end around carry
- If carry = 0, Result is -ve, subtract 3.

Ex:- 8 - 5

XS-3 code

8 $\rightarrow$ 1011

5 $\rightarrow$ 1000

$\downarrow$ 1's complement

0111

0101

1101

0011

1011
+ 0111
<u>1111</u>
① 0010
<u>0011</u>
0011
<u>0110</u>

perform the subtraction $(645)_{10} - (319)_{10}$ in XS-3 code using the 9's complement.

9's complement for 319 is 999

$$\begin{array}{r} 319 \\ \hline 680 \end{array}$$

XS-3 Code

$$\begin{array}{r} 645 \rightarrow 1001 \quad 0111 \quad 1000 \\ 680 \rightarrow +1001 \quad 1011 \quad 0011 \\ \hline \textcircled{1} \quad 0011 \quad 0010 \quad 1011 \\ \quad +0011 \quad +0011 \quad -0011 \\ \hline \quad 0110 \quad 0101 \quad 1000 \rightarrow +1 \\ \hline 0110 \quad 0101 \quad 1001 \end{array}$$

Gray Code

Gray code is a special case of unit-distance code. In unit-distance code, bit patterns for two consecutive numbers differ in only one bit position. These codes are also called cyclic codes.

* BINARY - To - GRAY CONVERSION:-

If an n -bit binary number is represented by $B_n, B_{n-1}, \dots, B_1$ and its gray code equivalent by $G_n, G_{n-1}, \dots, G_1$.

Where G_n and B_n are the MSBs. Then gray code bits are obtained from the binary code as follows.

$$G_n = B_n$$

$$G_{n-1} = B_n \oplus B_{n-1}$$

$$\vdots$$
$$G_1 = B_2 \oplus B_1$$

Where the symbol $\oplus$ stands for Exclusive OR (X-OR)

Ex:- 1001 to Gray code

$$\begin{array}{cccc} \text{Binary} & \rightarrow & 1 & \xrightarrow{\oplus} & 0 & \xrightarrow{\oplus} & 0 & \xrightarrow{\oplus} & 1 \\ & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ \text{Gray} & \rightarrow & 1 & & 1 & & 0 & & 1 \end{array}$$

The gray code for $(1001)_2$ is 1101

Decimal code	Binary code	Gray 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

* GRAY - To - BINARY CONVERSION:

If an n -bit gray number is represented by $G_n G_{n-1} \dots G_1$ and its binary equivalent by $B_n B_{n-1} \dots B_1$, then binary bits are obtained from Gray bits as follows.

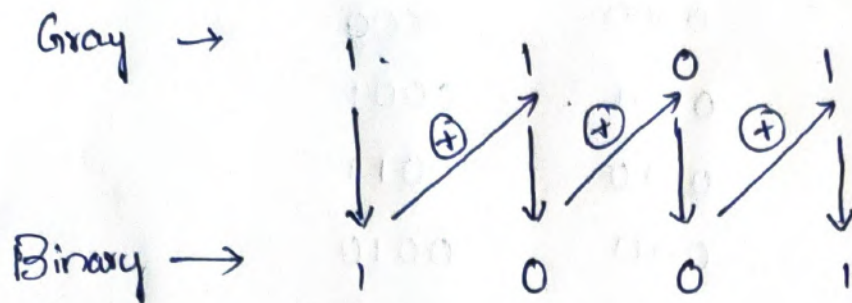
$$B_n = G_n$$

$$B_{n-1} = B_n \oplus G_{n-1}$$

$$\vdots$$

$$B_1 = B_2 \oplus G_1$$

Ex:- Convert the gray code 1101 to binary



The binary code is 1001.

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

Logic Gates

Logic gates are the fundamental building blocks of digital systems.

There are 3 basic types of gates AND, OR and NOT.

Logic gates are electronic circuits because they are made up of a number of electronic devices and components.

Inputs and outputs of logic gates can occur only in 2 levels. These two levels are termed HIGH and LOW, or TRUE and FALSE, or ON and OFF or simply 1 and 0.

AND Gate

An AND gate has two or more inputs but only one output.

The output is logic 1 state only when each one of its inputs is at logic 1 state.

The output is logic 0 state even if one of its inputs is at logic 0 state.



Logic Symbol

Truth Table

Inputs		Output
A	B	$Y = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate:

- An OR gate may have two or more inputs but only one output.
- The output is logic 1 state, even if one of its input is in logic 1 state.
- The output is logic 0 state, only when each one of its inputs is in logic 0 state.

Logic Symbol



Truth Table

Inputs		Output
A	B	$Y = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

NOT Gate

- A NOT gate, also called an Inverter, has only one input and one output.
- It is a device whose output is always the complement of its input.
- The output of a NOT gate is the logic 1 state when its input is in logic 0 state and the logic 0 state when its input is in logic 1 state.

Logic Symbol



Truth Table

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

NAND Gate

- NAND gate is a combination of an AND gate and a NOT gate.
- The output is logic 0 when each of the input is logic 1 and for any other combination of inputs, the output is logic 1.

Logic Symbol



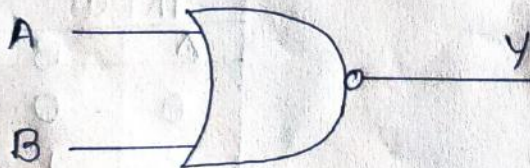
Truth Table

INPUT		OUTPUT
A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate

- NOR gate is a combination of an OR gate and a NOT gate.
- The output is logic 1, only when each one of its input is logic 0 and for any other combination of inputs, the output is a logic 0 level.

Logic Symbol



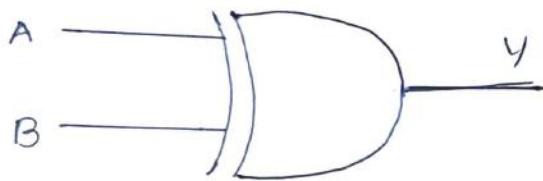
Truth Table

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

X-OR Gate

- An X-OR gate is a two input, one output logic circuit. The output is logic 1 when one and only one of its two inputs is logic 1. When both the inputs is logic 0 or when both the inputs is logic 1, the output is logic 0.

Logic Symbol



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

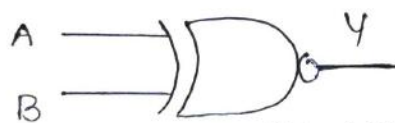
Truth Table

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

X-NOR Gate

- An X-NOR gate is the combination of an X-OR gate and a NOT gate.
- An X-NOR gate is a two input, one output logic circuit.
- The output is logic 1 only when both the inputs are logic 0 or when both the inputs is 1.
- The output is logic 0 when one of its inputs is logic 0 and other is 1.

Logic Symbol



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

Truth Table

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

Boolean Algebra

Boolean Algebra is used to analyze and simplify the digital circuits. Because, it uses only the binary numbers, i.e., 0 and 1, it is also called as Binary Algebra or Logical Algebra.

Boolean Laws

1. Commutative law
2. AND law
3. Associative law
4. Distributive law
5. OR law
6. Inversion law

Commutative Law

Any binary operation which satisfies the following expression is called as Commutative operation.

i) $A \cdot B = B \cdot A$

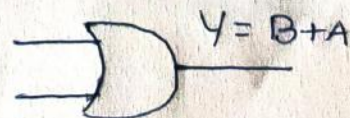
ii) $A + B = B + A$



$\Rightarrow$



$\Rightarrow$

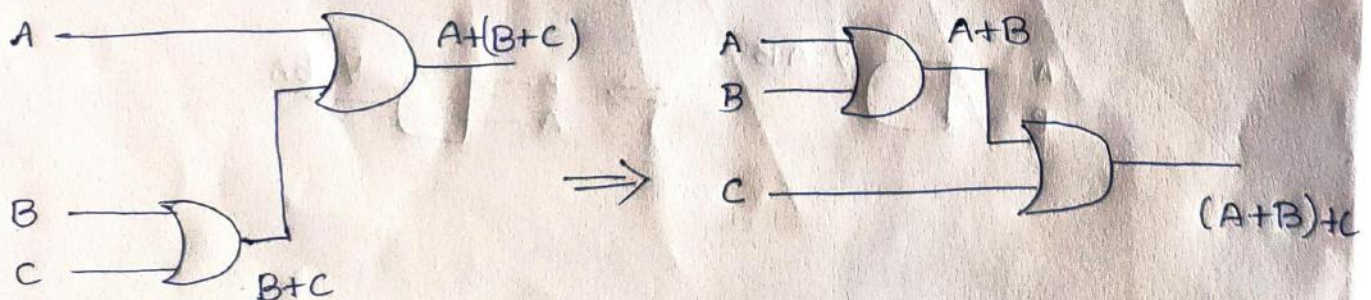
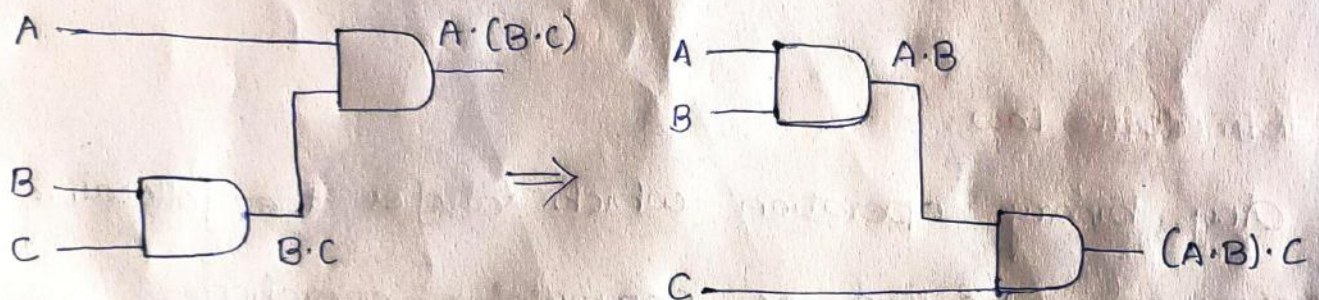


Inputs					
A	B	$A \cdot B$	$B \cdot A$	$A+B$	$B+A$
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	1	1	1

Associative law

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

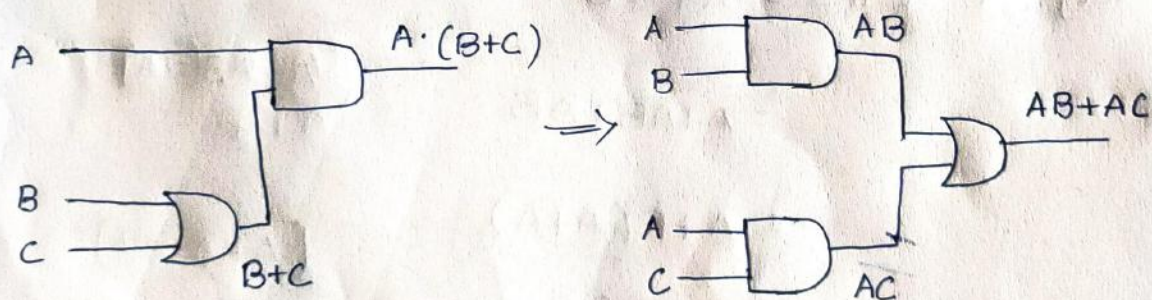
$$(A+B)+C = A+(B+C)$$



INPUTS						
A	B	C	B.C	A.(B.C)	A.B	(A.B).C
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	1	0
1	1	1	1	1	1	1

Distributive Law:

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



A	B	C	B+C	A.(B+C)	AB	AC	AB+AC
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

AND Laws

$$* A \cdot 0 = 0$$

$$* A \cdot 1 = A$$

$$* A \cdot A = A$$

$$* A \cdot \bar{A} = 0$$

OR Laws

$$* A + 0 = A$$

$$* A + 1 = 1$$

$$* A + A = A$$

$$* A + \bar{A} = 1$$

Inversion Law

$$* \text{If } A=0 \text{ then } \bar{A}=1, \bar{\bar{A}}=0$$

$$* \text{If } A=1 \text{ then } \bar{A}=0, \bar{\bar{A}}=1$$

Redundant Laws:-

$$\text{i), } A + AB = A$$

(Absorption law)

Proof:-

$$\text{L.H.S} = A + AB$$

$$= A(1+B)$$

$$= A(1)$$

$$= A$$

$$\therefore [1+B=1]$$

$$\text{ii), } A + \bar{A}B = A + B$$

Proof:-

$$\text{L.H.S} = A + \bar{A}B$$

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

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

$$= A + B(1)$$

$$= A + B$$

$$\therefore [A + AB = A]$$

$$\therefore [A + \bar{A} = 1]$$

$$\text{iii), } (A+B)(A+C) = A + BC$$

Proof:-

$$\text{L.H.S} = (A+B)(A+C)$$

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

$$[\because A \cdot A = A, B \cdot A = A \cdot B]$$

$$= A + AC + A \cdot B + B \cdot C$$

$$[\because A + A \cdot B = A]$$

$$= A + AC + BC$$

$$[\because 1 + C = 1]$$

$$= A(1 + C) + BC$$

$$= A(1) + BC$$

$$= A + BC$$

Consensus Theorem

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

$$\text{proof:- LHS} = AB + \bar{A}C + BC$$

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

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

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

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

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

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

$$\text{Proof: LHS} = (A + B)(\bar{A} + C)(B + C)$$

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

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

$$= ABC + BB\bar{A} + B \cdot B \cdot C + A \cdot AC + B\bar{A}C + B \cdot CC$$

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

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

$$[\because B \cdot B = B]$$

$$C \cdot C = C$$

$$A \cdot A = A$$

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

$$\therefore [A + A = A]$$

$$R.H.S = (A+B)(\bar{A}+C)$$

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

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

$$\therefore L.H.S = R.H.S$$

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

$$\text{proof:- } LHS = \bar{A} + AB$$

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

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

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

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

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

$$\begin{aligned} L.H.S &= \bar{A} + (AB) \quad (\because \text{Distributive Law}) \\ (or) &= (\bar{A} + A)(\bar{A} + B) \\ &= \bar{A} + B \\ &= R.H.S \end{aligned}$$

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

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

$$\text{proof:- } LHS = \bar{A} + A\bar{B}$$

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

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

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

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

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

$$L.H.S = \bar{A} + (A\bar{B})$$

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

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

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

Transposition theorem:

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

Proof:- R.H.S = $(A+C)(\bar{A}+B)$

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

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

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

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

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

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

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

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

Duality:-

It states that in a two valued Boolean algebra, the dual of an algebraic expression can be obtained simply by interchanging OR and AND operators and by replacing 1s by 0s and 0s by 1s.

Given Expression

Dual Expression

1. $0 = 1$

$1 = 0$

2. $0 \cdot 1 = 0$

$1 + 0 = 1$

3. $0 \cdot 0 = 0$

$1 + 1 = 1$

4. $1 \cdot 1 = 1$

$0 + 0 = 0$

5. $A \cdot 0 = 0$

$A + 1 = 1$

6. $A \cdot 1 = A$

$A + 0 = A$

7. $A \cdot A = A$

$A + A = A$

$$8. A \cdot \bar{A} = 0$$

$$9. A \cdot B = B \cdot A$$

$$10. A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

$$11. A \cdot (B + C) = AB + AC$$

$$12. A(A + B) = A$$

$$13. A(A \cdot B) = A \cdot B$$

$$14. \overline{AB} = \bar{A} + \bar{B}$$

$$15. (A + B)(\bar{A} + C)(B + C) = (A + B)(\bar{A} + C)$$

$$16. A + \bar{B}C = (A + \bar{B})(A + C)$$

$$17. (A + C)(\bar{A} + B) = AB + \bar{A}C$$

$$18. (A + B)(C + D) = AC + AD + BC + BD$$

$$19. A + B = AB + \bar{A}B + A\bar{B}$$

$$20. \overline{AB + \bar{A} + AB} = 0$$

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

$$A + B = B + A$$

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

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

$$A + AB = A$$

$$A + A + B = A + B$$

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

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

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

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

$$(AB + CD) = (A + C)(A + D)(B + C)$$

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

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

DeMorgans Theorem:

Theorem 1: $\overline{AB} = \bar{A} + \bar{B}$

This theorem states that the complement of a product is equal to addition of the complements.

S.no	A	B	$\overline{AB}$	$\bar{A}$	$\bar{B}$	$\bar{A} + \bar{B}$
1	0	0	1	1	1	1
2	0	1	1	1	0	1
3	1	0	1	0	1	1
4	1	1	0	0	0	0

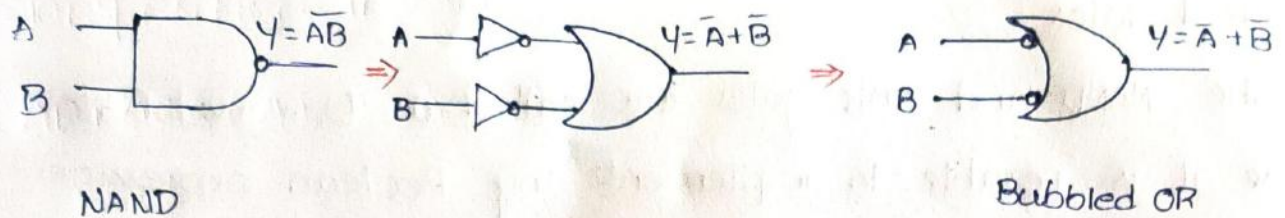


fig:- DeMorgan's First Theorem

Theorem 2: $\overline{A+B} = \bar{A} \cdot \bar{B}$

This theorem states that the complement of a sum is equal to the product of complements.

S.no	A	B	$\overline{A+B}$	$\bar{A}$	$\bar{B}$	$\bar{A} \cdot \bar{B}$
1	0	0	1	1	1	1
2	0	1	0	1	0	0
3	1	0	0	0	1	0
4	1	1	0	0	0	0

Truth table to verify DeMorgan's Second theorem

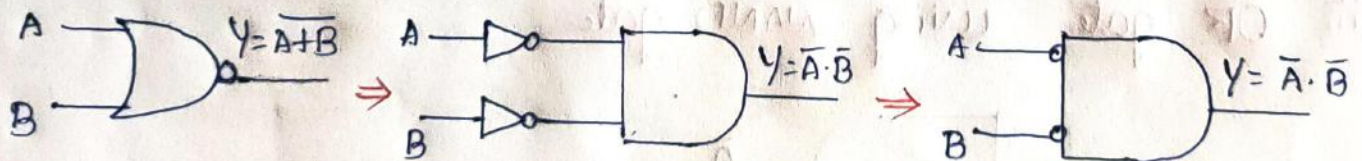


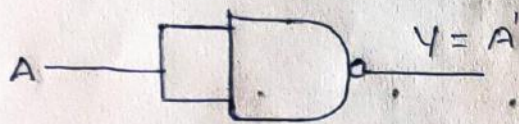
fig:- DeMorgan's Second Theorem

Universal Gates

The NAND and NOR gates are called as Universal Gates because it is possible to implement any Boolean expression with the help of only NAND or only NOR gates.

Realization of gates using NAND gate

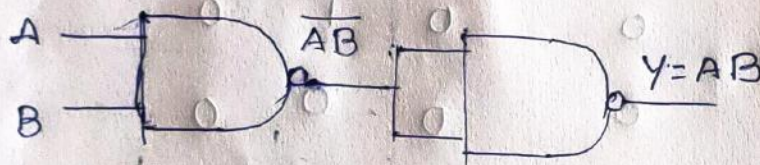
i) NOT gate using NAND gate



NOT gate

A	Y
0	1
1	0

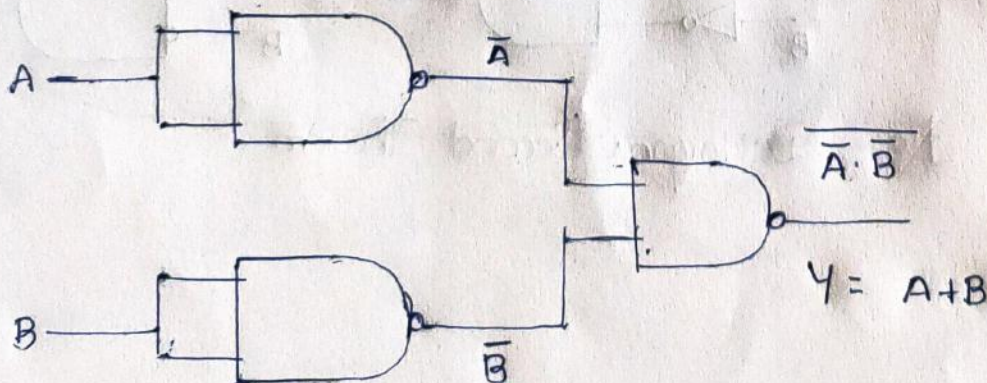
ii) AND gate using NAND gate



AND gate

A	B	$Y = AB$
0	0	0
0	1	0
1	0	0
1	1	1

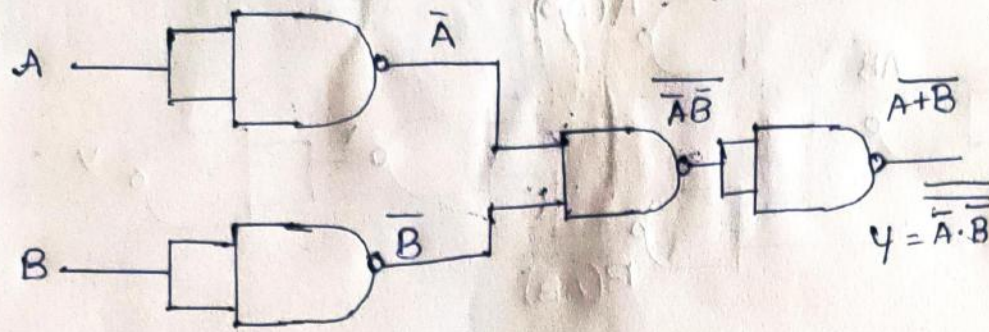
iii) OR gate using NAND gate



OR gate

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

iv, NOR gate using NAND gate

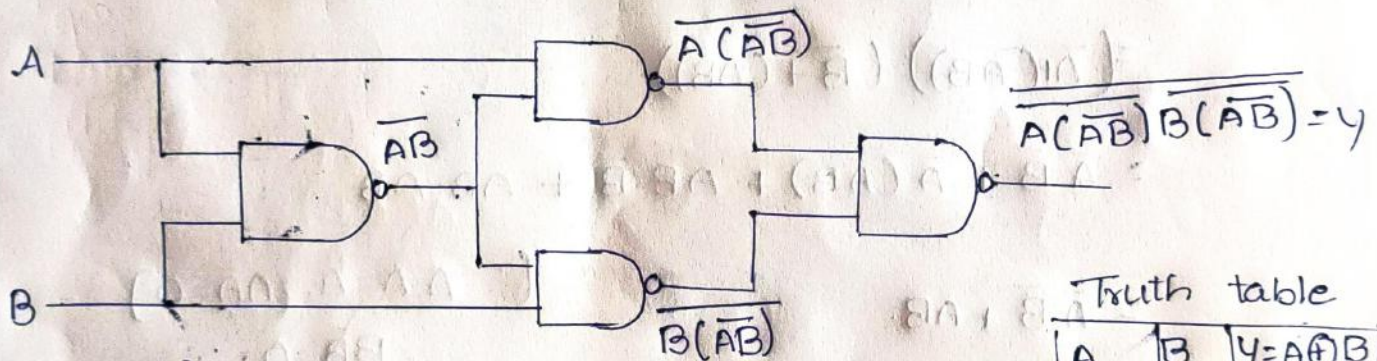


Truth table

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

NOR gate

v, EX-OR gate using NAND gate



Truth table

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

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

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

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

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

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

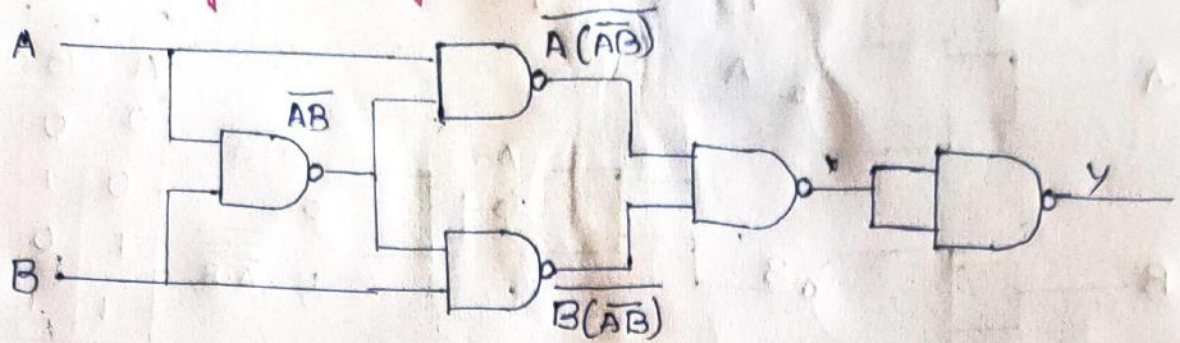
($\therefore$ DeMorgan's theorem)

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

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

($\therefore A\bar{A} = 0, B\bar{B} = 0$)

vi, X-NOR using NAND gate



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

$$Y = \overline{A(\overline{AB})} \overline{B(\overline{AB})} \quad (\because \overline{\overline{A}} = A)$$

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

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

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

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

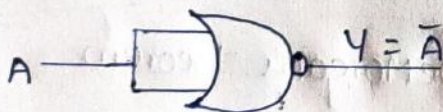
$$(\because A \cdot A = A, A \overline{A} = 0) \\ B \overline{B} = 0$$

Truth table

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

Realization of gates using only NOR gate

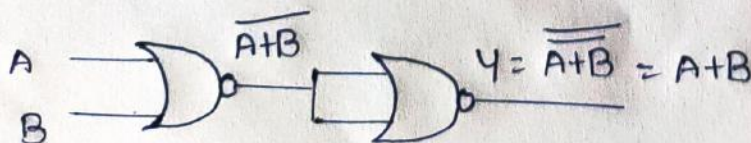
i, NOT gate using NOR gate



Truth table

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

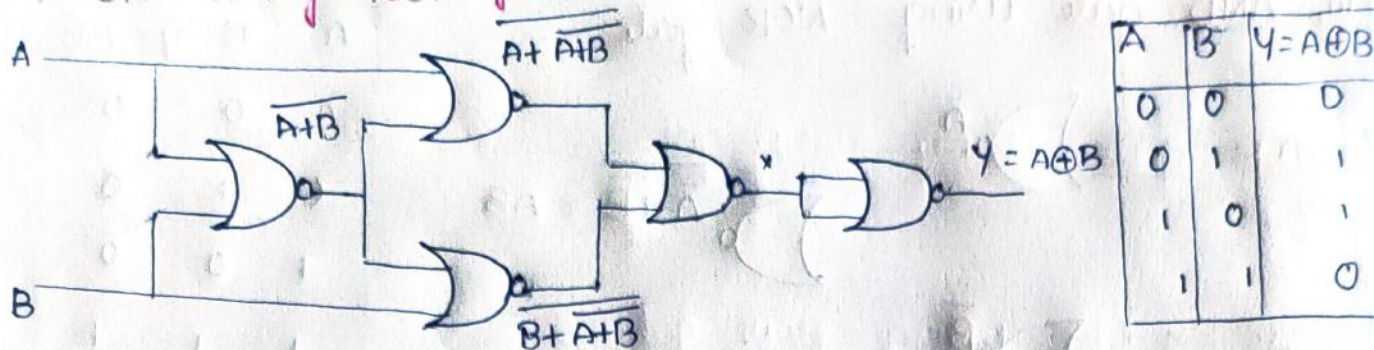
ii, OR gate using NOR gate



Truth table

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

vi, X-OR using NOR gate.



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

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

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

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

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

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

$$\left[\because \bar{\bar{A}} = A, \overline{A+B} = \bar{A} \cdot \bar{B}, \right. \\ \left. A\bar{A} = 0, B\bar{B} = 0 \right]$$

Boolean function and Expression:

Boolean algebra deals with binary variables and logic operations. A Boolean function is described by an algebraic expression called Boolean expression which consists of binary variables, the constants 0 and 1 and the logic operation symbols.

$$\text{Ex:- } F(A, B, C, D) = A + B\bar{C} + ADC$$

Various ways to represent a given function

• Sum of Products form

• Product of Sums form

Sum of Product (Sop) Form:-

- This is also called disjunctive canonical form (DCF) or Expanded Sum of products Form or Canonical sum of products form.
- In this form, the function is the sum of a number of products terms where each product term contains all variables of the function either in complemented or uncomplemented form.
- This can also be derived from the truth table by finding the sum of all the terms that corresponds to those combinations for which 'f' assumes the value 1.

Ex:- $f(A, B, C) = \bar{A}B + \bar{B}C$

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

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

- The product term which contains all the variables of the functions either in complemented or uncomplemented form is called a minterm.

- The minterm is denoted as $m_0, m_1, m_2, \dots$

- Another way of representing the function in canonical sop form is by showing the sum of minterms for which the function equals to 1.

for ex:-

$$f(A, B, C) = m_1 + m_2 + m_3 + m_6$$

(or)

$$f(A, B, C) = \sum m(1, 2, 3, 6)$$

Where $\sum m$ represents the sum of all the minterms.

Product of Sum (POS) Forms

- This form is also called as Conjunctive Canonical Form or Expanded product-of-sum Form or Canonical product of Sums Form.
- This is by considering the combinations for which $f=0$
- Each term is a sum of all the variables.
- The function $f(A, B, C) = (\bar{A} + \bar{B})(A + B)$
$$= (\bar{A} + \bar{B} + C\bar{C})(A + B + C\bar{C})$$
$$= (\bar{A} + \bar{B} + C)(\bar{A} + \bar{B} + \bar{C})(A + B + C)$$
$$(A + B + \bar{C})$$
- The sum term which contains each of the 'n' variables in either complemented or uncomplemented form is called a maxterm.
- Maxterm is represented as $M_0, M_1, M_2, \dots$

Ex:- $f(A, B, C) = M_0 \cdot M_1 \cdot M_7$

or

$$f(A, B, C) = \prod M(0, 1, 7)$$

SOP standard form

- The Complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function.

Ex1- $f(A, B, C) = \sum m(0, 2, 6, 7)$

$$\overline{f(A, B, C)} = \sum m(1, 3, 4, 5) = m_1 + m_3 + m_4 + m_5$$

$$f = \overline{m_1 + m_3 + m_4 + m_5} = \overline{m_1} \cdot \overline{m_3} \cdot \overline{m_4} \cdot \overline{m_5}$$

$$= M_1 M_3 M_4 M_5$$

$$= \Pi M (1, 3, 4, 5)$$

1. Expand $A(\bar{A}+B)(\bar{A}+B+\bar{C})$ to maxterms and minterms

Given pos form is

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

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

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

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

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

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

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

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

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

$$(\because A \cdot A = A)$$

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

$$= (000)(001)(010)(011)(100)(101)(110)(111)$$

$$= M_0 \cdot M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot M_5$$

$$= \Pi M (0, 1, 2, 3, 4, 5)$$

The maxterms M_6, M_7 are missing in the pos form

So, the sop form will contain the minterms 6 and 7

$$= \Sigma m(6, 7)$$

2. Convert the expression into their standard SOP form.

$$Y = AB + AC + BC$$

Given that $Y = AB + AC + BC$

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

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

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

3. Expand $A + BC + ABC$ to maxterms and minterms

Given that $Y = A + BC + ABC$

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

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

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

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

$$= (000) (001) (010) (011) (100) (101) (110) (111)$$

$$= (111) + (110) + (101) + (100) + (011)$$

$$= m_7, m_6, m_5, m_4, m_3$$

$$= \sum m(3, 4, 5, 6, 7)$$

Missing minterms are m_0, m_1, m_2

So, the maxterms are M_0, M_1, M_2

$$= \prod M(0, 1, 2)$$

3. For the given truth table, write the logic expression in the standard form.

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

Sol:-

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

$$\rightarrow \bar{A}\bar{B}C \quad (m_1)$$

$$\rightarrow A\bar{B}\bar{C} \quad (m_4)$$

$$\rightarrow ABC \quad (m_7)$$

$\because A=1$
 $\bar{A}=0$
 for minterms

minterms

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

$$= \sum m(1, 4, 7)$$

5. For the given truth table write the logical expression in the standard pos form.

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

Sol: Given truth table

A	B	C	Y
---	---	---	---

$$0 \quad 0 \quad 0 \quad 0 \rightarrow A+B+C$$

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

$$0 \quad 1 \quad 0 \quad 0 \rightarrow A+\bar{B}+C$$

$$0 \quad 1 \quad 1 \quad 0 \rightarrow A+\bar{B}+\bar{C}$$

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

$$1 \quad 0 \quad 1 \quad 0 \rightarrow \bar{A}+B+\bar{C}$$

$$1 \quad 1 \quad 0 \quad 0 \rightarrow \bar{A}+\bar{B}+C$$

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

$$\left[\begin{array}{l} \because A=0 \\ \bar{A}=1 \end{array} \right.$$

for maxterm]

Standard pos form:

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

1. Simplify the expression given below

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

Given expression in SOP form

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

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

$$[\because AB + AB = AB,$$

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

$$B \cdot B = B]$$

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

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

$$= B + B \quad [\because B + B = B]$$

$$= B$$

2. Simplify the following three variable Boolean expression.

$$Y = \sum m(2, 4, 6)$$

Given expression in SOP form

$$Y = m_2 + m_4 + m_6$$

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

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

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

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

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

$$[\because m_2 = 010$$

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

$$m_4 = 100$$

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

$$m_6 = 110$$

$$AB\bar{C}]$$

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

Simplify the following three variable logic expression

$$Y = \Pi M(1, 3, 5)$$

Given expression in pos form

$$Y = M_1 \cdot M_3 \cdot M_5$$

$$(\because M_1 = 001)$$

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

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

$$M_3 = 011$$

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

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

$$M_5 = 101$$

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

$$(\because A\bar{A}=0, \bar{C}\bar{C}=\bar{C}, B\bar{B}=0)$$

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

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

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

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

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

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

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

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

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

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

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

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

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

Karnaugh Maps (or) K-maps:

The K-map is a chart or a graph, composed of an arrangement of adjacent cells, each representing a particular combination of variables in sum or product form.

- The K-map is systematic method of simplifying the Boolean expression.

Two Variable K-map:-

A two variable expression can have $2^2 = 4$ possible combinations of the input variables A and B.

SOP Form:-

- * The 2 variable K-map has $2^2 = 4$ squares. These squares are called cells.
- * A '1' is placed in any square indicates that corresponding minterm is included in the output expression, and a 0 or no entry in any square indicates that the corresponding minterm does not appear in the expression for output.

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

POS form:-

Each sum term in the standard pos expression is called a Maxterm. A function in two variables (A, B) has 4 possible maxterms, $A+B$, $A+\bar{B}$, $\bar{A}+B$ and $\bar{A}+\bar{B}$. They are

represented as M_0, M_1, M_2 and M_3 respectively.

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

SOP Form

The possible minterm grouping in a two variable k-map are shown below

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

$$f = \bar{A}$$

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

$$f = \bar{B}$$

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

$$f = B$$

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

$$f = A$$

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

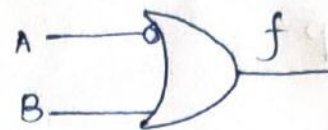
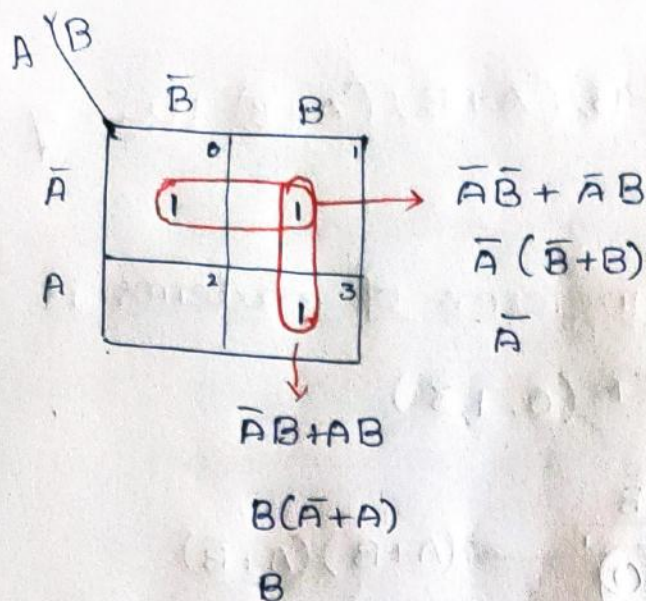
$$f = 1$$

Reduce the expression $f = \bar{A}\bar{B} + \bar{A}B + AB$ using K-map

Given expression $f = \bar{A}\bar{B} + \bar{A}B + AB$ expressed in minterms

$$f = m_0 + m_1 + m_3$$

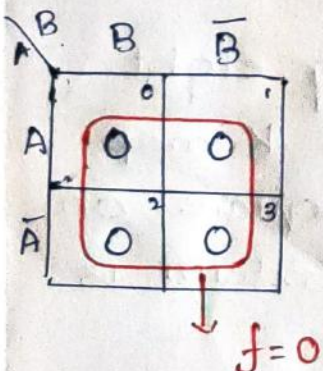
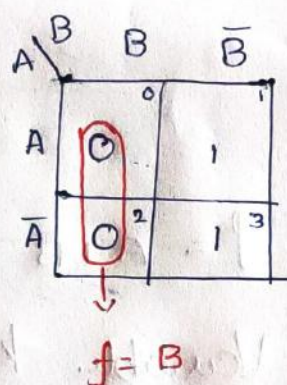
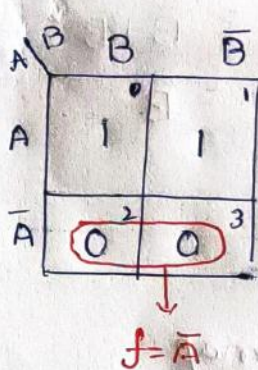
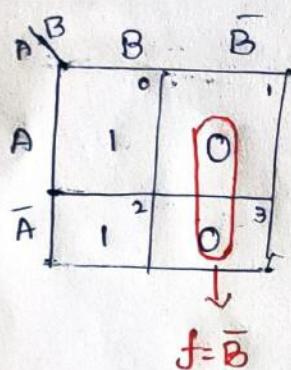
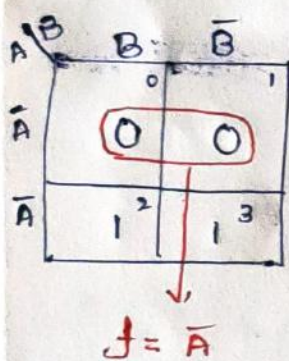
$$f = \sum m(0, 1, 3)$$



$f = \bar{A} + B$

POS form

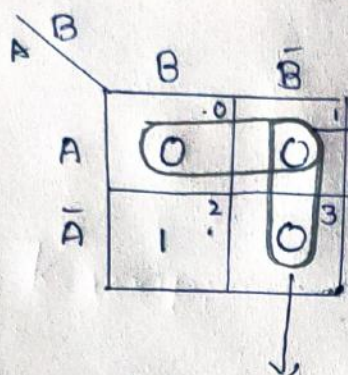
The possible minterm grouping in a two variable k-map are shown below.



Reduce the expression $f = (A + \bar{B})(\bar{A} + \bar{B})(A + B)$ using K-map

Sol:- The given expression in terms of maxterms is

$$f = \prod M(0, 1, 3)$$



$$\begin{aligned} & (A + B)(A + \bar{B}) \\ &= AA + A\bar{B} + AB + B\bar{B} \\ &= A + A(B + \bar{B}) \\ &= A + A \\ &= A \end{aligned}$$

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

$$\begin{aligned} &= A\bar{A} + \bar{A}\bar{B} + A\bar{B} + \bar{B}\bar{B} \\ &= \bar{B}(\bar{A} + A) + \bar{B} \\ &= \bar{B} + \bar{B} \\ &= \bar{B} \end{aligned}$$



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

Three Variable k-map

A function in three variables (A, B, C) can be expressed in SOP and POS form having eight possible combination. A three variable K-map have 8 squares or cells and each square on the map represents a minterm or maxterm is shown in the figure below.

A \ BC	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$ (m_0)	$\bar{A}\bar{B}C$ (m_1)	$\bar{A}BC$ (m_3)	$\bar{A}\bar{B}\bar{C}$ (m_2)
1	$A\bar{B}\bar{C}$ (m_4)	$A\bar{B}C$ (m_5)	ABC (m_7)	$A\bar{B}\bar{C}$ (m_6)

minterms

A \ BC	00	01	11	10
0	$A+B+C$ (M_0)	$A+B+\bar{C}$ (M_1)	$A+\bar{B}+\bar{C}$ (M_3)	$A+\bar{B}+C$ (M_2)
1	$\bar{A}+B+C$ (M_4)	$\bar{A}+B+\bar{C}$ (M_5)	$\bar{A}+\bar{B}+\bar{C}$ (M_7)	$\bar{A}+\bar{B}+C$ (M_6)

maxterms

Four Variable k-map

A four variable (A,B,C,D) expression can have $2^4 = 16$ possible combinations of input variables. A four variable k-map has $2^4 = 16$ squares or cells and each square on the map represents either a minterm or a maxterm as shown in figure below.

AB \ CD		00	01	11	10
AB	00	0 $\bar{A}\bar{B}\bar{C}\bar{D}$ (m_0)	1 $\bar{A}\bar{B}\bar{C}D$ (m_1)	3 $\bar{A}\bar{B}CD$ (m_3)	2 $\bar{A}\bar{B}C\bar{D}$ (m_2)
	01	4 $\bar{A}B\bar{C}\bar{D}$ (m_4)	5 $\bar{A}B\bar{C}D$ (m_5)	7 $\bar{A}BCD$ (m_7)	6 $\bar{A}BC\bar{D}$ (m_6)
	11	12 $AB\bar{C}\bar{D}$ (m_{12})	13 $AB\bar{C}D$ (m_{13})	15 $ABCD$ (m_{15})	14 $ABC\bar{D}$ (m_{14})
	10	8 $AB\bar{C}\bar{D}$ (m_8)	9 $AB\bar{C}D$ (m_9)	11 $ABCD$ (m_{11})	10 $ABC\bar{D}$ (m_{10})

Sop form (minterms)

AB \ CD		00	01	11	10
AB	00	0 $A+B+C+D$ (M_0)	1 $A+B+C+\bar{D}$ (M_1)	3 $A+B+\bar{C}+D$ (M_3)	2 $A+B+\bar{C}+\bar{D}$ (M_2)
	01	4 $A+\bar{B}+C+D$ (M_4)	5 $A+\bar{B}+C+\bar{D}$ (M_5)	7 $A+\bar{B}+\bar{C}+D$ (M_7)	6 $A+\bar{B}+\bar{C}+\bar{D}$ (M_6)
	11	12 $\bar{A}+\bar{B}+C+D$ (M_{12})	13 $\bar{A}+\bar{B}+C+\bar{D}$ (M_{13})	15 $\bar{A}+\bar{B}+\bar{C}+D$ (M_{15})	14 $\bar{A}+\bar{B}+\bar{C}+\bar{D}$ (M_{14})
	10	8 $\bar{A}+B+C+D$ (M_8)	9 $\bar{A}+B+C+\bar{D}$ (M_9)	11 $\bar{A}+B+\bar{C}+D$ (M_{11})	10 $\bar{A}+B+\bar{C}+\bar{D}$ (M_{10})

pos form (maxterms)

For the k-map shown in figure Write the simplified Boolean expression in SOP form and pos form

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

SOP form

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

Groupings for SOP form:

- Group 1: $\bar{C}AB$ (cells 1, 3, 2)
- Group 2: $BC\bar{C}$ (cells 1, 5)
- Group 3: AB (cells 3, 7)

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

pos form

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

Groupings for POS form:

- Group 1: $A+B$ (cells 0, 1, 4, 5)
- Group 2: $(A+\bar{C})$ (cells 1, 3, 5, 7)
- Group 3: $B+\bar{C}$ (cells 1, 3, 2, 6)

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

Write down the simplified Boolean expression for the k-map shown in figure.

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

sol:-

SOP form

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

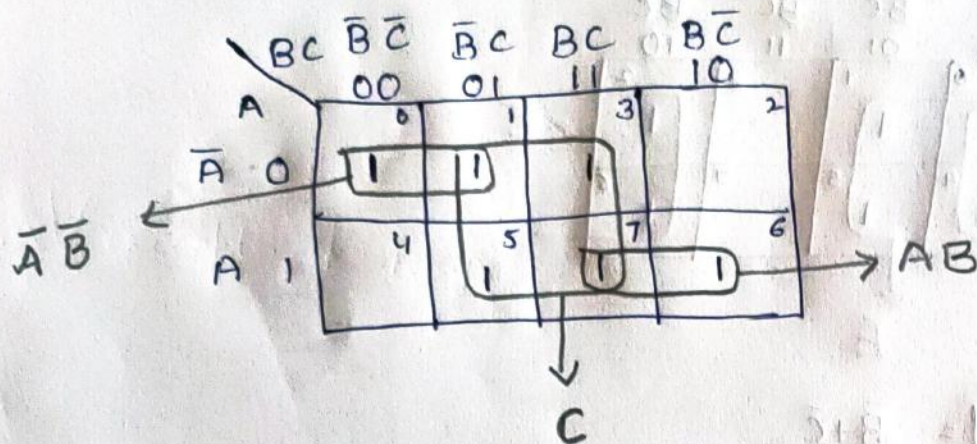
Groupings for SOP form:

- Group 1: $\bar{A}\bar{B}\bar{C}$ (cell 0)
- Group 2: $AB\bar{C}$ (cell 3)
- Group 3: $\bar{A}BC$ (cell 5)
- Group 4: $A\bar{B}C$ (cell 6)

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

Using k-map, realize the following expression

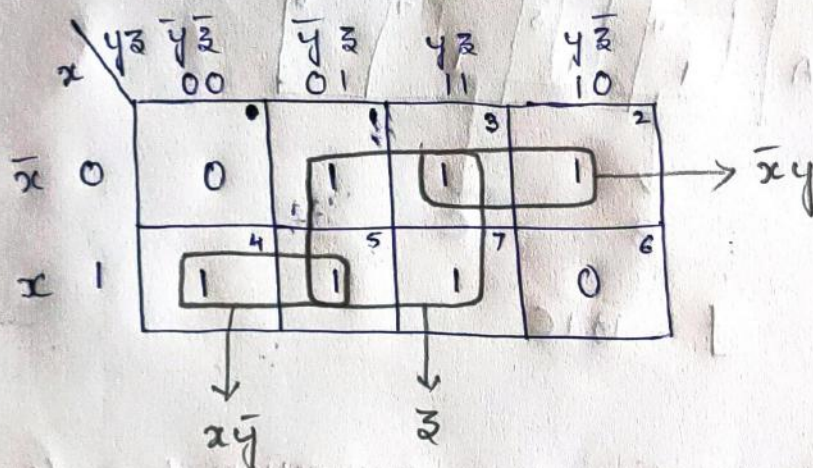
$$f(A, B, C) = \sum m(0, 1, 3, 5, 6, 7)$$



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

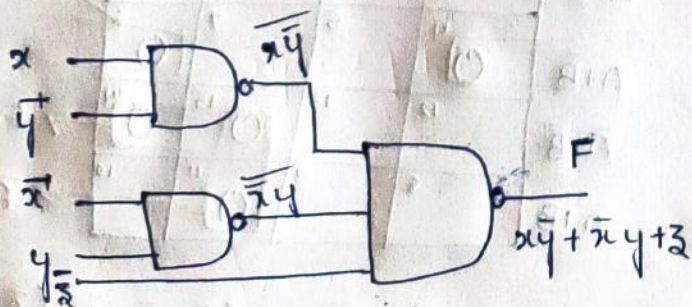
Implement the following Boolean function with NAND gates

$$F(x, y, z) = \sum(1, 2, 3, 4, 5, 7)$$

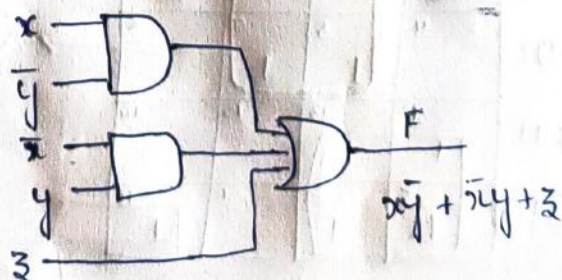


$$F = x\bar{y} + \bar{x}y + z$$

NAND gate implementation

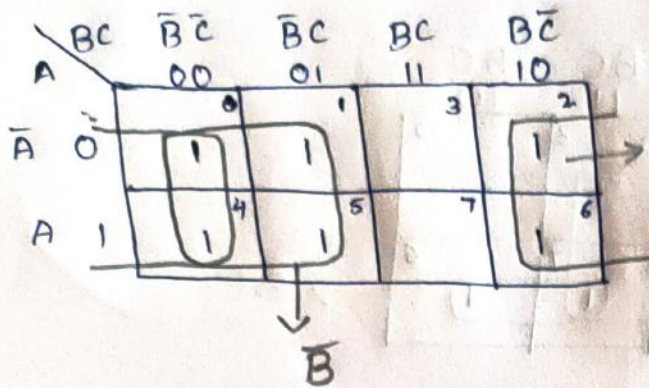


using AND OR gates



• Simplify the Boolean function using K-maps

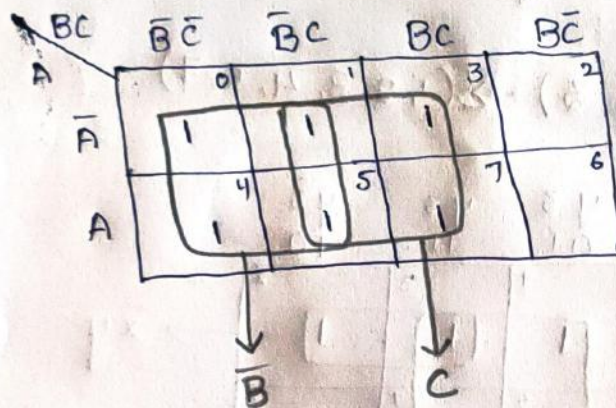
$$f = \sum m(0, 1, 2, 4, 5, 6)$$



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

• Simplify the following expression using K-maps

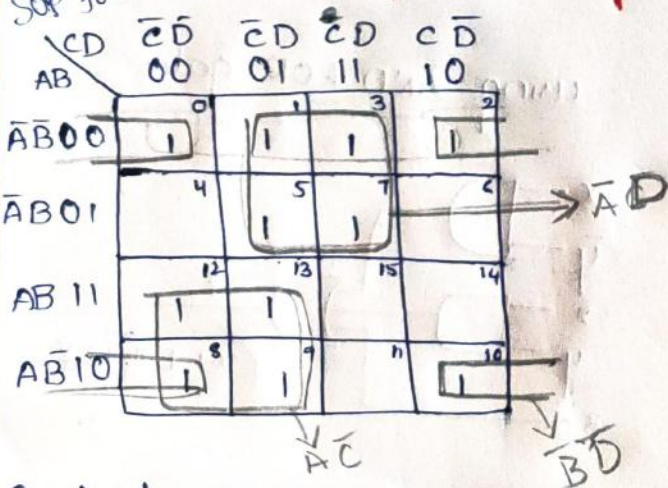
$$f = \sum m(0, 1, 3, 4, 5, 7)$$



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

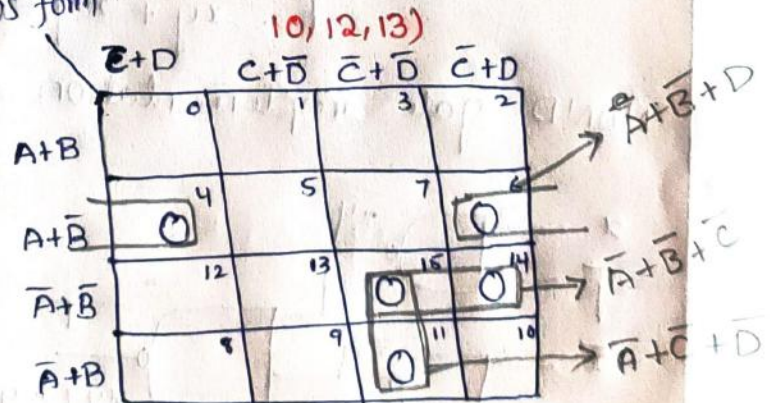
Reduce using mapping the expression $f = \sum m(0, 1, 2, 3, 5, 7, 8, 9, 10, 12, 13)$

SOP form



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

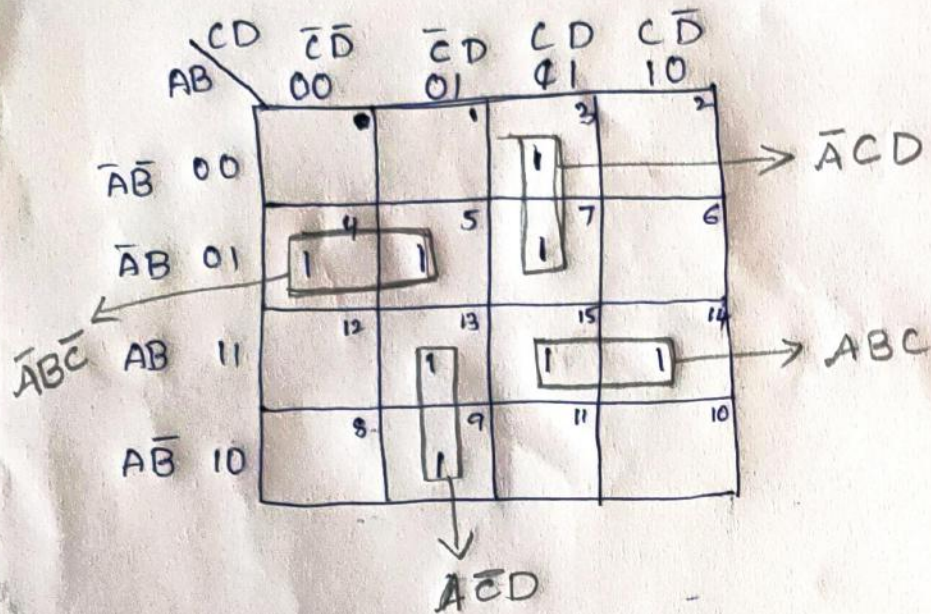
POS form



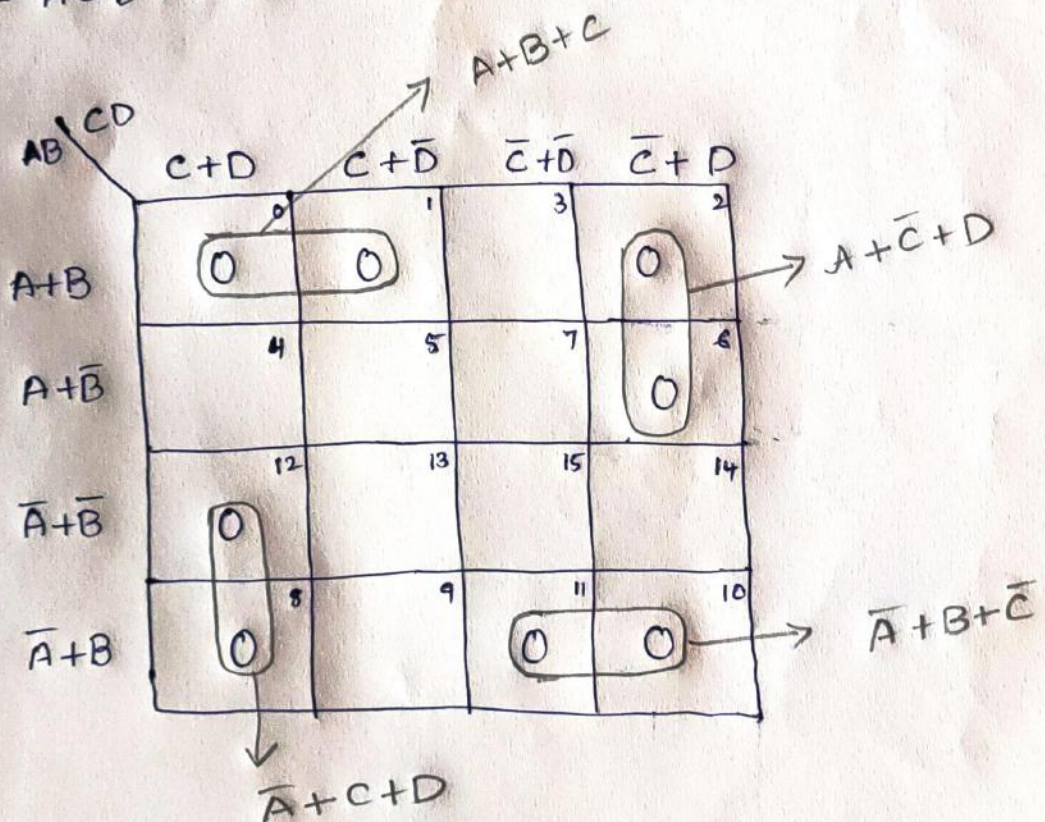
$$\text{POS form } f = (A+B+D)(\bar{A}+\bar{B}+C)(\bar{A}+B+\bar{C})(\bar{A}+\bar{C}+\bar{D})$$

Simplify the following boolean function by using four-variable k-map.

$$f = \sum (3, 4, 5, 7, 9, 13, 14, 15)$$



Sop form $f = A\bar{C}D + ABC + \bar{A}B\bar{C} + \bar{A}CD$



POS form $\Rightarrow f = (\bar{A} + \bar{C} + D)(\bar{A} + B + \bar{C})(A + \bar{C} + D)(A + B + C)$

1. Convert the Gray code 1101 to binary

21

Gray	1	1	0	1
	↓	↓	↓	↓
Binary	1	0	0	1

2. 1 0 1 1 0 1

Gray	1	0	1	1	0	1
	↓	↓	↓	↓	↓	↓
Binary	1	1	0	1	1	0

Error Detecting and Correcting Codes:

→ Codes which allow only error detection are called "Error Detecting codes".

→ Codes which allow error detection and correction are called "Error Detecting and Correcting codes".

1. Parity Bit

3-bit Message

Message with odd parity

Msg. with even parity

A B C

Message parity

Message parity

0 0 0

0 0 0 1

000 0

0 0 1

0 0 1 0

001 1

0 1 0

0 1 0 0

010 1

0 1 1

0 1 1 1

011 0

1 0 0

1 0 0 0

100 1

1 0 1

1 0 1 1

101 0

1 1 0

1 1 0 1

110 0

1 1 1

1 1 1 0

111 1

Hamming code:

No. of parity bits,

$$2^P \geq r + P + 1$$

$P \rightarrow$ No. of parity bits

$r \rightarrow$ No. of information bits.

Example:

- for 4 bit information, let $r = 4$

$P \rightarrow$ trial & error let $p = 2$

$$2^2 \geq 4 + 2 + 1$$

$$4 \geq 7 \quad \times$$

let $p = 3$

$$2^3 \geq 4 + 3 + 1$$

$$8 \geq 8 \quad \checkmark$$

$\therefore$ Three parity bits are required to provide single error correction for four information bits.

Error-Correcting codes:

1. 7-bit Hamming code

$P_1 \quad P_2 \quad D_3 \quad P_4 \quad D_5 \quad D_6 \quad D_7$

$$P_1 \rightarrow 1, 3, 5, 7$$

$$P_2 \rightarrow 2, 3, 6, 7$$

$$P_4 \rightarrow 4, 5, 6, 7$$

2) The 12-bit Hamming code

28

$P_1 \ P_2 \ D_3 \ P_4 \ D_5 \ D_6 \ D_7 \ P_8 \ D_9 \ D_{10} \ D_{11} \ D_{12}$

$P_1 \rightarrow 1, 3, 5, 7, 9, 11$

$P_2 \rightarrow 2, 3, 6, 7, 10, 11$

$P_4 \rightarrow 4, 5, 6, 7, 12$

$P_8 \rightarrow 8, 9, 10, 11, 12$

3) The 15-bit Hamming code

$P_1 \ P_2 \ D_3 \ P_4 \ D_5 \ P_6 \ D_7 \ P_8 \ D_9 \ D_{10} \ D_{11} \ D_{12} \ D_{13} \ D_{14} \ D_{15}$

$P_1 \rightarrow 1, 3, 5, 7, 9, 11, 13, 15$

$P_2 \rightarrow 2, 3, 6, 7, 10, 11, 14, 15$

$P_4 \rightarrow 4, 5, 6, 7, 12, 13, 14, 15$

$P_8 \rightarrow 8, 9, 10, 11, 12, 13, 14, 15$

1. Encode the binary coord $\underbrace{1011}_{x=4}$ into seven bit even parity hamming code.

Step 1: find the no. of parity bits required

let $p=3$

$$2^p \geq x+p+1$$

$$2^3 \geq 4+3+1$$

$$8 \geq 8 \checkmark$$

Three parity bits are sufficient

$$\text{Total code bits} = x+p = 4+3=7$$

Step 2) Construct a bit location table

Bit designation	D ₇	D ₆	D ₅	P ₄	D ₃	P ₂	P ₁
Bit location	7	6	5	4	3	2	1
Binary location number	111	110	101	100	010	010	001
Information bits	1	0	1		1		
parity bits				0		0	1

Step 3) Determine the parity bits

$$1 \quad P_1 \rightarrow 3 \quad 5 \quad 7 \rightarrow 1 \quad 1 \quad 1$$

even

$$\downarrow 0 \quad P_2 \rightarrow 3 \quad 6 \quad 7 \rightarrow 1 \quad 0 \quad 1$$

$$0 \quad P_3 \rightarrow 5 \quad 6 \quad 7 \rightarrow 1 \quad 0 \quad 1$$

$\therefore$ Seven bit Hamming code is 1 0 1 0 1 0 1

2. Determine the single error-correcting code for the information code 10111 for odd parity.

Step 1: Find the number of parity bits required

$$\text{let } p=3$$

$$x=5$$

$$2^p \geq x+p+1$$

$$2^3 \geq 5+3+1$$

$$8 \geq 9 \quad \times$$

$$\text{let } p=4$$

$$2^p \geq x+p+1$$

$$2^4 \geq 5+4+1$$

$$16 \geq 10 \quad \checkmark$$

Hence four bits are sufficient

$$\text{Total code bit} = 5 + 4 = 9$$

Step 2: Construct a bit location table

Bit designation	D ₉	P ₈	D ₇	D ₆	D ₅	P ₄	D ₃	P ₂	P ₁
Bit location	9	8	7	6	5	4	3	2	1
Binary location number	1001	1000	0111	0110	0101	0100	0011	0010	0001
Information bits	1		0	1	1		1		1 0
Parity bits		0				1			<u>odd</u>

Step 3: Determine the parity Bits

0 For P₁ : 3 5 7 9 1 1 0 1

1 For P₃ : 3 6 7 1 1 0

1 For P₄ : 5 6 7 1 1 0

For P₈ : Bit P₈ checks bit locations 8 and 9.
and must be 0 to have an odd parity.

Step 4: Enter the parity bits into the table to form a nine bit Hamming code

1 0 0 1 1 1 1 1 0

Detecting and Correcting an Error:

1. Assume that the even parity Hamming code in example 011001 is transmitted and that 0100011 is received

The receiver does not know what was transmitted.

Determine bit location where error has occurred using received code.

Step 1: Construct the bit location table

Bit designation	D ₇	D ₆	D ₅	P ₄	D ₃	P ₃	P ₁
Bit location	7	6	5	4	3	2	1
Binary location number	111	110	101	100	011	010	001
Received code	0	1	0	0	0	1	1

Step 2: Check for parity bits even

- 1 For P₁: checks locations 1, 3, 5, 7 → 1 0 0 0 (LSB)
- 0 For P₃: checks locations 2, 3, 6, 7 → 1 0 1 0
- 1 For P₄: checks locations 4, 5, 6, 7 → 0 0 1 0

∴ The resultant word } 1 0 1

This says that the bit in the number 5 location is in error. It is 0 & it should be 1.

∴ The correct code 0 1 1 0 0 1 1.

The Hamming Code 1011 01101 is received. Correct it if any errors. There are four parity bits and odd parity is used.

ALPHANUMERIC CODES

→ Alphabetic codes are codes used to encode the characters of alphabet in addition to the decimal digits.

The ASCII code - American Standard Code for Information Interchange

→ It is basically a 7-bit code.

$$n = 2^7 = 128$$

Title: The ASCII code.

SB ₆	MSBs							
	000	001	010	011	100	101	110	111
0000	NUL	DEL	space	0	@	P	p	
0001	SOH	DC1	!	1	A	a	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	END	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	^	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	w	w
1000	BS	CAN	(	8	H	X	x	z
1001	HT	EM	)	9	I	Y	y	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DLE

Abbreviations:

ACK - Acknowledge	FF - Form Feed
BEL - Bell	FS - Form Separator
BS - Backspace	GS - Group Separator
CAN - Cancel	HT - Horizontal Tab
CR - Carriage Return	LF - Line Feed
DC1 - Direct Control 1	NAK - Negative Acknowledge
DC2 - Direct Control 2	NUL - Null
DC3 - " " 3	RS - Record Separator
DC4 - " " 4	SI - Shift In
DEL - Delete Idle	SO - Shift Out
DLE - Data Link Escape	SOTI - Start of Heading
EM - End of Medium	STX - Substitute
END - Enquiry	SYN - Synchronous Idle
EOT - End of Transmission	US - Unit Separator
ESC - Escape	VT - Vertical Tab
ETB - End of Transmission Block	
ETX - End of Text	

The EBCDIC Code : Extended Binary Coded Decimal Interchange Code.

$$\therefore 2^8 = 256$$

		MISD (hex)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	NUL	DEL	DC	SP	*								E	T	V	0
	1	SOH	DC1	QOS			/			a	j	~		A	T		1
	2	SIX	DC2	LS	MS					b	k	s		B	K	S	2
	3	TX	DC3							c	l	t		C	L	T	3
	4	PF	RTS	BYT	IN					d	m	u		D	M	U	4
	5	LST(hex)	HT	NL	IF	RV				e	n	v		E	N	V	5
	6	LC	BS	EDB	YC					f	o	w		F	O	W	6
	7	DEL	IL	PRE	LOT					g	p	x		G	P	X	7
	8									h	q	y		H	Q	Y	8
	9		CNN							i	r	z		I	R	Z	9
	A		EM														
	B	SMM	CC	SM		ø	!	I	:								
	C	VT				.	¢	,	#								
	D	FF	IFS		DEY	<	†	%	@								
	E	CR	IQS	END	MOI	(	)	-	'								
	F	SD	IRS	PCR		+	;	>	=								
		SJ	IUS	BEL	SUB	I	'	?	,								

UNIT-2

Combinational Circuit:-

A Combinational circuit may be defined as a logic circuit the output of which depends only upon the combination of the inputs. The output does not depend on the past value of inputs or outputs. Therefore, Combinational circuits do not need any memory.

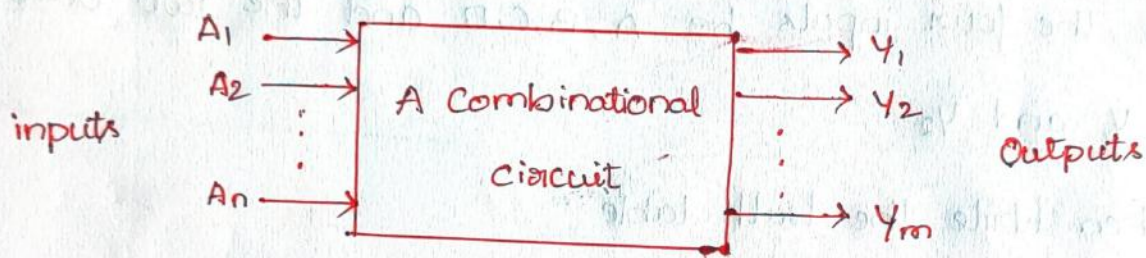


fig:- Block diagram of a Combinational circuit.

A Combinational circuit can have a number of inputs and a number of outputs. The above figure has n inputs and m outputs. Between the inputs and outputs, logic gates are connected, and hence, combinational circuit basically consists of logic gates.

The Various Steps Involved in Designing Procedure of a Combinational logic may be listed under:

- we will be given a problem
- Then, we determine the number of inputs and outputs and assign letter symbols to input and output variables.
- After that we write a truth table relating the inputs and

outputs.

Then, Write k-map for each output and obtain the simplified Boolean expression for each output.

Lastly draw the logic diagram.

Ex:- A circuit has four inputs and two outputs. One of the outputs is high when majority of inputs are high. The second output is high only when all inputs are of same type. Design the combinational circuit.

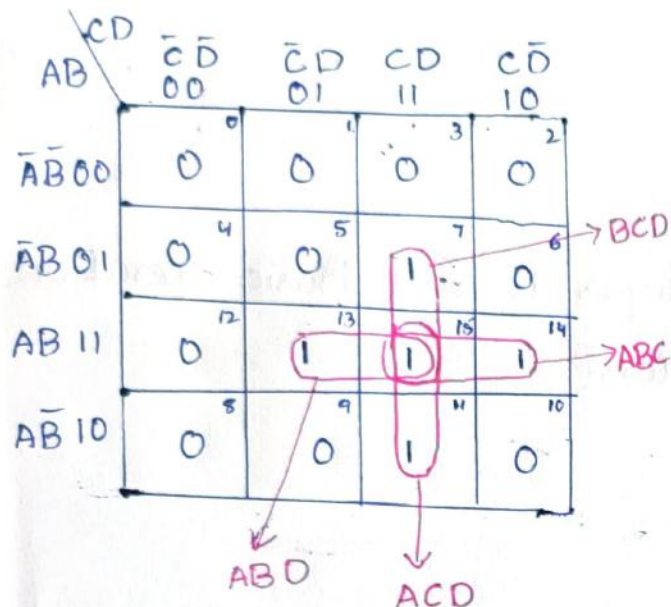
Sol:- i, let the four inputs be A, B, C, D and the two outputs be Y_1 and Y_2 .

ii, Then Write the truth table

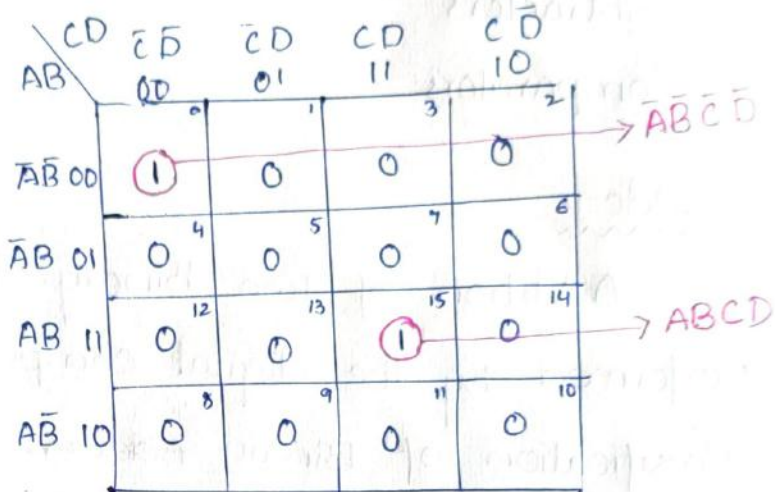
Decimal	Inputs				outputs	
	A	B	C	D	Y_1	Y_2
0	0	0	0	0	0	1
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	0	1	1	0	0
4	0	1	0	0	0	0
5	0	1	0	1	0	0
6	0	1	1	0	0	0
7	0	1	1	1	1	0
8	1	0	0	0	0	0
9	1	0	0	1	0	0
10	1	0	1	0	0	0
11	1	0	1	1	1	0
12	1	1	0	0	0	0
13	1	1	0	1	1	0
14	1	1	1	0	1	1
15	1	1	1	1	1	1

iii, Then, Write k-map for each output and get simplified expression.

for output y_1



for output y_2

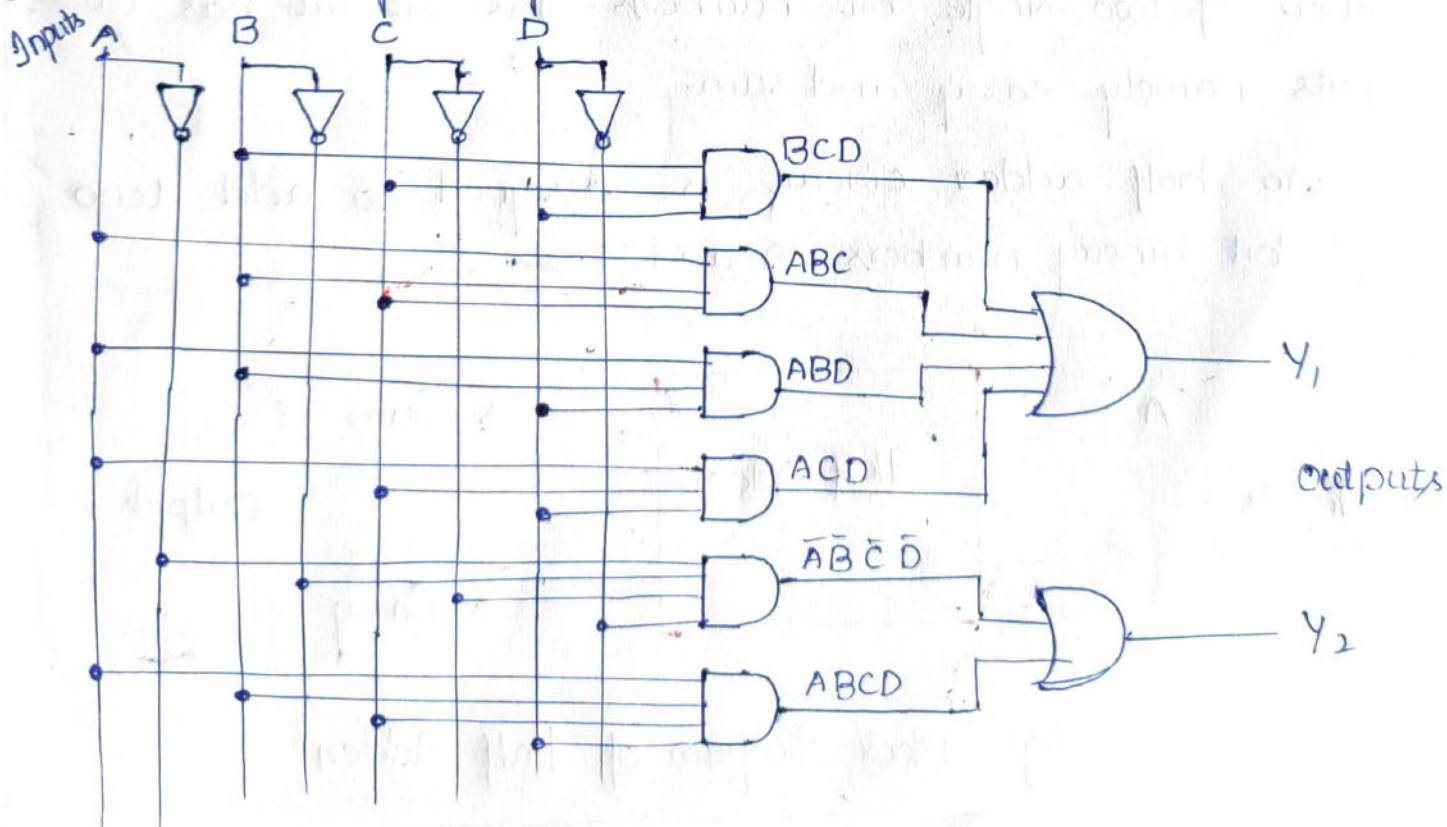


Simplified expressions

$$y_1 = BCD + ABC + ABD + ACD$$

$$y_2 = \bar{A}\bar{B}\bar{C}\bar{D} + ABCD$$

iv, Draw the logic diagram.



Classification of Combinational Circuits

The Combinational circuits may be classified as

- i) Code converters
- ii) Adders
- iii) Subtractors
- iv) Comparators

• Adders:

Addition of two binary digits is most basic operation performed by the digital computers.

Classification of Binary Adders:

- i) Half adder, and
- ii) Full adder

Half Adder:-

Half adder is a combinational logic circuit with two inputs and two outputs. It is the basic building block for addition of two single bit numbers. This circuit has two outputs namely carry and sum.

An half adder circuit is designed to add two single bit binary numbers A and B.

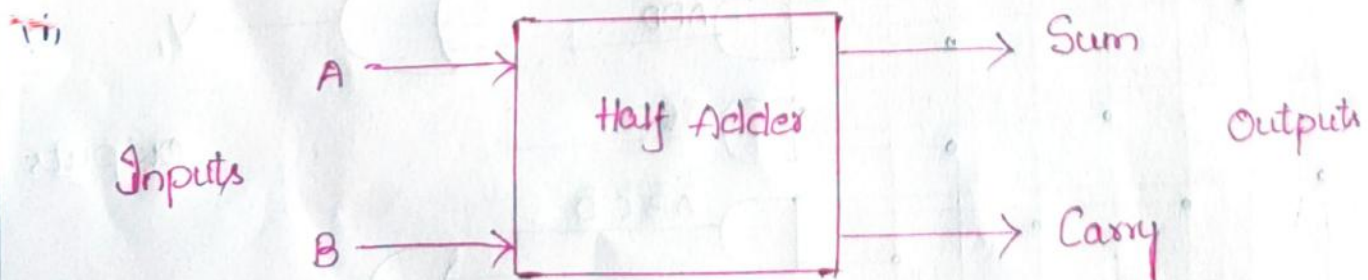
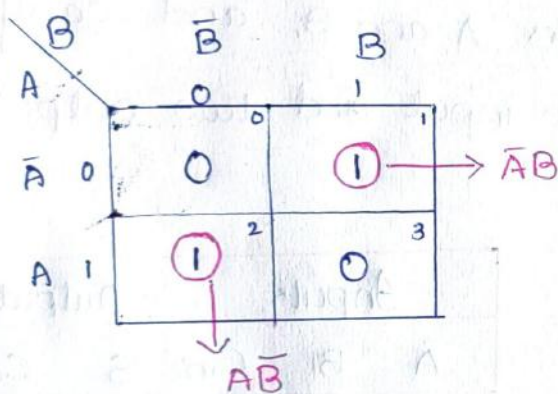


fig:- Block diagram of Half Adder

ii) Truth Table

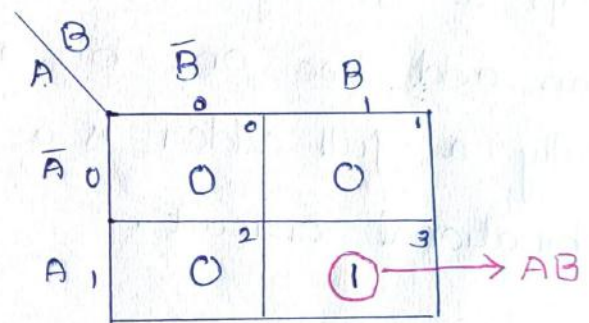
S.NO	Inputs		Outputs	
	A	B	Sum	Carry
1	0	0	0	0
2	0	1	1	0
3	1	0	1	0
4	1	1	0	1

iii) K-maps for Carry and Sum outputs



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

fig:- K-map for Sum output



$$\text{Carry} = AB$$

fig:- K-map for Carry output

iv) Implementation of logic diagram

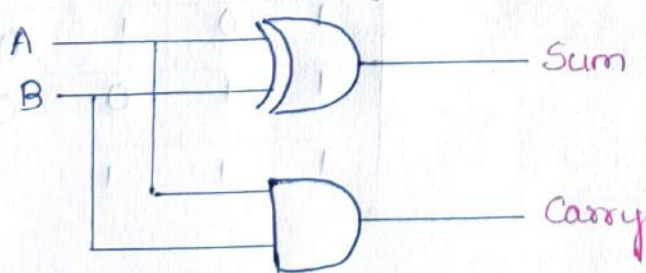


fig:- Half Adder Circuit

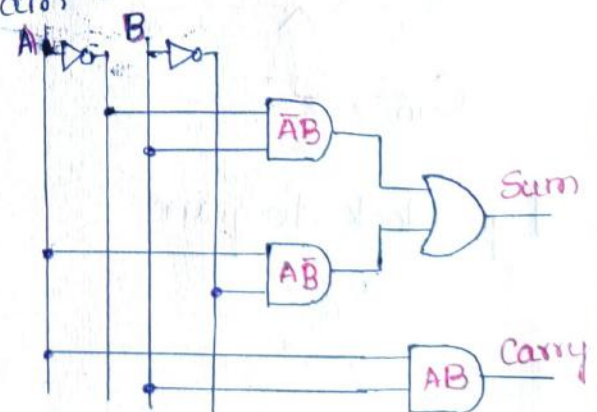


fig:- Half Adder using Basic gates

Drawback:

A half adder can add A_0 and B_0 to produce S_0 and C_0 . However, the addition of next bits requires the addition of

A_1, B_1 and C_0 . The addition of three bits is not possible to perform by using an half adder circuit. Therefore, practically we cannot use a half adder.

$$A = A_1 A_0$$

$$B = B_1 B_0$$

$$\begin{array}{r} + C_0 \\ \hline C_1 \quad S_1 \quad S_0 \end{array} \rightarrow \text{Carry generated from the addition } (A_0 + B_0)$$

Full Adder:

To overcome the drawback of an Half Adder circuit, we develop a 3 single bit adder circuit called Full Adder. It can add two one-bit numbers A and B , and carry C_{in} . Basically, a full adder is a three input and two output combinational circuit.

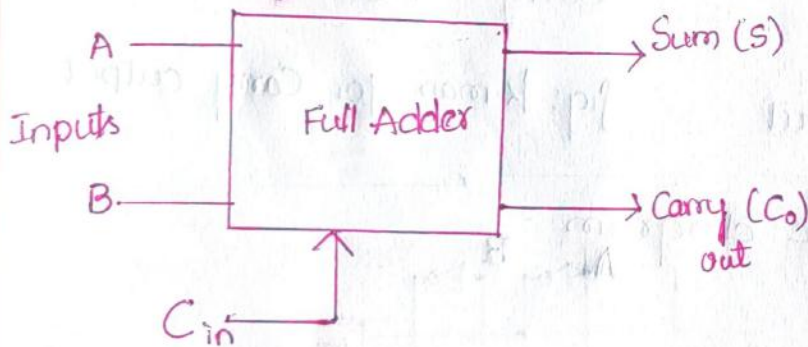


fig1- Block diagram

Inputs			Outputs	
A	B	C_{in}	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

fig1- Truth Table

k-maps for the Sum(S) and Carry out (C_o) outputs.

k-map for Sum

		BCin			
A	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$	
	0	1	3	2	
$\bar{A}$	0	1	0	1	$\bar{A}BC_{in}$
A	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$	
	4	5	7	6	
A	1	0	1	0	$A\bar{B}\bar{C}_{in}$

k-map for Carry out

		BCin			
A	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$	
	0	1	3	2	
$\bar{A}$	0	0	1	0	BC_{in}
A	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$	
	4	5	7	6	
A	0	1	1	1	AC_{in}

$$\text{Carry out} = BC_{in} + AC_{in} + AB$$

$$\text{Sum} = \bar{A}\bar{B}C_{in} + \bar{A}BC_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$$

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

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

$$\text{let } x = A \oplus B$$

$$\text{Sum} = C_{in}(\bar{x}) + \bar{C}_{in}(x)$$

$$= x \oplus C_{in}$$

$$\therefore \text{Sum} = A \oplus B \oplus C_{in}$$

Logic Diagram for full Adder

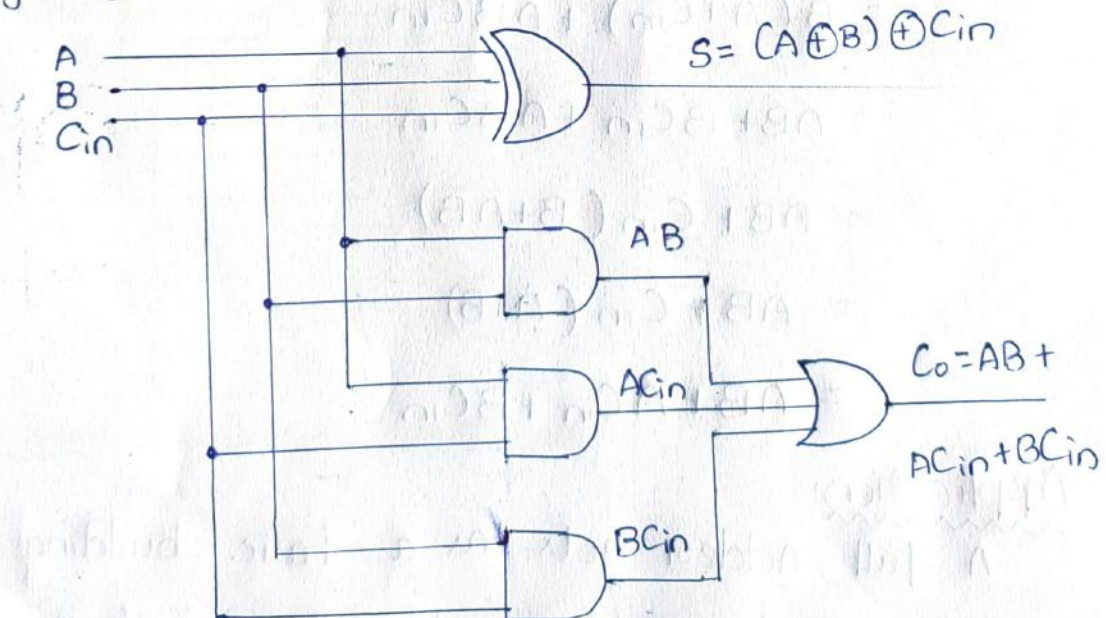


fig:- full Adder Circuit

Full Adder using Half Adders:-

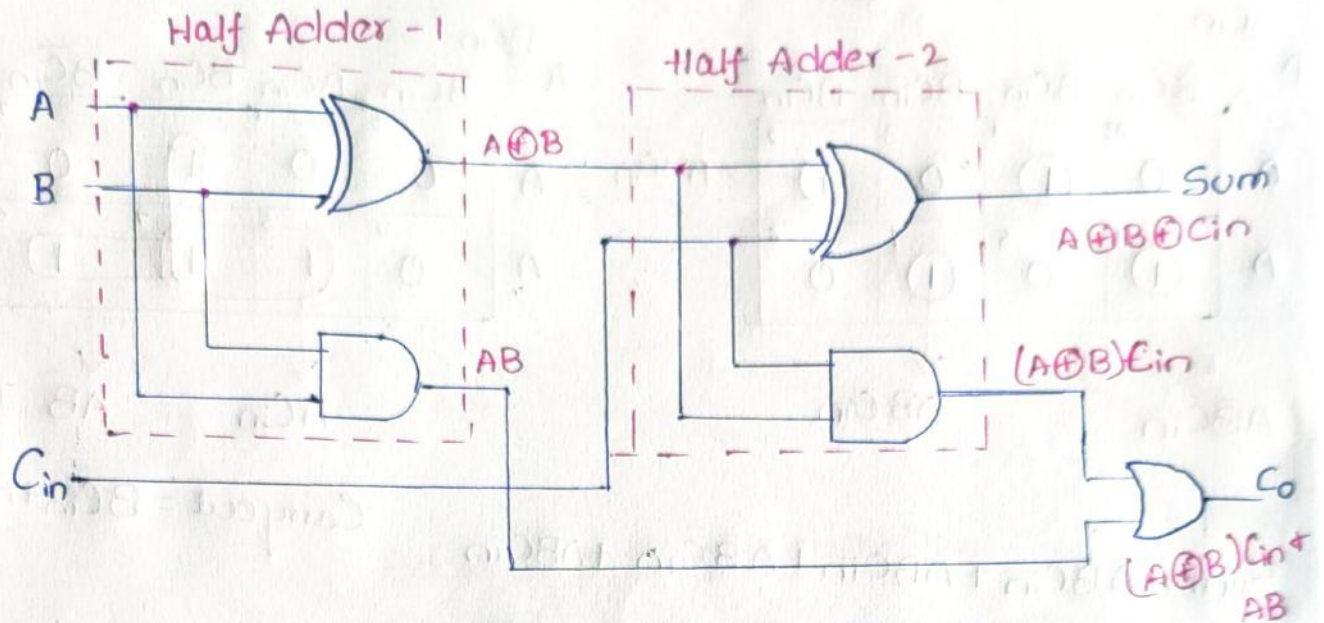


Fig:- Full Adder using two Half Adders

$$\text{Sum} = A \oplus B \oplus C_{in}$$

$$\begin{aligned} C_o &= (A \oplus B)C_{in} + AB \\ &= (\bar{A}B + A\bar{B})C_{in} + AB \\ &= \bar{A}BC_{in} + A\bar{B}C_{in} + AB \\ &= B(\bar{A}C_{in} + A) + A\bar{B}C_{in} \\ &= B(A + C_{in}) + A\bar{B}C_{in} \\ &= AB + BC_{in} + A\bar{B}C_{in} \\ &= AB + C_{in}(B + A\bar{B}) \\ &= AB + C_{in}(A + B) \\ &= AB + AC_{in} + BC_{in} \end{aligned}$$

Application:-

A full adder acts as a basic building block of the 4 bit / 8 bit binary / BCD adder ICs such as 7483.

1. Implement Half Adder using only NAND gates.

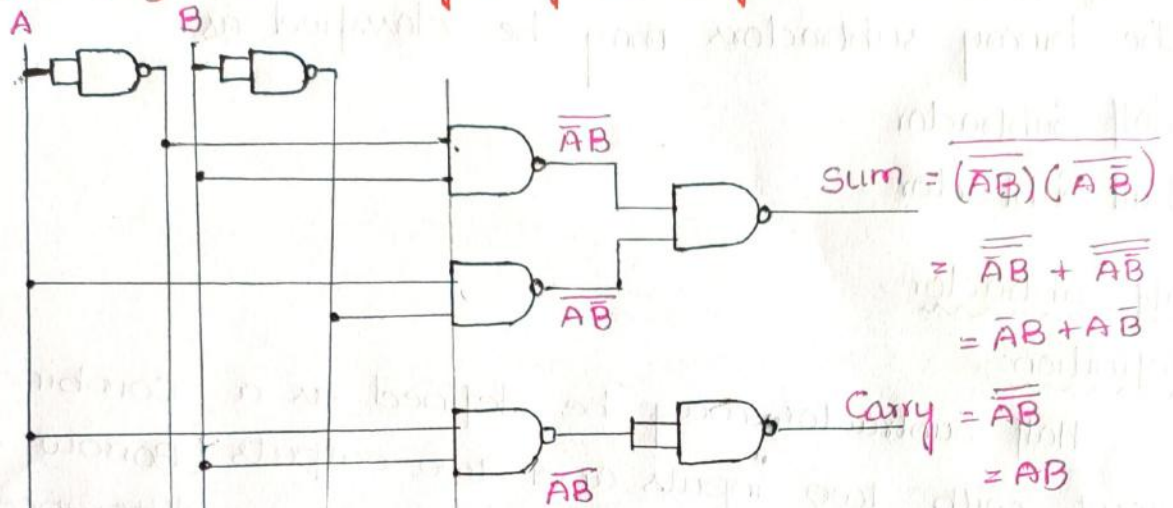


fig 1:- Half Adder Using only NAND gates.

2. Implement Full Adder using only NAND gates

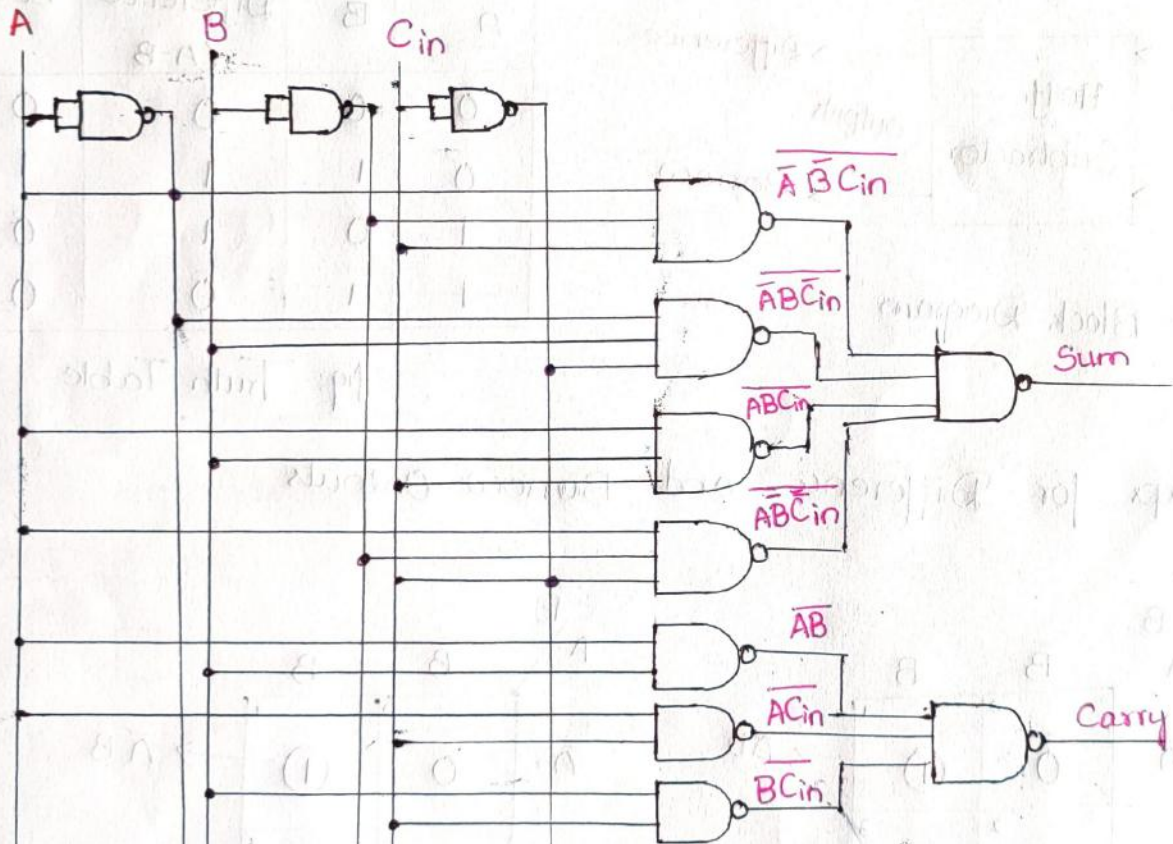


fig 2:- Full Adder Using only NAND gates

Binary Subtractor

The binary subtractors may be classified as:

- Half Subtractor
- Full Subtractor

Half Subtractor:-

Definition:-

Half subtractor may be defined as a combinational circuit with two inputs and two outputs (Borrow and difference). A half subtractor produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed.

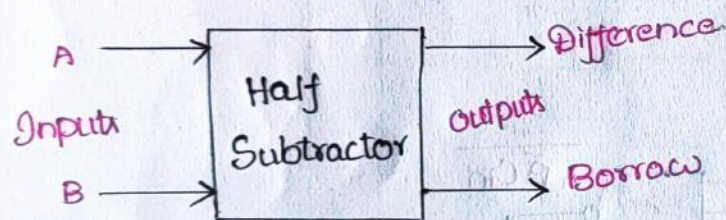
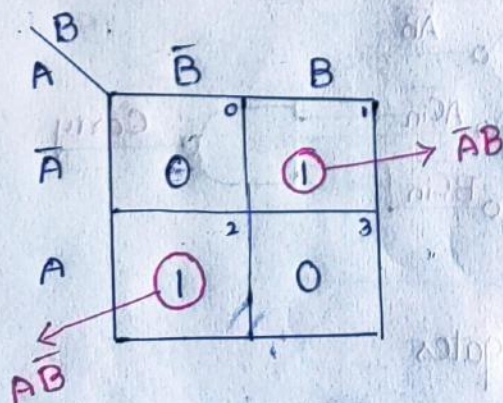


fig:- Block Diagram

Inputs		Outputs	
A	B	Difference (A-B)	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

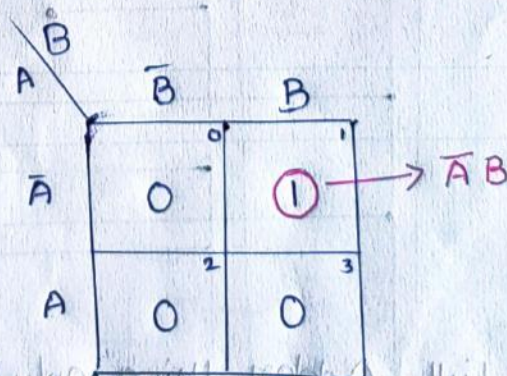
fig:- Truth Table

K-map for Difference and Borrow Outputs



$$\begin{aligned}\text{Difference} &= \bar{A}B + AB \\ &= A \oplus B\end{aligned}$$

fig:- k-map for Difference



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

fig:- k-map for Borrow

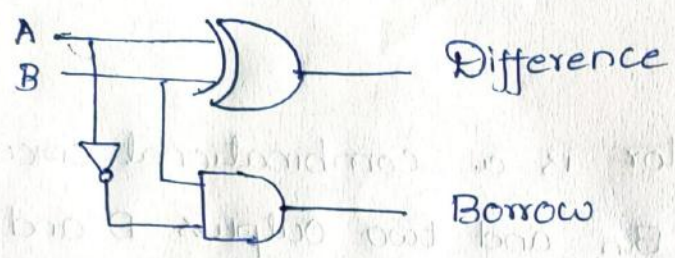


fig:- Half Subtractor Circuit

Drawback

An half subtractor can only perform the subtraction of two binary bits. However, while performing the subtraction, it does not take into account the borrow of the lower significant stage.

Implementation of Half Subtractor using Basic Gates.

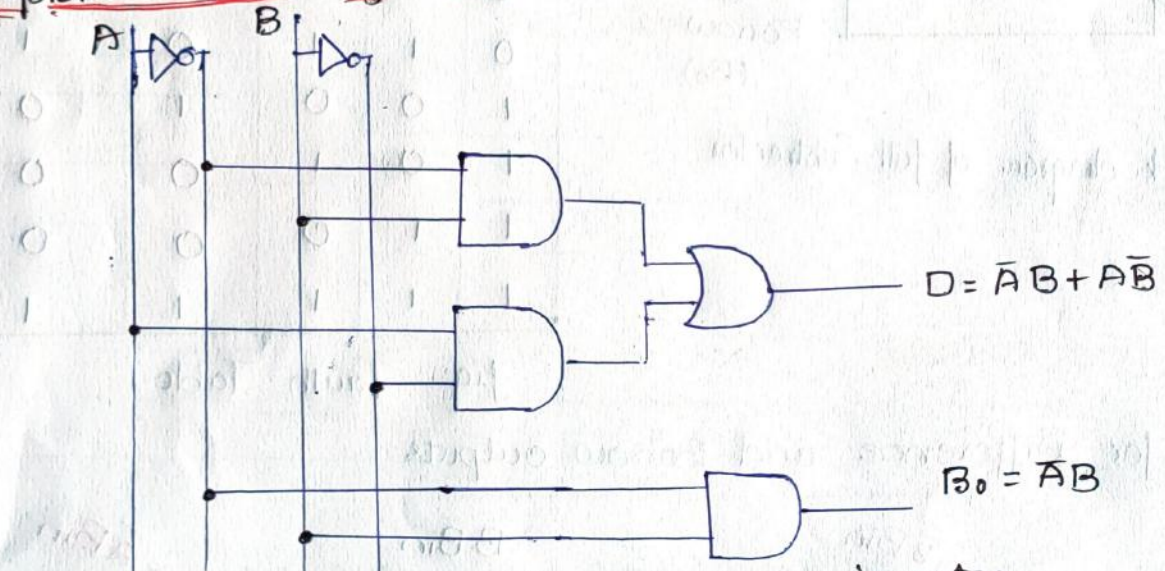


fig:- Half Subtractor using basic gates

Implement Half Subtractor using only NAND gates

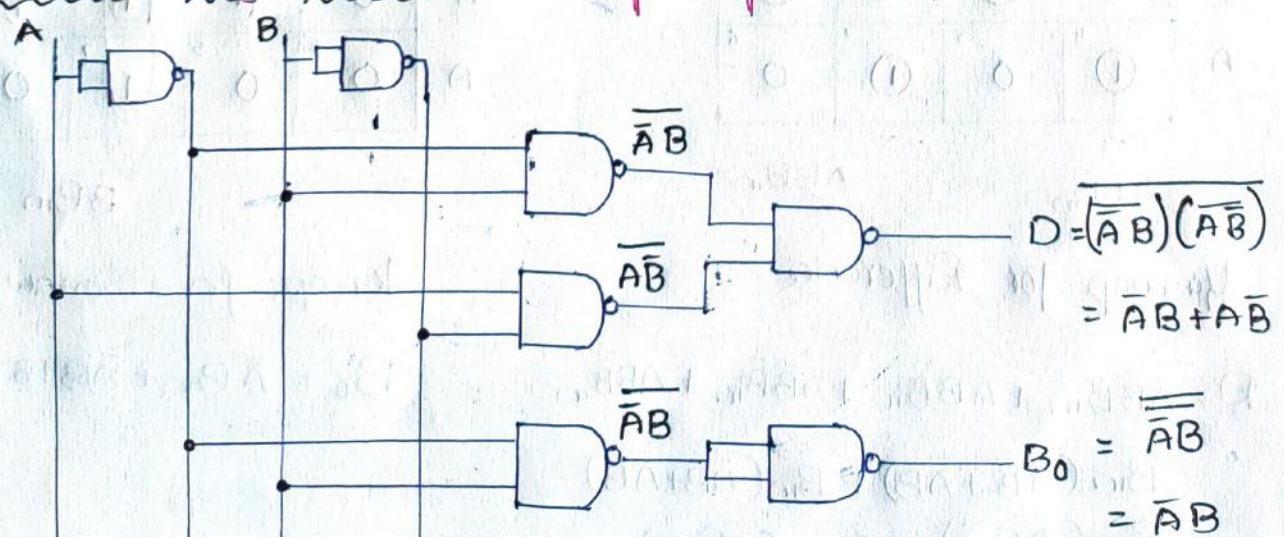


fig:- Half Subtractor using only NAND Gates

Full Subtractor

Definition:-

A full subtractor is a combinational circuit with three inputs A, B and B_{in} and two outputs D and B_o . Here, A is the minuend, B is subtrahend, B_{in} is the borrow produced by the previous stage, D is the difference output and B_o is the Borrow output.

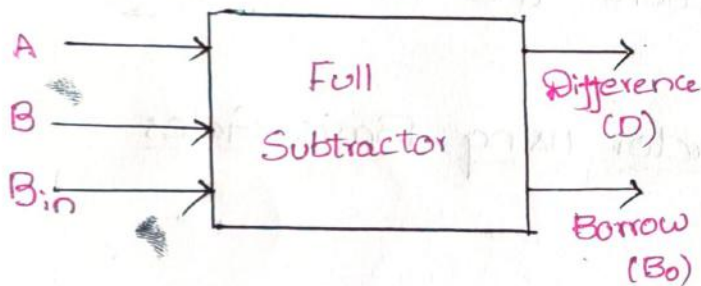
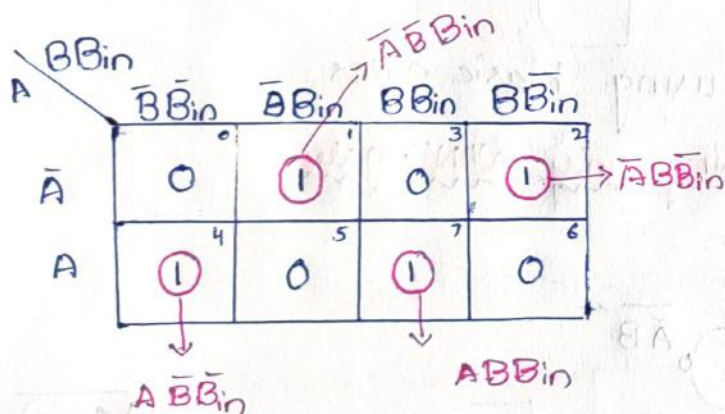


fig:- Block diagram of full Subtractor

Inputs			Outputs	
A	B	B_{in}	D	B_o
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

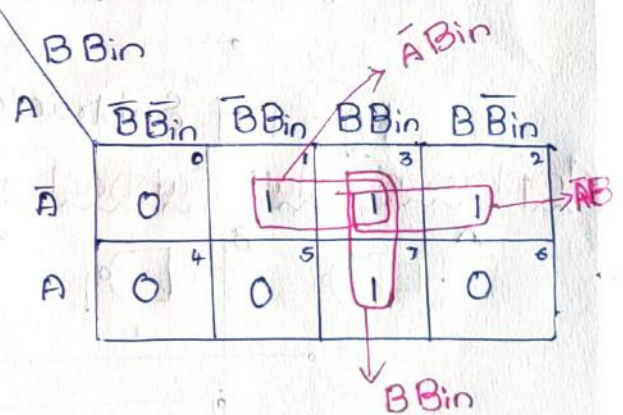
fig:- Truth Table

k-maps for Difference and Borrow outputs



K-map for Difference

$$\begin{aligned}
 D &= \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}B_{in} + AB\bar{B}_{in} \\
 &= B_{in}(\bar{A}\bar{B} + AB) + \bar{B}_{in}(\bar{A}B + A\bar{B}) \\
 &= B_{in}(A \oplus B) + \bar{B}_{in}(A \oplus B) \\
 &= A \oplus B \oplus B_{in}
 \end{aligned}$$



k-map for Borrow

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

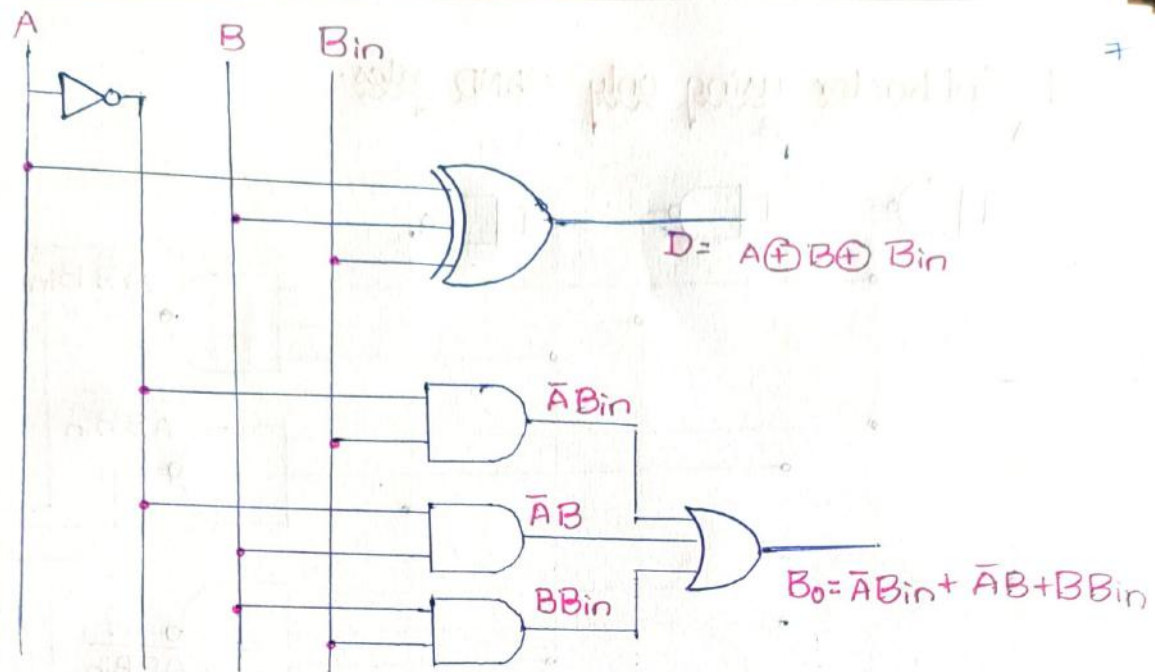


Fig:- Logic Diagram for a full Subtractor

Full Subtractor using Half Subtractors

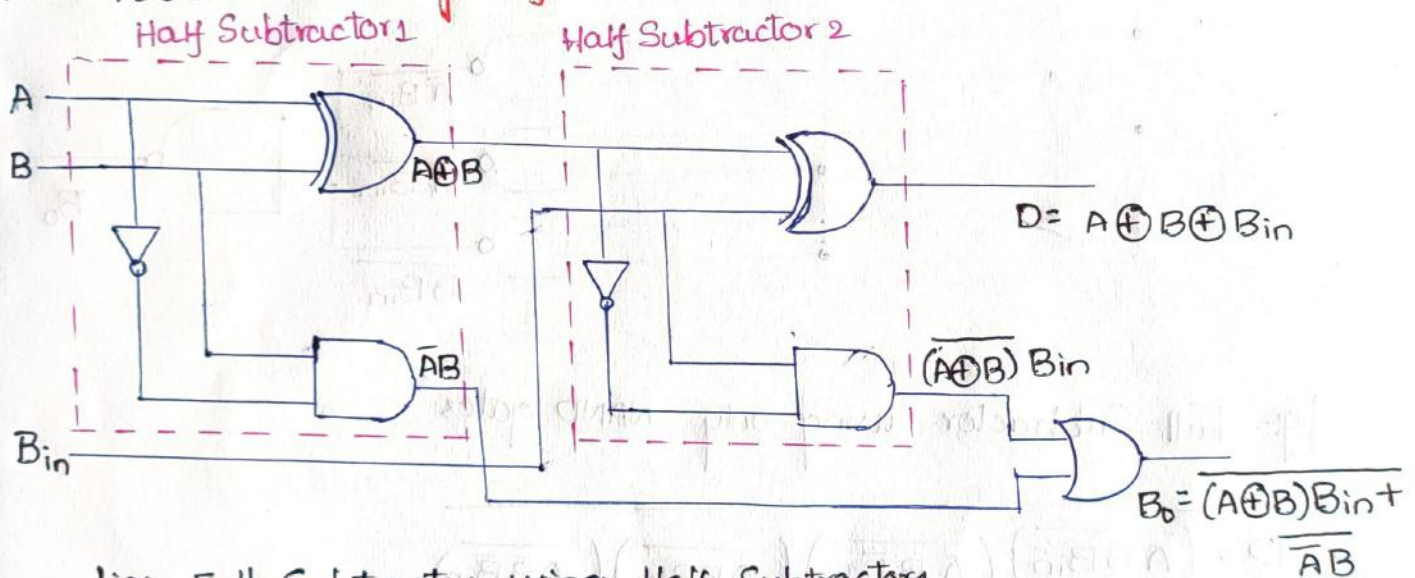


Fig:- Full Subtractor using Half Subtractors

$$\begin{aligned}
 B_0 &= (\overline{A \oplus B})B_{in} + \overline{A}B \\
 &= (\overline{A \oplus B})B_{in} + \overline{A}B \\
 &= \overline{A \oplus B}B_{in} + \overline{A}B + AB_{in} \\
 &= \overline{A \oplus B}B_{in} + B(A_{in} + \overline{A}) \\
 &= \overline{A \oplus B}B_{in} + B(\overline{A} + B_{in}) \\
 &= \overline{A \oplus B}B_{in} + \overline{A}B + BB_{in} \\
 &= \overline{A}(\overline{B}B_{in} + B) + BB_{in} \\
 &= \overline{A}(B + B_{in}) + BB_{in} \\
 &= \overline{A}B + \overline{A}B_{in} + BB_{in}
 \end{aligned}$$

$$[\because \overline{A} + AB = \overline{A} + B]$$

Full Subtractor using only NAND gates.

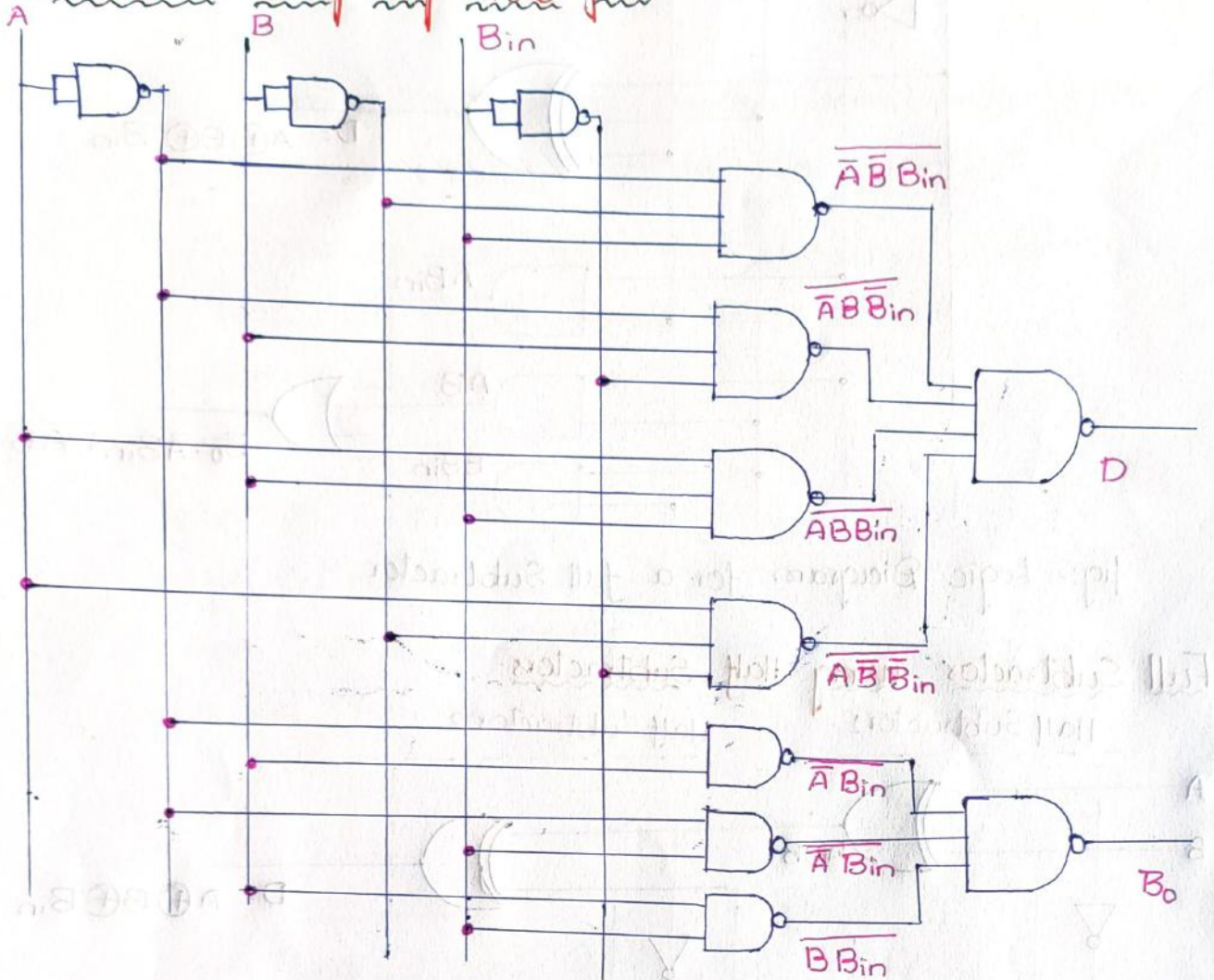


fig:- Full Subtractor using only NAND gates

$$D = (\overline{A} \overline{B} B_{in}) (\overline{A} B \overline{B}_{in}) (\overline{A} B B_{in}) (\overline{A} \overline{B} \overline{B}_{in})$$

$$= \overline{\overline{A} \overline{B} B_{in}} + \overline{\overline{A} B \overline{B}_{in}} + \overline{\overline{A} B B_{in}} + \overline{\overline{A} \overline{B} \overline{B}_{in}}$$

$$= \overline{A} \overline{B} B_{in} + \overline{A} B \overline{B}_{in} + \overline{A} B B_{in} + \overline{A} \overline{B} \overline{B}_{in}$$

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

$$B_0 = (\overline{A} \overline{B}) (\overline{A} B_{in}) (\overline{B} B_{in})$$

$$= \overline{\overline{A} \overline{B}} + \overline{\overline{A} B_{in}} + \overline{\overline{B} B_{in}}$$

$$= \overline{A} \overline{B} + \overline{A} B_{in} + \overline{B} B_{in}$$

MULTIPLEXERS:

A Multiplexers (MUX) is a combinational logic component that has several inputs and only one output.

- Mux directs one of the inputs to its output line by using a control bit word to its select lines.
- MUX contains
 - * 2^n inputs
 - * n Selection inputs
 - * a single output
 - * Selection input determines the input that should be connected to the output.

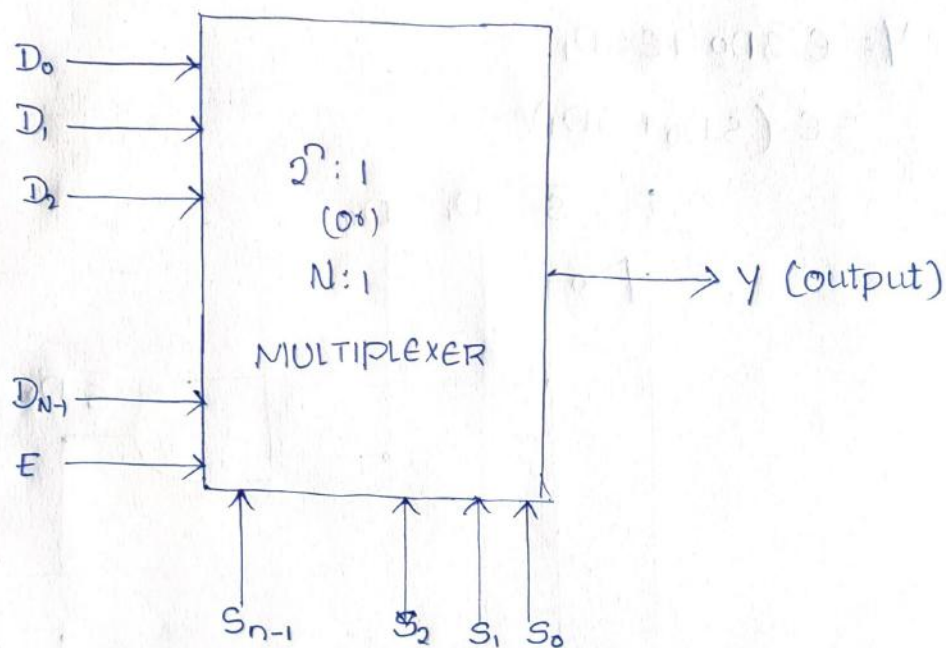
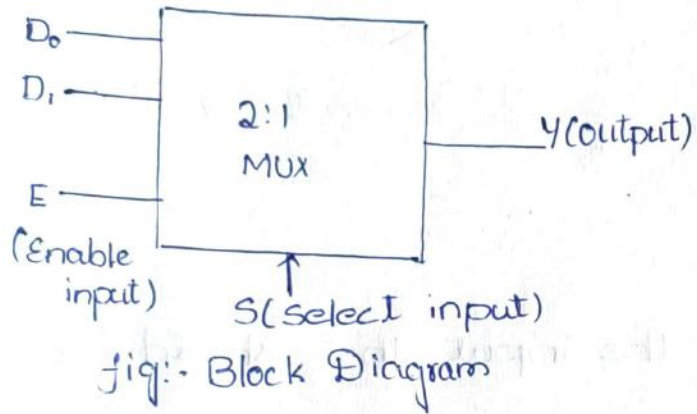


Fig:- Block diagram of N:1 Multiplexer

- The Multiplexer is also called as data Selector.
- The Multiplexer acts like an electronic switch that selects one from different.
- A multiplexer may have an enable input to control the operation of the unit.

2:1 MULTIPLEXER

2:1 Multiplexer has two data inputs D_0 and D_1 , one select input S , an enable input and one output. The block diagram of 2:1 multiplexer is shown in figure.



Enable	Select i/p S	Output Y
0	x	0
1	0	D_0
1	1	D_1

Truth Table

$$Y = E \bar{S} D_0 + E S D_1$$
$$= E (\bar{S} D_0 + S D_1)$$

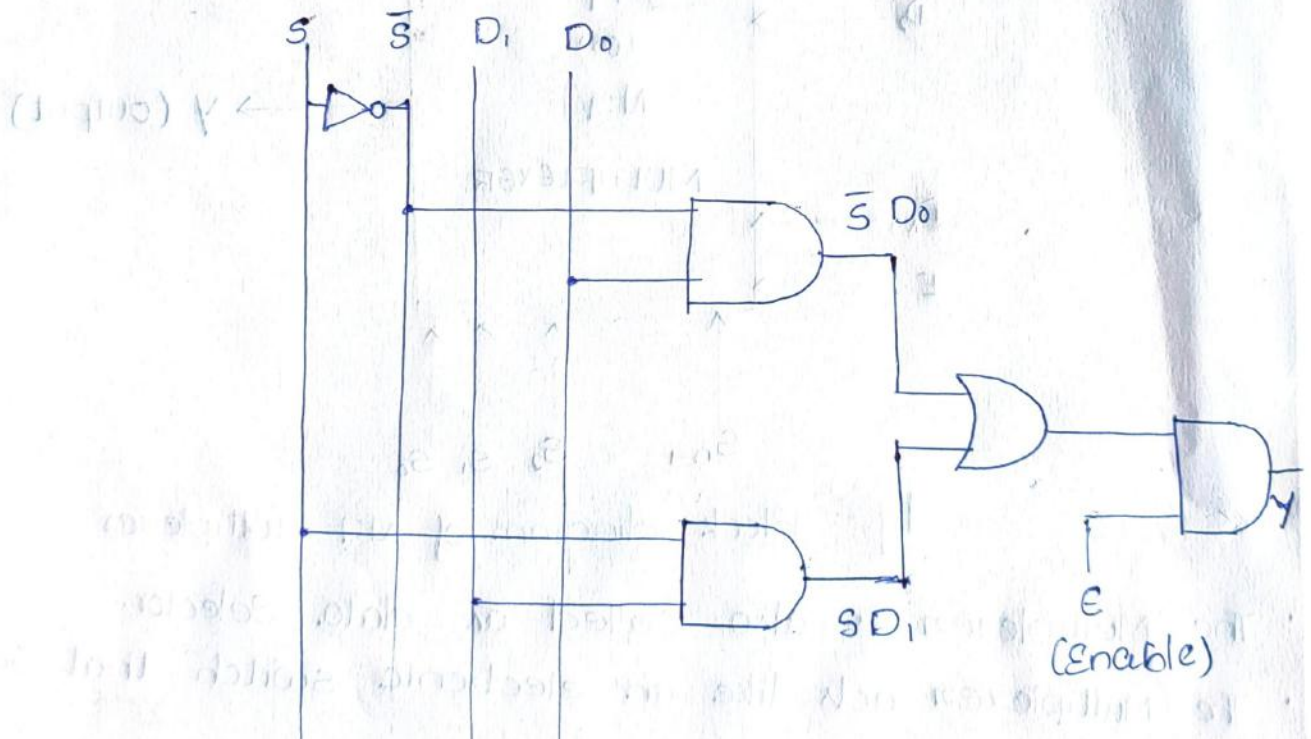
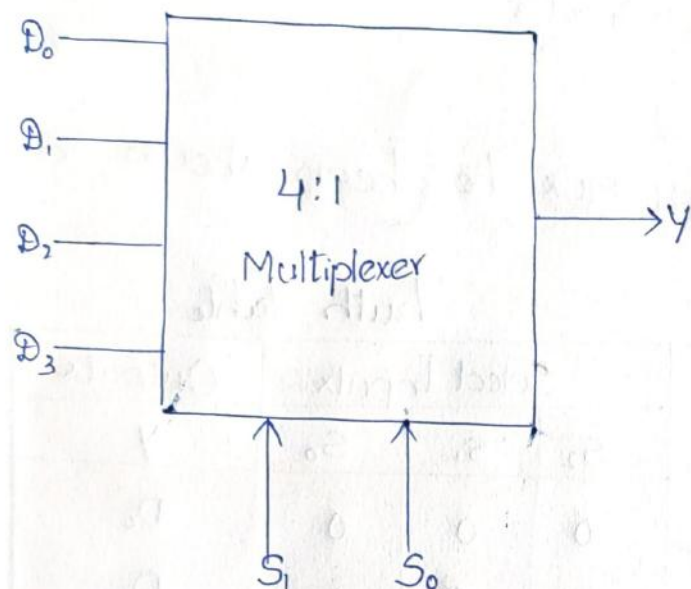


fig:- Realization of 2:1 MUX using gates

4:1 MULTIPLEXER:



Select inputs		Output Y
S_1	S_0	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Truth table

fig:- Block diagram

It has four data inputs, two select lines and one output.

Truth table tells us

$$Y = D_0 \text{ when } S_1 S_0 = 00$$

$$Y = D_1 \text{ when } S_1 S_0 = 01$$

$$Y = D_2 \text{ when } S_1 S_0 = 10$$

$$Y = D_3 \text{ when } S_1 S_0 = 11$$

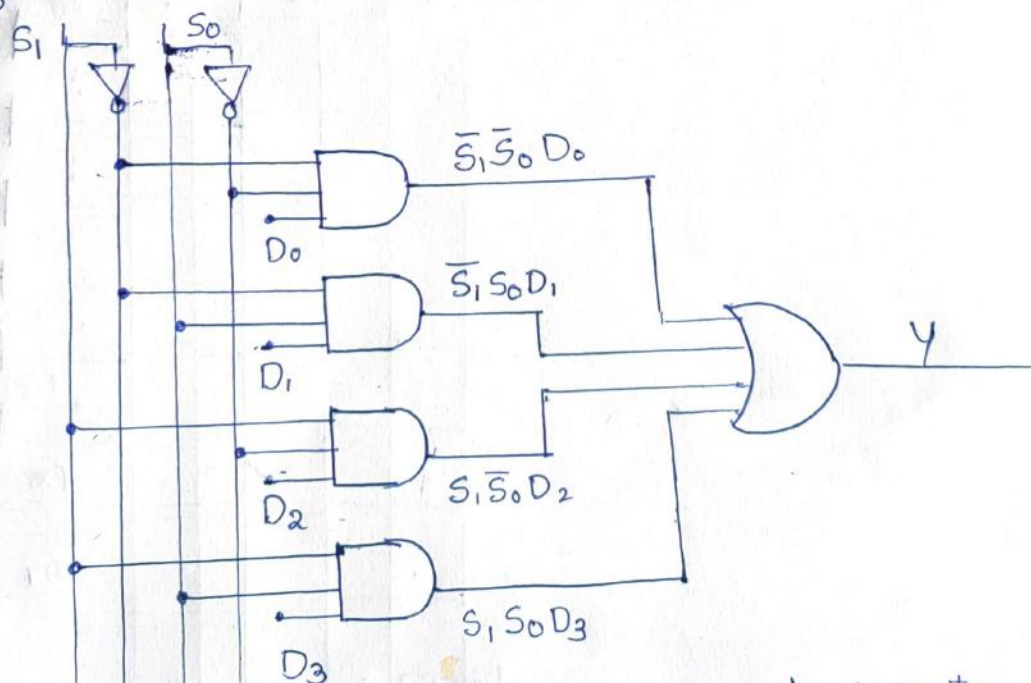


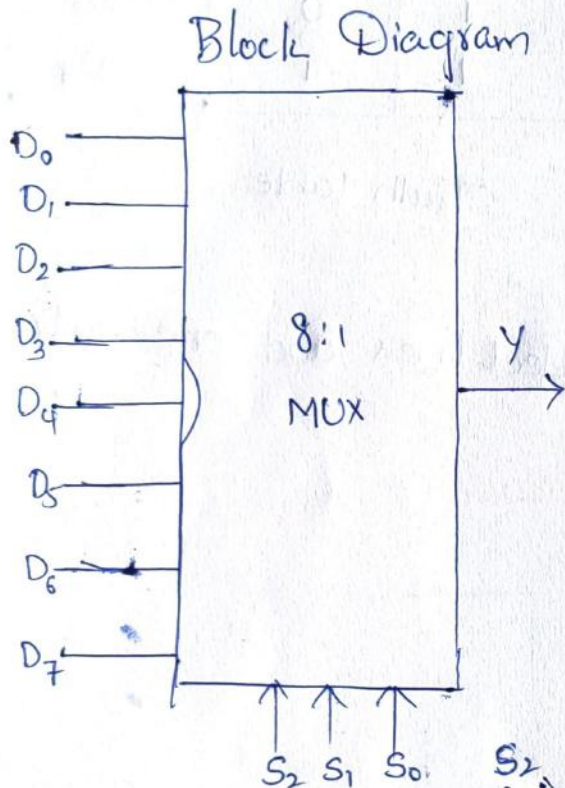
fig:- Realization of 4:1 multiplexer using basic gates

The output will be high when the selected input is 1.
Hence, the logical expression for output in the SOP form will be

$$Y = \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3$$

8:1 MULTIPLEXER:-

The block diagram of an 8:1 MUX has been shown in below figure and its Truth table.



Truth Table

Select Inputs			Outputs
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

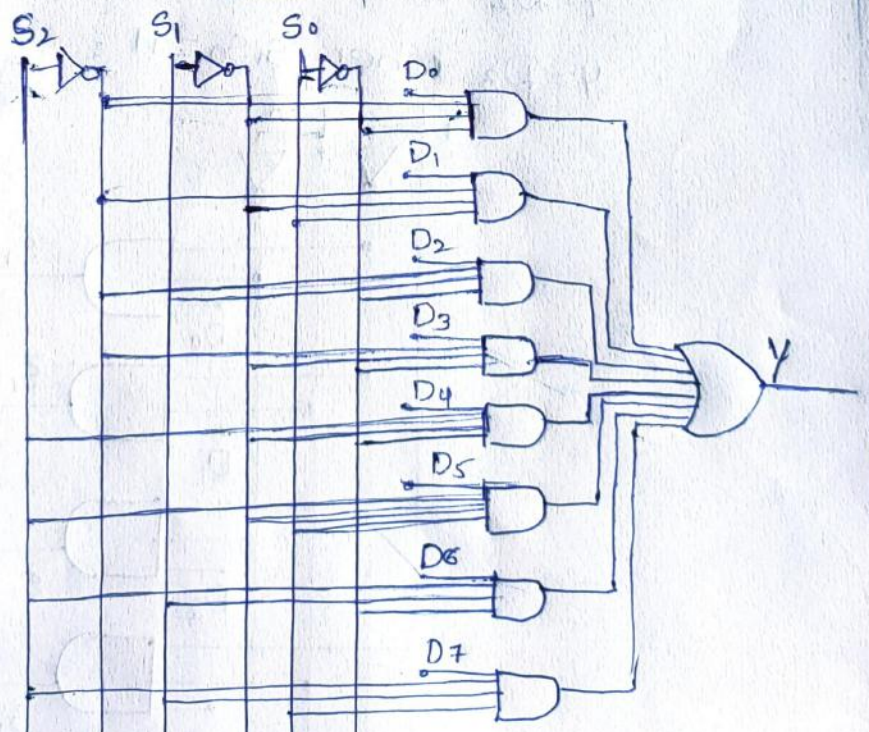


Fig- Realization of 8:1 MUX

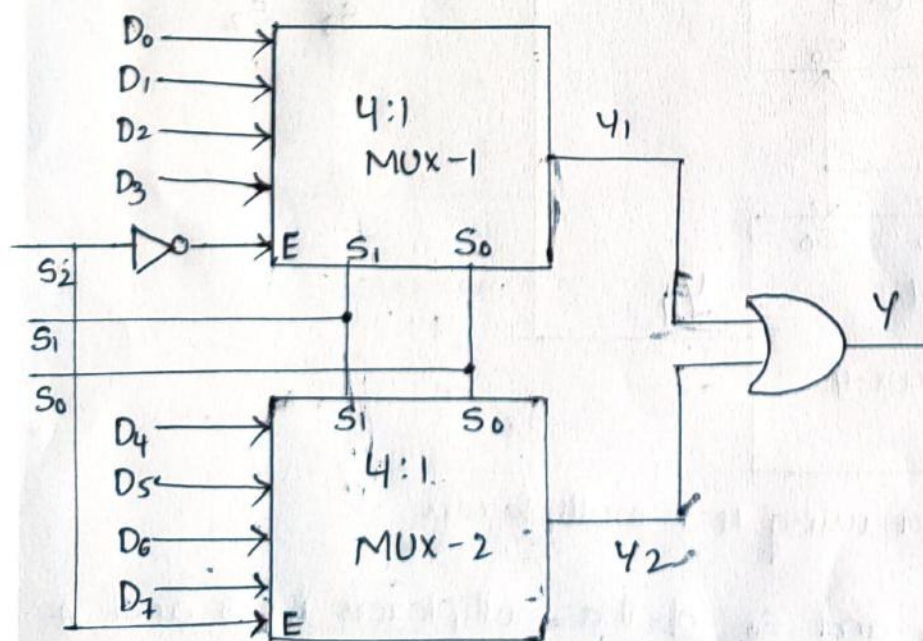
Applications of a Multiplexer:-

- It is used as a data selector to select one out of many data inputs.
- It is used for simplification of logic design.
- In the data acquisition system.
- In designing the combinational circuits.
- In the D/A converters.
- To minimize the number of connections.

Multiplexer Tree (Expanding Multiplexers)

The multiplexers having more number of inputs can be obtained by cascading two or more multiplexers with less number of inputs. This is known as a multiplexer tree.

1. Implement an 8:1 multiplexer using two 4:1 multiplexers.



Select inputs			output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

MUX-1 is Enable
MUX-2 is Enable

fig 1- Multiplexer by cascading two 4:1 Multiplexers

Truth Table

2. Implement a 16:1 multiplexer using 4:1 multiplexers.

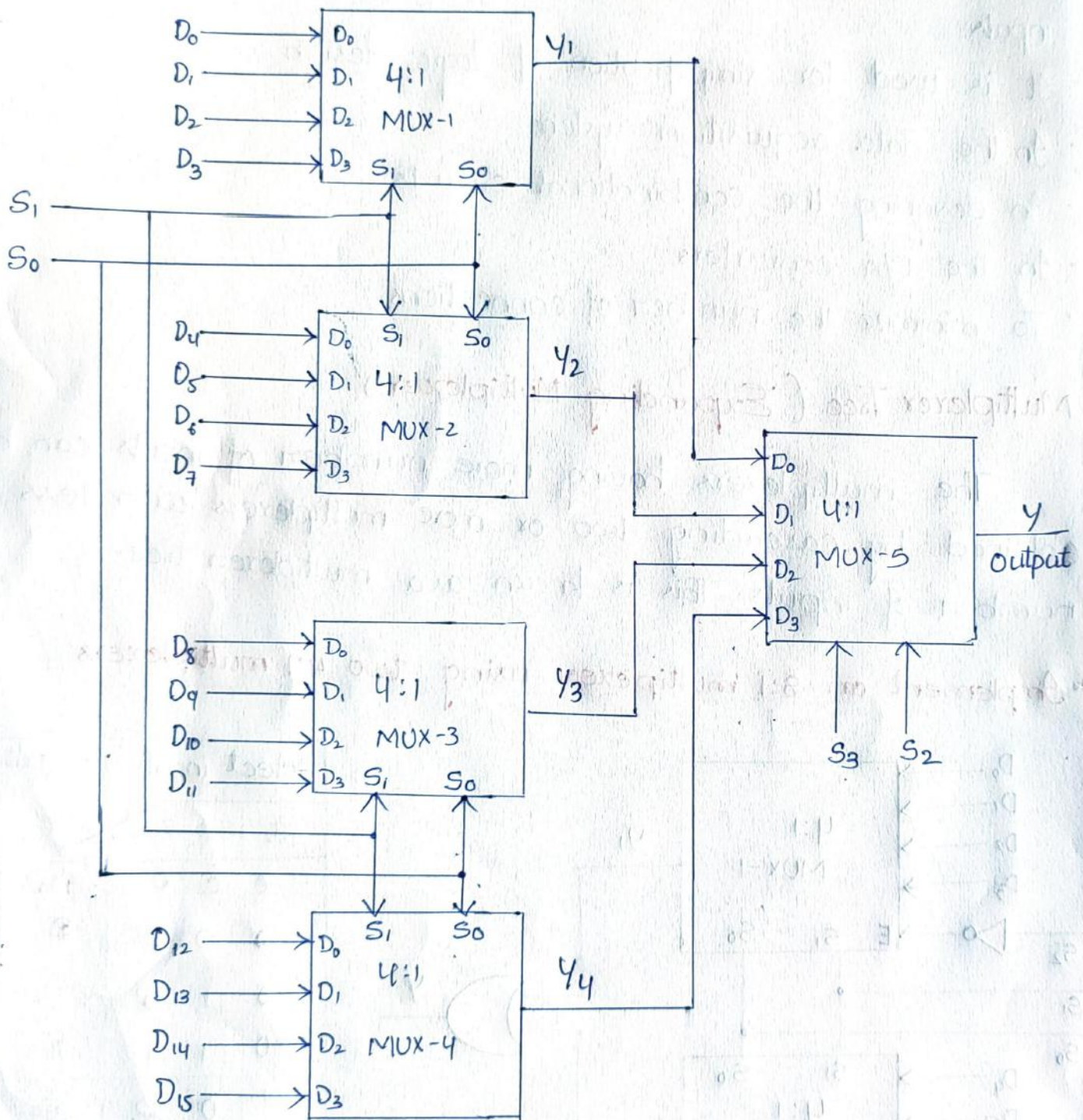


fig-1 16:1 Multiplexer using 4:1 multiplexers

The select input S_1 and S_0 of the multiplexers 1, 2, 3 and 4 are connected together. The select inputs S_3 and S_2 are applied to the data inputs D_0 , D_1 , D_2 and D_3 of MUX-5 as shown in figure.

Select inputs				MUX Outputs				Final output Y
S_3	S_2	S_1	S_0	Y_1	Y_2	Y_3	Y_4	
0	0	0	0	D_0	D_4	D_8	D_{12}	D_0
0	0	0	1	D_1	D_5	D_9	D_{13}	D_1
0	0	1	0	D_2	D_6	D_{10}	D_{14}	D_2
0	0	1	1	D_3	D_7	D_{11}	D_{15}	D_3
$S_3 S_2 = 00$ MUX-5 Selects Y_1								
0	1	0	0	D_0	D_4	D_8	D_{12}	D_4
0	1	0	1	D_1	D_5	D_9	D_{13}	D_5
0	1	1	0	D_2	D_6	D_{10}	D_{14}	D_6
0	1	1	1	D_3	D_7	D_{11}	D_{15}	D_7
$S_3 S_2 = 01$ MUX-5 Selects Y_2								
1	0	0	0	D_0	D_4	D_8	D_{12}	D_8
1	0	0	1	D_1	D_5	D_9	D_{13}	D_9
1	0	1	0	D_2	D_6	D_{10}	D_{14}	D_{10}
1	0	1	1	D_3	D_7	D_{11}	D_{15}	D_{11}
$S_3 S_2 = 10$ MUX-5 Selects Y_3								
1	1	0	0	D_0	D_4	D_8	D_{12}	D_{12}
1	1	0	1	D_1	D_5	D_9	D_{13}	D_{13}
1	1	1	0	D_2	D_6	D_{10}	D_{14}	D_{14}
1	1	1	1	D_3	D_7	D_{11}	D_{15}	D_{15}
$S_3 S_2 = 11$ MUX-5 Selects Y_4								

Table:- Summary of Operation.

Select inputs				MUX Outputs				Final Output Y
S ₃	S ₂	S ₁	S ₀	Y ₁	Y ₂	Y ₃	Y ₄	
0	0	0	0	D ₀	D ₄	D ₈	D ₁₂	D ₀
0	0	0	1	D ₁	D ₅	D ₉	D ₁₃	D ₁
0	0	1	0	D ₂	D ₆	D ₁₀	D ₁₄	D ₂
0	0	1	1	D ₃	D ₇	D ₁₁	D ₁₅	D ₃
$S_3 S_2 = 00$ MUX-5 Selects Y ₁								
0	1	0	0	D ₀	D ₄	D ₈	D ₁₂	D ₄
0	1	0	1	D ₁	D ₅	D ₉	D ₁₃	D ₅
0	1	1	0	D ₂	D ₆	D ₁₀	D ₁₄	D ₆
0	1	1	1	D ₃	D ₇	D ₁₁	D ₁₅	D ₇
$S_3 S_2 = 01$ MUX-5 Selects Y ₂								
1	0	0	0	D ₀	D ₄	D ₈	D ₁₂	D ₈
1	0	0	1	D ₁	D ₅	D ₉	D ₁₃	D ₉
1	0	1	0	D ₂	D ₆	D ₁₀	D ₁₄	D ₁₀
1	0	1	1	D ₃	D ₇	D ₁₁	D ₁₅	D ₁₁
$S_3 S_2 = 10$ MUX-5 Selects Y ₃								
1	1	0	0	D ₀	D ₄	D ₈	D ₁₂	D ₁₂
1	1	0	1	D ₁	D ₅	D ₉	D ₁₃	D ₁₃
1	1	1	0	D ₂	D ₆	D ₁₀	D ₁₄	D ₁₄
1	1	1	1	D ₃	D ₇	D ₁₁	D ₁₅	D ₁₅
$S_3 S_2 = 11$ MUX-5 Selects Y ₄								

Table:- Summary of Operation.

4. Implement a full adder using 8:1 multiplexers

The truth table of a full adder is

Inputs			Outputs	
A	B	C _{in}	Sum(S)	Carry(C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The Sum and Carry outputs can be expressed in the standard SOP form is

$$S = \sum m(1, 2, 4, 7) \text{ and } C = \sum m(3, 5, 6, 7)$$

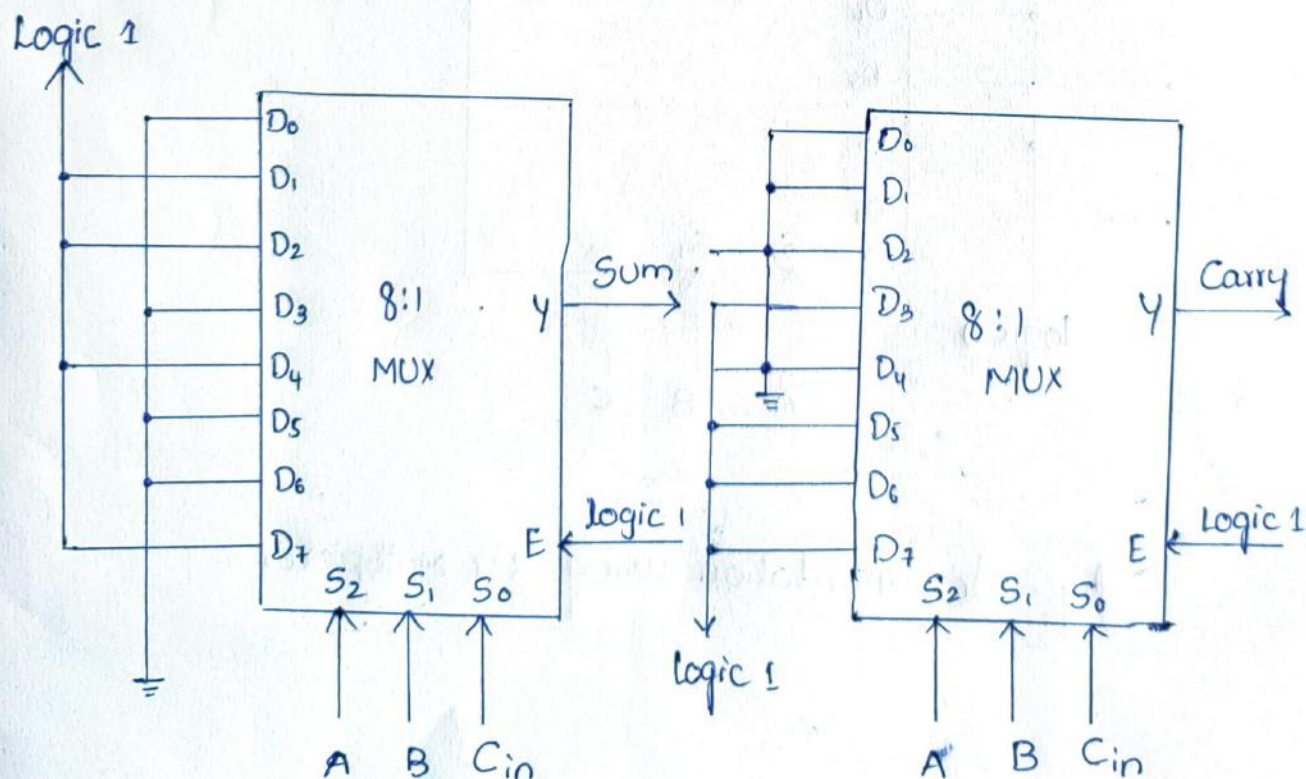


Fig:- Full Adder using 8:1 MUX

Multiplex

5. Implement the following function using an 8:1 multiplexer.

$$f(A, B, C, D) = \sum m(0, 2, 4, 6, 8, 10, 12, 14)$$

Inputs	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇
$\bar{A}$	0	①	2	③	4	⑤	6	⑦
A	8	⑨	10	⑪	12	⑬	14	⑮
Input to MUX	0	1	0	1	0	1	0	1

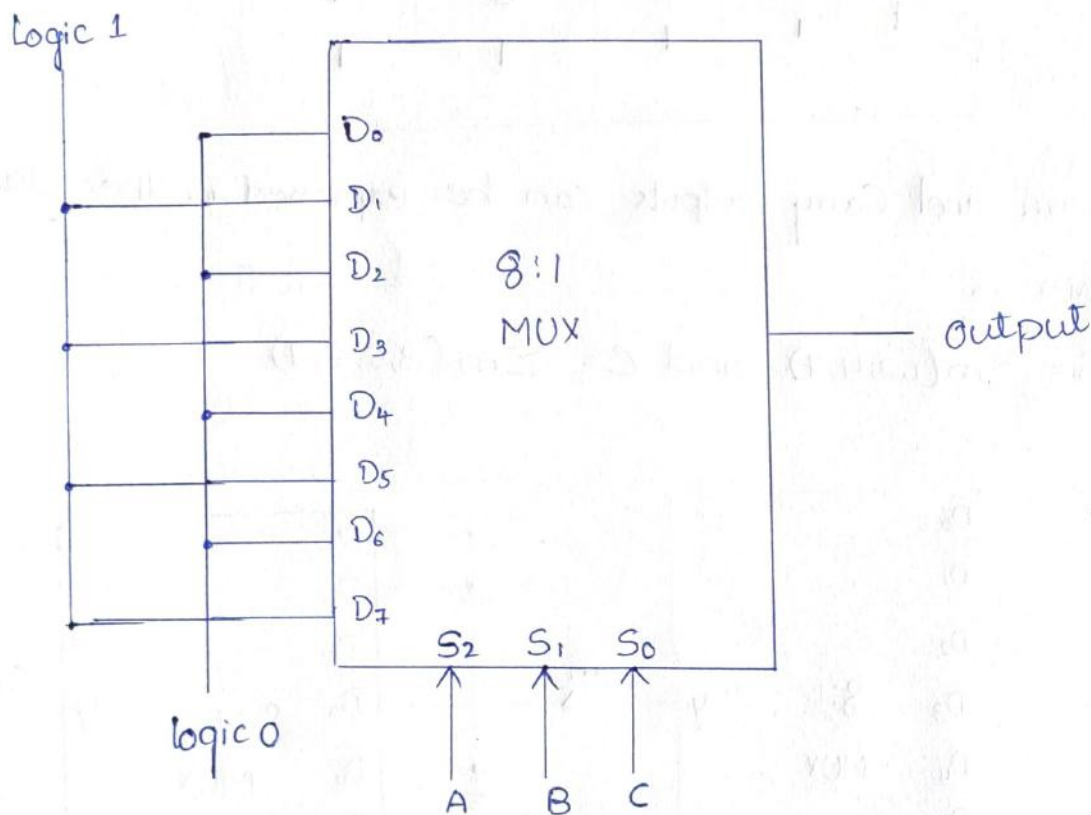


Fig:- Implementation using 8:1 Multiplexer

DEMULTIPLEXER

→ It is also called a "data distributor".

→ A demultiplexer is a device that takes a single input line and routes it one of several digital output lines.

→ A demultiplexer has 2^n outputs and n select lines which are used to select which input line to send to the output.

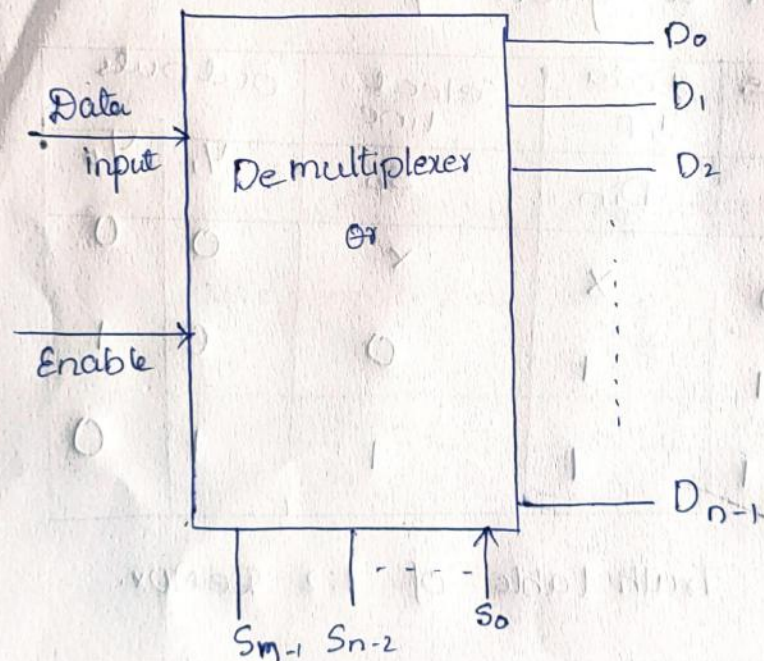


Fig:- 1:n demultiplexer

1:2 Demux

A 1:2 demultiplexer has one data input D_{in} , One Select input S_0 , One Enable (E) input and two outputs Y_0 and Y_1 .

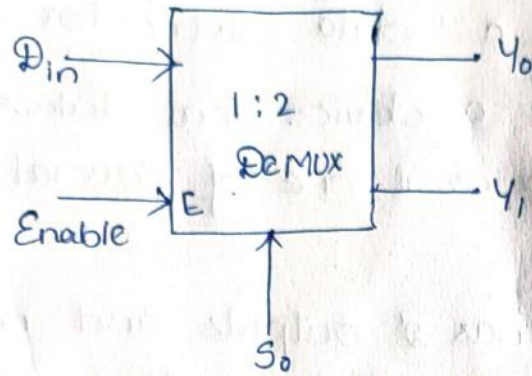


fig:- Block diagram

- If Enable = 0, then all the outputs are 0.
- If Enable input = 1, one of the outputs Y_0 (or) Y_1 is active for a given input.

Enable	Data i/p	Select line	Outputs	
E	Din	S_0	Y_1	Y_0
0	X	X	0	0
1	1	0	0	1
1	1	1	1	0

Truth table of 1:2 Demux.

$$Y_0 = E \text{ Din } \overline{S_0}$$

$$Y_1 = E \text{ Din } S_0$$

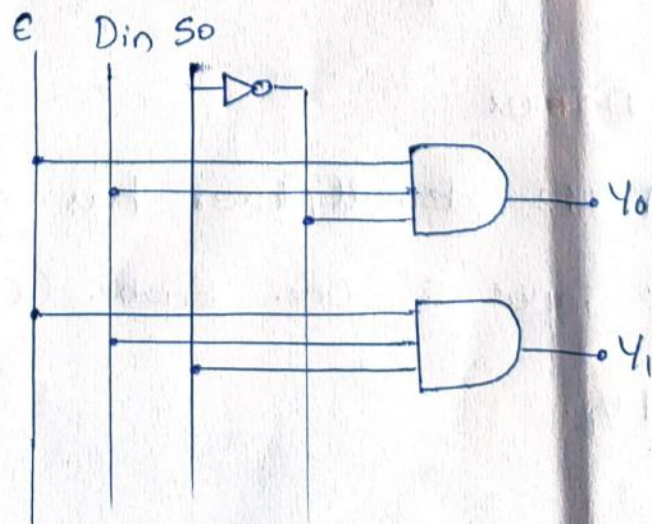


fig:- 1:2 Demux using gates

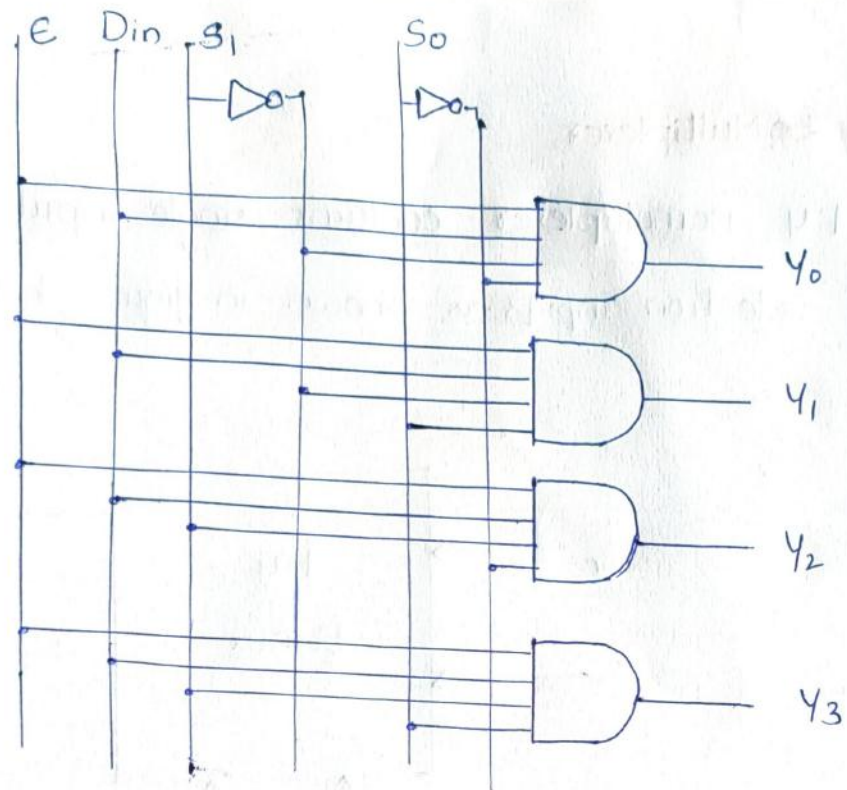
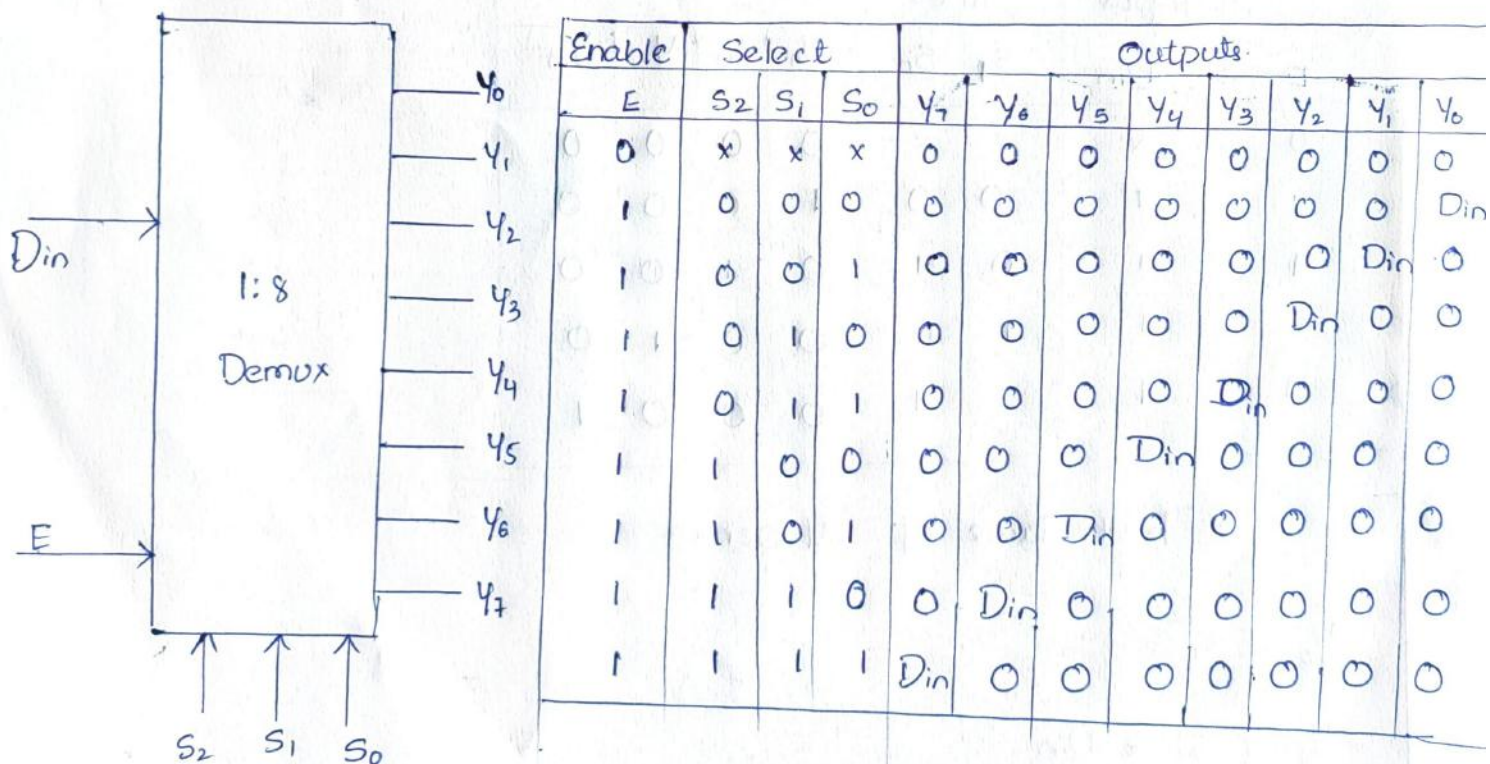


Fig:- 1:4 Demultiplexer.

1:8 Demultiplexer

1:8 Demultiplexer has one data input, eight outputs, three Select inputs and an enable input E as shown in block diagram.



Truth table of 1:8 Demux

Fig:- Block diagram of 1:8 Demux

1:4 Demultiplexer

1:4 Demultiplexer contains single input, four outputs and two selection inputs as shown in figure below.

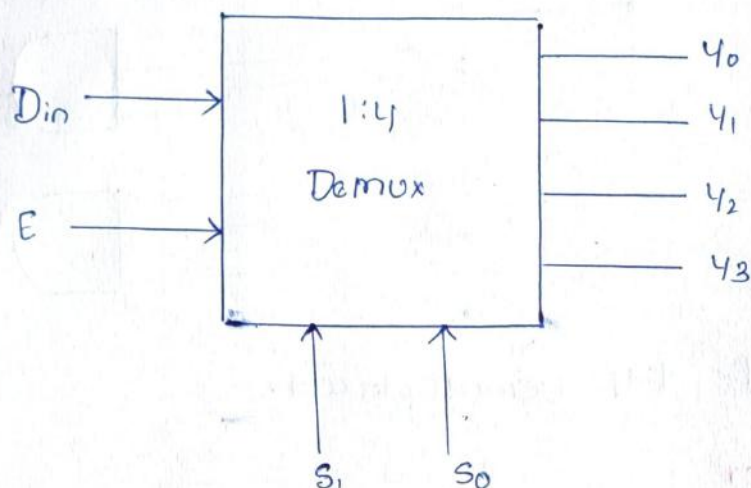


fig:- Block diagram of 1:4 Demux

Enable	Data Input	Select lines		Outputs			
		S_1	S_0	Y_0	Y_1	Y_2	Y_3
E	Din						
0	x	x	x	0	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	0	1	0	0
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

Truth table of 1:4 Demux

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

$$Y_1 = E \bar{S}_1 S_0 \text{ Din}$$

$$Y_2 = E S_1 \bar{S}_0 \text{ Din}$$

$$Y_3 = E S_1 S_0 \text{ Din}$$

$$Y_0 = E \bar{S}_2 \bar{S}_1 \bar{S}_0 \text{Din}$$

$$Y_1 = E \bar{S}_2 \bar{S}_1 S_0 \text{Din}$$

$$Y_2 = E \bar{S}_2 S_1 \bar{S}_0 \text{Din}$$

$$Y_3 = E \bar{S}_2 S_1 S_0 \text{Din}$$

$$Y_4 = E S_2 \bar{S}_1 \bar{S}_0 \text{Din}$$

$$Y_5 = E S_2 \bar{S}_1 S_0 \text{Din}$$

$$Y_6 = E S_2 S_1 \bar{S}_0 \text{Din}$$

$$Y_7 = E S_2 S_1 S_0 \text{Din}$$

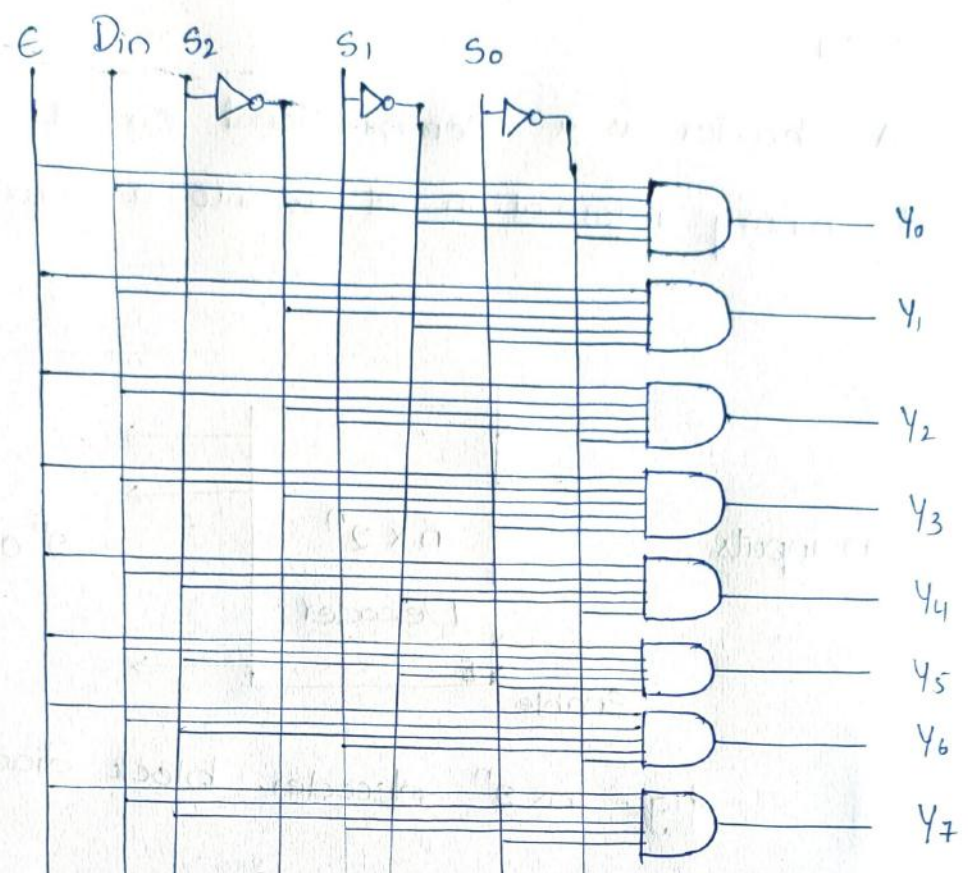
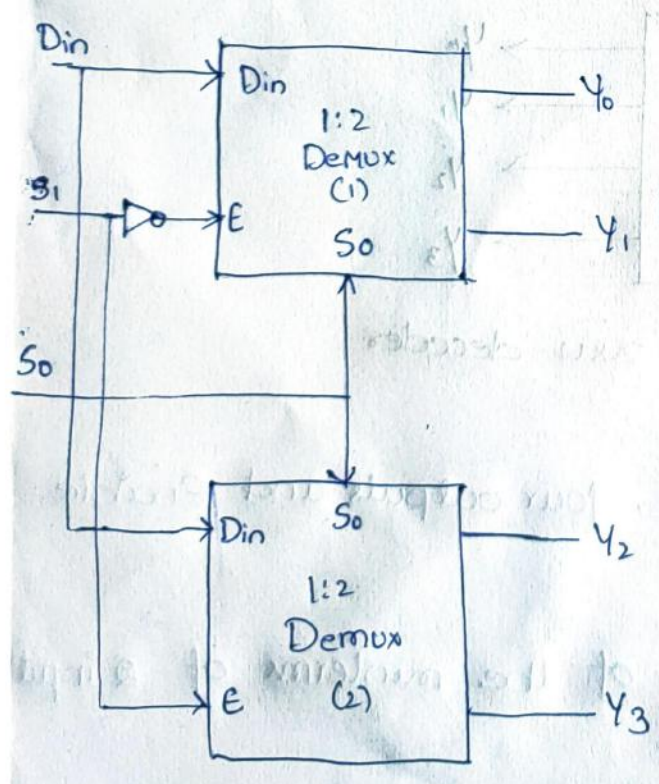


Fig1- Logic Circuit of 1:8 demultiplexer

Obtain a 1:4 line demux using 1:2 demultiplexers.



Select Inputs		Outputs				
S ₁	S ₀	Y ₃	Y ₂	Y ₁	Y ₀	
0	0	0	0	0	D _{in}	Demux (1) Enabled
0	1	0	0	D _{in}	0	
1	0	0	D _{in}	0	0	Demux (2) Enabled
1	1	D _{in}	0	0	0	

Fig:- 1:4 Demux using 1:2 Demux

Decoder:-

A decoder is a combinational circuit. It converts the n -bit binary information at its input into a maximum of 2^n output lines.

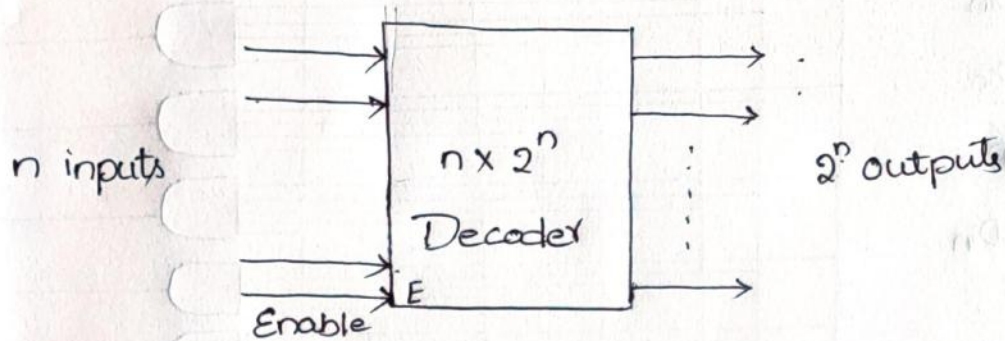


fig:- $n \times 2^n$ decoder block diagram

2x4 Decoder:-

The block diagram of 2x4 decoder has shown in figure.

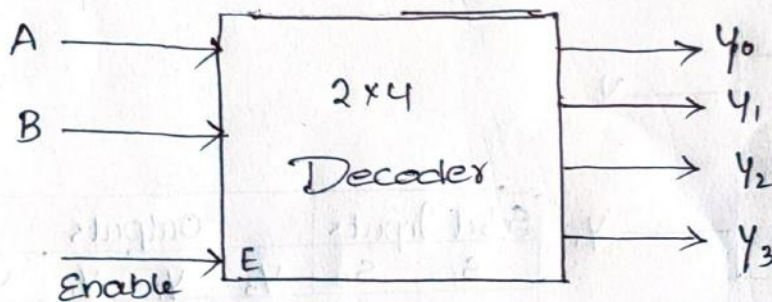


fig:- Block diagram of 2x4 decoder

- It contains 2 inputs, four outputs and Enable input.
- Each output represents one of the minterms of 2 input variables.
- If $\text{Enable} = 0$, then all the outputs are zero.
- If $\text{Enable} = 1$, one of the outputs Y_0 to Y_3 is active.

for a given input.

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

output Y_0 is active, $Y_0=1$ when inputs $A=0, B=0$

Y_1 is Active, $Y_1=1$ when inputs $A=0, B=1$

Y_2 is Active, $Y_2=1$ when inputs $A=1, B=0$

Y_3 is Active, $Y_3=1$ when inputs $A=1, B=1$

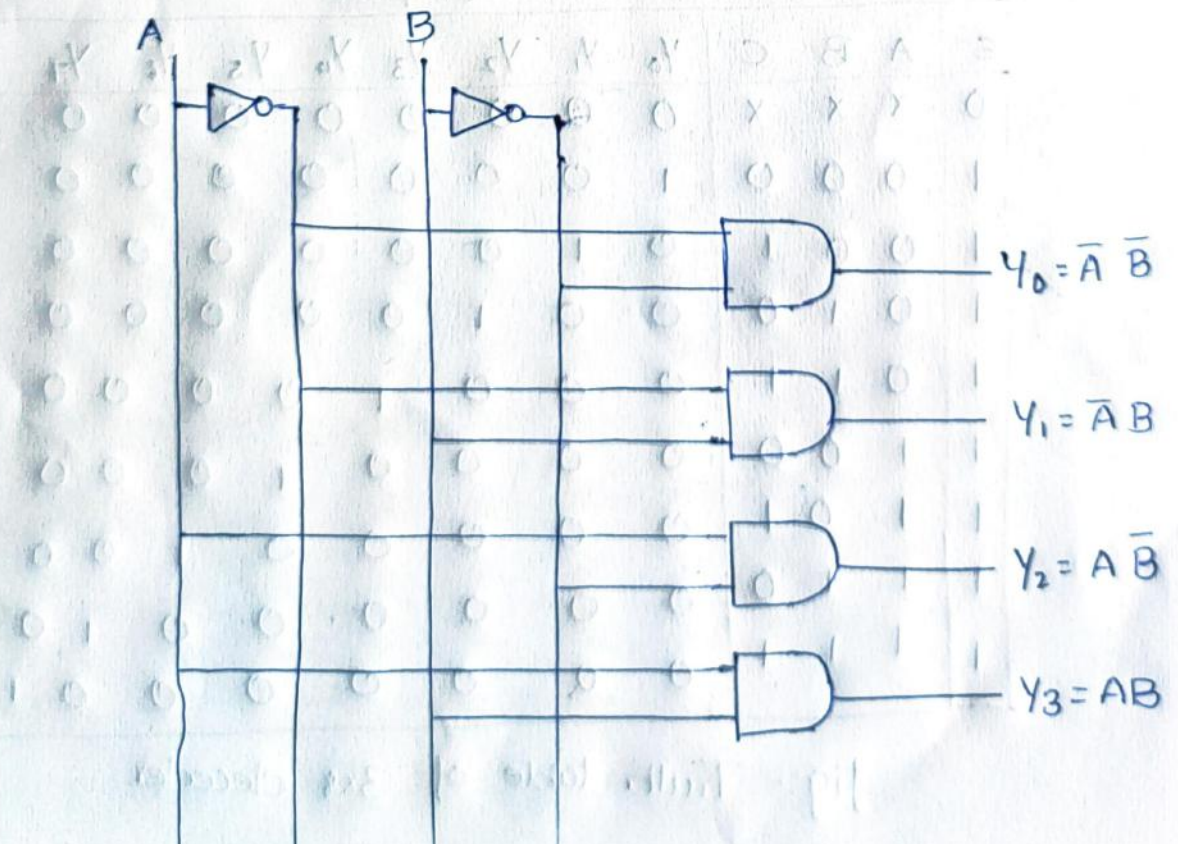
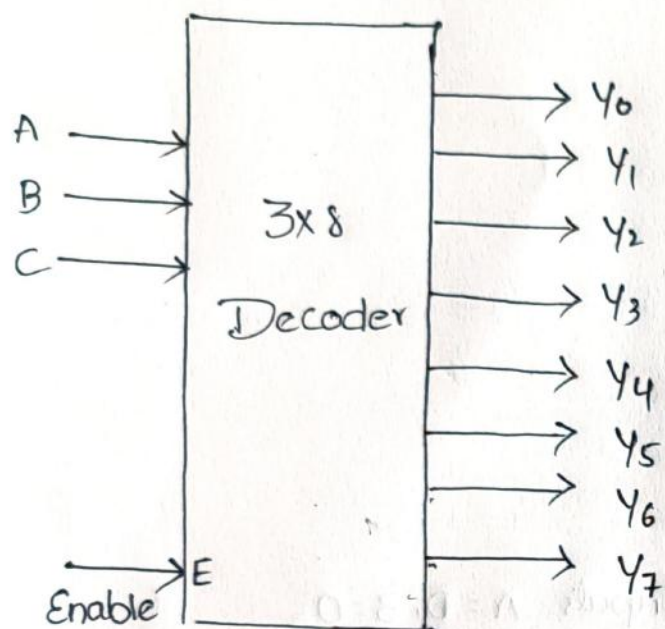


Fig 1- Logic Diagram of 2x4 decoder

3x8 Decoder:-

The block diagram of 3x8 decoder is shown in figure.



Input				Output			
A	B	C	D	Y0	Y1	Y2	Y3
0	0	0	0	1	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	0	1	0
0	1	1	0	0	0	0	1
1	0	0	0	0	1	0	0
1	0	1	0	0	0	1	0
1	1	0	0	0	0	0	1
1	1	1	0	0	0	0	0

fig:- Block diagram of 3x8 decoder.

Inputs				Outputs							
E	A	B	C	Y ₀	Y ₁	Y ₂	Y ₃	Y ₄	Y ₅	Y ₆	Y ₇
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

fig:- Truth table of 3x8 decoder

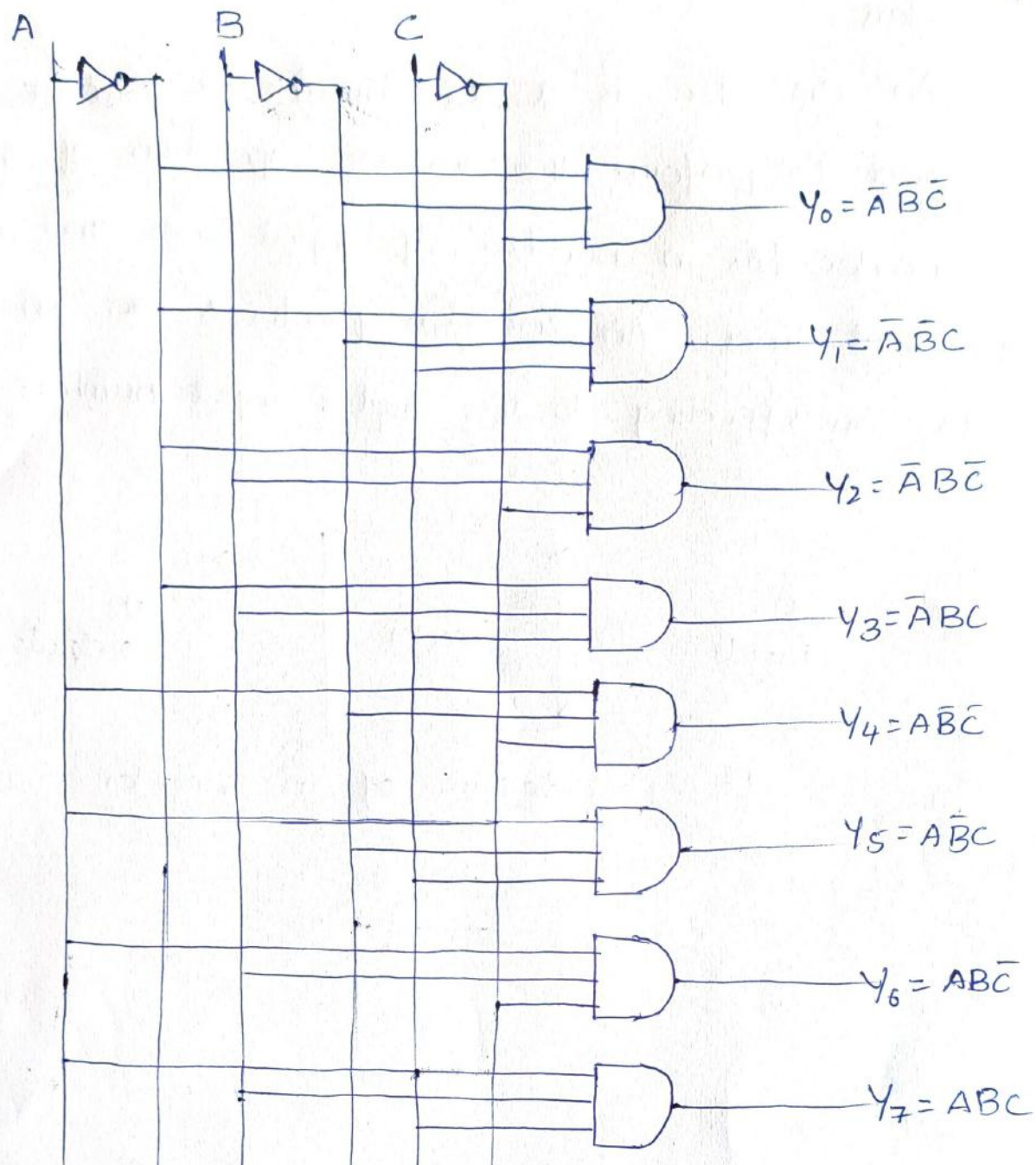


Fig:- Logic Diagram of 3x8 decoder

Applications of Decoders:

- They can be used as code converters

Ex:- BCD to 7-Segment display decoder

Binary to octal decoder

- They are used for data distribution
- Used as building blocks in implementing switching function.

Sequential Circuits :-

The output of a sequential circuit depends upon the present time inputs, the previous output and the sequence in which the inputs are applied.

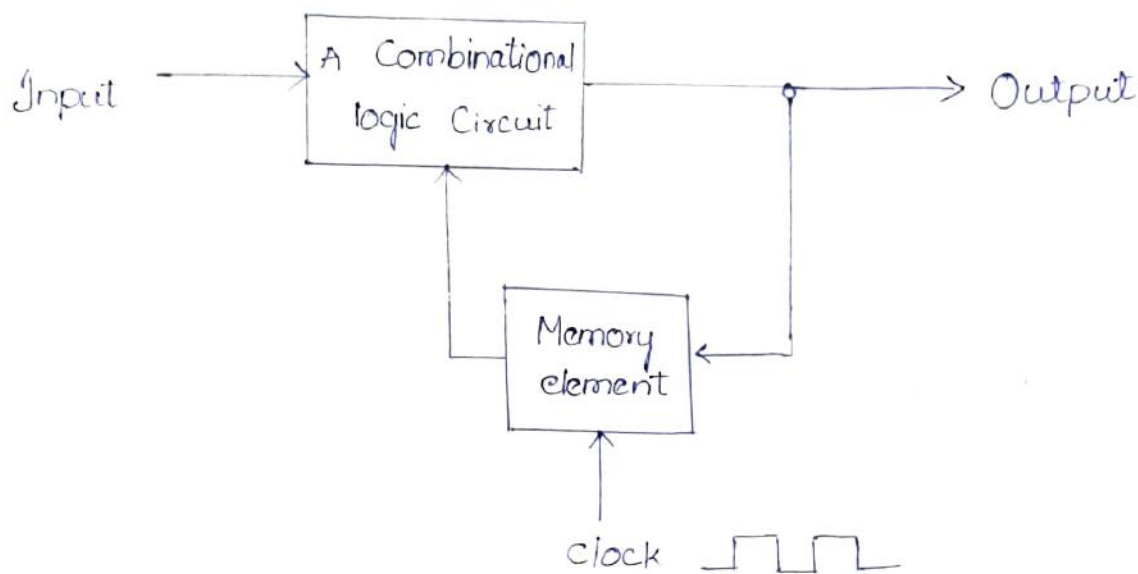


fig:- Block diagram of a sequential circuit.

① sequential circuit requires a memory element.

Present State of a Sequential Circuit:-

The data stored by the memory element at any given instant of time is known as the present state of the sequential circuit.

Next State:-

The Combinational circuit operates on the external inputs and the present state to produce new outputs. Some of those new outputs are stored in memory element and known as the next state of the sequential circuit.

Clock Signal:-

The clock signal is a timing signal. Clock is a rectangular signal as shown in figure with a duty cycle equal to 50%. The clock signal repeats itself after every T seconds. Therefore, the clock frequency

$$F = 1/T.$$

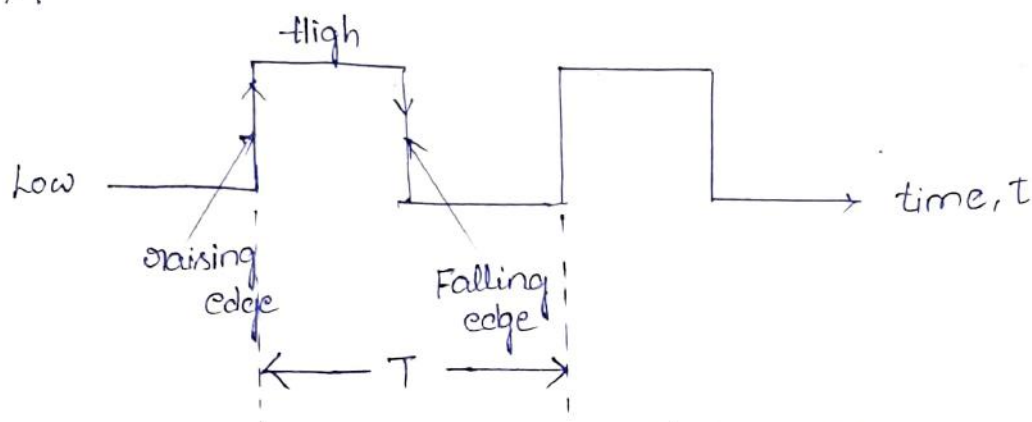


Fig:- Illustration of a Clock signal.

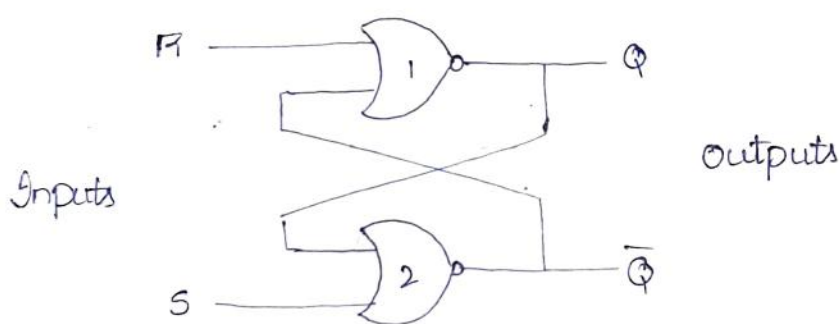
Comparison between combinational and sequential circuits:-

Combinational circuit	Sequential Circuit
* In combinational circuits, the output variables at any instant of time are dependent only on the present input variables. * Memory unit is not required in combinational circuit. * These circuits are faster because the delay between the i/p and o/p is due to propagation delay of gates only. * Easy to design.	* In sequential circuits, the output variables at any instant of time are dependent not only on the present input variables, but also on the present state. * Memory unit is required to store the past history of the input variables. * Sequential circuits are slower than combinational circuits. * Comparatively hard to design.

SR LATCH:-

The latch has two outputs Q and Q' . When the circuit is switched on the latch may enter into any state. If $Q=1$, then $Q'=0$, which is called SET state. If $Q=0$, then $Q'=1$, which is called ~~same~~ RESET state. Whether the latch is in SET state or RESET state, it will continue to remain in the same state, as long as the power is not switched off. But the latch is not an useful circuit, since there is no way of entering the desired input.

NOR Latch:-



NOR Truth Table

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

fig:- SR Latch using NOR gate

Case 1:- $S=1, R=0$,

output of NOR gate 2 is 0 i.e., $\bar{Q}=0$,

hence the inputs of NOR gate 1 are 0 $\Rightarrow Q=1$.

$\therefore$ When $S=1, R=0 \Rightarrow Q=1, \bar{Q}=0$.

Case 2:- $S=0, R=1$,

Output of NOR gate 1 is 0 because one of the input

is 1 in NOR truth table then the output is zero.

$\therefore \bar{Q}=0$

hence the both inputs of NOR gate 1 are 0 so the output $Q = 1$.

$$\therefore S = 0, R = 1 \Rightarrow Q = 1, \bar{Q} = 0$$

Case 3:- $S = 0, R = 0$

We know that $\bar{Q} = \overline{S + Q}$ and $Q = \overline{R + \bar{Q}}$

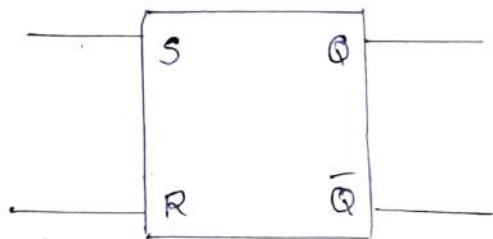
$$\therefore \bar{Q} = \overline{0 + Q} \quad \text{and} \quad Q = \overline{0 + \bar{Q}}$$

$$\bar{Q} = \overline{0 \cdot Q} \quad Q = \overline{0 \cdot \bar{Q}}$$

$$= \bar{Q} \quad = Q$$

$\therefore S = R = 0$, then Q and $\bar{Q}$ do not change their states.

Case 4:- $S = 1, R = 1$, then the outputs Q and $\bar{Q}$ both are forced to 0. Actually, this is an indeterminate state which must be avoided.

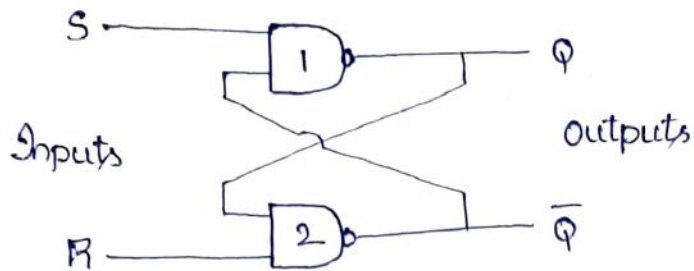


Symbol

S	R	Q	$\bar{Q}$	State
0	0	Q^+	$\bar{Q}^+$	Storage state
0	1	0	1	Reset
1	0	1	0	Set
1	1	0	0	Invalid

Truth table

NAND LATCH



NAND Truth table

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

Fig:- SR Latch using NAND gates

Case i):- $S=0, R=0$; Then the outputs Q and $\bar{Q}$ both are forced to 1. This is an undeterminate state which must be avoided.

Case ii):- $S=0, R=1$

Output of NAND gate 1 is 1 i.e., $Q=1$

hence the inputs of NAND gate 2 are 1 $\therefore \bar{Q}=0$.

Case iii):- $S=1, R=0$

Output of NAND gate 2 is 1 i.e., $\bar{Q}=1$

hence the inputs of NAND gate 1 are 1 $\therefore Q=0$.

Case iv):- $S=1, R=1$

We know that $Q = \overline{S \cdot \bar{Q}}$ and $\bar{Q} = \overline{R \cdot Q}$

$$Q = \overline{\bar{Q} \cdot S}$$

$$= \bar{S} + Q$$

$$= 0 + Q$$

$$= Q$$

$$\bar{Q} = \overline{R \cdot Q}$$

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

$$= 0 + \bar{Q}$$

$$= \bar{Q}$$

$\therefore S=R=1$, then Q and $\bar{Q}$ do not change their states.

S	R	Q	$\bar{Q}$	State
0	0	1	1	Invalid
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q^+	$\bar{Q}^+$	Memory

S-R Flip-flop

The basic flip-flop is a one bit memory cell that gives the fundamental idea of memory device. The logic diagram and the block diagram of S-R flip-flop with clocked input.

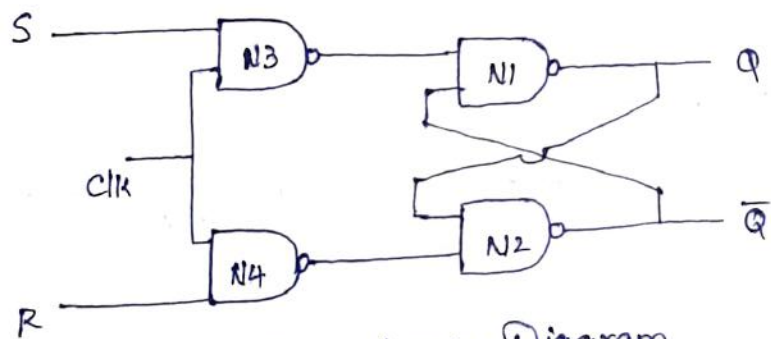


fig:- Logic Diagram

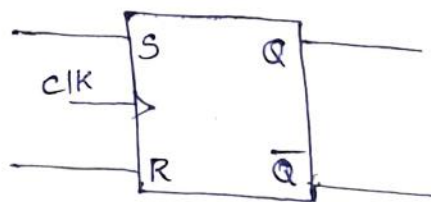


fig:- Block diagram

The flip-flop can be made to respond only during the occurrence of clock pulse by adding two NAND gates to the input latch. So synchronization is achieved. i.e., flipflops are allowed to change their states only at particular instant of time. The clock pulses are generated by a clock pulse generator. The flip-flops are affected only with the arrival of

clock pulse.

- When $clk=0$ the output of $N3$ and $N4$ are 1 regardless of the value of S and R . This is given as input to $N1$ and $N2$. This makes the previous value of Q and $\bar{Q}$ unchanged.
- When $clk=1$ the information at S and R inputs are allowed to reach the latch and change of state in flipflop takes place.
- $clk=1, S=0, R=1$ gives the RESET state i.e., $Q=0, \bar{Q}=1$
- $clk=1, S=1, R=0$ gives the SET state i.e., $Q=1, \bar{Q}=0$
- $clk=1, S=1, R=1$ is not allowed, because it is not able to determine the next state. This condition is said to be a "Race Condition".

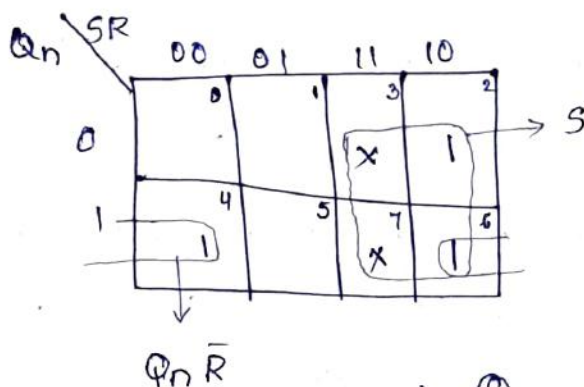
clk	S	R	Q_{n+1}
0	x	x	Q_n Nochange
1	0	0	Q_n No change
1	0	1	0 RESET
1	1	0	1 SET
1	1	1	x Not allowed.

Truth table

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

Characteristic table

from characteristic table



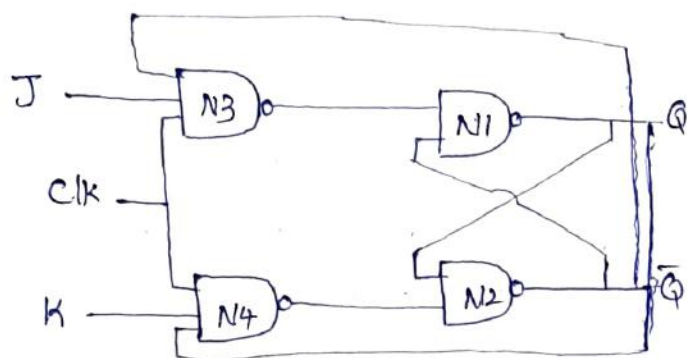
Q_n	Q_{n+1}	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

Excitation table

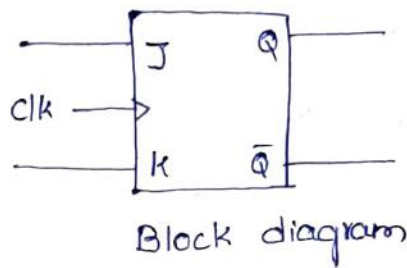
$$\Rightarrow Q_{n+1} = S + Q_n \bar{R}$$

JK flip-flop

The Invalid condition in SR flip-flop, when $S=R=1$ is eliminated in J-K flipflop. There is a feedback from the output to the inputs.



Logic diagram



The J and K are called control inputs, because they determine what the flipflop does when a positive clock arrives.

→ When $J=0, K=0$ then both N3 and N4 will produce high output and the previous value of Q and $\bar{Q}$ retained as it is.

→ When $J=0, K=1$, N3 will get an output as 1 and Q of N4 depends on the value of $\bar{Q}$. The final output is $Q=0, \bar{Q}=1$ i.e., Reset state.

→ When $J=1, K=0$ the output of N4 is 1 and N3 depends on the value of Q . The final output is $Q=1$ and $\bar{Q}=0$ i.e., Set state.

→ When $J=1, K=1$ it is possible to set (or) reset the flip-flop depending on the current state of output.

If $Q=1, \bar{Q}=0$ then N4 passes '0' to N2 which produces $\bar{Q}=1, Q=0$ which is reset state. When $J=1, K=1, Q$ changes to the complement of the last

State. The flipflop is said to be in the toggle state.

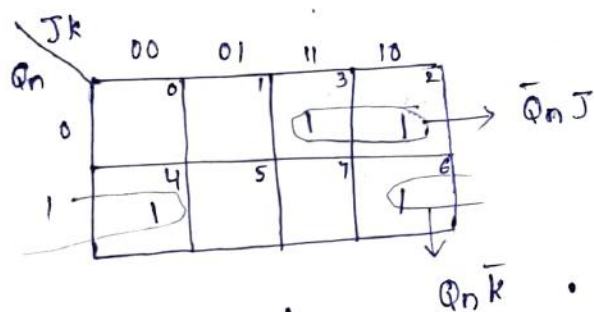
J	K	Q_{n+1}	
0	0	Q_n	Nochange
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}_n$	Complement/ Toggle

Truth Table

Q_n	J	K	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

Characteristic table

from characteristic Table



$$Q_{n+1} = \bar{Q}_n J + Q_n \bar{K}$$

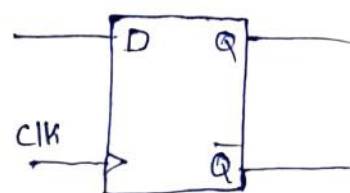
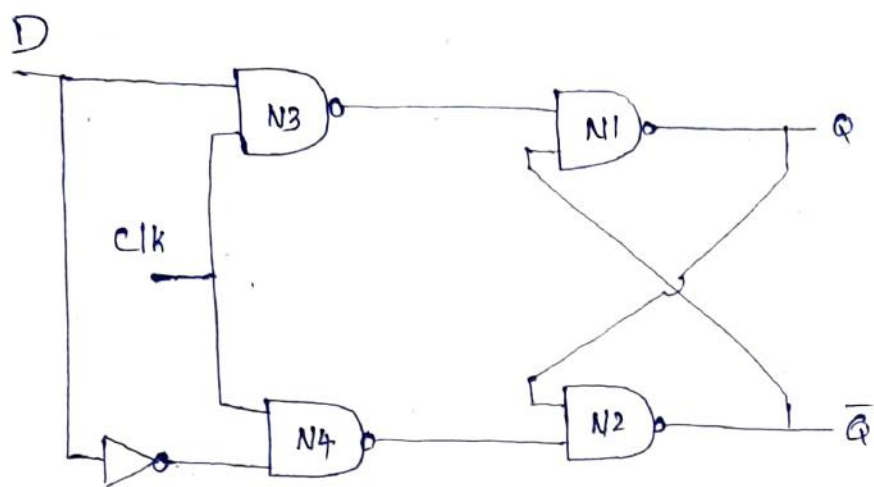
Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Excitation table

D flip-flop:-

The D flip-flop is the modified form of S-R flipflop, S-R flip-flop is converted to D flip-flop by adding an inverter between S and R and only one input D is taken instead of S and R. so one input is D and complement of D is given as another input. The logic diagram and the block

diagram of D flip-flop with clocked input.



Block diagram

Logic diagram

when the clock is low both the NAND gates (N1 and N2) are disabled and Q retains its last value. when clock is high both the gates are enabled and the input value at D is transferred to its output Q. D flipflop is also called "Data flip-flop".

clk	D	Q_{n+1}
0	x	Q_n
1	0	0
1	1	1

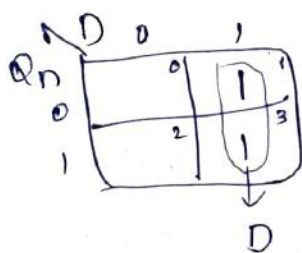
Truth Table

Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Characteristic Table

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

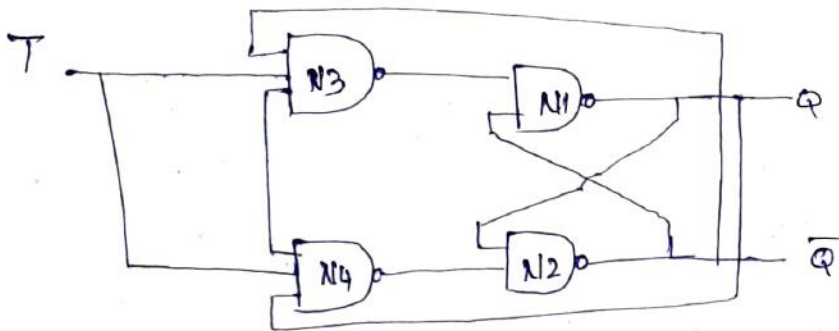
Excitation Table



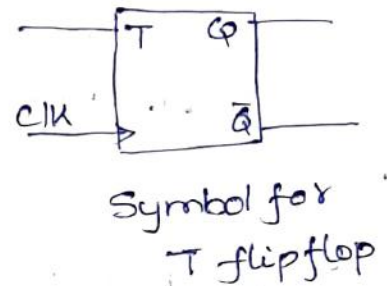
$$Q_{n+1} = D$$

T flipflop:-

If the T input is high, the T flip-flop changes state ("toggles") whenever the clock input is strobed. If the T input is low, the flip-flop holds the previous value.



Logic diagram

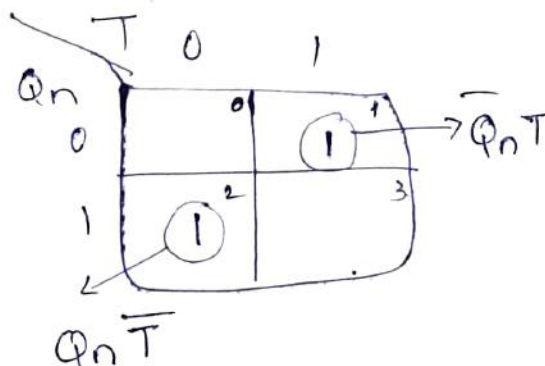


clk	T	Q_{n+1}
0	x	Q_n
1	0	Q_n
1	1	$\overline{Q_n}$

Truth Table

Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Characteristic Table



$$Q_{n+1} = \overline{Q_n}T + Q_n\overline{T}$$

Applications of Flip-Flops:-

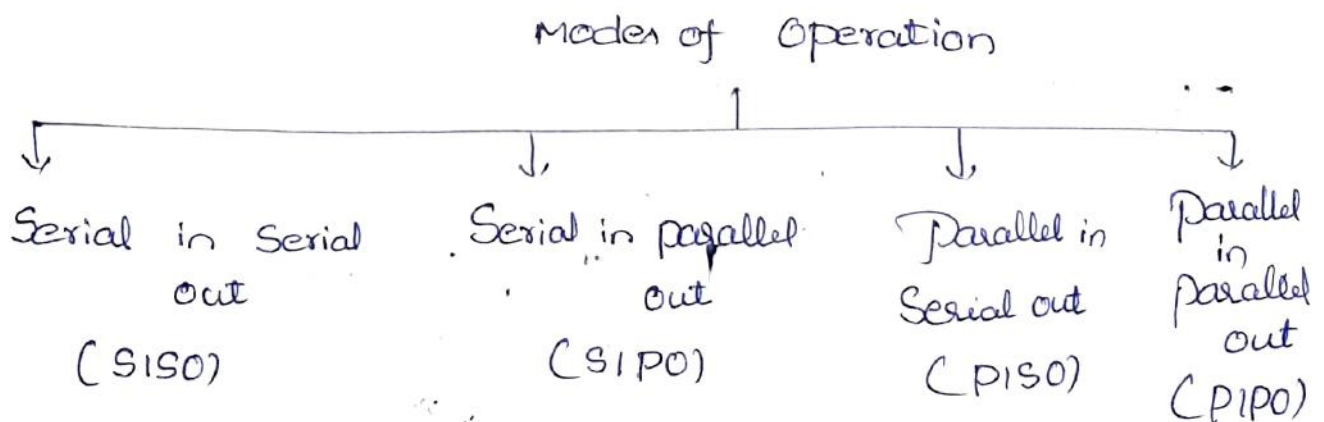
- Frequency Division
- Parallel data storage
- Serial data storage.
- Transfer of data.

Registers

To increase the storage capacity in terms of number of bits, we have to use no. of flipflops. Such a group of flip-flops is known as a Register.

The n -bit register will consist of n number of flipflops and it is capable of storing an n bit word.

Registers may be classified based on the way in which data are entered and taken out from a register. There may be following four possible modes:-



Shift Registers:-

The binary data in a register can be moved within the register from one flipflop to the other or outside it with application of clock pulses. Those registers which allow such data transfers are known as Shift Registers.

Modes of operation of a Shift Register.

- Serial input Serial output (Serial shift Right)
- Serial input Serial output (Serial shift left)
- Serial input Parallel output
- Parallel input Serial output

Serial Input Serial Output (Shift Left Mode)

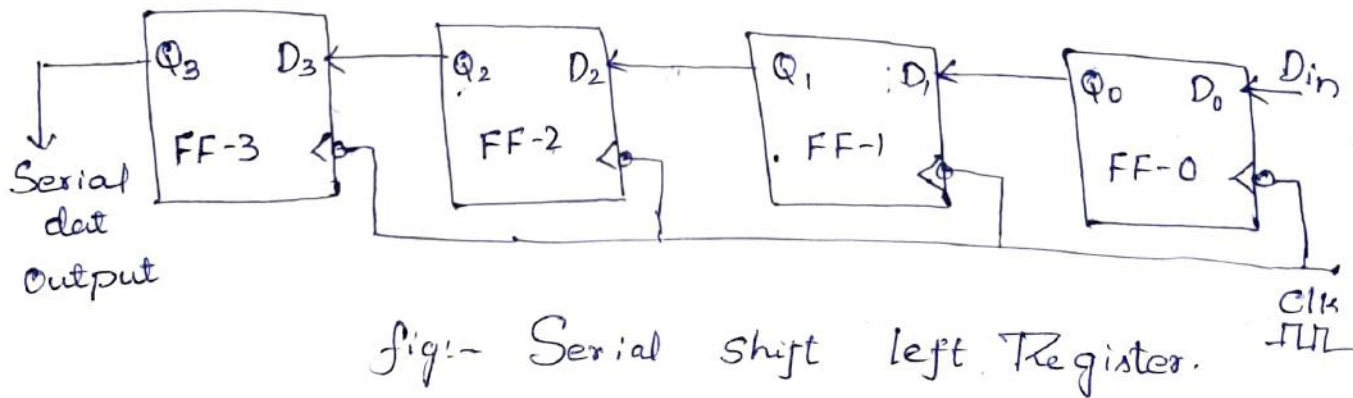


Fig:- Serial shift left Register.

→ Let us consider that all the flipflops initially are in the rest condition. i.e., $Q_3 = Q_2 = Q_1 = Q_0 = 0$.

→ Let us illustrate the entry of a four bit binary number 1111 into the register.

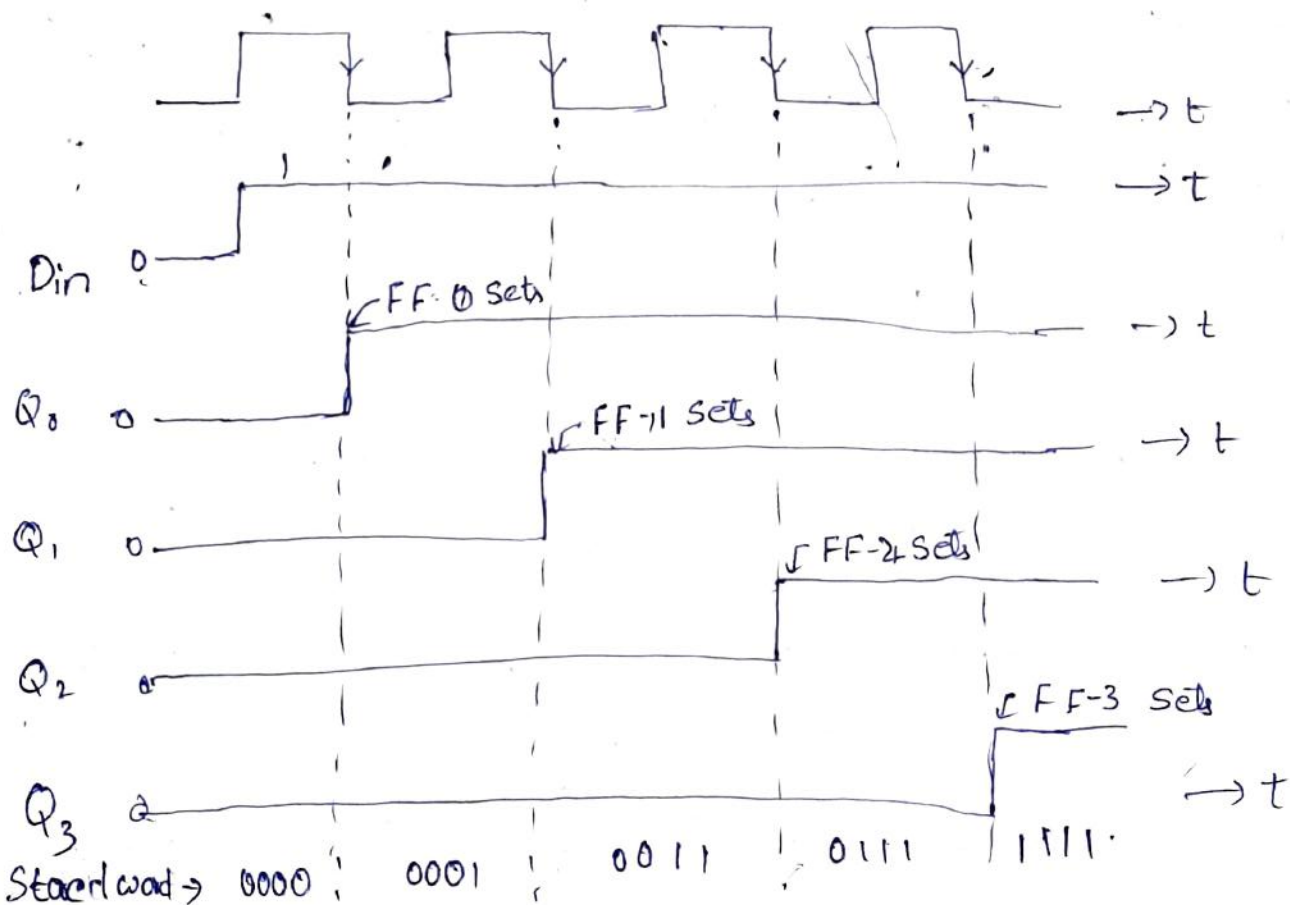
This Number must be applied to D_{in} bit-by-bit

with the MSB bit applied first.

CLK	Q_3	$Q_2 = D_3$	$Q_1 = D_2$	$Q_0 = D_1$	Serial input $D_{in} = D_0$
	0	0	0	0	
↓	0	0	0	1 ←	1
↓	0	0	1 ←	1 ←	1
↓	0	1 ←	1 ←	1 ←	1
↓	1 ←	1 ←	1 ←	1 ←	1

Direction of data travel ←

Waveforms for Shift Left operation



Serial in Serial Out (Shift Right Mode)

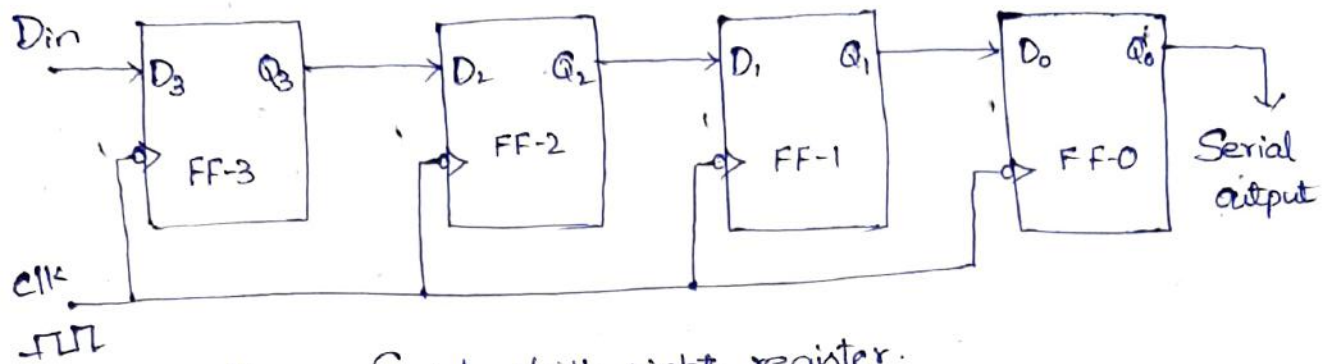


fig:- Serial shift right register.

Before application of clock signal, let $Q_3 Q_2 Q_1 Q_0 = 0000$ and we apply LSB bit of the number to be entered to D_{in} . Hence $D_{in} = D_3 = 1$. Then, we apply the clock. On the first falling edge of clock, the FF-3 is set, and the stored word in the register will be given by

$$Q_3 Q_2 Q_1 Q_0 = 1000$$

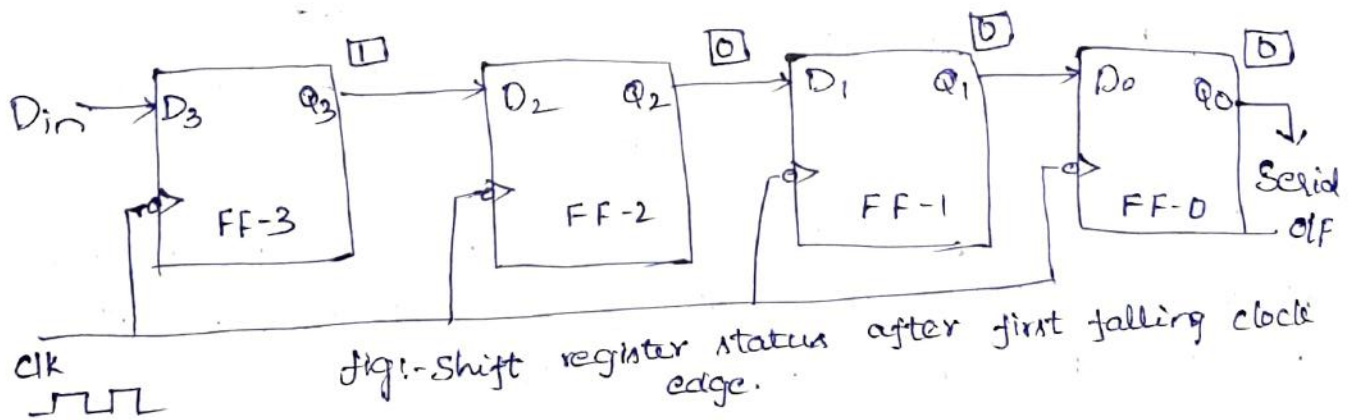


fig:- Shift register status after first falling clock edge.

Apply the next bit to D_{in} . Hence, $D_{in} = 1$. As soon as the next negative edge of the clock hits, FF-2 sets and the stored word will change

to, $Q_3 \ Q_2 \ Q_1 \ Q_0 = 1 \ 1 \ 0 \ 0$

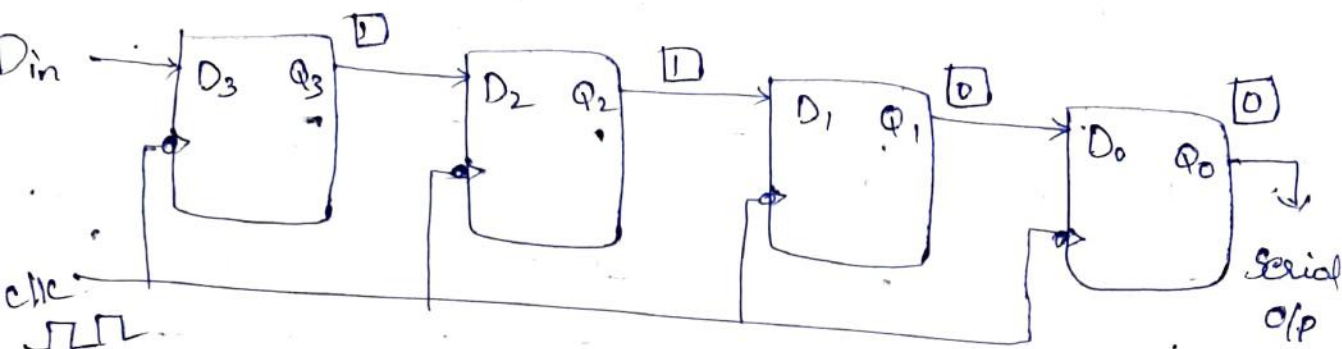


fig:- Shift register status after second falling edge of clock.

Apply next bit to be stored i.e., 1 to D_{in} . Then apply the clock pulse. As soon as the third negative clock edge hits. FF-1 will set and the output will get modified to

$Q_3 \ Q_2 \ Q_1 \ Q_0 = 1 \ 1 \ 1 \ 0$

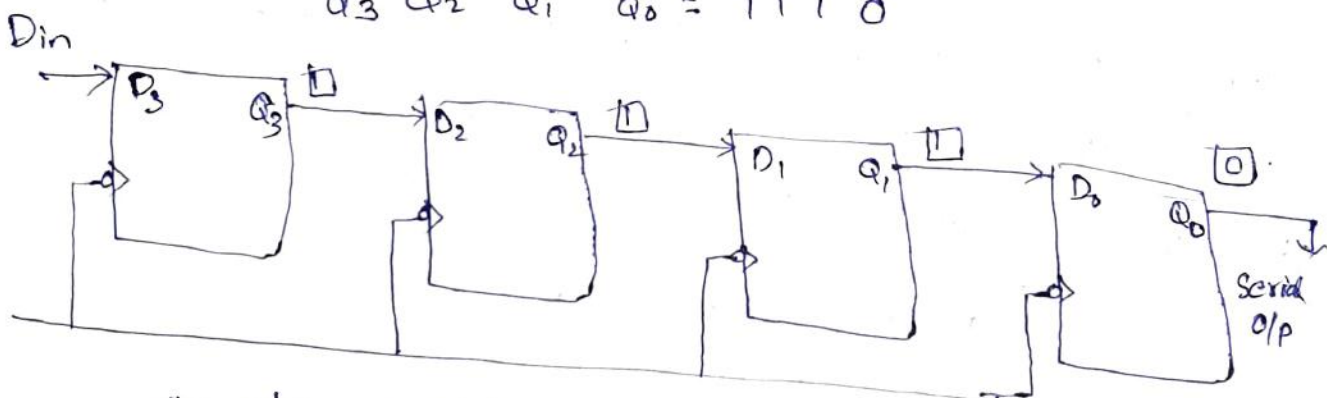
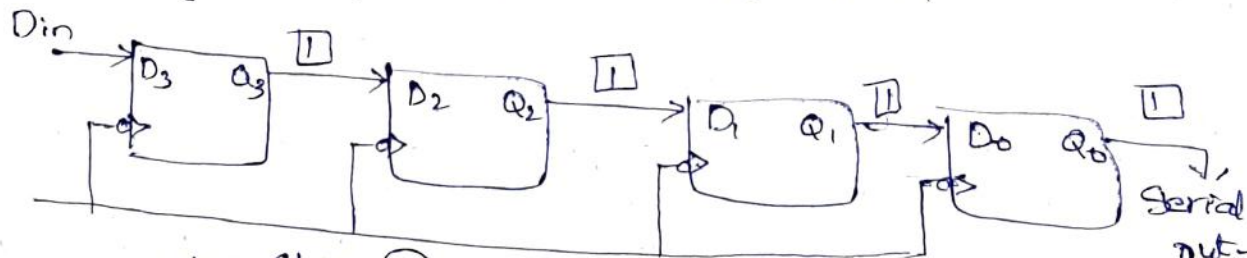


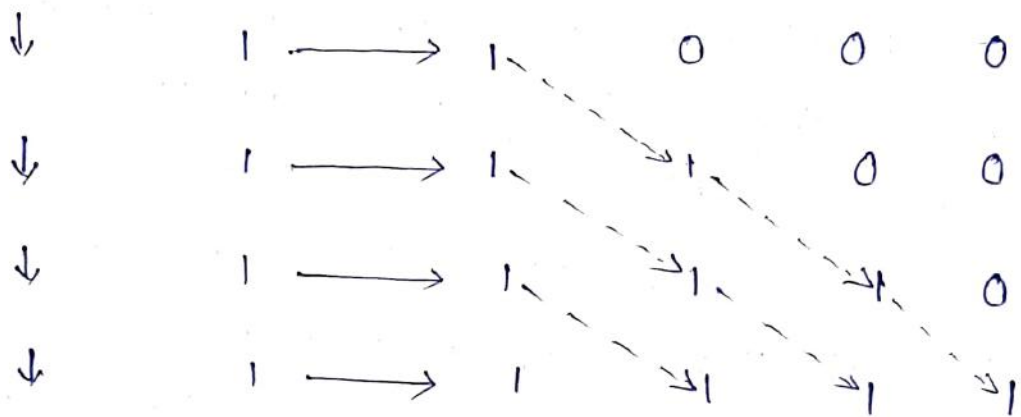
fig:- Shift register status after the third falling edge of clock.

Similarly with $D_{in} = 1$ with the fourth negative clock edge arriving, the stored word in the register will be given by $Q_3 \ Q_2 \ Q_1 \ Q_0 = 1 \ 1 \ 1 \ 1$



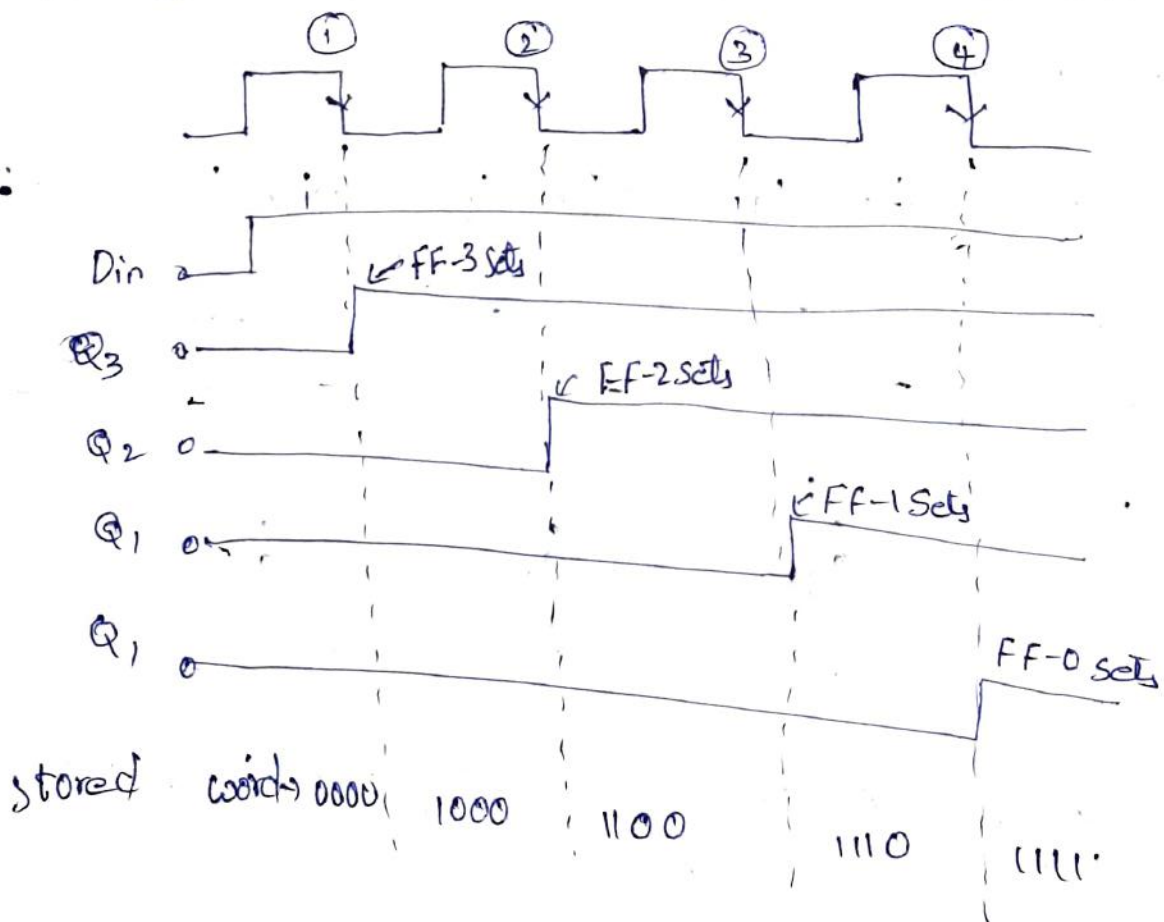
of clock. fig:- Shift Register status after the fourth falling edge

clk $D_{in} = Q_3$ $Q_3 = D_2$ $Q_2 = D_1$ $Q_1 = D_0$ $Q_0 = D_{in}$



→ Direction of data travel.

Waveforms for Shift Right Operation



Serial IN Parallel OUT (SIPO):-

In this data is entered serially and then taken out in parallel. This means that first the data is loaded bit-by-bit. The outputs are disabled as long as the loading is taking place. As soon as the loading is complete, and all the flipflops consist of their required data, the outputs are enabled. So that all the loaded data is made available over all the output lines simultaneously. Also, number of clock cycles required to load a four bit word is 4. Therefore, the speed of operation of SIPO mode will remain same as that of SISO mode.

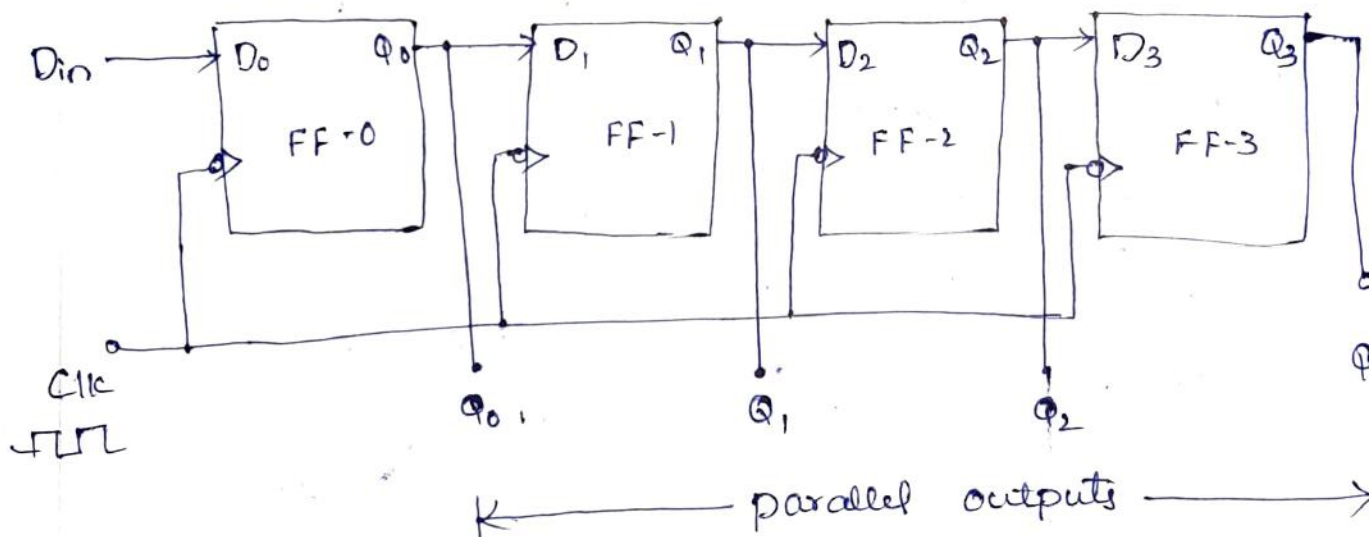
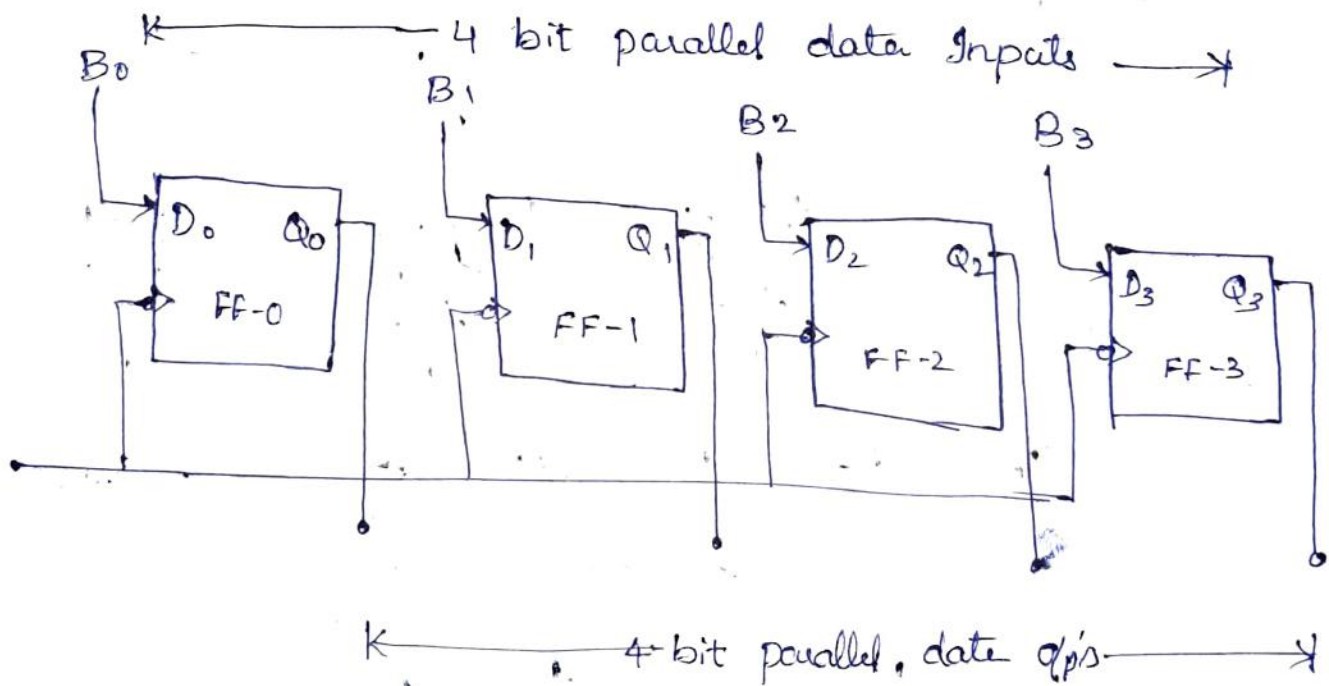


fig:- Illustration of Serial input parallel output mode.

PARALLEL IN PARALLEL OUT (PIPO)

Figure shows the parallel in parallel out mode of operation. The 4-bit binary input B_0, B_1, B_2, B_3 is applied to the data inputs D_0, D_1, D_2 and D_3 respectively of the four flip-flops. As soon as a negative clock edge is applied, the input binary bits shall be loaded into the flip-flops simultaneously. The loaded bits appear simultaneously to the output side. Here, only one clock pulse is essential to load all the bits.



PARALLEL IN SERIAL OUT MODE (PISO)

In this mode, the bits are entered in parallel i.e., The circuit shown in figure is a four bit parallel input serial output register. Output of previous FF is connected to the input of the next one with the help of a combinational circuit. Then the binary input word B_0, B_1, B_2, B_3 is applied through the same combinational circuit. There are two modes in which this circuit can work. These modes are shift mode and load mode.

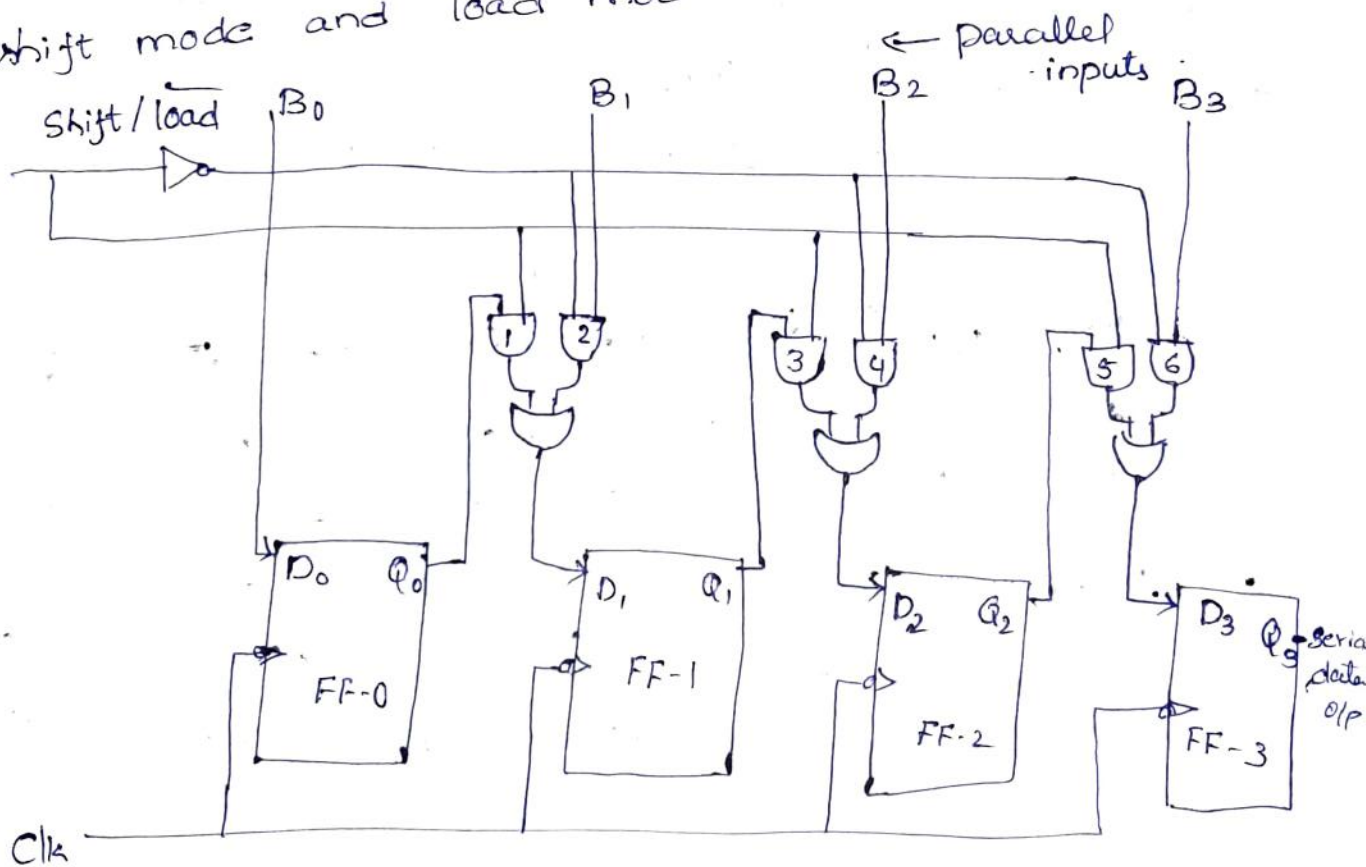


Fig. 1. Parallel in Serial out Shift Register

Shift Mode

When the Shift/Load line is high (1), the AND gates 2, 4, 6 become inactive. Hence, the parallel

loading of the data becomes impossible. But, the AND gates 1, 3 and 5 become active. Therefore, the shifting of data from left to right bit-by-bit on application of clock pulses. Thus, the parallel in Serial out operation takes place.

Load Mode:

When the shift/load line is low (0), the AND gates 2, 4 and 6 become active. They pass B_1, B_2 and B_3 bits to the corresponding flip-flops. On low going edge of clock, the binary inputs B_0, B_1, B_2, B_3 get loaded into the corresponding flip-flops. Therefore, Parallel loading takes place.

Counters:-

The digital circuit used for counting pulses is known as counter.

It is a Sequential circuit. Counter is the widest applications of flip-flops. It is a group of flip-flops with a clock signal applied. Basically, counters count the number of clock pulses. It can be used for measuring frequency or time period.

Classification of Counters

counters are basically of following two types:

- i. Asynchronous or ripple counters.
- ii. Synchronous counters.

i) Asynchronous / Ripple up Counter

for these counters, the external clock signal is applied to one flip-flop and then the output of preceding flip-flop is connected to the clock of next flip-flop.

figure shows a 2 bit Ripple up counter. The no. of flip-flops used is 2. The Toggle flip-flops are being used. But, we can use the JK flip-flop.

Here T is connected permanently to logic 1.

External clock is applied to the clock input of flipflop A and Q_A output is applied to the clock input of the next flip-flop i.e., FF-B.

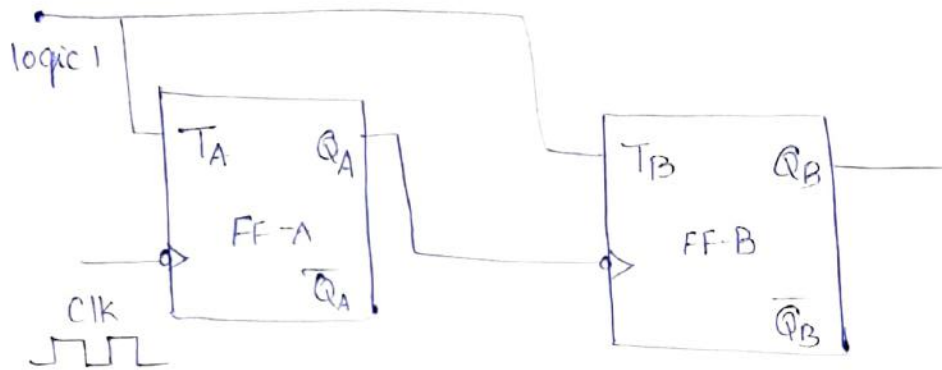


fig:- A two bit asynchronous binary up counter.

→ Initially both the flipflops be in reset condition.

$$\text{Therefore, } Q_B Q_A = 00$$

→ The first negative going clock edge hits FF-A, it will toggle as $T_A = 1$. Hence, Q_A will be equal to 1. Also, Q_A is connected to clock input of FF-B. Since Q_A has changed from 0 to 1, it is treated as the positive clock edge by FF-B. There is no change in Q_B because FF-B is a negative edge triggered FF. Hence after the first clock pulse the counter outputs are $Q_B Q_A = 01$

→ At the Second falling edge of clock,

FF-A toggles again, to make $Q_A = 0$.

∴ Q_B will become 1.

The counter outputs are

$$Q_B Q_A = 10$$

→ At the ^{Third} falling edge of clock

FF-A toggles and Q_A changes from 0 to 1.

∴ Q_B will remain same

The counter outputs are

$$Q_B Q_A = 11$$

→ At the fourth falling edge of clock

FF-A toggles and Q_A becomes 0 and Q_B also toggles from 1 to 0.

∴ The counter outputs are

$$Q_B Q_A = 00.$$

Clk	Counter outputs		State number	Decimal equivalent of counter o/p
	Q_B (MSB)	Q_A (LSB)		
Initially	0	0	—	0
1 st (↓)	0	1	1	1
2 nd (↓)	1	0	2	2
3 rd (↓)	1	1	3	3
4 th (↓)	0	0	4	0

from the above table, this counter has four distinct states of output namely 00, 01, 10 and 11. In general, the number of states $= 2^n$, where n is equal to the number of flip-flops.

Down Counter:-

The counters which can count in the downward direction, i.e., from the maximum count to zero are called down counters.

A 3-bit asynchronous down counter has been shown in figure. The clock input is applied directly to FF-A. But, $\bar{Q}_A$ is connected to clock of FF-B, $\bar{Q}_B$ to clock of FF-C and so on.

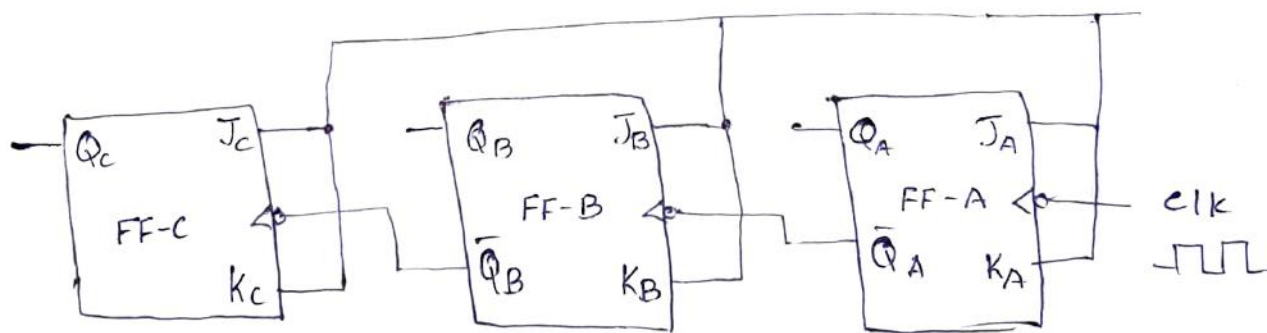


Fig:- A 3-bit asynchronous down counter.

Let initially all the flip-flops be in the reset condition.

$$\therefore Q_C \ Q_B \ Q_A = 000$$

As soon as the first falling clock pulse arrives,

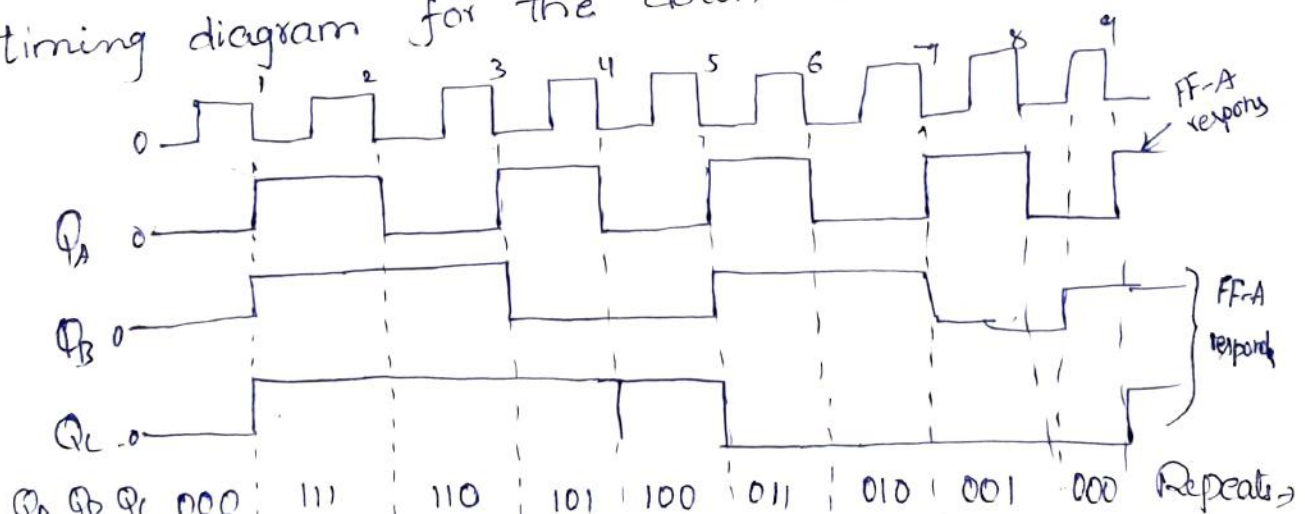
FF-A toggles. So, Q_A becomes 1 and $\overline{Q}_A$ becomes 0 from 1. Also, $\overline{Q}_A$ acts as a clock to FF-B. Hence, FF-B will change its state. Hence, Q_B becomes 1 and $\overline{Q}_B$ becomes 0 from 1. $\overline{Q}_B$ acts as clock to FF-C. Hence FF-C will change its state. Therefore Q_C becomes 1 and $\overline{Q}_C$ becomes 0. Hence, after the first clock pulse, the output of counter are,

$$Q_C Q_B Q_A = 111$$

Corresponding to the second falling clock edge, FF-A toggles, Q_A becomes 0 and $\overline{Q}_A$ becomes 1. This positive going change in $\overline{Q}_A$ does not alter the state of FF-B. So, Q_B remains 1 and $\overline{Q}_B$ remains 0. Hence, there is no change in the state of FF-C. Hence, after the second clock pulse, the counter outputs as under:

$$Q_C Q_B Q_A = 110$$

The down counting will thus take place. The timing diagram for the down counter is shown.



Synchronous Counters

If the clock pulses are applied to all the flipflops in a counter simultaneously, then such a counter is called as Synchronous counter.

2-Bit Synchronous UP Counter

A 2-bit synchronous counter has shown in fig:

The J_A and K_A inputs of FF-A are connected to logic 1. Hence FF-A works as a toggle flip-flop. The J_B and K_B inputs are connected to Q_A . Hence, FF-B toggles if $Q_A = 1$ and there won't be any state change if $Q_A = 0$.

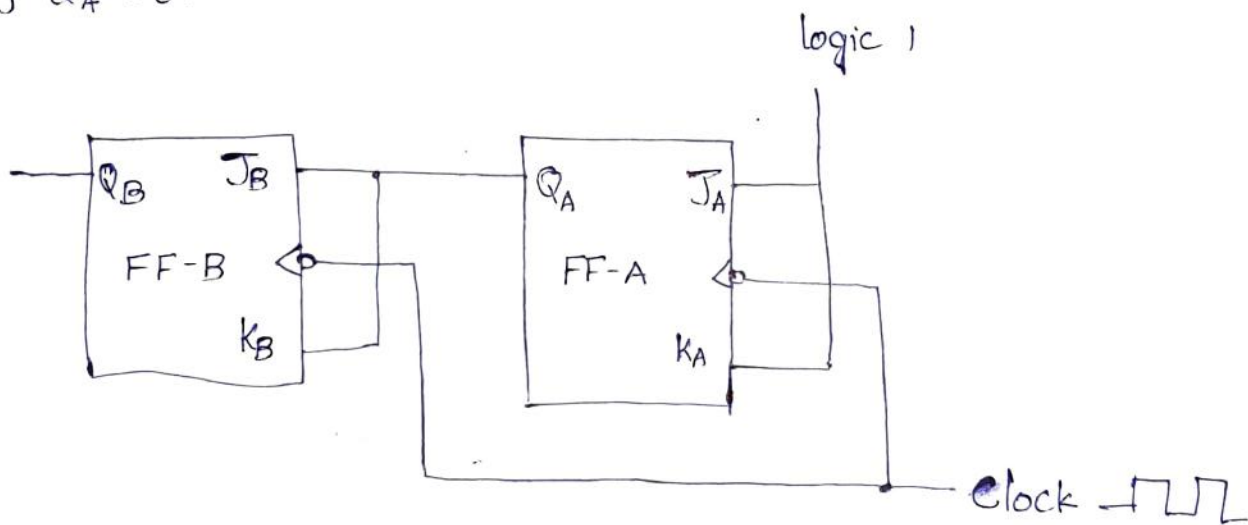


fig:- A 2-bit Synchronous counter.

→ Initially $Q_B Q_A = 0 0$

→ After first negative clock edge is applied. FF-A toggles and Q_A will change from 0 to 1. But, at the instant of applications of negative

clock edge, $Q_A = 0$, therefore, $J_B = K_B = 0$. Hence FF-B will not change its state. Thus, Q_B will remain 0.

$$\therefore Q_B Q_A = 01$$

→ On the arrival of second negative clock edge, FF-A toggles again and Q_A changes from 1 to 0. But, at this instant, Q_A was 1. Hence, $K_B = 1$ and FF-B toggles. Hence, Q_B changes from 0 to 1.

$$\therefore Q_B Q_A = 10$$

→ Similarly on the next clock pulse FF-A toggles Q_A from 0 to 1. But there is no change of state for Q_B .

$$Q_B Q_A = 11$$

→ On the next clock pulse Q_A changes from 1 to 0 as Q_B will also change from 1 to 0.

$$Q_B Q_A = 00$$

CLOCK	Counter outputs	
	Q_B (MSB)	Q_A (LSB)
Initially	0	0
1 st (↓)	0	1
2 nd (↓)	1	0
3 rd (↓)	1	1
4 th (↓)	0	0 =

S.no	parameter of comparison	Asynchronous Counter	Synchronous Counter
1.	Circuit Complexity	Logic circuit is simple	with increase in no. of states, the logic circuit becomes complicated.
2.	Connection pattern	output of the preceding FF is connected to clock of the next FF.	There is no connection between output of preceding FF and CLK of next one.
3.	Clock input	All the FFs are not clocked simultaneously.	All the FFs receive clock signal simultaneously.
4.	Propagation Delay	$P.D = n \times (t_d)$ where n is no. of FFs and t_d is p.d per F-F	$P.D = (t_d)_{FF} + (t_d)_{gate}$ It is much shorter than that of asynchronous counter.
5.	Maximum frequency of operation	Low because of the long propagation delay.	High due to shorter propagation delay.