* NUMBER SYSTEM :-

- The binary digits have a great significance in digital system.

- Also, the decimal system is the universally used system to represent quantities in day to day life.

- But along with these, there exists more number systems. So, the numbers systems are:

~~1] Digital Number System:~~

1] Decimal number system
2] Binary number system
3] Octal number system
4] Hexadecimal number system

1] Decimal number system:

- It has a base of 10.

- It has 10 numbers, from 0 to 9 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

- The decimal position values are represented as power of 10.

10^3	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}

2] Binary number System:

- Its base is 2.
- It uses two symbols, 0 and 1.
- Each symbol is known as a bit. A group of 4 bits is called a nibble while a group of 8 bits is called a byte.
- The binary values are represented as power of 2.

2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}

3] Octal number system:

- It has a base of 8.
- It uses eight symbols from 0 to 7 (0, 1, 2, 3, 4, 5, 6, 7).
- The octal value positions are represented as powers of 8.

8^4	8^3	8^2	8^1	8^0	8^{-1}	8^{-2}	8^{-3}

4]

4] Hexadecimal number system:

- Its base is 16.
- It contains 16 symbols. Out of them, 10 are digits 0 to 9 and other 6 are the alphabets A to F.

* Relationship betⁿ number systems:

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

* Number System Conversions:

1] Decimal to Binary
2] Binary to Decimal
3] Octal to Binary
4] Octal to Decimal
5] Decimal to Octal
6] Binary to Octal
7] Hexadecimal to decimal
8] Decimal to Hexadecimal.
9] Hexadecimal to Binary
10] Binary to Hexadecimal
11] Octal to Hexadecimal
12] Hexadecimal to octal

1] Binary to Decimal Conversion:

Ex. 1 Convert $(110110)_2 = (?)_{10}$

1	1	0	1	1	0
2^5	2^4	2^3	2^2	2^1	2^0

A $= 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

$= 32 + 16 + 0 + 4 + 2 + 0$

$= 54$

$\therefore (110110)_2 = (54)_{10}$

Ex. 2 Convert $(11.0111)_2 = (?)_{10}$

→ 1 1 . 0 1 1 1

2^1 2^0 2^{-1} 2^{-2} 2^{-3} 2^{-4}

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

$= 2 + 1 + 0 + 1/4 + 1/8 + 1/16$

$= 3.4375$

$\therefore (11.0111)_2 = (3.4375)_{10}$

2] Decimal to Binary Conversion

Ex 1 Convert $(29)_{10} = (?)_2$

2	29	1
2	14	0
2	7	1
2	3	1
2	1	1
	0	

↑

$\therefore (29)_{10} = (11101)_2$

Ex 2 convert (0.6875) to binary.

Decimal fraction		Product	Integer bit
0.6875	× 2	1.375	1
0.375	× 2	0.75	0
0.75	× 2	1.5	1
0.5	× 2	1.0	1

↓

$\therefore (0.6875)_{10} = (0.1011)_2$

3] Octal to Binary Conversion :

Octal Digit	Binary Equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Ex.1 Convert $(652)_8$ to binary

→

6 5 2

110 101 010

$= (110101010)_2$

Ex.2 Convert $(5432.76)_8$ to binary.

5 4 3 2 . 7 6

101 100 011 010 . 111 110

$= (101100011010.111110)_2$

4] Binary to Octal Conversion:

1. Convert $(110111010)_2$

→ 110 111 010

6 7 2 $= (672)_8$

2. Convert $(1101 \cdot 11011)$ to octal.

→ 001 101 · 110 110

1 5 · 6 6 $= (15.66)_8$

5] Octal to Decimal Conversion:

1. Convert $(250)_8$ to decimal.

→ 2 5 0

8^2 8^1 8^0

$= 2 \times 8^2 + 5 \times 8^1 + 0 \times 8^0$

$= 128 + 40 + 0$

$= (168)_{10}$

2. Convert $(35.7)_8$ to decimal.

→ 3 5 · 7

8^1 8^0 · 8^{-1}

$= 3 \times 8^1 + 5 \times 8^0 + 7 \times 8^{-1}$

$= 24 + 5 + 0.875$

$= (29.875)_{10}$

6] Decimal to Octal Conversion:

1. Convert $(6458)_{10}$ to octal.

8	6458	2
8	807	7
8	100	4
8	12	4
8	1	1
	0	

$= (14472)_8$

2. Convert $(0.615)_{10}$ to octal.

Decimal fraction	Product	Integer digit
0.615×8	4.92	4
0.92×8	7.36	7
0.36×8	2.88	2
0.88×8	7.04	7
0.04×8	0.32	0
0.32×8	2.56	2

$\therefore (0.615)_{10} = (0.472702)$

7] Hexadecimal to Decimal conversion:

1. Convert $(268)_{16}$ to decimal.

→ 2 6 8

16^2 16^1 16^0

$= 2 \times 16^2 + 6 \times 16^1 + 8 \times 16^0$

$= 512 + 96 + 8$

$= (616)_{10}$

Convert $(456E)_{16}$ to decimal.

4	5	6	E
16^3	16^2	16^1	16^0

$= 4 \times 16^3 + 5 \times 16^2 + 6 \times 16^1 + 14 \times 16^0$ (E = 14)

$= (17774)_{10}$

Convert $(11A.62)_{16} = (?)_{10}$

1	1	A	.	6	2
16^2	16^1	16^0		16^{-1}	16^{-2}

$= 1 \times 16^2 + 1 \times 16 + 10 \times 16^0 + 6 \times 16^{-1} + 2 \times 16^{-2}$

$= 256 + 16 + 10 + \frac{6}{16} + \frac{2}{256}$

$= (282.3828)_{10}$

Decimal to Hexadecimal Conversion:

Convert $(200)_{10}$ to hexadecimal.

16	200	8
16	12	12 → C
	0	

LSB ↑ MSB

$\Rightarrow (200)_{10} = (C8)_{16}$

Convert (2001.43) to hexadecimal

16	2001	1
16	125	13 → D
16	7	7
	0	

LSB ↑ MSB

$(2001)_{10} = (7D1)_{16}$

Decimal Fraction	Product	Integer digit
0.43 × 16	6.88	6
0.88 × 16	14.08	14 (E)
0.08 × 16	1.28	1
0.28 × 16	4.48	4
0.48 × 16	7.68	7
0.68 × 16	10.88	10 (A)

$(0.43)_{10} = (0.6E147A)_{16}$

$\therefore (2001.43)_{10} = (7D1.6E147A)_{16}$

9] Hexadecimal to Binary Conversion:

1. Convert $(D283)_{16}$ to binary.

→ D (13) 2 8 3

1101 0010 1000 0011

$= (1101\ 0010\ 1000\ 0011)_2$

2. Convert $(57EB:AD)_{16}$ to binary

5 (14) 7 E (11) B : A (10) D (13)

0101 0111 1110 1011 : 1010 1101

$= (01010111\ 11101011 : 10101101)_2$

10] Binary to Hexadecimal:

1. Convert $(10111010)_2$ to hexadecimal.

0001 0111 1010

1 7 10

1 7 A

$= (17A)_{16}$

2. Convert $(100101110.11101)_2 = (?)_{16}$

0001 0010 1110 . 1110 1000

1 2 14 . 14 8

2 E E 8

$= (12E.E8)_{16}$

11] Octal to Hexadecimal Conversion:

1. convert $(537)_8$ to hexadecimal.

5 3 7

101 011 111

~~$= (010100111111)$~~

$= (1,0101,1111)_2$

0001 0101 1111

1 5 15

1 5 F

$= (15F)_{16}$

2. Convert $(146)_8$ to $(?)_{16}$

1 4 6
001 100 110

$= (001100110)_2$

$= 0\,0110\,0110$
0 6 6

$= (66)_{16}$

3.12] Hexadecimal to Octal Conversion:

1. Convert 3DB into octal.

3 D B
3 13 11
0011 1101 1011

$= (0011\,1101\,1011)_2$

$= 001\,111\,011\,011$
1 7 3 3

$= (1733)_8$

2. $(78.4B)_{16} = (?)_8$

7 8 . 4 B
7 8 . 4 11
0111 1000 . 0100 1011

$= (1111000.01001011)$

= 001 111 000 . 010 010 110

= 1 7 0 . 2 2 6

$= (170.226)_8$

* 1's Complement:

1's complement of a number can be obtained simply by changing all 1's to 0's and 0's to 1's.

E.g., ~~convert~~ find 1's complement of 101010 and 110101

→ 1) 101010 → 1 0 1 0 1 0
↓ ↓ ↓ ↓ ↓ ↓
0 1 0 1 0 1

2) 110101 → 1 1 0 1 0 1
↓ ↓ ↓ ↓ ↓ ↓
0 0 1 0 1 0

* 2's complement:

2's complement of a binary number is obtained by adding 1 to 1's complement of the number.

Negative numbers are represented by 2's complement of their absolute value.

• Example: Find 2's complement of $(1010\ 0101)_2$

→ 1 0 1 0 0 1 0 1 → 0 1 0 1 1 0 1 0
+ 1
0 1 0 1 1 0 1 1

* Binary Arithmetics:

- The arithmetic operations such as addition, substraction, multiplication & division are performed in binary number system.

- **Rules for addition:**

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 (carry 1)
1 + 1 + 1 = 1 (carry 1)
```

e.g.
```
  1011
+ 1001
------
 10100
```

- **Rules for substraction:**

```
0 - 0 = 0
(borrow 1) 0 - 1 = 1
1 - 0 = 1
1 - 1 = 0
```

```
  1110
- 1011
------
  0011
```

- **Rules for binary multiplication:**

```
0 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1
```

```
   101
 × 010
------
   000
+ 101x
+ 000xx
------
 01010
```

2. Logic Gates

- The digital circuits operate in binary mode, where input or output is either 0 or 1. Thus, boolean algebra can be used for designing and symplifying digital circuits.

- Logic circuits are fundamental circuits used for realizing digital equations and circuits.

* Logic Gates:

"Logic gates are devices that have ~~two~~ one or more inputs and one output."

The output produced will be high or low depending upon the combination of high and low inputs used and the type of gate used.

There are three types of logic gates:

1. Basic Gates
2. Universal Gates
3. Derived Gates

1. Basic Gates:

There are three basic logic gates as follows:

1) NOT Gate
2) OR Gate
3) AND Gate

1) Inverter / NOT Gate:

- This circuit performs a basic logic function called "inversion" or complementation. It has one input and one output.

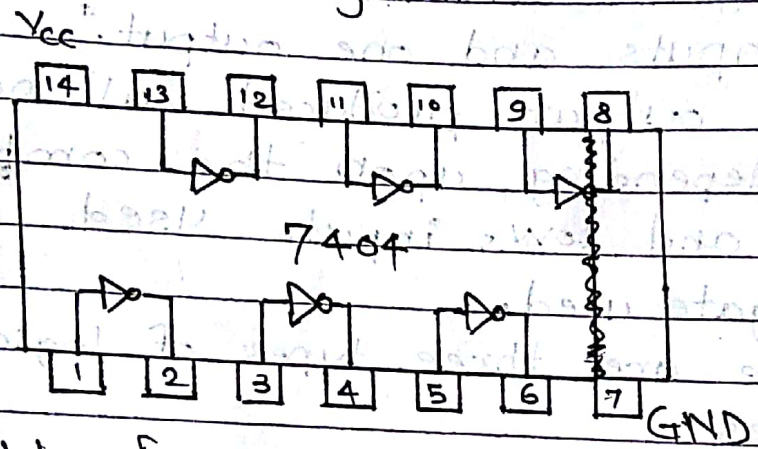
- It is expressed as $Y = \bar{A}$

- Truth table:

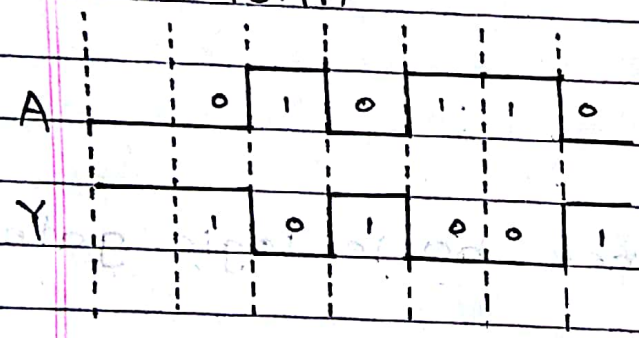
I/P	O/P
A	$Y = \bar{A}$
0	1
1	0

- Symbol: A —▷○— Y

- Circuit diagram:

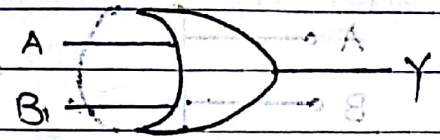


- Waveform:



2) OR Gate:

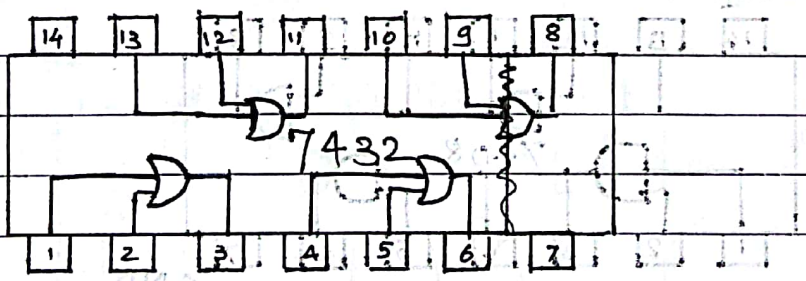
- This circuit performs the function of logical addition. It is expressed as $Y = A + B$
- The O/P is high if any of the input / both the inputs are high. It is low for both inputs low.
- Symbol:



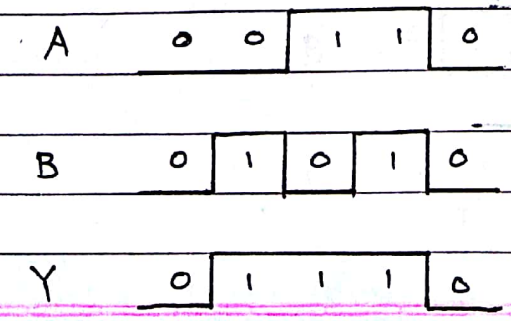
- Truth table:

Input		Output
A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

- Circuit diagram:

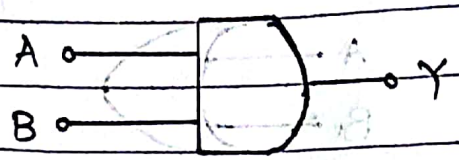


- Waveform:



3) AND Gate:

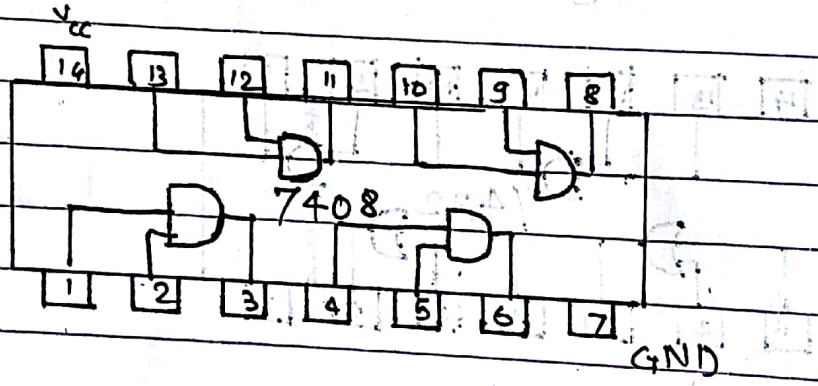
- This circuit performs the function of logical multiplication and is expressed as $Y = A \cdot B$.
- The output is high only when both the inputs are high.
- Symbol:



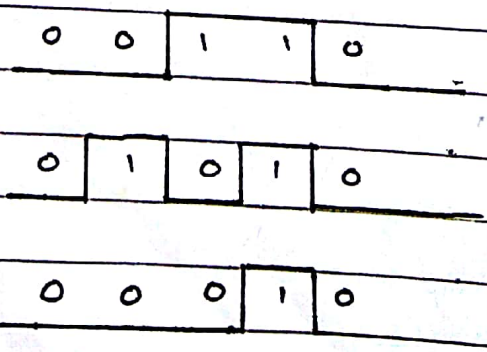
- Truth table:

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

- Circuit diagram:



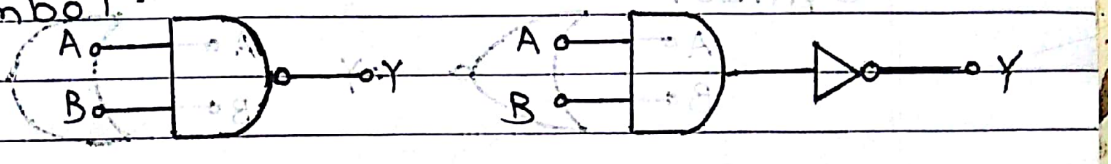
- Waveform:



2. Universal Gates:

1) NAND Gate:

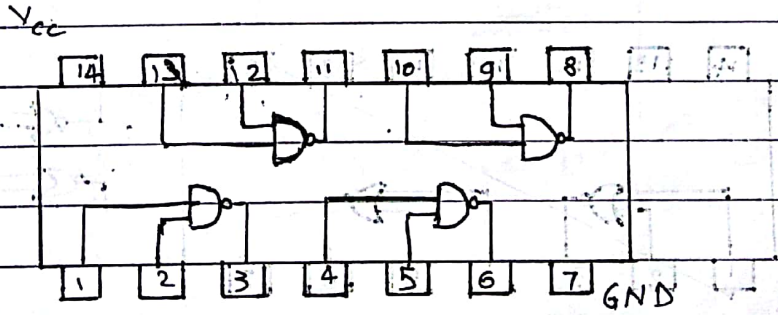
- NAND means NOT-AND. Its O/P is complement of AND. It is represented as: $Y = \overline{AB}$
- The output is high only when one of the inputs is low.
- Symbol:



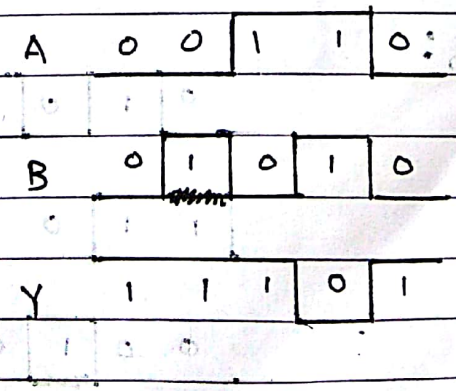
- Truth table:

Input		Output
A	B	$Y = \overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

- Circuit diagram:

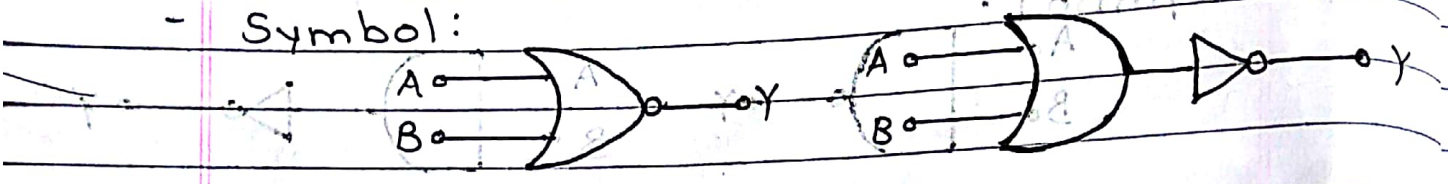


- Waveform:



2) NOR Gate:

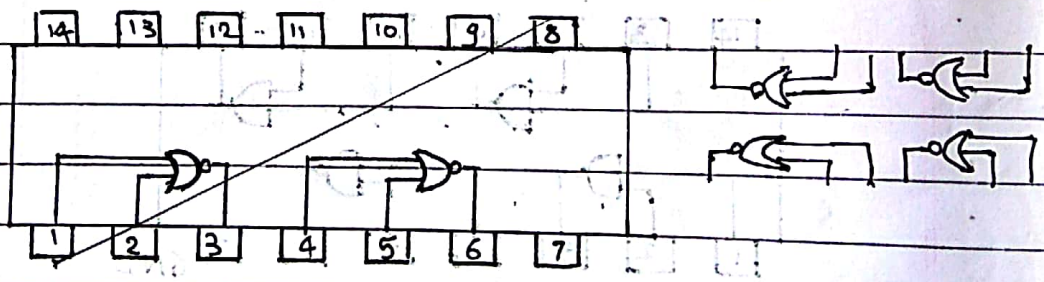
- NOR gate actually means NOT-OR. Its O/P is complement of OR. It is represented as $Y = \overline{A+B}$.
- The output is high only when both the inputs are low.
- Symbol:



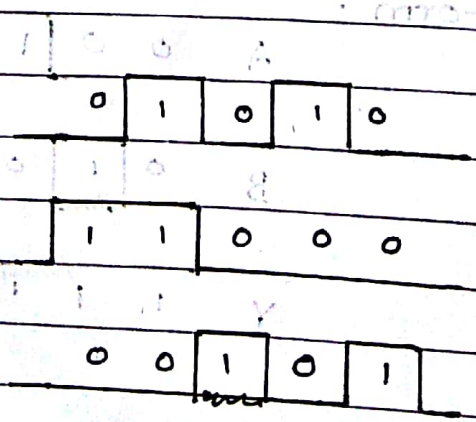
- Truth table:

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

- Circuit diagram:



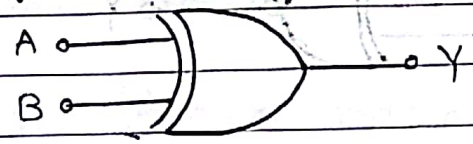
- Waveform:



3. Derived Gates:

1) EX-OR Gate:

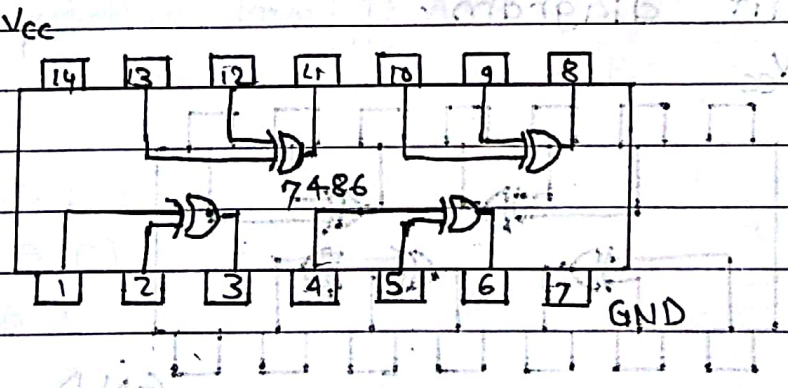
- The EX-OR gives high O/P when any of the input is high.
- It is represented as $Y = A \oplus B$ i.e. $Y = \bar{A}B + A\bar{B}$
- Symbol:



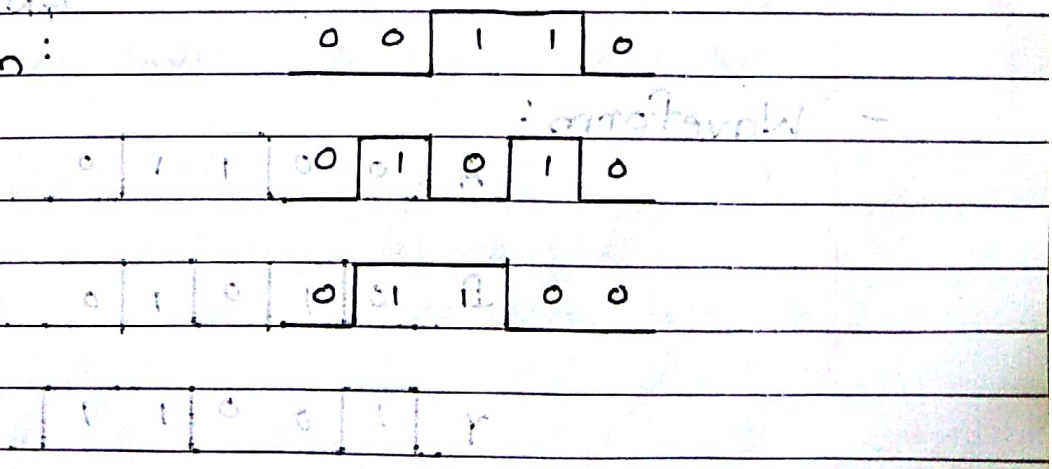
- Truth table:

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

- Circuit diagram:

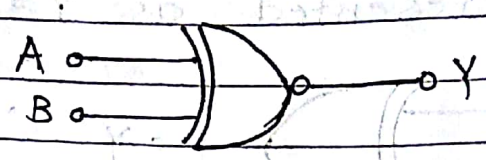


- Waveform:



2) EX-NOR Gate:

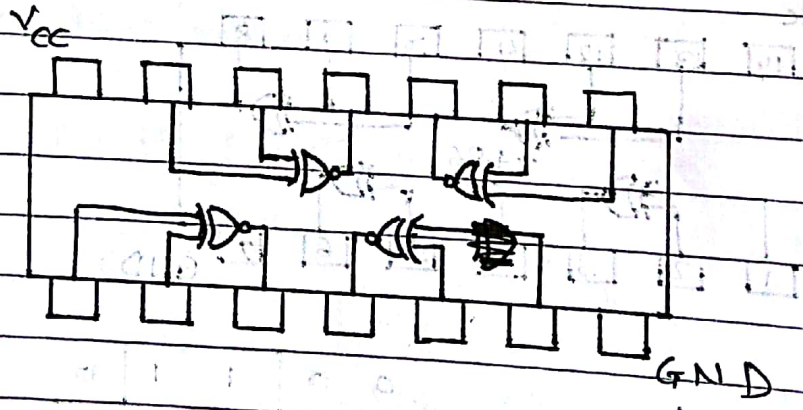
- EX-NOR can be represented by $Y = \overline{A \oplus B}$ i.e. $Y = AB + \bar{A} \cdot \bar{B}$.
- Its output is high if both the inputs are high or both the inputs are low.
- Symbol:



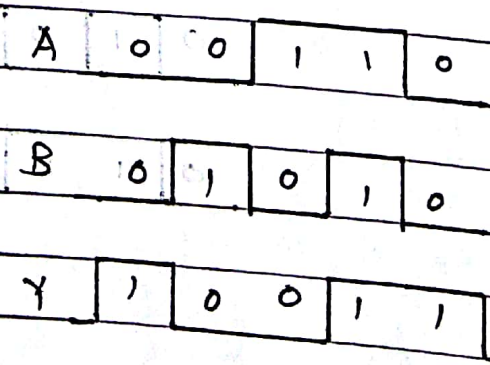
- Truth table:

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

- Circuit diagram:



- Waveform:



* Boolean Algebra:

- Boolean Algebra differs from ordinary algebra in a way that boolean constants and variables can have only two values (0 or 1). These values represent voltage level not actual numbers.

- There are three basic operations in boolean algebra:

1) Logical addition (+) or OR operation
2) Logical ~~subtraction~~ multiplication (·) or AND operation
3) Logical inversion (⁻) or NOT operation

* Boolean Laws:

1) Commutative law: i) $A+B = B+A$
   ii) $AB = BA$

2) Associative law:
   i) $A+(B+C) = (A+B)+C$
   ii) $A(BC) = (AB)C$

3) Distributive law: $A(B+C) = AB+AC$

4) AND laws:
   i) $A \cdot 0 = 0$
   ii) $A \cdot 1 = A$
   iii) $A \cdot A = A$
   iv) $A \cdot \bar{A} = 0$

5) OR laws:
   i) $A + 0 = A$
   ii) $A + 1 = 1$
   iii) $A + A = A$
   iv) $A + \bar{A} = 1$

6) Inversion law:

i) $\bar{\bar{A}} = A$

7) $A + AB = A$

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

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

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

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

* DUALITY THEOREM:

- The duality theorem states that -
  "The dual of an algebraic expression can be obtained by replacing binary 1 by binary 0 and by interchanging the AND and OR operators."

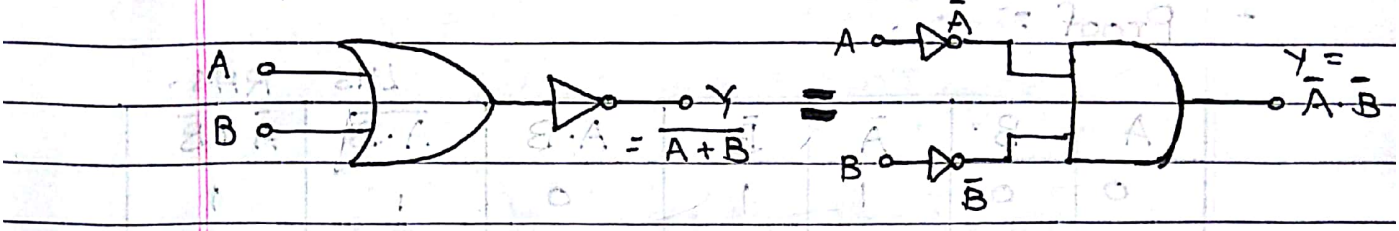
- Consider $(A+B)(A+C) = A + BC$ is given expression. Then the dual of the expression is $AB + AC = A(B+C)$

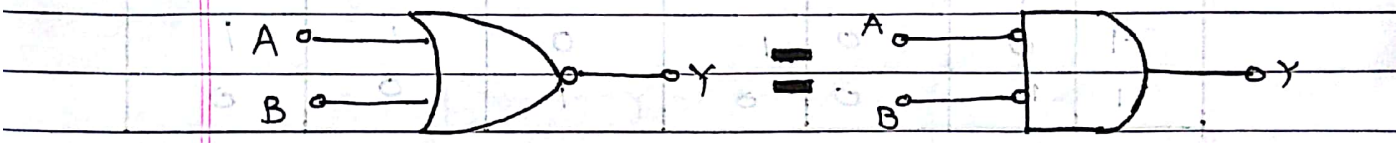
- Duality theorem can be used for generating new expressions from the boolean expressions.

* DE MORGAN'S THEOREM:

- De Morgan's First Theorem: $\overline{A+B} = \bar{A} \cdot \bar{B}$

- It states that complement of sum is equal to the product of complements.





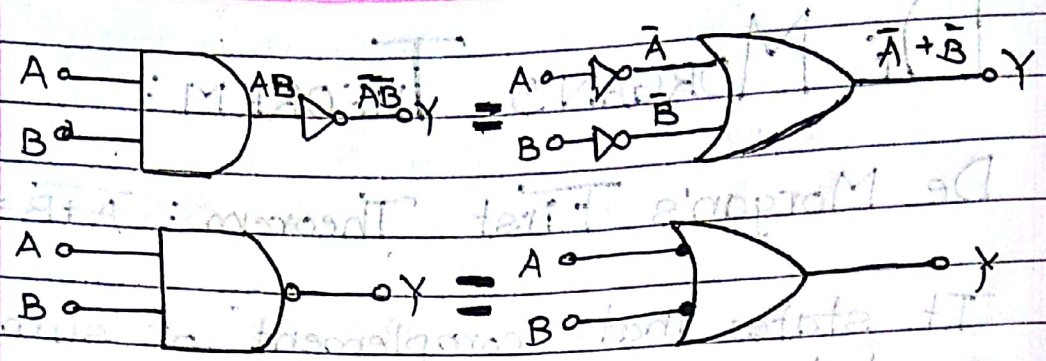
- Proof:

A	B	$\bar{A}$	$\bar{B}$	A+B	(LHS) $\overline{A+B}$	(RHS) $\bar{A} \cdot \bar{B}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Hence, LHS = RHS

- De Morgan's Second Theorem: $\overline{A \cdot B} = \bar{A} + \bar{B}$

- It states that the complement of product is equal to the sum of complements.



- Proof:

					LHS	RHS
A	B	$\bar{A}$	$\bar{B}$	$A \cdot B$	$\overline{A \cdot B}$	$\bar{A}+\bar{B}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

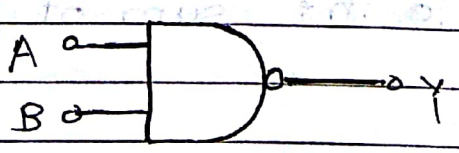
Hence,

LHS = RHS.

* UNIVERSAL Gates:

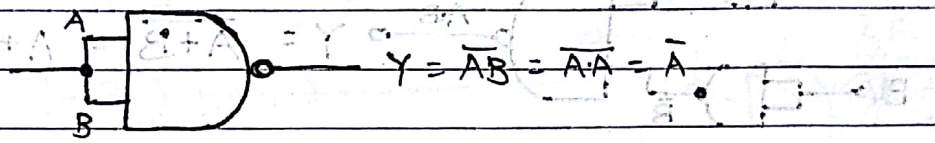
- The NAND and NOR gates are known as universal gates because any logic function and any gate can be implemented using NAND and NOR gates.

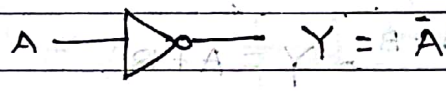
1. NAND as a universal gate:



1) NOT Gate :

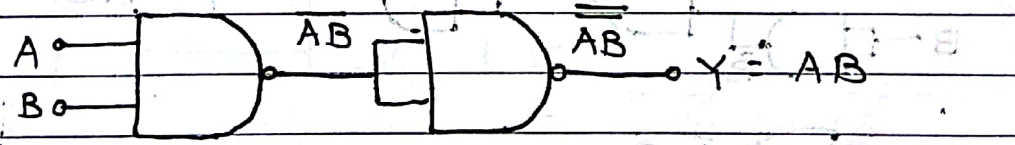
- NOT gate can be constructed using NAND gate by tying all its inputs together.
- The expression of NOT gate is $Y = \bar{A}$
  The expression of NAND gate is $Y = \overline{AB}$
  So, $A = B$
  $\therefore Y = \overline{A \cdot A} = \bar{A}$ = NOT gate

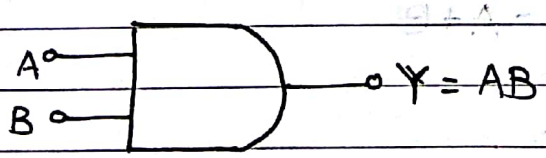




2) AND Gate :

- AND gate is generated by inverting the output of NAND gate
- Expⁿ for AND gate : $Y = AB = \overline{\overline{AB}}$
  Expⁿ for NAND gate : $Y = \overline{A \cdot B}$





3) OR Gate:

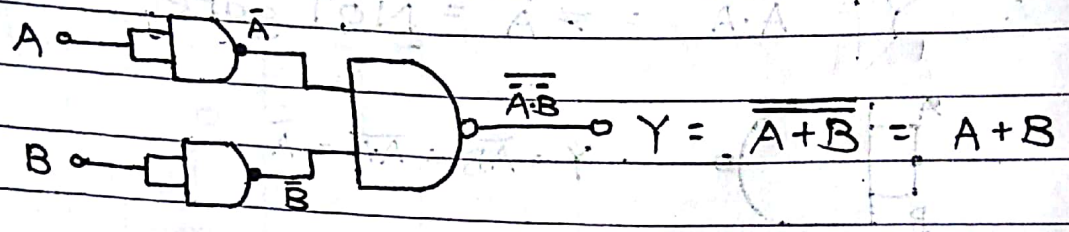
- Exp^n for OR gate : $Y = A + B$

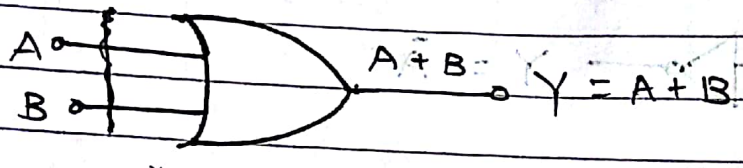
$= \overline{\overline{A+B}}$

Applying De Morgan's first theorem:

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

- Exp^n for NAND gate : $Y = \overline{A \cdot B}$

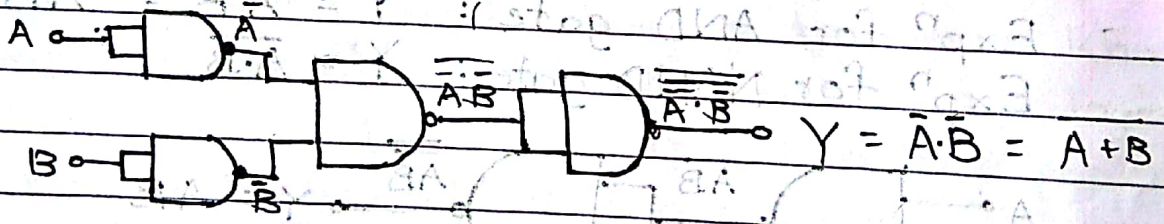


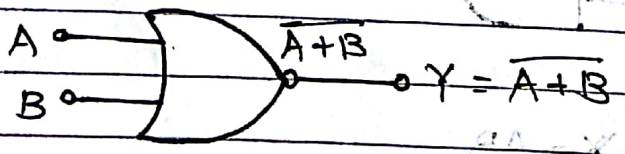


4) NOR Gate:

- Exp^n for NOR gate : $Y = \overline{A+B} = \bar{A} \cdot \bar{B} = \overline{\bar{A} \cdot \bar{B}}$

Exp^n for NAND gate : $Y = \overline{A \cdot B}$





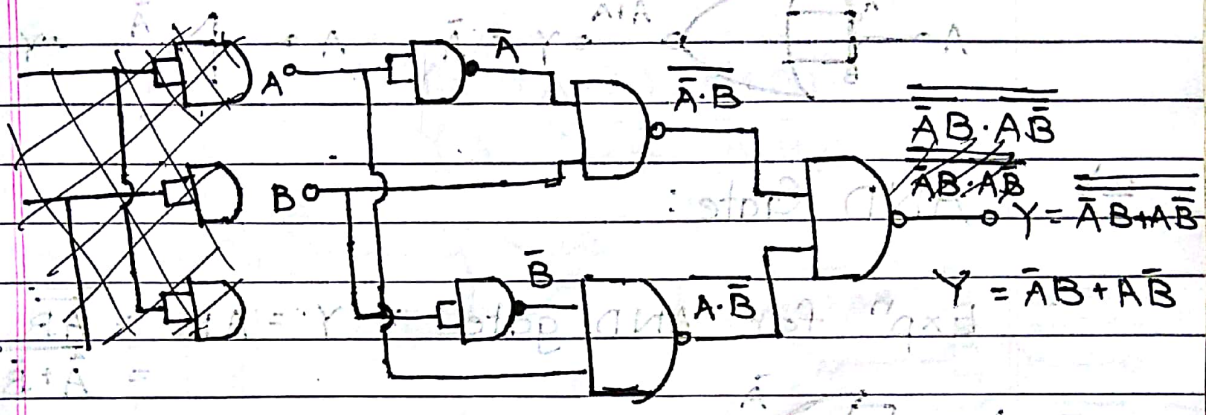
5) EX-OR Gate :

- Expression for EX-OR gate is :

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

$$\text{i.e. } Y = \overline{\bar{A}B + A\bar{B}}$$

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

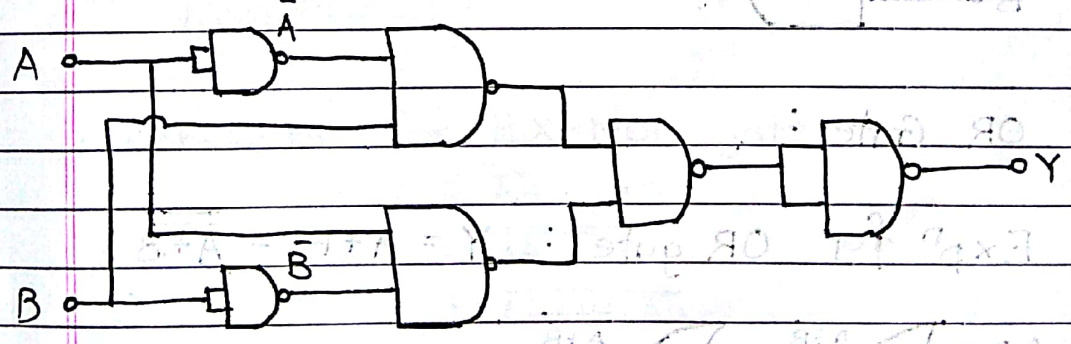


6) EX-NOR Gate :

- Expression for Ex-OR gate is:

$$Y = \bar{A}\bar{B} + AB \qquad \overline{\overline{\bar{A}\bar{B}} \cdot \overline{AB}}$$

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



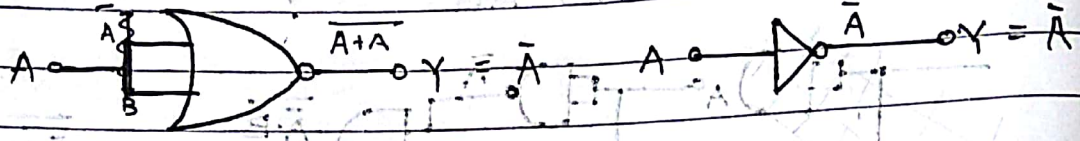
2. NOR as a Universal Gate:

1) NOT Gate:

- Expⁿ for NOT gate is : $Y = \bar{A}$
- Expⁿ for NOR gate is : $Y = \overline{A+B}$

If both I/P are A

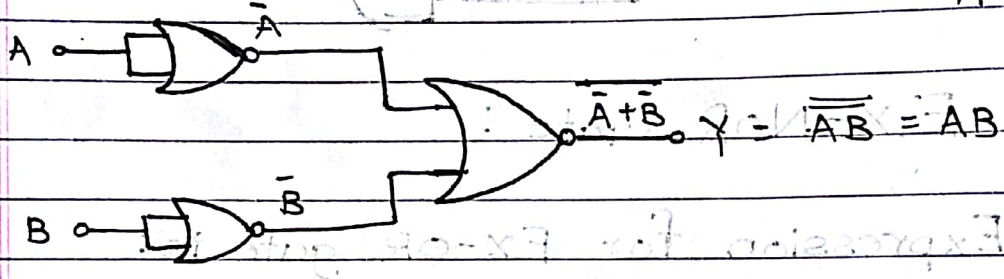
$\therefore Y = \overline{A+A} = \bar{A}$

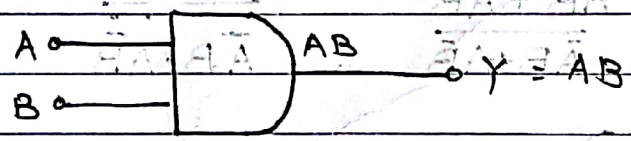


2) AND Gate:

- Expⁿ for AND gate : $Y = AB = \overline{\overline{AB}}$

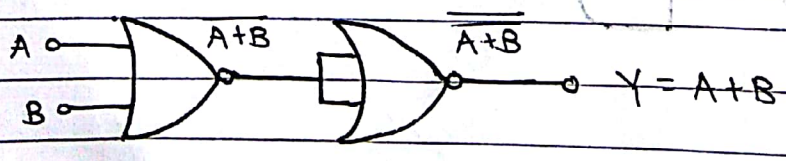
$= \bar{A} + \bar{B}$

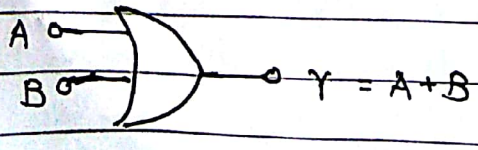




3) OR Gate:

- Expⁿ for OR gate : $Y = A+B = \overline{\overline{A+B}}$





4) EX-OR Gate

Expⁿ for EX-OR gate : $Y = \bar{A}B + A\bar{B}$

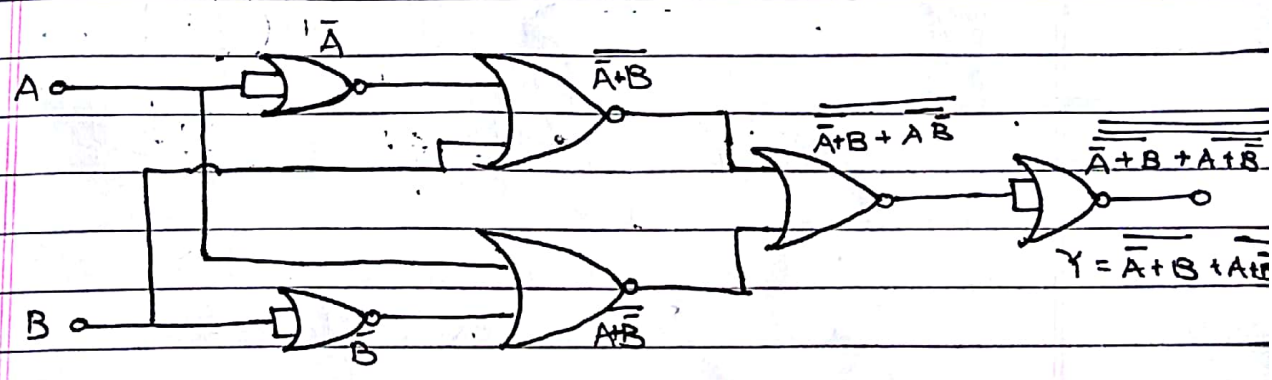
$$Y = \overline{\bar{A}B + A\bar{B}} = \overline{\bar{A}B} \cdot \overline{A\bar{B}}$$

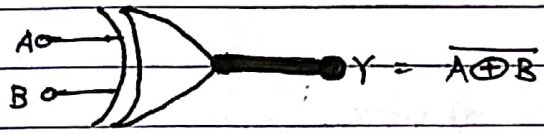
$$\overline{\bar{A}B} = \bar{\bar{A}} + \bar{B} = A + \bar{B}$$

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

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

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





5) EX-NOR Gate :

- Expression for EX-NOR gate :

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

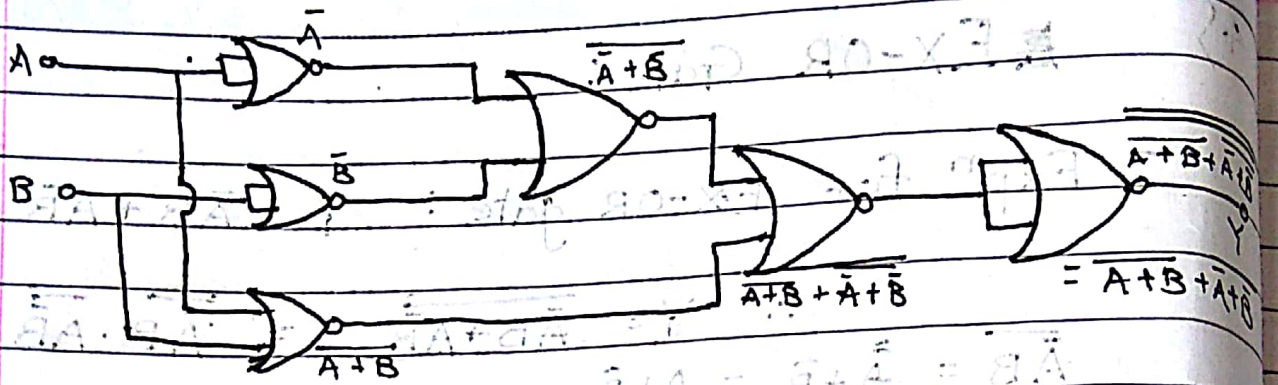
$$= \overline{\bar{A}\bar{B} + AB}$$

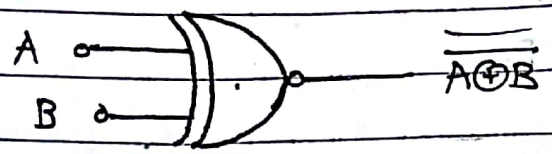
$$= \overline{(A+B) \cdot \bar{A}\bar{B}}$$

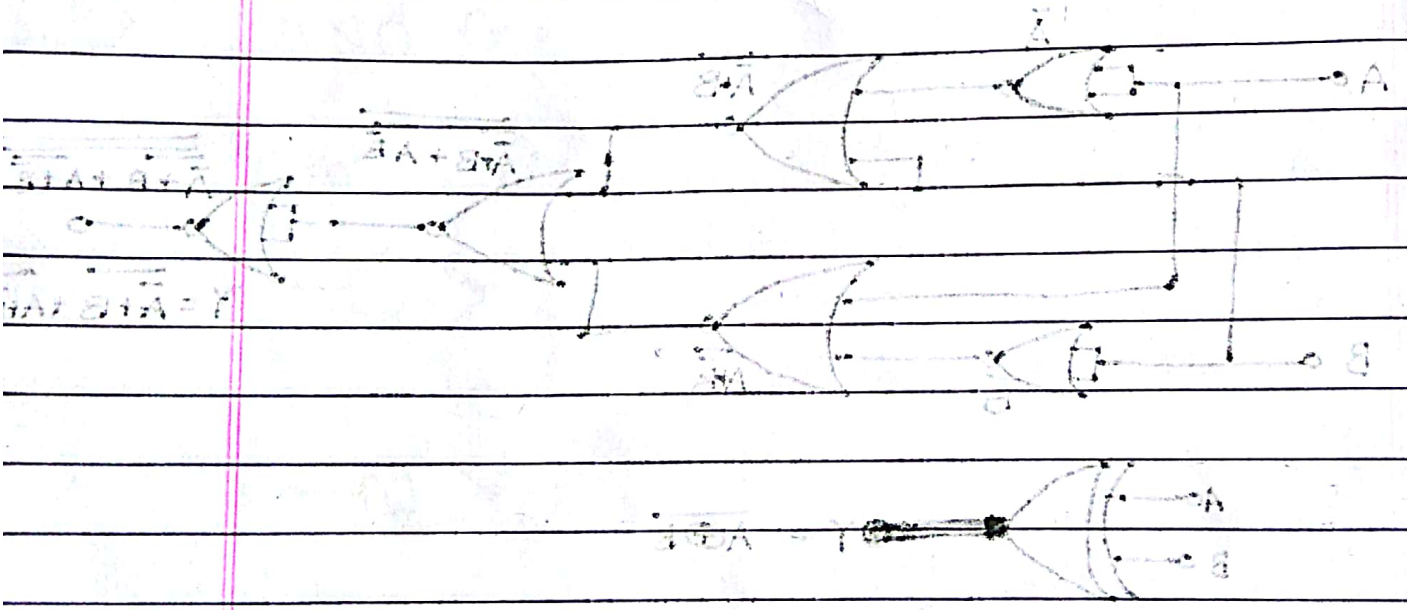
$$= (A+B)(\bar{A}+\bar{B})$$

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

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







3. Combinational Logic Circuits

- Boolean algebra and De Morgan's laws are used for simplifying the logical expressions. It can be implemented using logic gates.
- The no. of gates required can be reduced by simplifying the logical expression.
- The circuits made by using multiple logic gates, where the o/p is based on the combination of input gates are known as combinational logic circuits.
- In order to design a combinational circuit, it is necessary to specify the following requirements of combinational circuits:

1) A set of statements
2) Boolean expressions
3) Truth table

- There are two unique representations for each boolean function:

1) Canocial or standard sum of products form
2) Canocial or standard product of sums form:

1. Standard Sum of Products form (SOP):

- Standard SOP is only the product expression that consists all input variables in complemented or uncomplemented form
- An I/P variable in complemented form has ~~variable~~ value 0 while in uncomplemented form has value 1.

- A sum of product (SOP) term is a group of product terms (ANDed terms) ORed together.

- Each individual term in standard SOP form is called minterm.

Input			Standard product form	Minterm integration
A	B	C		
0	0	0	$\bar{A}\bar{B}\bar{C}$	m_0
0	0	1	$\bar{A}\bar{B}C$	m_1
0	1	0	$\bar{A}B\bar{C}$	m_2
0	1	1	$\bar{A}BC$	m_3
1	0	0	$A\bar{B}\bar{C}$	m_4
1	0	1	$A\bar{B}C$	m_5
1	1	0	$AB\bar{C}$	m_6
1	1	1	ABC	m_7

- For examples:

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

2) $Y = \bar{X}Z + X\bar{Z} + XZ$

2. Canonical SOP:

- When any switching expression is to be expressed in canonical SOP form, then each term of the expression should be examined and if it is a minterm, then it should be kept as it is.

- If any particular variable does not occur in any term, then for each variable A/B/C which do not occur, multiply that term by $(A+\bar{A})$, $(B+\bar{B})$ or $(C+\bar{C})$ as required.
- Multiply all the terms & represent the repeated terms only once.

Ex. Find the canonical SOP form for the expression:

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

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

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

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

$$Y = \Sigma m(7,6,1,5,2,0)$$

$$Y = \Sigma m(0,1,2,5,6,7)$$

3. Standard Product of Sum Form (POS):

- Standard product of sum form of function means a function which uses standard sum terms.
- A POS is a group of sum terms (ORed terms) ANDed together.

Input			Standard sum term	Maxterm designated
A	B	C		
0	0	0	$A+B+C$	M_0
0	0	1	$A+B+\bar{C}$	M_1
0	1	0	$A+\bar{B}+C$	M_2
0	1	1	$A+\bar{B}+\bar{C}$	M_3
1	0	0	$\bar{A}+B+C$	M_4
1	0	1	$\bar{A}+B+\bar{C}$	M_5
1	1	0	$\bar{A}+\bar{B}+C$	M_6
1	1	1	$\bar{A}+\bar{B}+\bar{C}$	M_7

- For example:

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

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

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

4. Canonical POS:

- When any switching expⁿ has to be expressed in canonical form, if a certain variable does not occur in any of the sum terms, then for each variable A, B or C, add $A\bar{A}$, $B\bar{B}$ or $C\bar{C}$ as required.

Ex. Convert the following function into canonical form: $Y = \cancel{A\bar{B}+B\bar{C}}\ (\bar{A}+B)(B+\bar{C})+(\bar{A}+C)$

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

$=$

* Methods for simplifying the 'boolean ~~expression~~ equations :

1. Algebraic method
2. Karnaugh-map simplification
3. Quine-Mccluskey method

1. Algebraic Method :

- The algebraic methods utilize the Boolean laws & De Morgan's theorems to simplify the given boolean expression.

- Drawbacks :

1) It is a time consuming method.
2) It uses a trial & error, manipulative method.

* KERNAUGH MAP (K-Map) :

- Kernaugh Map is a graphical method used to simplify logic eq's or to convert a truth table to its corr. logic circuit in a simple & systematic manner.

- The K-map is made up of squares. Each square represents one term. It is a systematic method for combining terms & obtaining the minimal expression.

- Each 'n' variable map consists of 2^n cells or squares. Thus; 3-variable map has total 8 cells while 4-variable map contains $2^4 = 16$ cells.

- Each cell in a K-map corr. to the particular combination of input variables.

• 2-variable K-map:

A \ B	$\bar{B}$	B
$\bar{A}$	m_0	m_1
A	m_2	m_3

$m_0 = \bar{A}\bar{B}$ $m_1 = \bar{A}B$
$m_2 = A\bar{B}$ $m_3 = AB$

• 3-variables K-map:

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

	$\bar{C}$	OR	C
$\bar{A}\bar{B}$	m_0		m_1
$\bar{A}B$	m_2		m_3
AB	m_6		m_7
$A\bar{B}$	m_4		m_5

• 4-variable K-map:

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	m_0	m_1	m_3	m_2
$\bar{A}B$	m_4	m_5	m_7	m_6
AB	m_{12}	m_{13}	m_{15}	m_{14}
$A\bar{B}$	m_8	m_9	m_{11}	m_{10}

CD \ AB	$\bar{A}\bar{B}$	$\bar{A}B$	$A\bar{B}$	AB
OR $\bar{C}\bar{D}$	m_0	m_1	m_3	m_2
$\bar{C}D$	m_4	m_5	m_7	m_6
CD	m_{12}	m_{13}	m_{15}	m_{14}
$C\bar{D}$	m_8	m_9	m_{11}	m_{10}

☆ Plotting a Boolean Expression:

- Following are the steps to plot a boolean expression on the K-map:

1) Transform the boolean expression to a SOP expression.

2) Fill in the appropriate cells of the K-map by placing a '1' in each cell corr. to a ~~sum~~ term in SOP expression.

x. Plot the given exp^n ~~truth table~~ on K-map.

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

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

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

→ $\bar{A}\bar{B}\bar{C} = m_0$

→ $\bar{A}\bar{B}C = m_1$

→ $\bar{A}BC = m_3$

K-map representation

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

☆ Simplification using K-map

- After plotting the logic function / truth table on a K-map, we use the grouping technique to simplify the logic expression.
- To group means to combine the terms in adjacent cells.
- If adjacent 1's are grouped, the result of simplification is SOP form.
- If adjacent 0's are grouped, the result of simplification is POS form.
- There are various types of combinations for simplification. They are:

1. The pair
2. The quad
3. The octet.

1. The pair

- A pair is a group of two adjacent 1's in a K-map. A pair leads to the elimination of one variable in SOP form.

Ex:

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

i.e. $Y = \bar{A}BC + A\bar{B}C$

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

Here, in the pair $\bar{A}$ & C are common. So, variable B is eliminated and the answer is $\bar{A}C$.

2. The quad:

- A group of four adjacent 1's on K-map is known as a quad.
- A quad eliminates two variables.

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$			1	1
$\bar{A}B$				
AB				
$A\bar{B}$			1	1

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

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

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

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

$Y = \bar{B}C$

Here, in the quad $\bar{B}$ & C are common. So, Variables A & D are eliminated to give $Y = \bar{B}C$.

3. The octet:

- An octet is a group of eight adjacent 1's.
- It leads to the elimination of three variables.

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1		
$\bar{A}B$	1	1		
AB	1	1		
$A\bar{B}$	1	1		

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

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

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

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

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

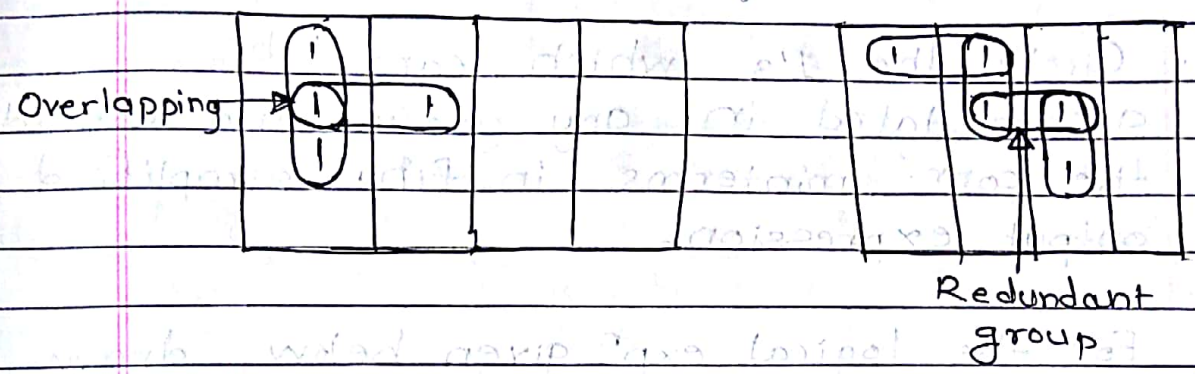
$$= \bar{C}(A+\bar{A})$$

$$\boxed{Y = \bar{C}}$$

★ Rules for K-map Simplification :

1. A K-map can have more than one pair, quad or octet.
2. The 1's left without groups must also be encircled.
3. While grouping, it may be remembered that overlapping of groups is allowed. i.e. two or more groups can have one or more 1's in common.

4. Redundancy is not allowed i.e. a group whose all 1's are overlapped by other groups.



5. For simplifying SOP equations, groups should include only cells containing 1's and for POS eq^{n}s, groups should contain only 0's.

6. Groups can be horizontal, vertical but not diagonal.

7. Groups should be as large as possible.

8. Groups can be wrapped around the table.

☆ Simplification of SOP Expression using K-map

1. Draw the K-map. Depending on the no. of variables, an eq^n with n variables will require a K-map with 2^n cells.
E.g., 3 variables K-map has $2^3 = 8$ cells.

2. Plot the K-map from the truth table or SOP expression. The minterms that are present are represented by a 1 in the K-map cell.

3. Create group of 1's. Try to form group to eliminate maximum variables.

4. Circle the 1's which cannot be accomodated in any group and consider the corr. minterms in final simplified output expression.

Ex. For the logical exp^n given below, draw the K-map & obtain the simplified logical expression, $Y = \Sigma m(1, 5, 7, 9, 11, 13, 15)$

⟶ $Y = \Sigma m(1, 5, 7, 9, 11, 13, 15)$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	1	1	0
$A\bar{B}$	0	1	1	0
$A\bar{B}$	0	1	1	0

$Y = \bar{C}D + BD + AD$

☆ POS Simplification using K-map

- POS eq^{n}s contain maxterm '0' in uncomplemented form, and '1' in complemented form.

- While plotting a K-map for simplifying POS equations, we represent a maxterm by writing a '0' in the cell.

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

A \ B	B	$\bar{B}$
A	M_0	M_1
$\bar{A}$	M_2	M_3

2 - variable K-maps

A \ BC	BC 00	B$\bar{C}$ 01	$\bar{B}\bar{C}$ 11	$\bar{B}$C 10
A 0	A + B + C M_0	A + $\bar{B}$ + C M_1	A + $\bar{B}$ + $\bar{C}$ M_3	A + $\bar{B}$ + C M_2
$\bar{A}$ 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

3 - variable K-map

AB \ CD	CD 00	C$\bar{D}$ 01	$\bar{C}\bar{D}$ 11	$\bar{C}$D 10
AB 00	A + B + C + D M_0	A + B + C + $\bar{D}$ M_1	A + B + $\bar{C}$ + $\bar{D}$ M_3	A + B + $\bar{C}$ + D M_2
A$\bar{B}$ 01	A + $\bar{B}$ + C + D M_4	A + $\bar{B}$ + C + $\bar{D}$ M_5	A + $\bar{B}$ + $\bar{C}$ + $\bar{D}$ M_7	A + $\bar{B}$ + $\bar{C}$ + D M_6
$\bar{A}\bar{B}$ 11	$\bar{A}$ + $\bar{B}$ + C + D M_{12}	$\bar{A}$ + $\bar{B}$ + C + $\bar{D}$ M_{13}	$\bar{A}$ + $\bar{B}$ + $\bar{C}$ + $\bar{D}$ M_{15}	$\bar{A}$ + $\bar{B}$ + $\bar{C}$ + D M_{14}
$\bar{A}$B 10	$\bar{A}$ + B + C + D M_8	$\bar{A}$ + B + $\bar{C}$ + $\bar{D}$ M_9	$\bar{A}$ + B + $\bar{C}$ + $\bar{D}$ M_{11}	$\bar{A}$ + B + $\bar{C}$ + D M_{10}

4 - variable K-map

- Simplification of POS equation:

1. Draw the K-map.
2. Enter a 0 corr. to every maxterm in the POS
3. Fill up the remaining cells in the K-map with 1
4. Group the adjacent 0's in pairs, quads or octets.
5. Write the simplified eqn.

Ex. Obtain the minimal expn for the function given below:

$f(A, B, C, D) = \pi(1, 4, 6, 8, 9, 10, 12, 14, 15)$

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

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

✳ ADDERS:

- Addition is the most fundamental operation. A simple addition consists of four possible elementary operations as follows:

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

binary add" in the first three operations results in a sum of one digit. But, in the last binary addition, the sum consists of two binary digits.

- In this, the lower significant bit is called sum & the most significant bit is called a carry bit.

- "The combinational circuit which is used to add binary digits is called an adder."

- It is classified into two types:
  1. Half adder
  2. Full adder

1. Half Adder:

- "A binary half adder is a combinational logic circuit which adds two bits of binary data, producing a sum bit and a carry bit as the two output signals."

- It has A & B as inputs and sum & carry as outputs.

- Truth table:

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

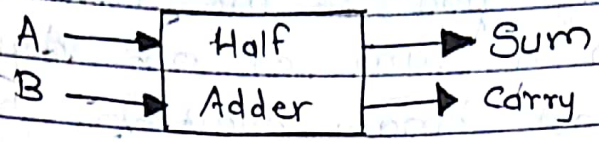
	$\bar{B}$	B
$\bar{A}$	0	0
A	0	(1)

Carry

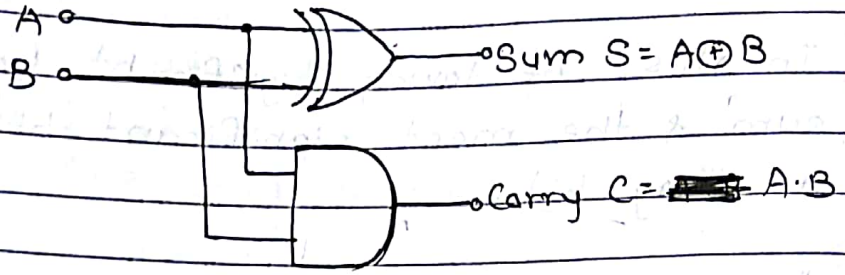
Sum

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

- Diagram:



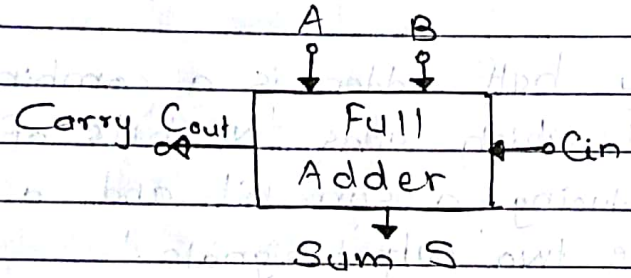
- Logical implementation:



2. Full Adder:

- "A full adder is a combinational circuit that performs sum of three input bits."

- This circuit has three inputs A, B, Cin and two outputs S & Cout.



- In this, A & B are the two bits to be added. The third input Cin represents the carry from the previous lower significant position. The two outputs are sum S and carry Cout.

- Truth table :

Input			output	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Sum

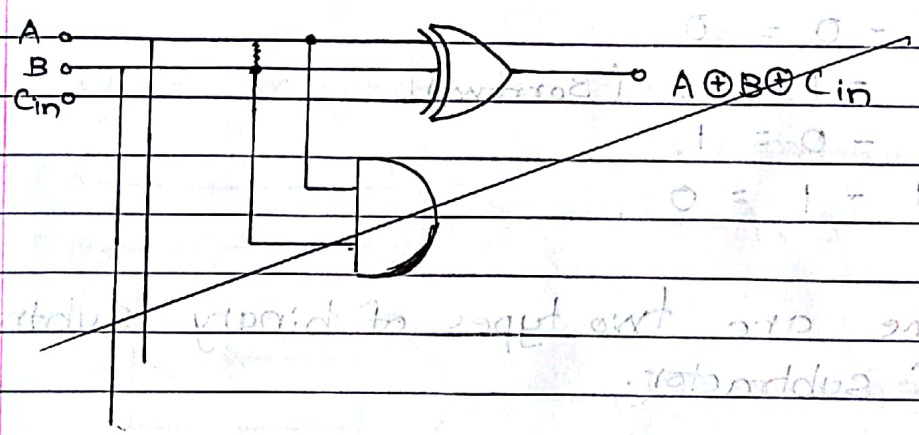
	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$
$\bar{A}$	0	(1)	0	(1)
A	(1)	0	(1)	0

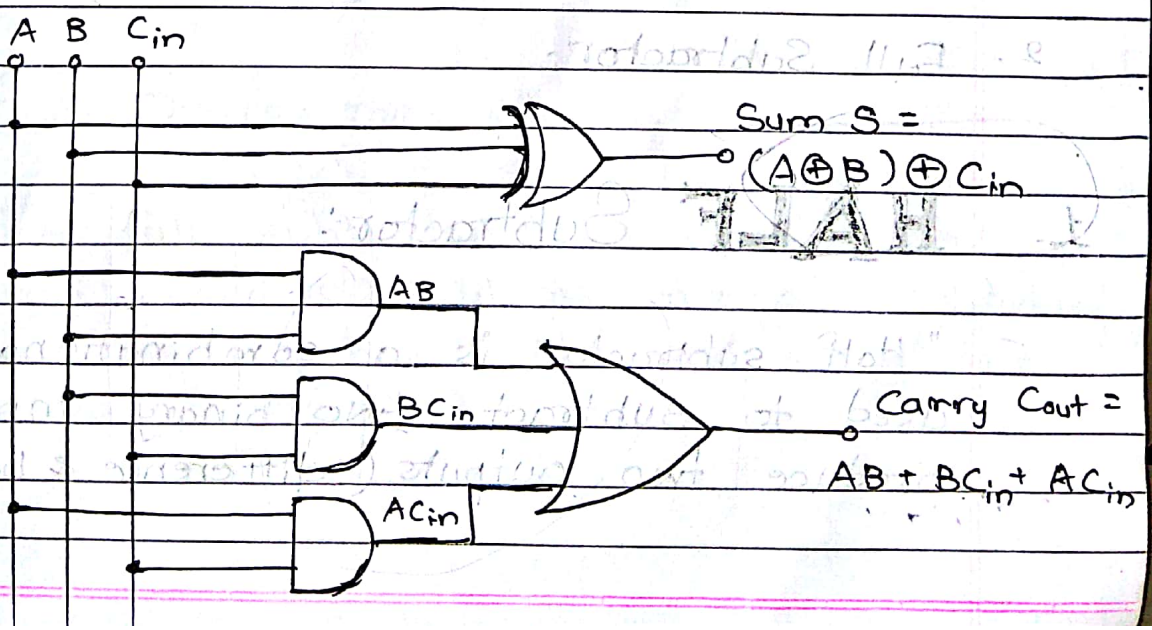
Carry

	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$
$\bar{A}$	0	0	1	0
A	0	1	1	1

$\therefore S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$

- Logical implementation : $C = AB + BC_{in} + AC_{in}$





- Applications:
1. It is used in digital computers
2. It acts as the basic building blocks of BCD adder IC 7483.

* Subtractor:

- The subtraction of two binary numbers can be done by taking the complement of the subtrahend & adding it to the minuend.

- The rules of ~~sb~~ subtraction are as follows:

0 - 0 = 0
0 - 1 = 1 (Borrow)
1 - 0 = 1
1 - 1 = 0

- There are two types of binary subtractors:
1. Half subtractor.
2. Full Subtractor.

1. HALF Subtractor:

- "Half subtractor is a combinational circuit used to subtract two binary inputs to produce two outputs (difference & borrow)."

- Truth table :

Input		Output	
A	B	A - B	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- K-map :

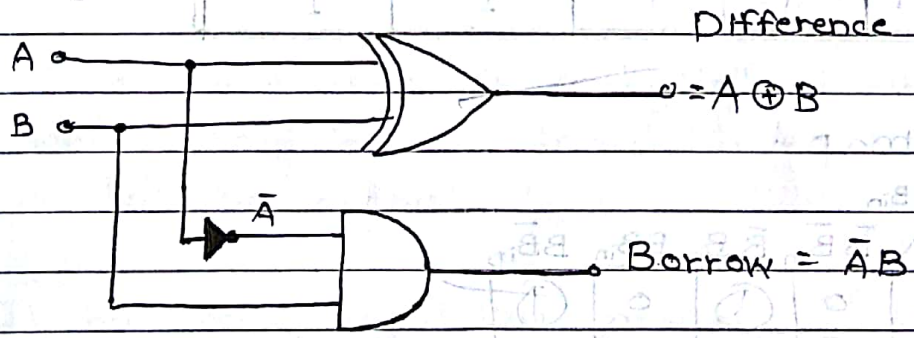
A \ B	$\bar{B}$	B
$\bar{A}$	0	(1)
A	(1)	0

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

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

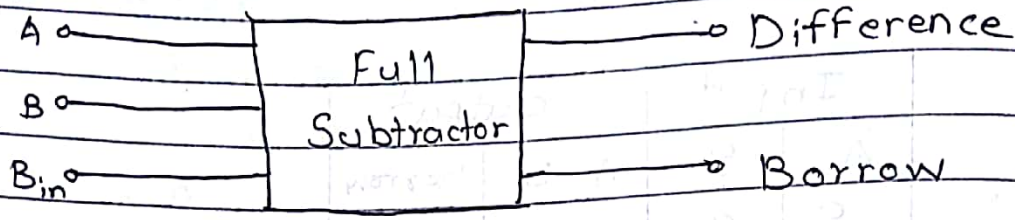
Borrow
$= \bar{A} \cdot B$

- Logical implementation :



2. Full Subtractor :

- "The full subtractor is a combinational circuit used to subtract two bits considering that a 1 has been borrowed from the previous stage."



- Truth table :

Input			output	
A	B	B_{in}	Difference $A-B-B_{in}$	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

- K-map :

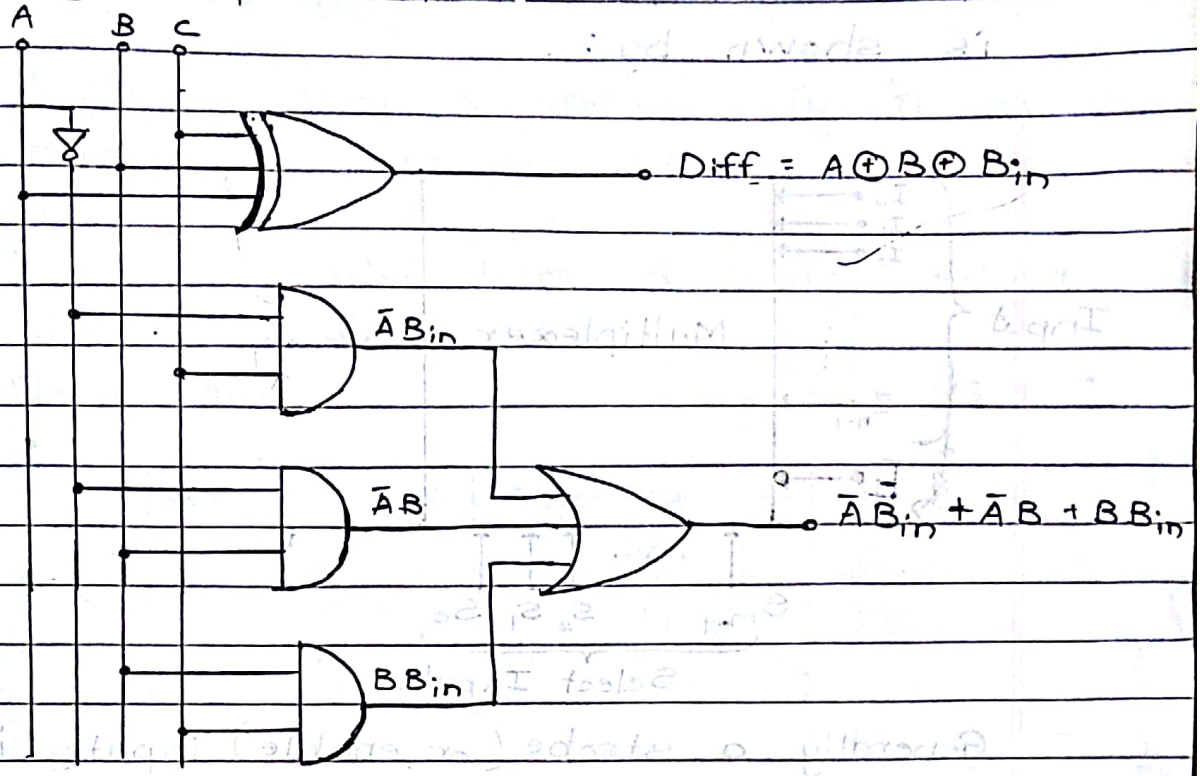
A \ $B\bar{B}_{in}$	$\bar{B}\bar{B}_{in}$	$\bar{B}B_{in}$	BB_{in}	$B\bar{B}_{in}$
$\bar{A}$	0	1	0	1
A	1	0	1	0

Difference = $\bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in} = A \oplus B \oplus$

A \ $B\bar{B}_{in}$	$\bar{B}\bar{B}_{in}$	$\bar{B}B_{in}$	BB_{in}	$B\bar{B}_{in}$
$\bar{A}$	0	1	1	1
A	0	0	1	0

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

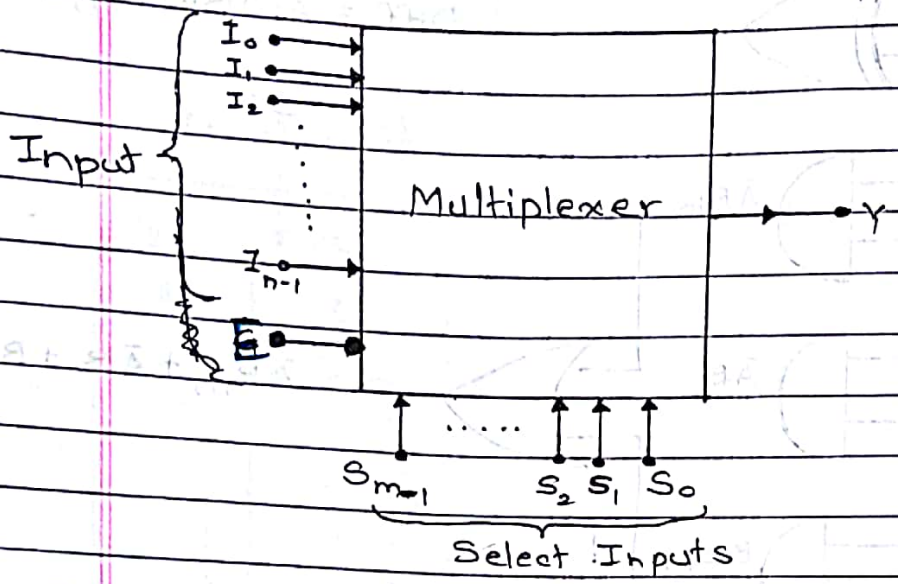
- Logic implementation :



* MULTIPLEXER :

- "The multiplexer is a digital circuit which selects one of the input lines & connect it to the output."
- It has many input lines and one output line.
- The multiplexer is also known as data selector. The selection of desired input is done by means of selection lines.
- Generally, there are 2^m input lines and 'm' selection lines whose bit combinations determine which input is to be selected.

- The functional block diagram for multiplexer is shown by:



- Generally, a strobe (or enable) inputs is incorporated which helps in cascading and it is generally active low which means it performs its intended operation when it is low.

- Standard ICs are available for:
1. 2:1 MUX
2. 4:1 MUX
3. 8:1 MUX
4. 16:1 MUX

- Advantages:
1. Simplification of logic expⁿ is not required.
2. It minimizes the IC package count.
3. Logic diagram is simplified.

1. 2:1 Multiplexer :

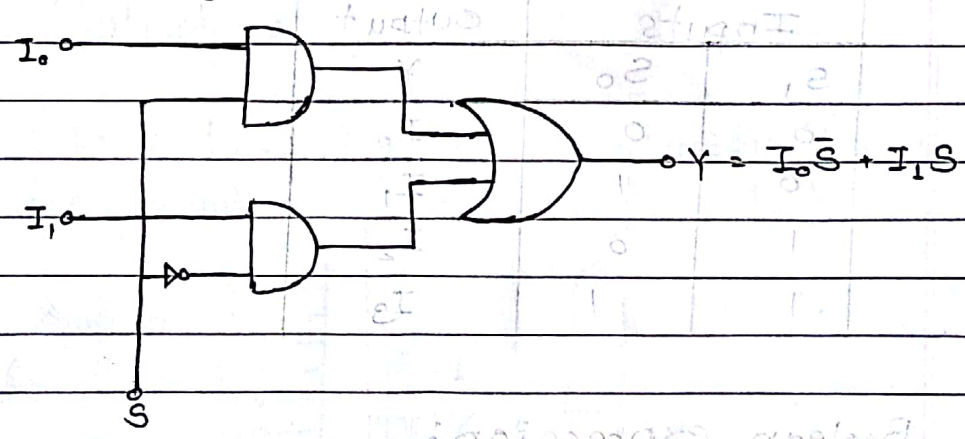
- A 2:1 multiplexer is shown in fig. It has two inputs I_0 & I_1 and one output Y.
- The no. of select lines required is only 1.
- ~~The truth table~~
- Truth table :

Select input	Output selected
S	Y
0	I_0
1	I_1

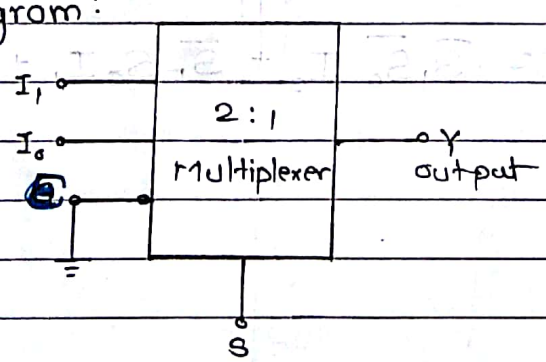
- Logic expression :

$$Y = I_0\bar{S} + I_1S$$

- Logic diagram :



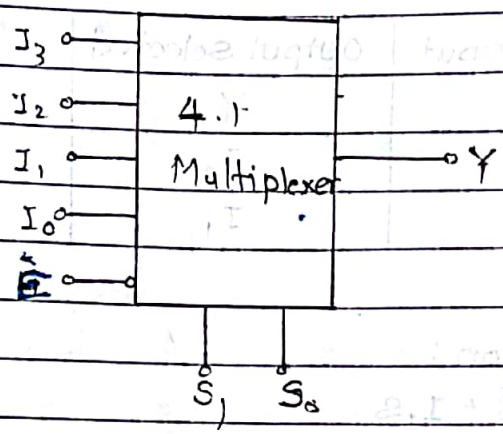
- Block diagram :



2. 4:1 Multiplexer :

- A 4:1 Multiplexer has four inputs I_0, I_1, I_2, I_3 which are selectively transmitted to output Y depending on the select input combinations S_0, S_1

- Block diagram :



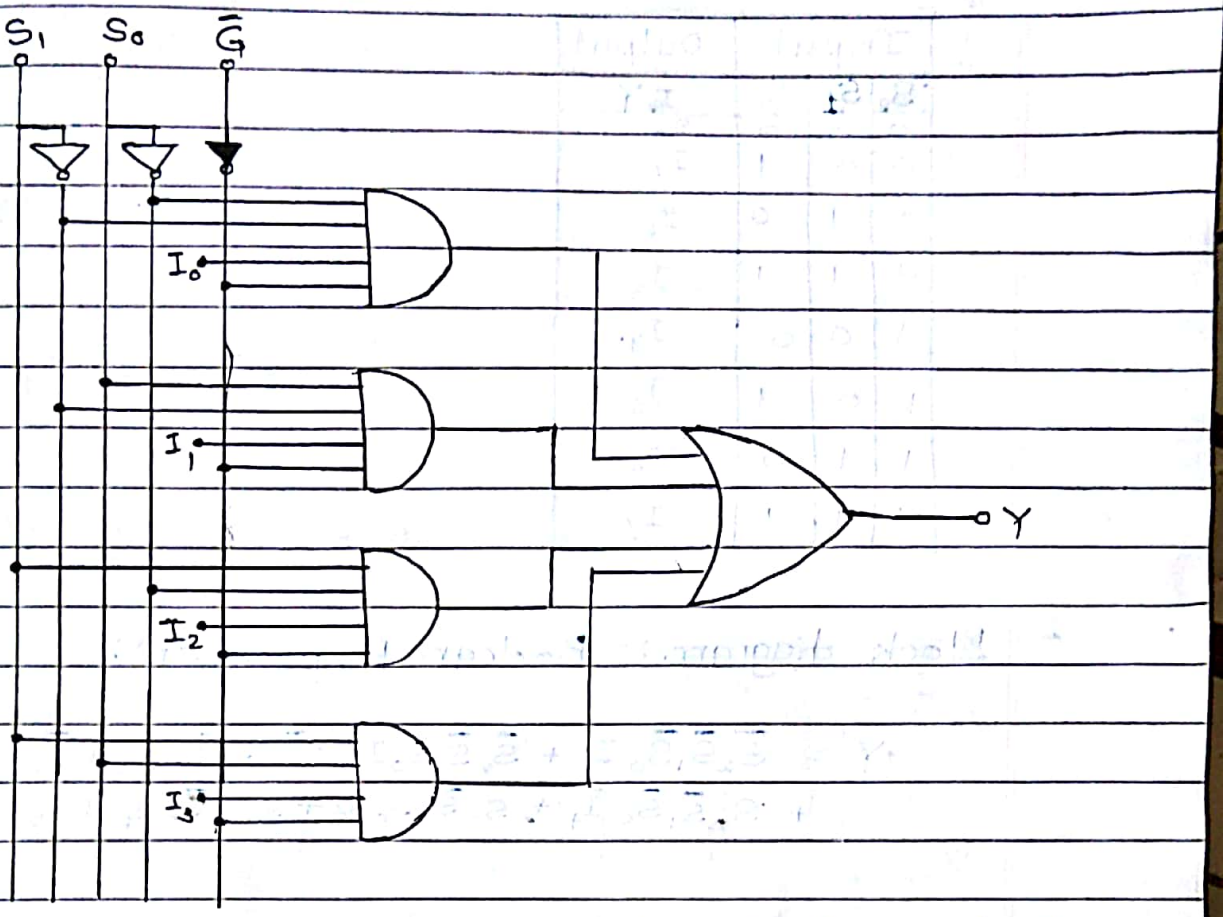
- Truth table :

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

- Boolean expression :

$$Y = \bar{S_1}\bar{S_0} I_0 + \bar{S_1} S_0 I_1 + S_1 \bar{S_0} I_2 + S_1 S_0 I_3$$

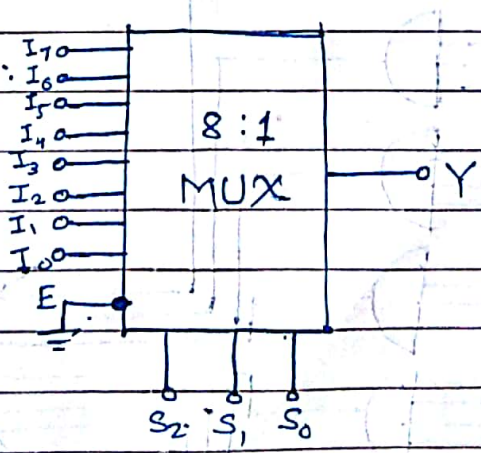
- Logic diagram:



3. 8:1 Multiplexer:

- A 8:1 MUX has 8 inputs & 3 selection lines. It has one output Y.

- Block diagram:



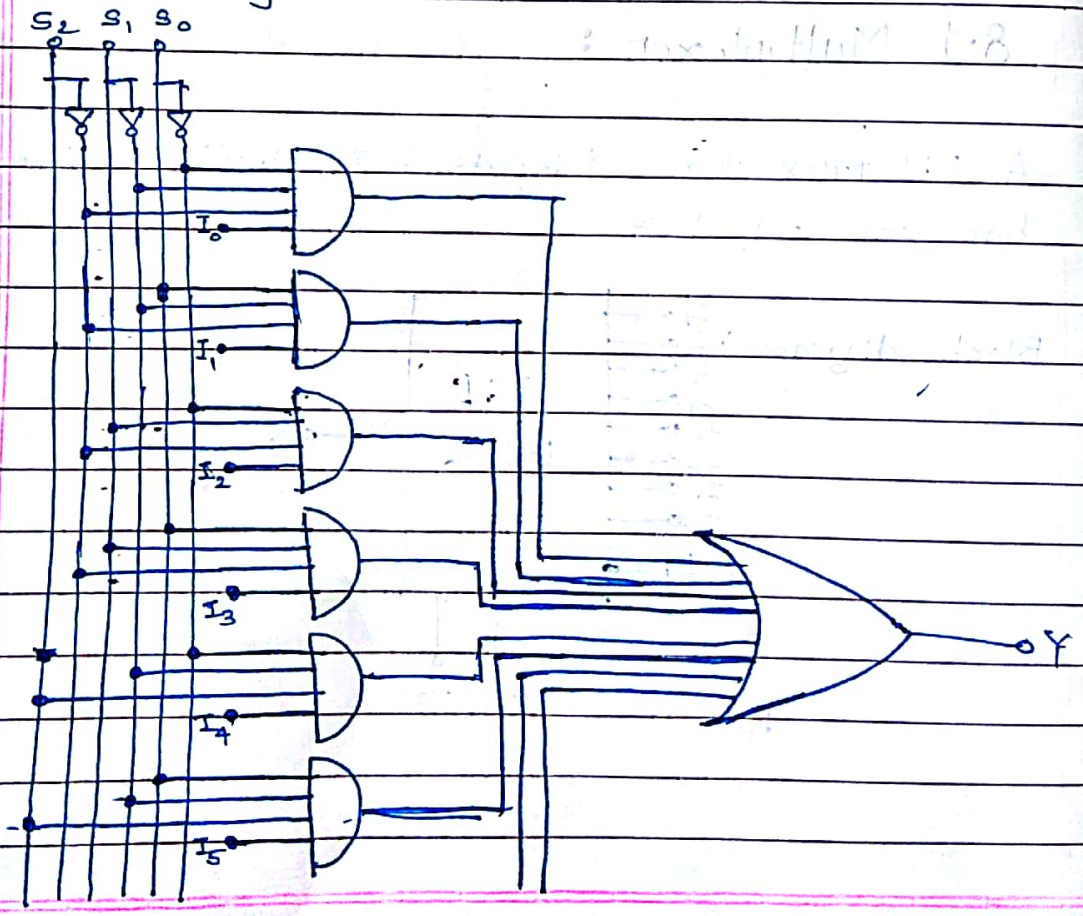
- Truth table:

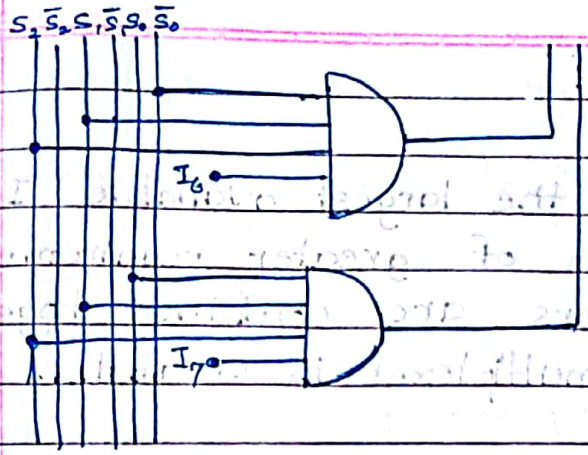
Input			Output
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

- ~~Block diagram:~~ Boolean Expression:

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

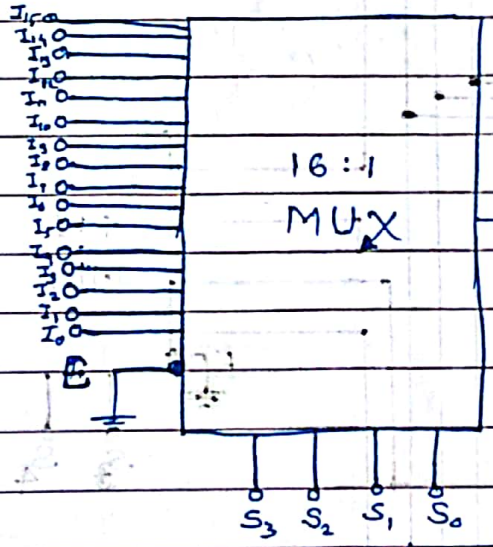
- Logic diagram:





4 16:1 MULTIPLEXER:

- It has 16 inputs & 4 selection lines.
- Block diagram:



- Truth table:

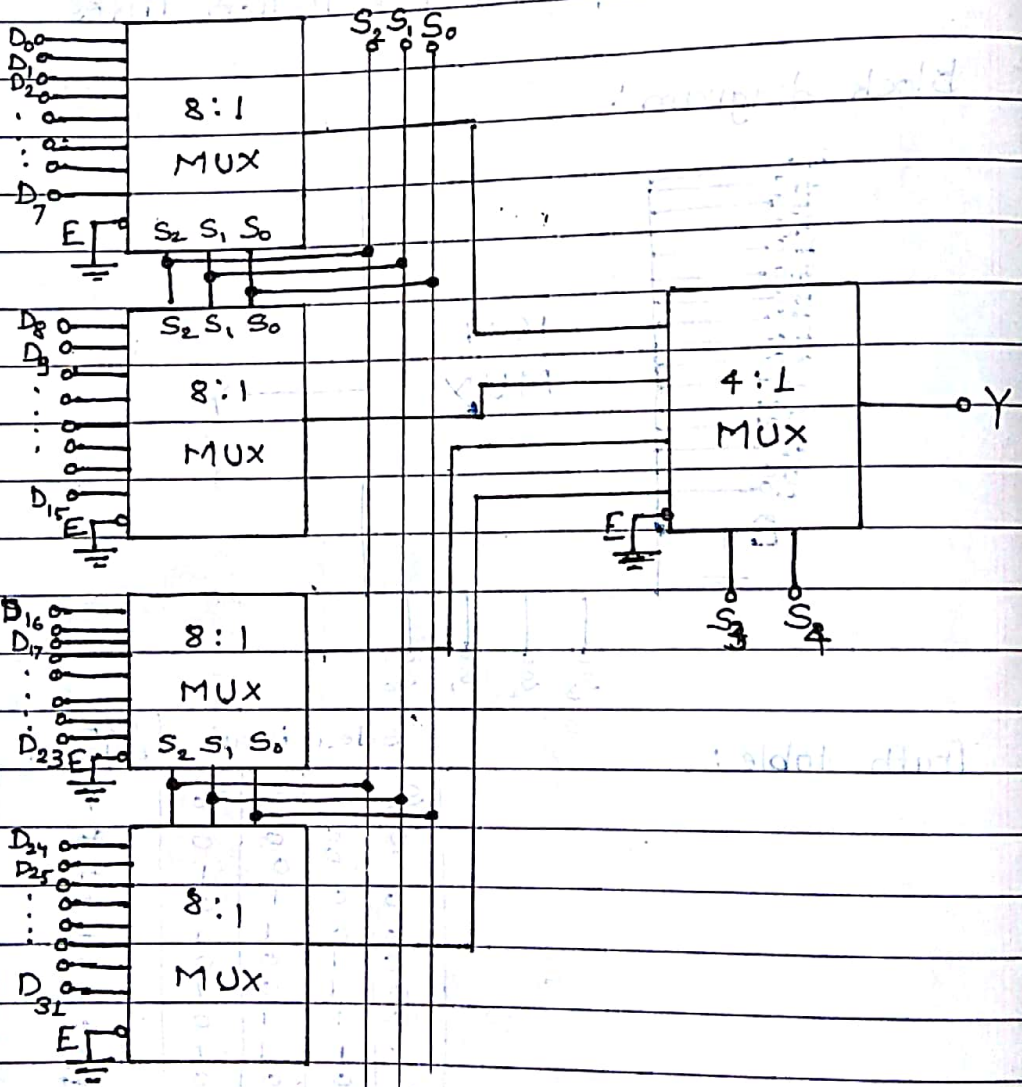
Select input				O/P selected
S_3	S_2	S_1	S_0	Y
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5
0	1	1	0	I_6
0	1	1	1	I_7
1	0	0	0	I_8
1	0	0	1	I_9
1	0	1	0	I_{10}
1	0	1	1	I_{11}
1	1	0	0	I_{12}
1	1	0	1	I_{13}
1	1	1	0	I_{14}
1	1	1	1	I_{15}

* Multiplexer Tree :

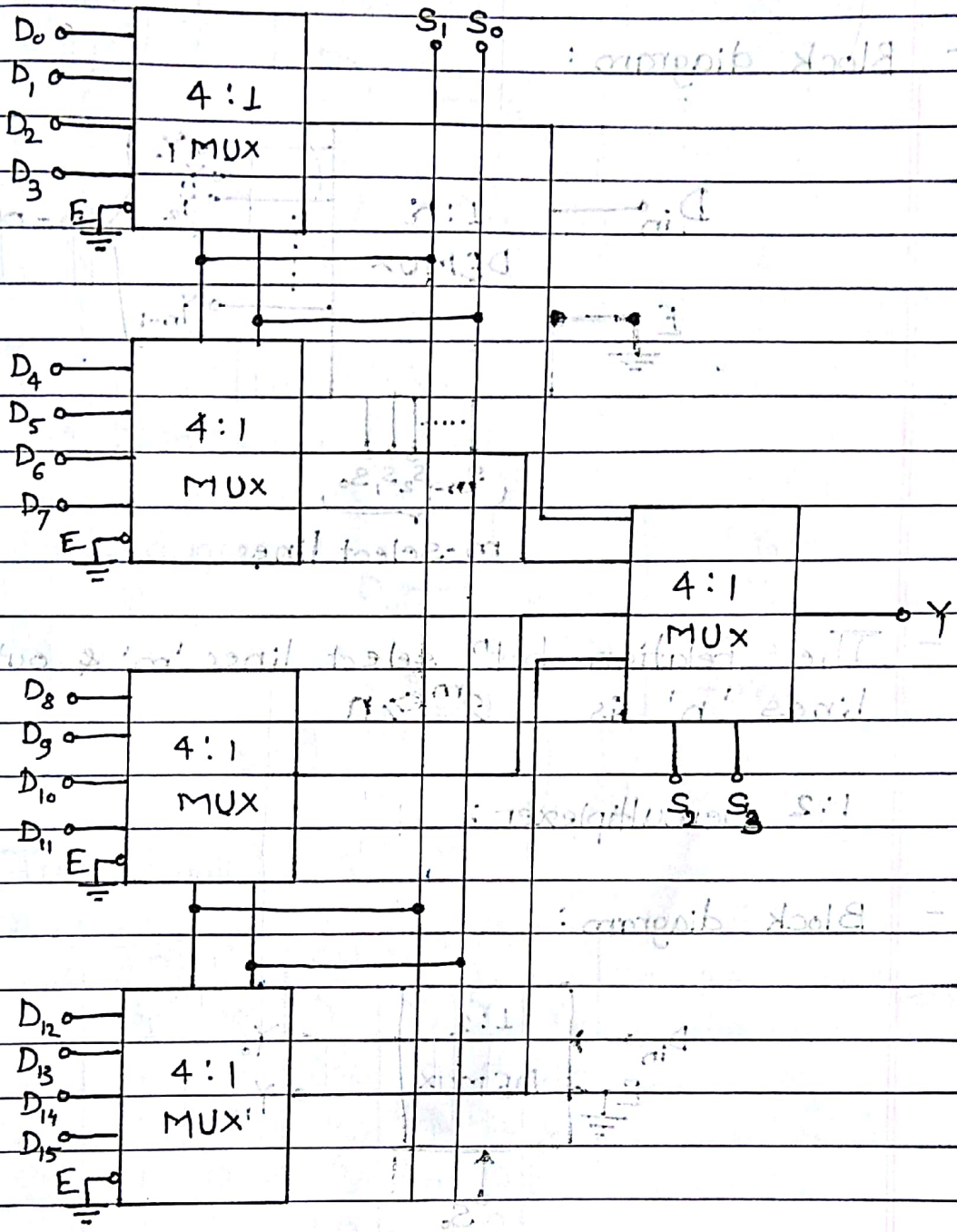
- 16:1 MUX IC is the largest available IC. So, to produce ICs of greater requirements, these multiplexers are combined together and a greater multiplexer is created.

- This circuit is known as multiplexer tree.

1. Design 32:1 MUX using 8:1 MUX

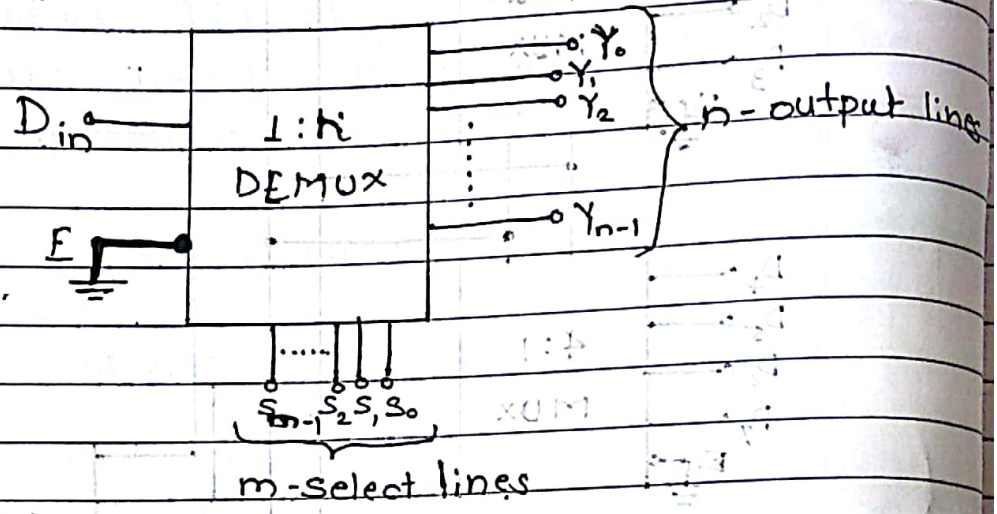


2. Design 16:1 MUX using 4:1 MUX.



* DEMULTIPLEXER:

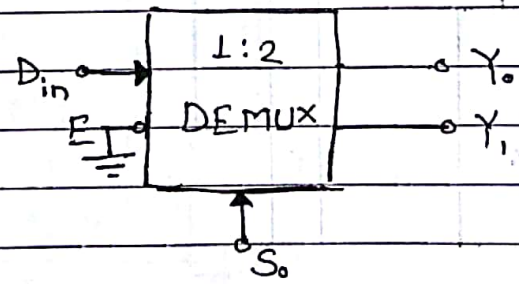
- Block diagram:



- The relation bet^n select lines 'm' & output lines 'n' is $2^m \geq n$

1. 1:2 Demultiplexer:

- Block diagram:

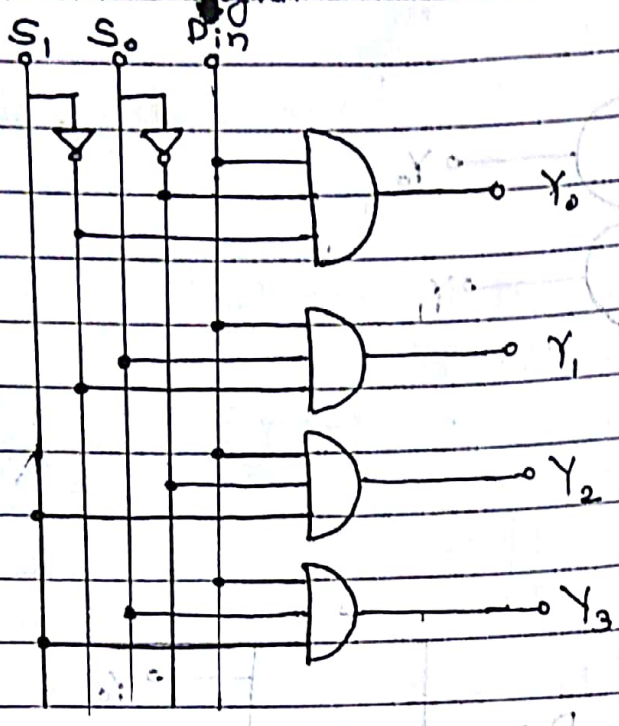


- Truth table:

Input	Output	
S_0	Y_0	Y_1
0	D_{in}	0
1	0	D_{in}

- Logic circuit: $Y_0 = \bar{S_0} D_{in}$ $Y_1 = S_0 \cdot D_{in}$

- Circuit diagram :



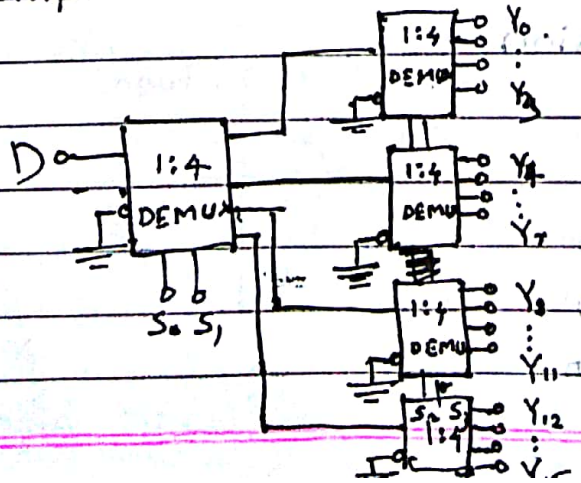
- Similar circuit is for 1:8 & 1:16 DEMUX.

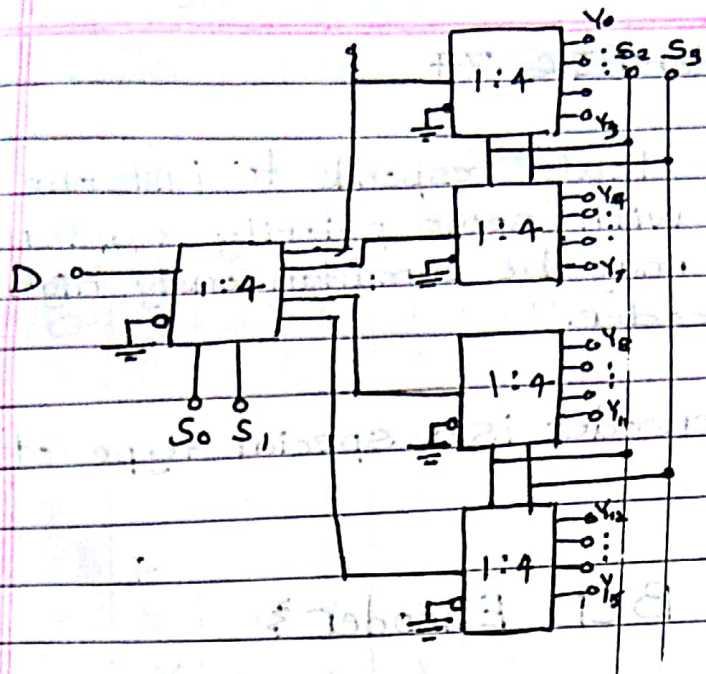
* DEMULTIPLEXER TREE

- For obtaining large output lines, we have to use two or more demultiplexers and ~~have~~ need expansion of a demultiplexer.

- This expansion of demultiplexer is called as demultiplexer tree.

Ex. 1 Implement a 1:16 DEMUX using 1:4 DEMUX.





* Difference between Multiplexer & Demultiplexer:

	Multiplexer	Demultiplexer
1	It is a combinational logic circuit, that works on 'many to one' concept	It is a combinational logic circuit that works on 'one to many' concept.
2	It has 'n' no. of data inputs and a single output.	It has a single data input & 'n' outputs.
3	The relation betn input lines & select lines is $n = 2^m$	The relation betn output line & select lines is $n = 2^m$
4	It is used as data selector	It is used as data distributor.
5	It is ~~serial~~ parallel to serial port converter	It is series to parallel port converter

* Priority Encoder ~~IC 74~~

- A logic circuit that responds to just one input in accordance with some priority system, among all those that may be simultaneously high is called as priority encoder.

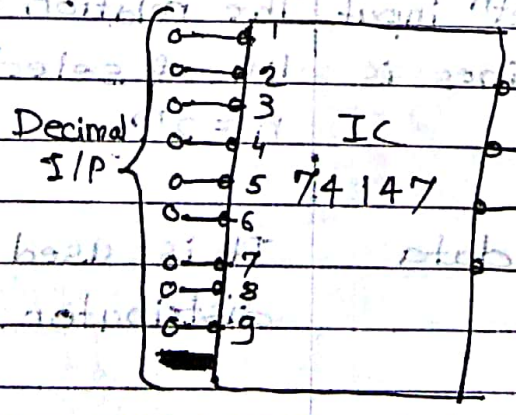
- The priority encoder is a special type of encoder.

- Types:
1. Decimal to BCD Encoder
2. ~~Binary to~~ Octal to Binary Encoder

1. Decimal-to-BCD Encoder (IC 74147):

- IC 74147 is a priority encoder IC which can be conveniently used for decimal to BCD conversion.

- It is called priority encoder because when more than one inputs are active, the input of higher priority is selected.



- Active low I/P & O/P.

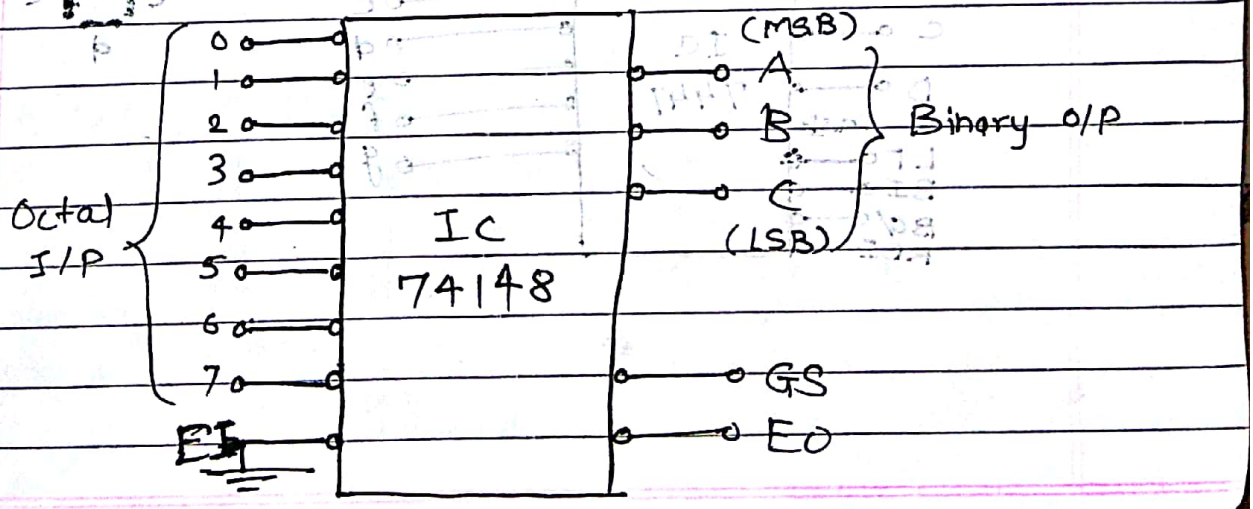
- Truth table:

Decimal inputs									BCD outputs			
1	2	3	4	5	6	7	8	9	A	B	C	D
1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0
x	0	1	1	1	1	1	1	1	1	1	0	1
x	x	0	1	1	1	1	1	1	1	1	0	0
x	x	x	0	1	1	1	1	1	1	0	1	1
x	x	x	x	0	1	1	1	1	1	0	1	0
x	x	x	x	x	0	1	1	1	1	0	0	1
x	x	x	x	x	x	0	1	1	1	0	0	0
x	x	x	x	x	x	x	0	1	0	1	1	1
x	x	x	x	x	x	x	x	0	0	1	1	0

2. Octal to Binary Encoder (IC 74148):

- IC 74148 is a priority encoder IC which can be conveniently used for octal to binary conversion.

- This circuit has active low inputs and active low outputs.

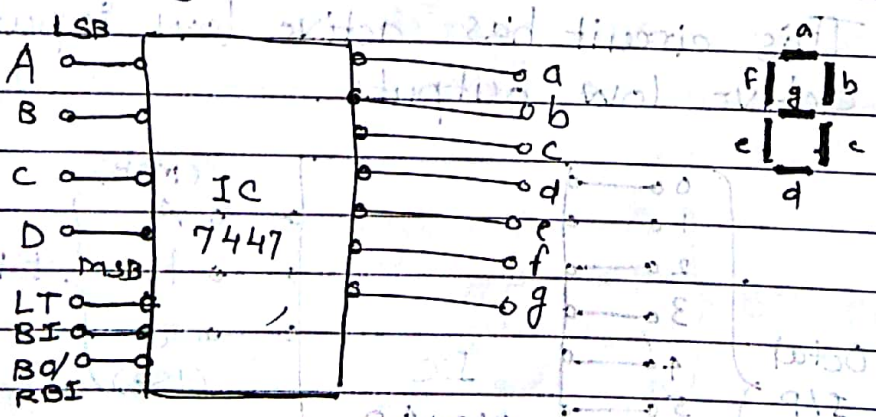


EI	Inputs								Outputs				
	0	1	2	3	4	5	6	7	A	B	C	GS	EO
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	x	0	1	1	1	1	1	1	1	1	0	0	1
0	x	x	0	1	1	1	1	1	1	0	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0	0	1
0	x	x	x	x	0	1	1	1	0	1	1	0	1
0	x	x	x	x	x	0	1	1	0	1	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	0	1
0	x	x	x	x	x	x	x	0	0	0	0	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0

* BCD-to-7 Segment Decoder (IC 7447):

- 7-segment display is the most popular display used in digital systems. For displaying data using this device, the data have to be converted from BCD-to-7 segment code.

- Block diagram:

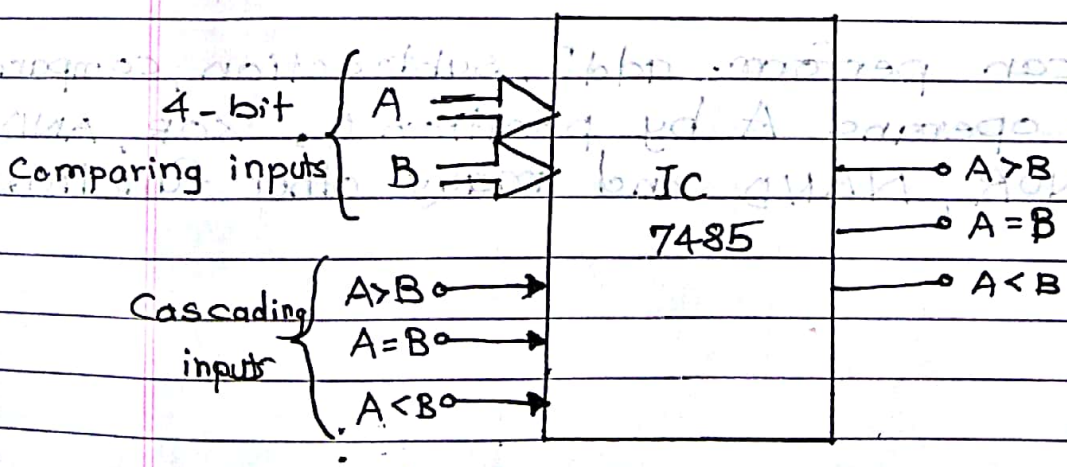


- Truth Table:

LT	BI	RBI /BO	Input D	C	B	A	Output a	b	c	d	e	f	g
0	1	X	x	x	x	x	0	0	0	0	0	0	0
x	0	x	x	x	x	x	1	1	1	1	1	1	1
1	1	0	0	0	0	0	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	1
1	1	1	0	0	0	1	1	0	0	1	1	1	1
1	1	1	0	0	1	0	0	0	1	0	0	1	0
1	1	1	0	0	1	1	0	0	0	0	1	1	0
1	1	1	0	1	0	0	1	0	0	1	1	0	0
1	1	1	0	1	0	1	0	1	0	0	1	0	0
1	1	1	0	1	1	0	0	1	0	0	0	0	0
1	1	1	0	1	1	1	0	0	0	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	0	0	0	1	0	0

* Digital Comparator (IC 7485):

- Digital comparator is used to compare two 4-bit input and produce output as A>B, A=B and A<B.



4. Sequential Logic Circuit

* Combinational Logic circuit:

"The circuit in which the output at any instant depends upon the present inputs is known as combinational logic circuit."

E.g., Adder, subtractor, multiplexer, demultiplexer.

* Sequential Logic Circuit:

"The circuit in which the output at any instant depends upon the present input as well as past input / output is known as sequential logic circuit."

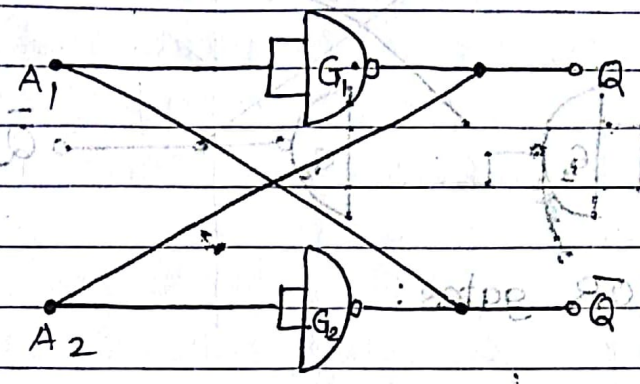
E.g., flip flop, counter, shift register, etc.

* Difference between Combinational & Sequential Logic circuit:

	Combinational Logic Circuit	Sequential Logic Circuit
1	In this circuit, output at an instant depends upon the present inputs only.	In this circuit, output at an instant depends upon present as well as past inputs/output.
2	E.g. adder, subtractor, multiplexer, demultiplexer.	Eg., Shift register counter
3	Memory element is not required.	Memory element is required.

* One Bit Memory Cell:

- It is also known as basic Flip-flop. It has two stable states : 1 and 0.

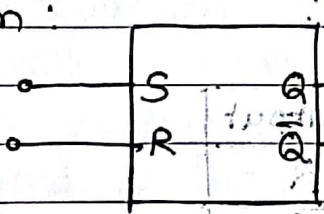


* SR Flip-Flop:

- It is the basic flip flop.

- It has two inputs S & R, where S stands for 'Set' and R stands for 'Reset'.

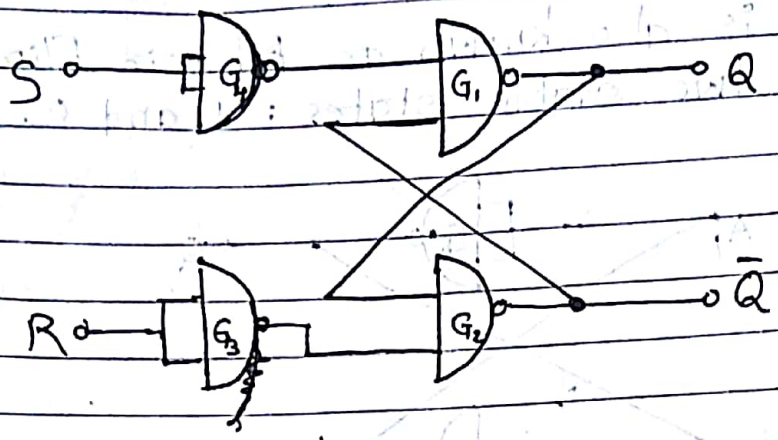
- Block diagram:



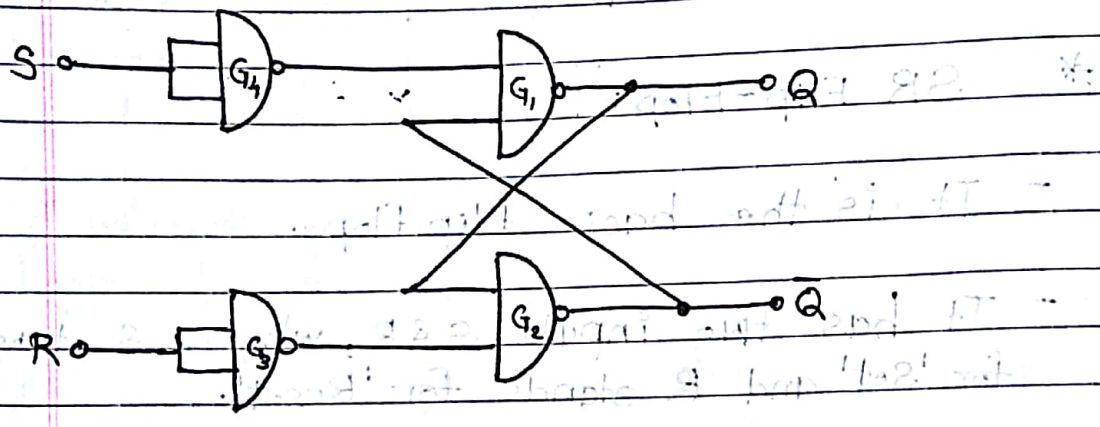
- It can be constructed in two ways:

1) Using NAND gate
2) Using NOR gate

1) Using NAND gates:



2) Using NOR gates:



- Truth table:

Input		Output	
S	R	Q	
0	0	Q_n	Hold
0	1	0	Reset
1	0	1	Set
1	1	?	Prohibit

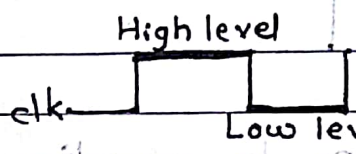
* Clock Triggering:

- The result of latches and flip-flops responding to clock input is called as clock pulse triggering.

- It is mainly of two types:
1. Level Triggering
2. Edge Triggering

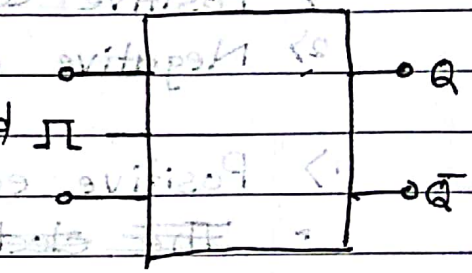
1. Level Triggering:

- The result of digital circuit responding to the level of clock input is called level triggering.

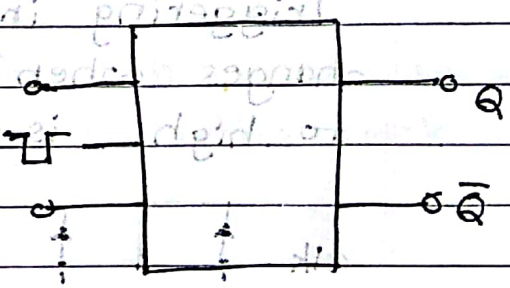


- It is of two types:
1) Positive level triggering
2) Negative level triggering

1) Positive level triggering:
- In this, the data is transferred from input to output when the clock signal is high.

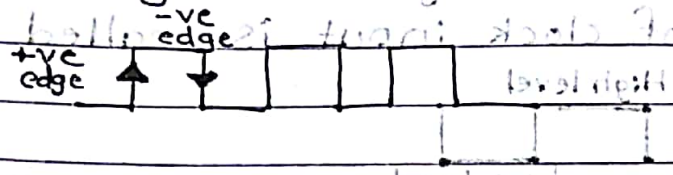


2) Negative level triggering:
- In this, the data is transferred from i/p to o/p when the clock signal is low.



2. Edge Triggering:

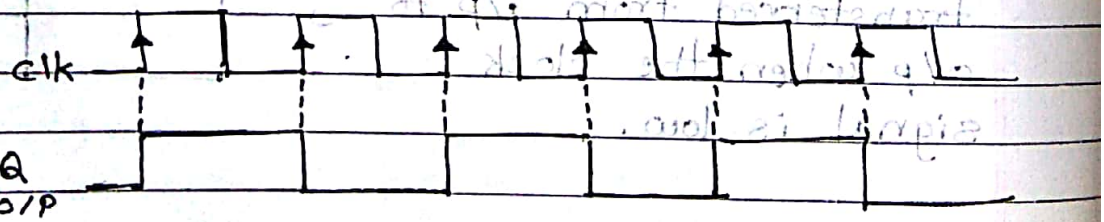
- A typical clock pulse train is as shown in fig. It can be positive or negative.
- The pulse goes through two transitions: 0 to 1 and 1 to 0.
- Positive transition is defined as positive edge and negative transition is called negative edge.

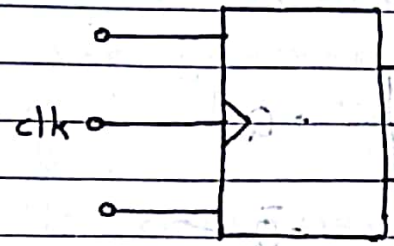


- The result of responding to negative edge or positive edge is called as edge triggering.
- It is of two types:
  1) Positive edge triggering
  2) Negative edge triggering

1) Positive edge triggering:

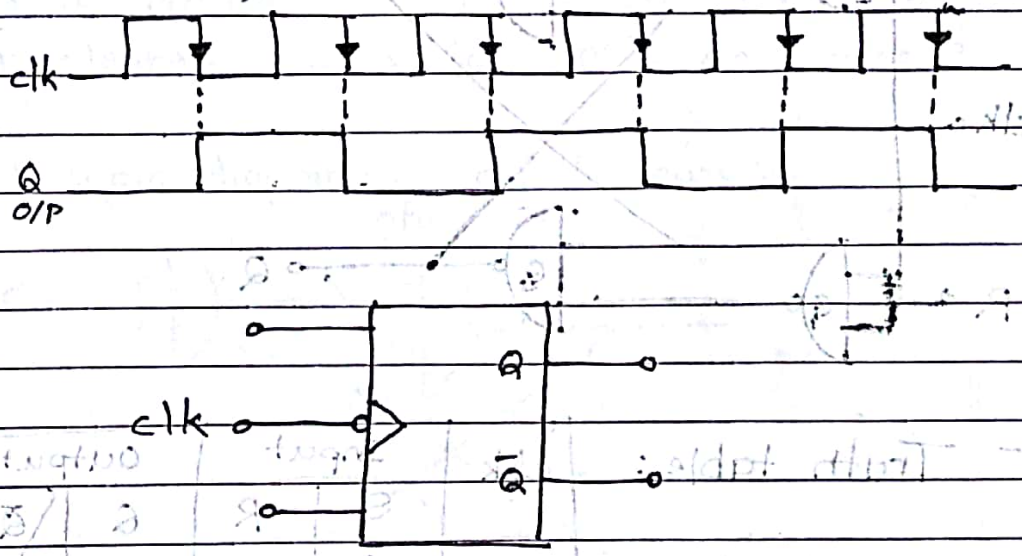
- ~~The clock pulse~~
- Triggering in which output of flip-flop changes when the pulse goes from low to high is called positive edge triggering.





2) Negative edge triggering:

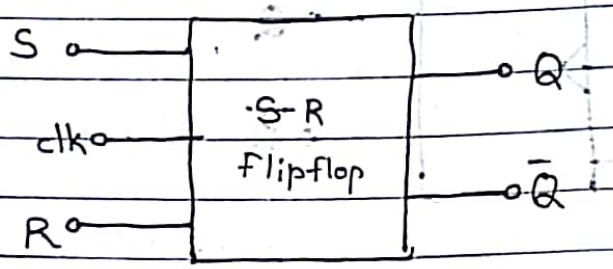
- Triggering in which output of flip-flop changes when the pulse goes from high to low, is called negative edge triggering.



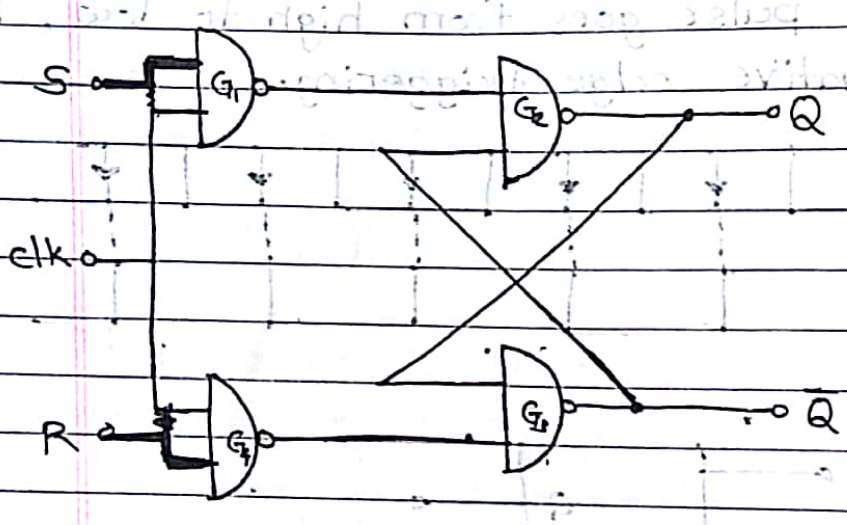
* Clocked SR Flip-Flop:

- In a clocked SR flip-flop, the circuit responds to inputs S and R only when the clock is present.
- Its truth table is exactly same as S-R flip-flop.

- Block diagram:



- Circuit Diagram:



- Truth table:

clk	Input		output	
	S	R	Q	$\bar{Q}$
1	0	0	Q_n	$\bar{Q}_n$
1	0	1	0	1
1	1	0	1	0
1	1	1	??	??

* Preset and Clear Inputs:

- When a flip-flop is switched ON, the state of circuit is uncertain. It can be set (1) or reset (0).

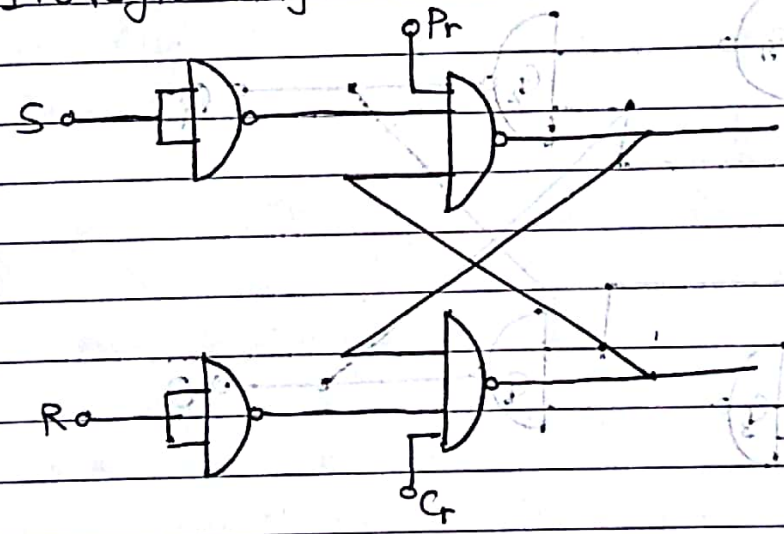
- In some applications, we need to set or reset the given flip-flop before use.

- This is done using preset (P_r) and clear (C_r) inputs.

- 'Preset' is used to store value 1 in the FF & 'clear' is used to store value 0 in the FF.

- It is active low i.e. a flip-flop is preset or cleared when logic '0' is applied to it.

- Its logic diagram is as follows:



* J-K Flip-Flop:

- In J-K flip-flop, the drawback of SR is removed.

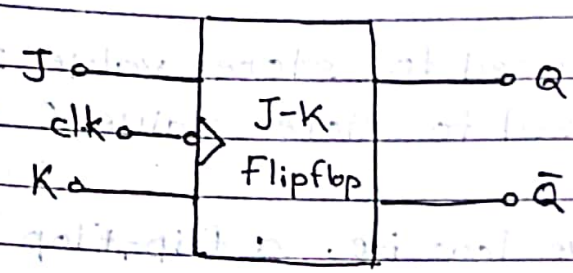
- In S-R flip-flop, when both inputs are 1, then the output is prohibited.

- To overcome this, J-K flip-flop is designed

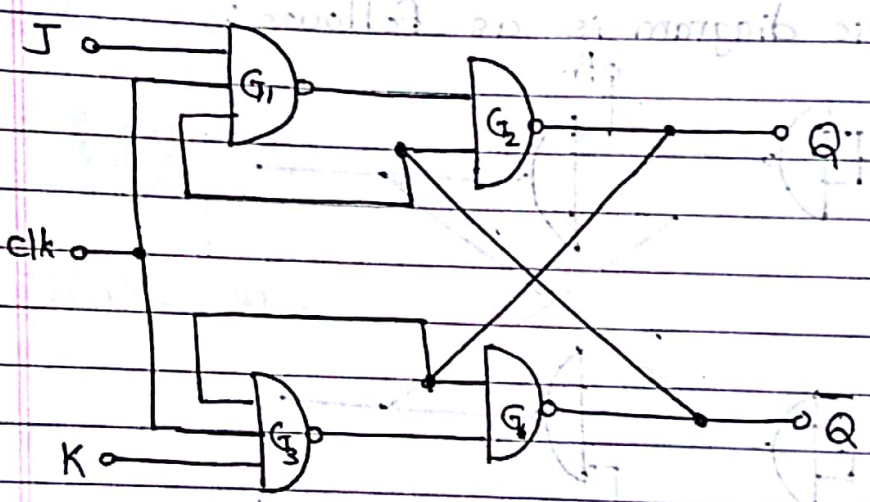
so that the output will be obtained.

- ~~Block~~ In this, J stands for 'Jump' and K stands for 'Keep'.

- Block diagram:



- Circuit Diagram:



- Truth table:

Input		Output	
J	K	Q	
0	0	Q_n	Hold prev. I/P
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}_n$	Complement of prev. I/P.

★ Race around condition:

- When a clock pulse is applied to a J-K flip-flop, during the clock pulse, the output of J-K flip flop keeps changing at J=K=1.

- When the pulse turns to zero, the o/p of Q is uncertain.

- This is known as race around condition.

- This drawback of J-K flip-flop can be overcome using another ~~technique~~ type of flip-flop known as 'Master-Slave J-K Flip-flop'.

* Master-Slave JK Flip-Flop:

- A master-Slave J-K flip flop is a cascade of two J-K flip-flops, in which the first half part acts as master JK flipflop whereas the other part acts as slave JK flip flop.

- The output of second flip-flop is the input of first one. A clock pulse is applied to first flip-flop and its inverted pulse is applied to the second flip-flop.

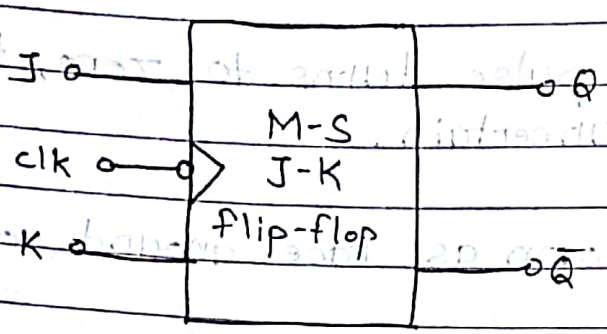
- So, when clk = 1, master flip-flop gets activated and outputs Q_m & $\bar{Q}_m$ act as Q & $\bar{Q}$. But, as clk = 0 in slave, flip-flop, its output remains

1) as it is.

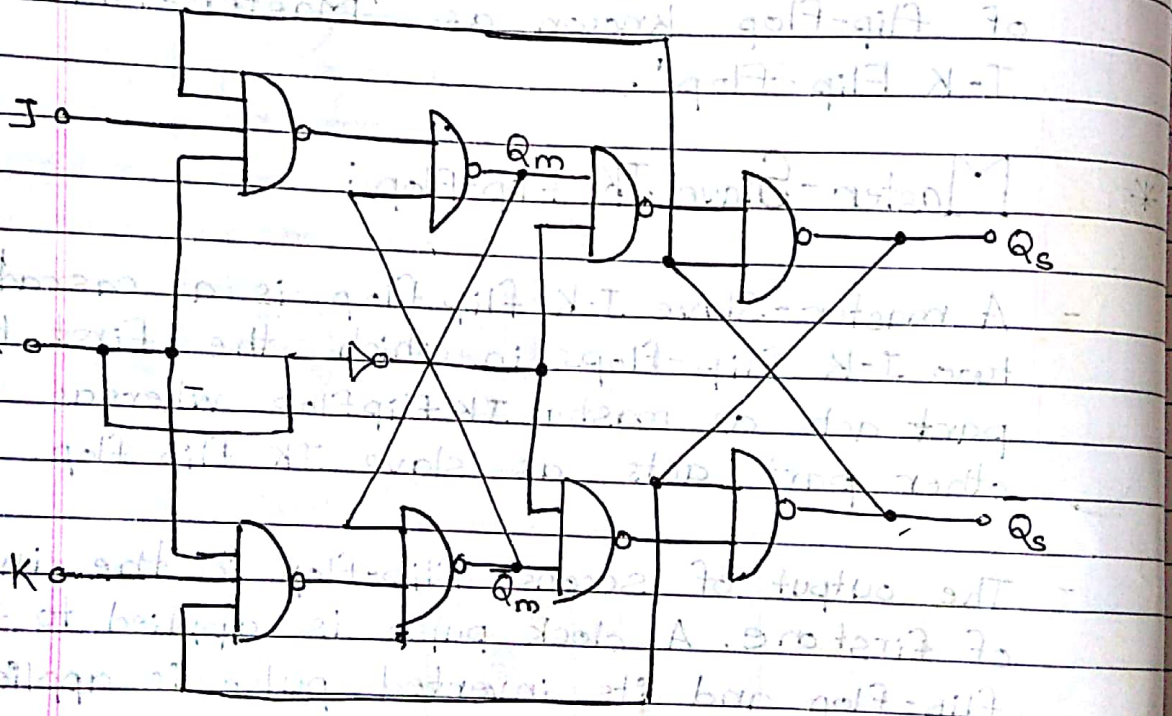
- When clk = 0 is applied, master flip-flop is deactivated. So, the output remains constant.

- Block Diagram:

2)



- Circuit Diagram:

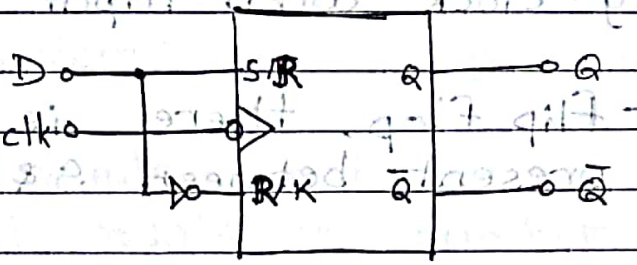


- Truth table:

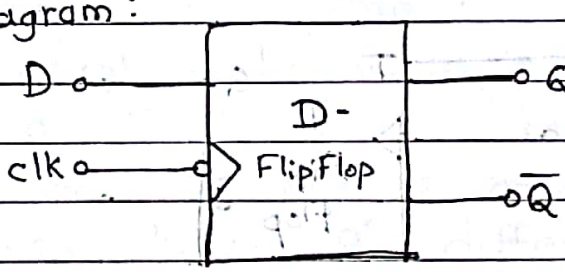
Input		Output
J	K	Q
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

* D-FlipFlop: *

- A D-flip flop can be prepared from a J-K or S-R flip-flop by just adding an inverter as shown in fig



- Block diagram:



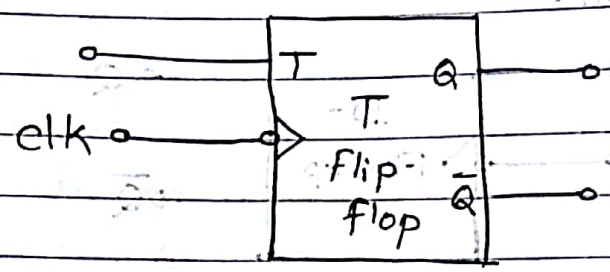
- Truth table:

Input D	Output Q
0	0
1	1

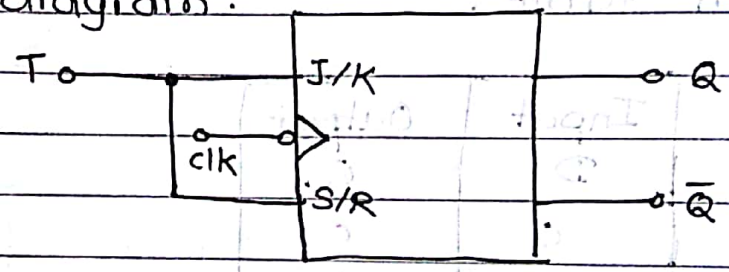
- A D Flip-Flop produces the output which is given to it as an input.
- It is also known as delay Flip flop as it produces the output after the clock is applied.

* T-type Flip-flop:

- A T-flip flop is just like a D-flip flo
- But, in T-flipflop the output changes with every clock when input is high.
- In a T-flip flop, there is no inverter present between S&R or J&K
- Block diagram:



- Circuit diagram:



- Truth table:

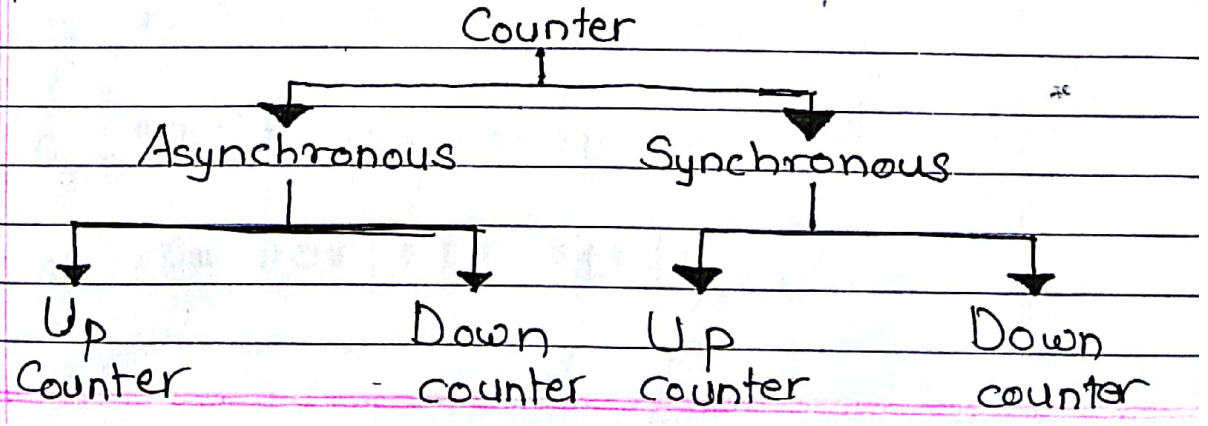
Input T	Output Q
0	Q_n
1	$\bar{Q}_n$

* Applications of Flip-Flop:

1. Counter
2. Shift Register
3. Memory

* Counters:

- "A counter can be defined as a logic circuit that counts the number of clock pulses applied at the input."
- Each count is known as a state of the counter. It is in the form of binary number. So, a counter counting in terms of n-bits have 2^n different states.
- The number of different states of the counter is known as modulus of a counter.
- So, a 2-bit counter has $2^n = 4$ states. i.e. MOD-4 counter.
- A counter can be classified as:



1] Asynchronous (Ripple) Counter:

- In an asynchronous counter, the first flip-flop is clocked by the external clock pulse and then each successive flip-flop is clocked by the Q and $\bar{Q}$ output of the previous flip-flop.

- In an asynchronous counter, clock pulse is not applied simultaneously.

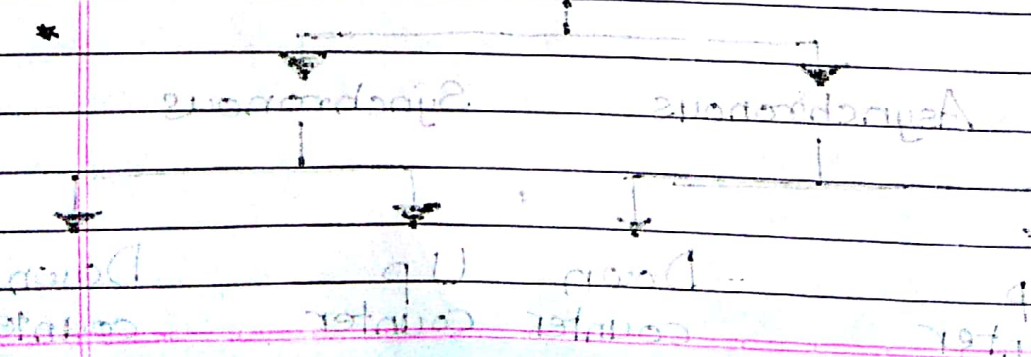
- In this counter, the input of each gate is 'logic 1'.

- In a counter, the flip-flops used are T-type flip-flops.

1. Asynchronous Up Counter:

- In this, the output Q of first flipflop is applied as clock input to the other flip-flop.

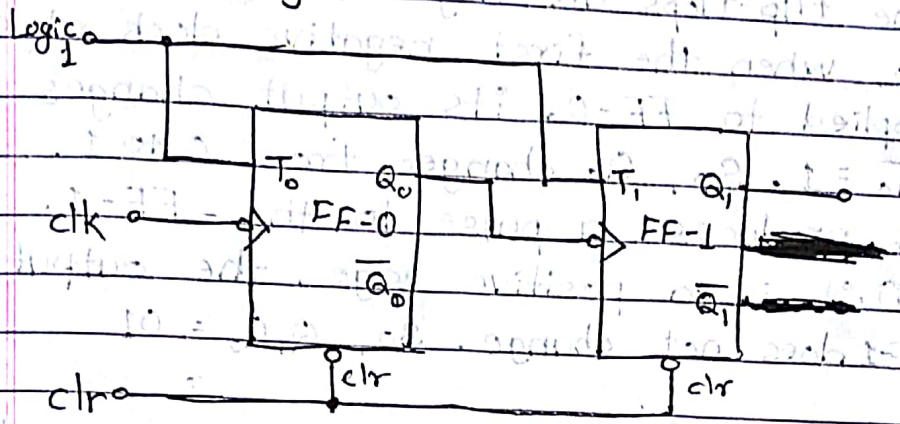
- In this, the decimal equivalent of the output increases with the successive clock pulses. So, it is known as up count



1) 2-bit Asynchronous Up Counter:

- It is also known as MOD-4 counter as it counts from 0 to 3.

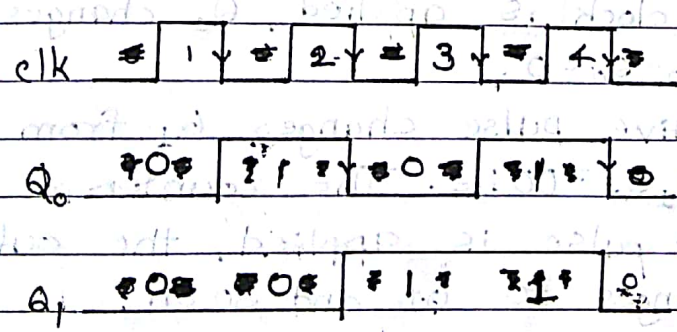
- Block diagram:



- Truth table:

Clock pulse	Counter output	
	Q_1	Q_0
Initially	0	0
First ↓	0	1
Second ↓	1	0
Third ↓	1	1
Forth ↓	0	0

- Waveform:



- Operation:

① First, both the outputs are reset using clr input. So, $Q_1Q_0 = 00$.

② The flip-flops are negative edge triggered. So, when the first negative clock edge is applied to FF-0, its output changes as $T_0 = 1$. So, Q_0 changes from 0 to 1. It produces a pulse to the FF-1. But, as it is a positive edge, the output of FF-1 does not change. So, $Q_1Q_0 = 01$

③ When 2nd clock is applied to FF-0, the output again changes from 1 to 0. So, $Q_0 = 0$. This output acts as a negative pulse for FF-1. So, Q_1 changes ~~from 1 to 0~~ 0 to 1. Thus, $Q_1Q_0 = 10$

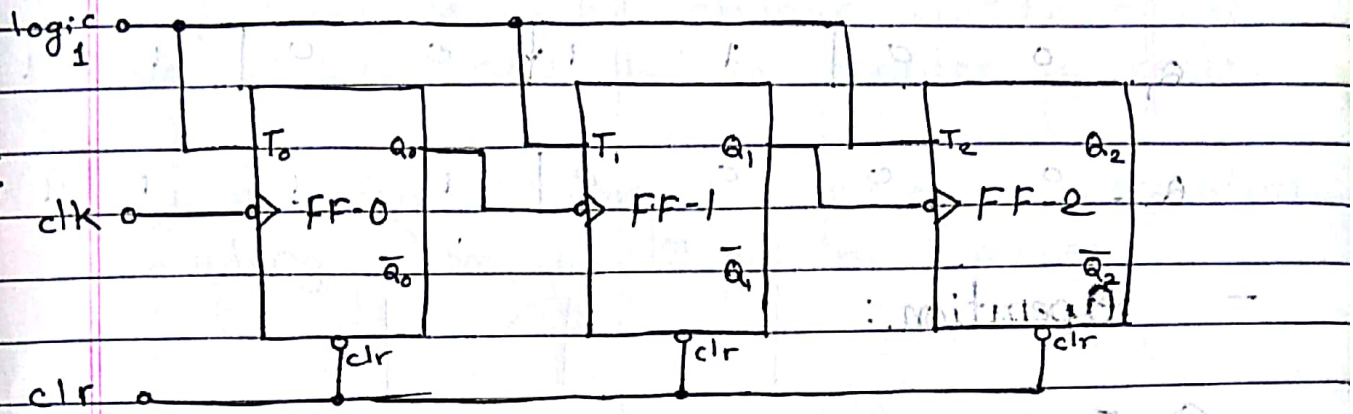
④ When 3rd clock is applied to FF-0, the o/p changes from 0 to 1. So, $Q_0 = 1$. The pulse produced is positive. So, Q_1 does not change. $Q_1 = 1$. Thus, $Q_1Q_0 = 11$.

⑤ When 4th clock is applied, Q_0 changes from 1 to 0. So, $Q_0 = 0$. This negative pulse changes Q_1 from 1 to 0. Thus, $Q_1Q_0 = 00$ i.e. The counter is reset. When 5th pulse is applied, the output again changes to 01 and so on.

2) 3-bit Asynchronous Up Counter:

- It is also known as MOD-8 counter

- Block diagram:

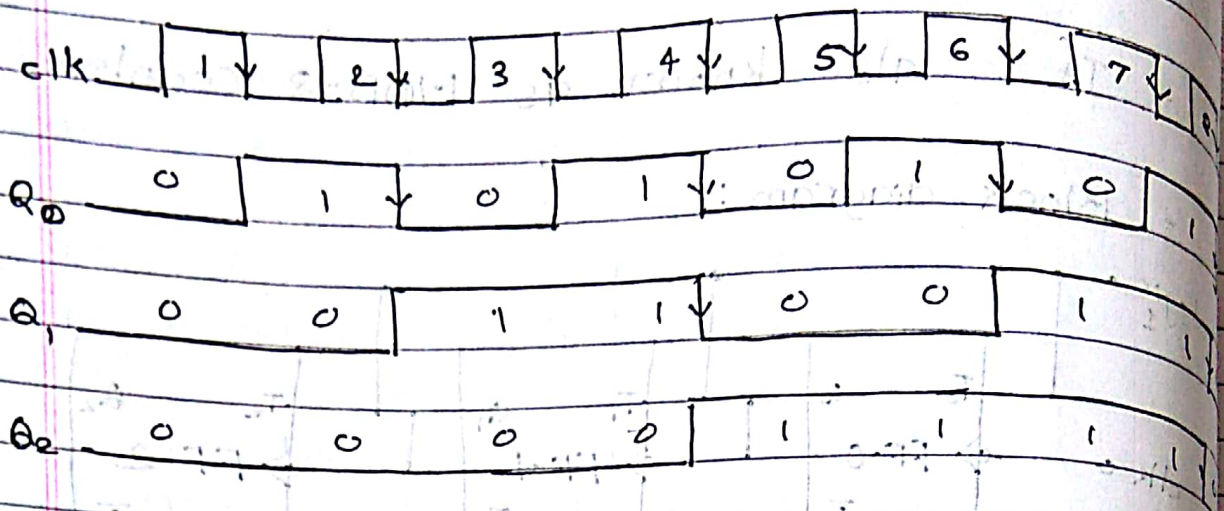


- Truth Table:

Clock pulse	Flip-flop O/P		
	Q_2	Q_1	Q_0
Initially	0	0	0
1st	0	0	1
2nd	0	1	0
3rd	0	1	1
4th	1	0	0
5th	1	0	1
6th	1	1	0
7th	1	1	1
8th	0	0	0

- ~~Waveform~~

- Waveform:



- Operation:

① In this, 3 negative edge triggered T-flip-flops are used. First, clear input is applied to all flip-flops. So, $Q_2Q_1Q_0 = 000$

② When 1st clock is applied, Q_0 changes from 0 to 1. But Q_1 & Q_2 remain 0. So, $Q_2Q_1Q_0 = 001$

③ When 2nd clock is applied, Q_0 changes from 1 to 0. This produces a negative pulse for FF-1. So, Q_1 changes from 0 to 1. But Q_2 remains same. So, $Q_2Q_1Q_0 = 010$.

④ When 3rd pulse is applied, Q_0 changes from 0 to 1. But Q_0 & Q_1 does not change. So, $Q_2Q_1Q_0 = 011$.

⑤ When 4th clock is applied, Q_0 changes from 1 to 0. So, the negative pulse produced changes Q_1 from 1 to 0. It changes Q_2 from 0 to 1. So, $Q_2Q_1Q_0 = 100$

⑤ This continues till 7th clock. So, at 7th clock, the output $Q_2Q_1Q_0 = 111$.

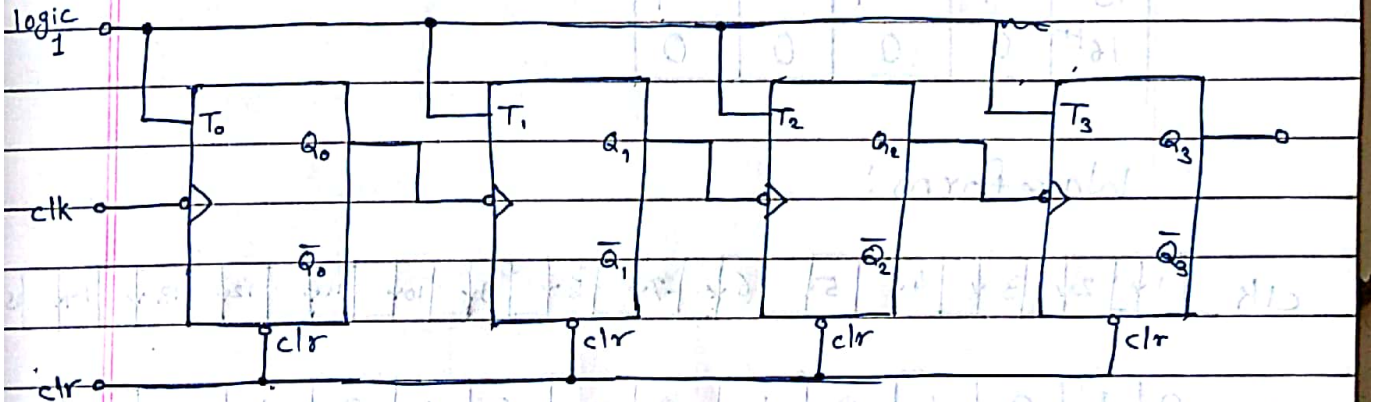
⑥ When 8th clock is applied, Q_0 changes from 1 to 0. As a result, Q_1 changes from 1 to 0. This negative pulse changes Q_2 from 1 to 0. So, $Q_2Q_1Q_0 = 000$. That is, counter is reset.

⑦ Again, when 9th clock is applied, counter starts counting from 1 to 7 for next pulses.

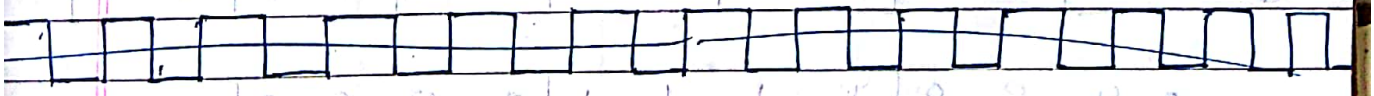
3) 4-bit Asynchronous Counter :

- It is also known as MOD-16 Counter.

- Block diagram:



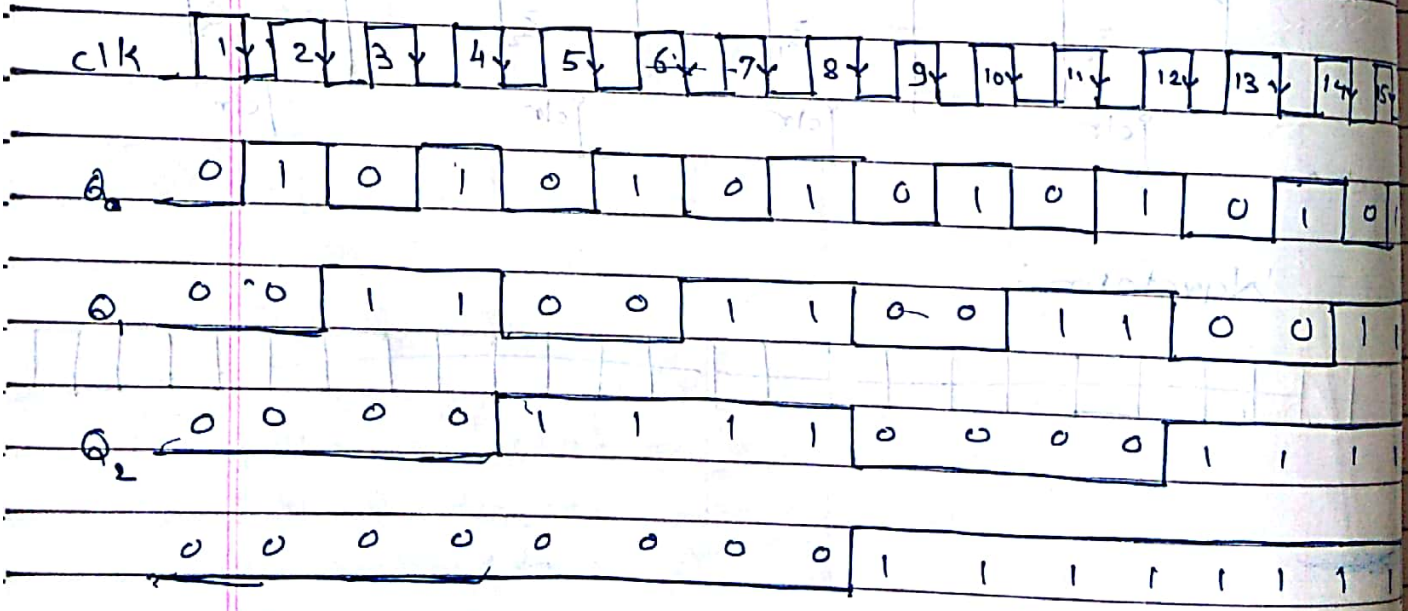
- Waveform:



- Truth table:

clock	Q_3	Q_2	Q_1	Q_0
Initial	0	0	0	0
1st	0	0	0	1
2nd	0	0	1	0
3rd	0	0	1	1
4th	0	1	0	0
5th	0	1	0	1
6th	0	1	1	0
7th	0	1	1	1
8th	1	0	0	0
9th	1	0	0	1
10th	1	0	1	0
11th	1	0	1	1
12th	1	1	0	0
13th	1	1	0	1
14th	1	1	1	0
15th	1	1	1	1
16th	0	0	0	0

- Waveform:

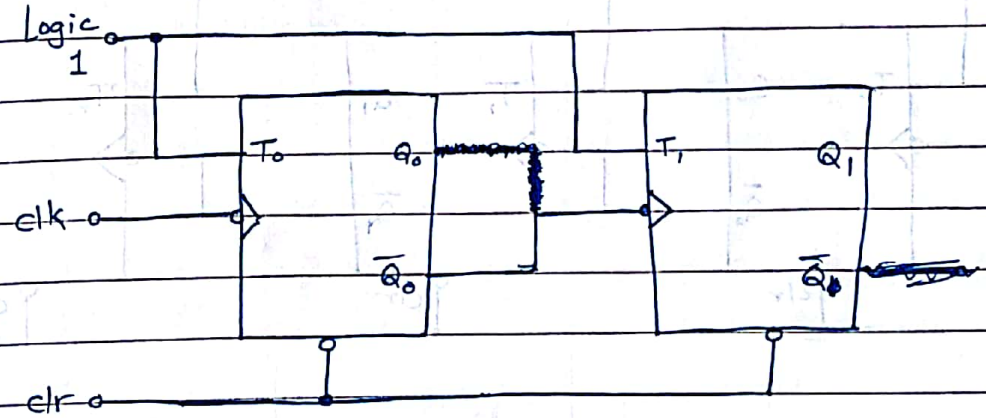


2. Asynchronous Down Counter:

- In this, the output complement $\bar{Q}$ is applied as ~~input~~ clock input for next ~~counter~~ flip-flop
- In this, decimal equivalent of the output decreases with successive clock pulses. So, it is known as down counter.

1) 2-bit Asynchronous Down Counter:

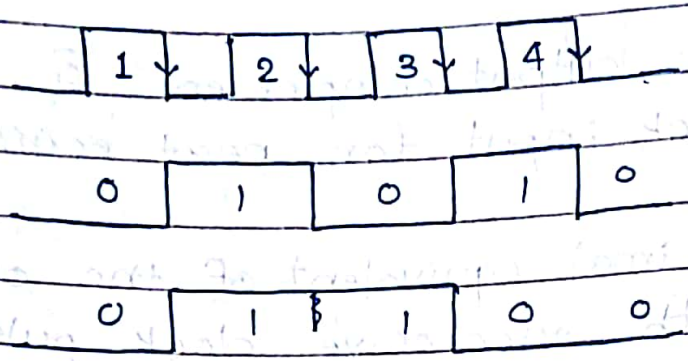
- Block Diagram:



- Truth table:

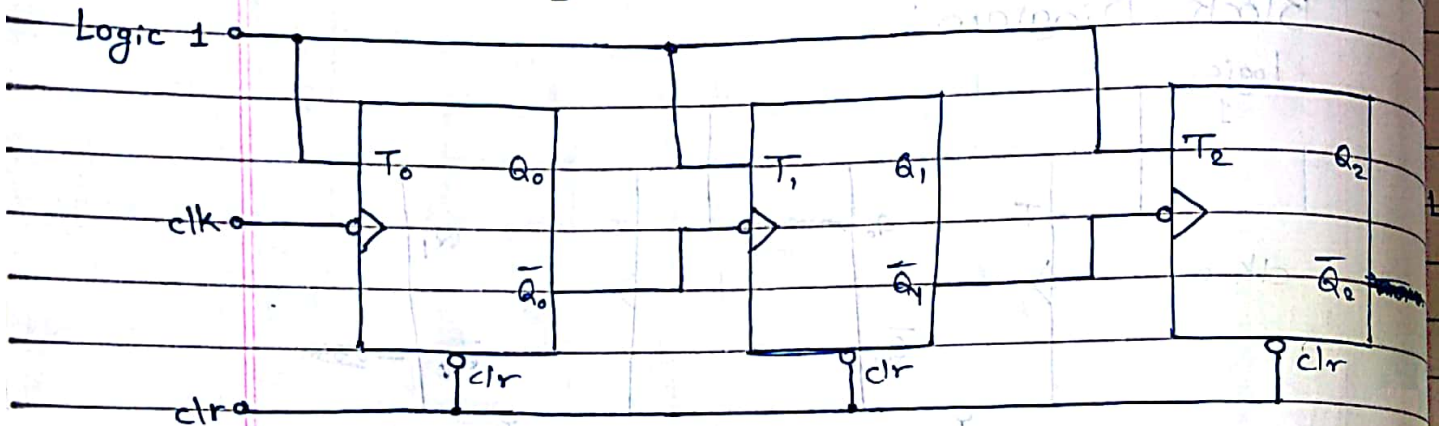
Clock	Q_1	Q_0
Initially	0	0
1st	1	1
2nd	1	0
3rd	0	1
4th	0	0

- Waveform:



2) 3-bit Asynchronous Down Counter:

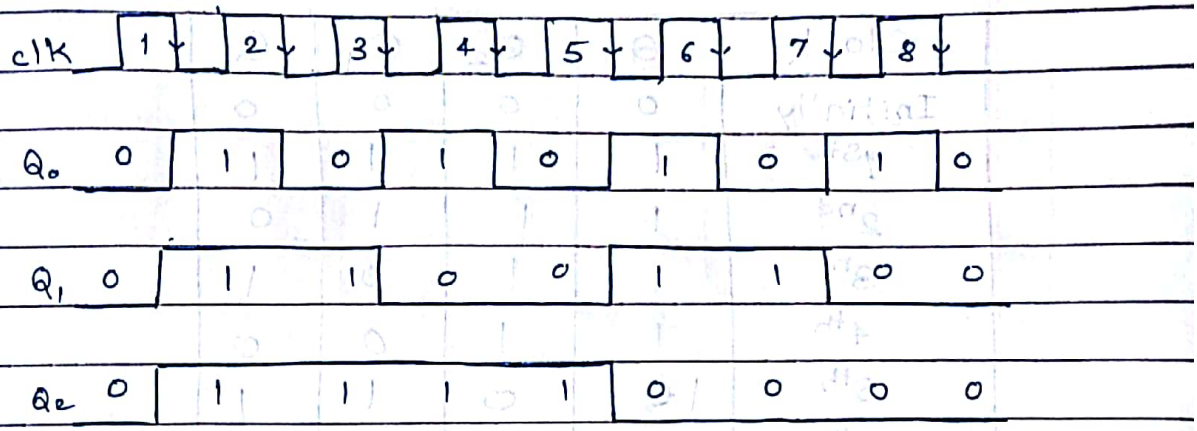
- Block Diagram:



- Truth table:

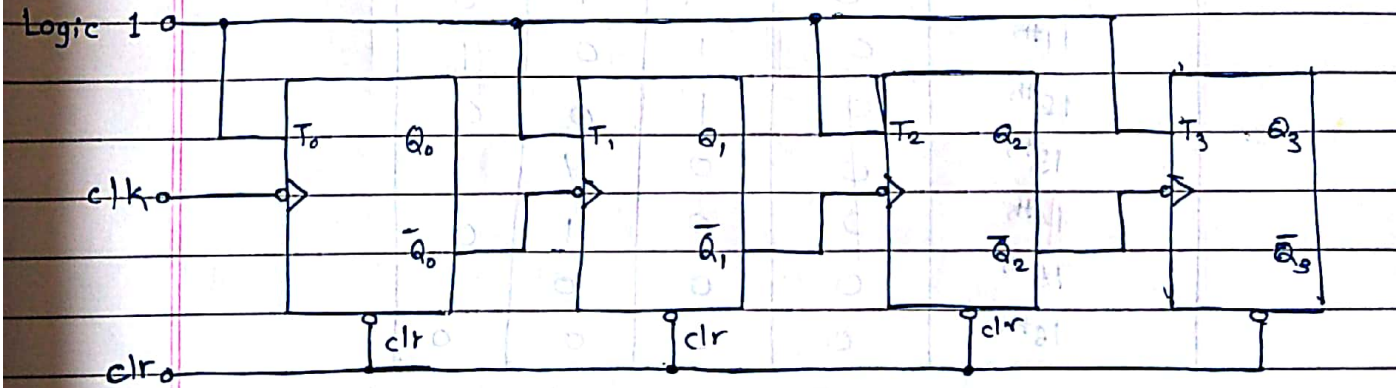
Clock	Q_2	Q_1	Q_0
Initially	0	0	0
1st	1	1	1
2nd	1	1	0
3rd	1	0	1
4th	1	0	0
5th	0	1	1
6th	0	1	0
7th	0	0	1
8th	0	0	0

- Waveform:

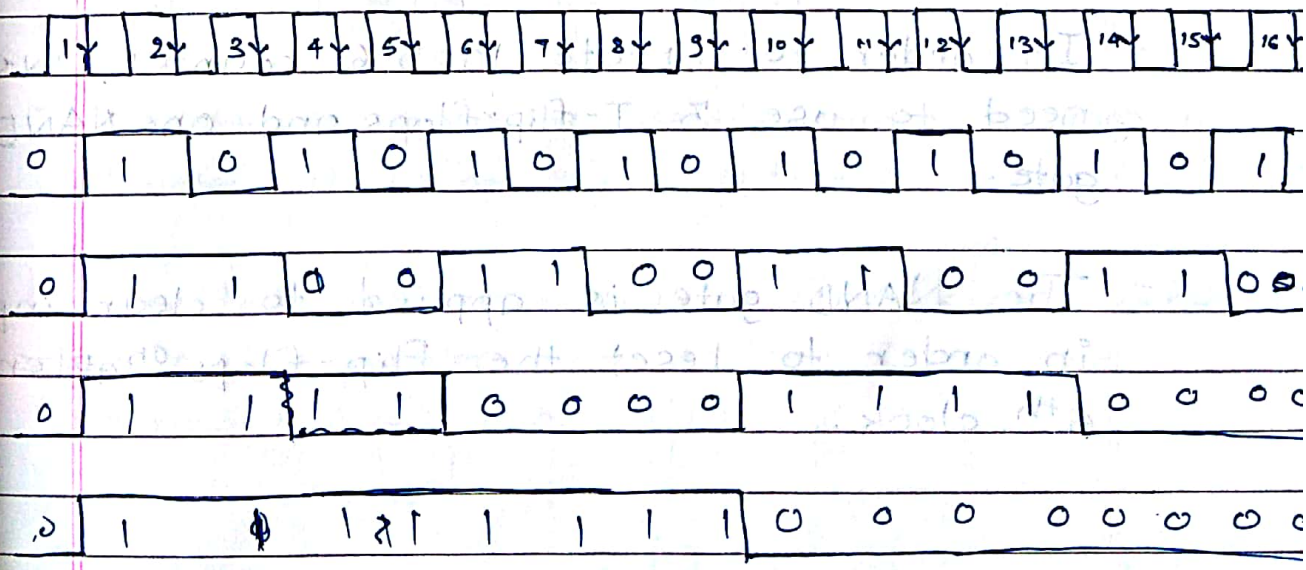


3) 4-bit Asynchronous Down Counter:

- Block Diagram:



- Waveform:



- Truth Table:

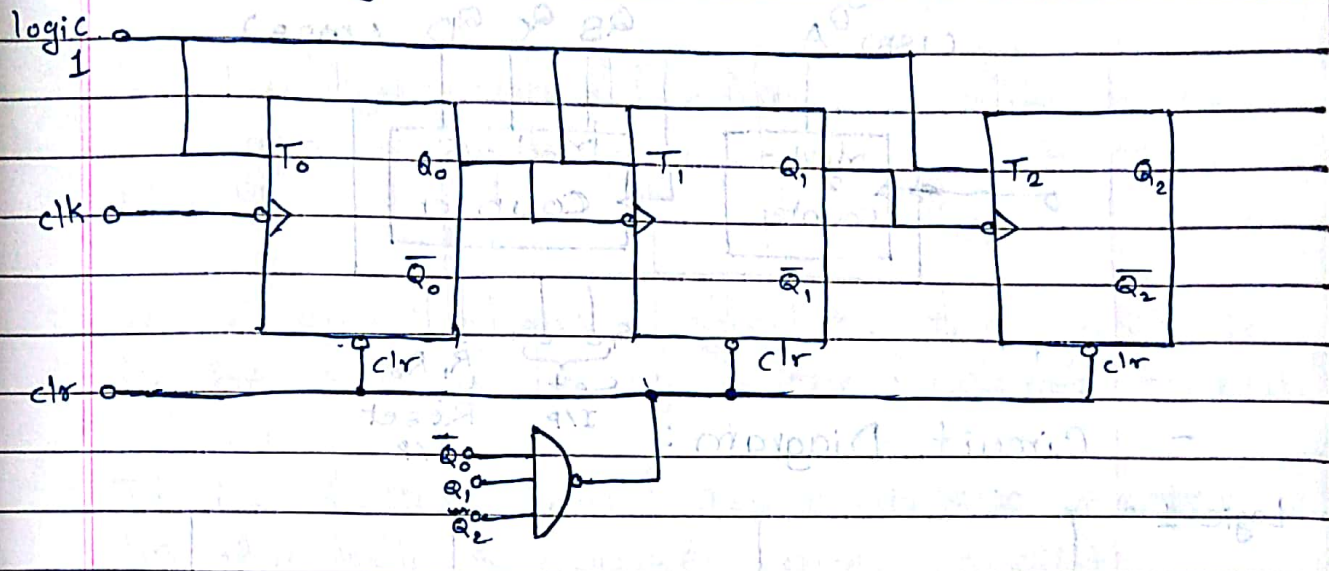
Clock	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1st	1	1	1	1
2nd	1	1	1	0
3rd	1	1	0	1
4th	1	1	0	0
5th	1	0	1	1
6th	1	0	1	0
7th	1	0	0	1
8th	1	0	0	0
9th	0	1	1	1
10th	0	1	1	0
11th	0	1	0	1
12th	0	1	0	0
13th	0	0	1	1
14th	0	0	1	0
15th	0	0	0	1
16th	0	0	0	0

★ MOD-6 Ripple Counter:

- In order to create MOD-6 counter, we need to use 3 T-flip-flops and one NAND gate.

- The NAND gate is applied to clear inpu in order to reset the flip-flops on ~~after~~ 6th clock.

- Block Diagram:



- Truth Table:

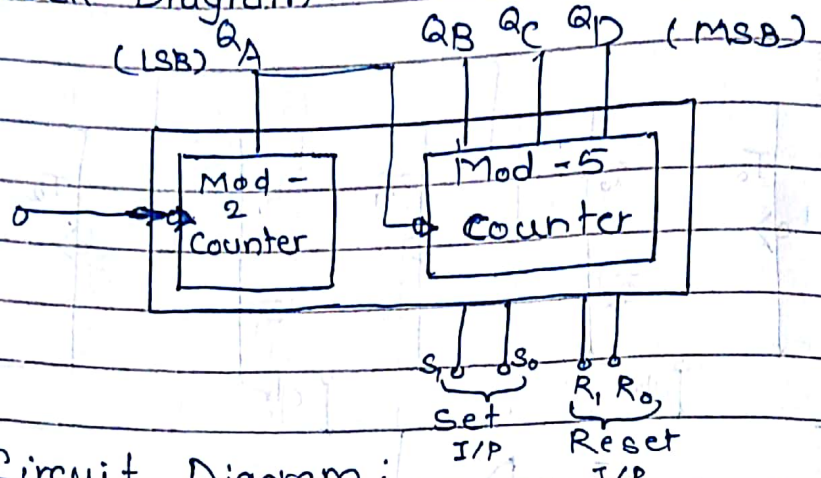
clock	Q_2	Q_1	Q_0
Initially	0	0	0
1st	0	0	1
2nd	0	1	0
3rd	0	1	1
4th	1	0	0
5th	1	0	1
6th	0	0	0

★ Decade Counter (IC 7490):

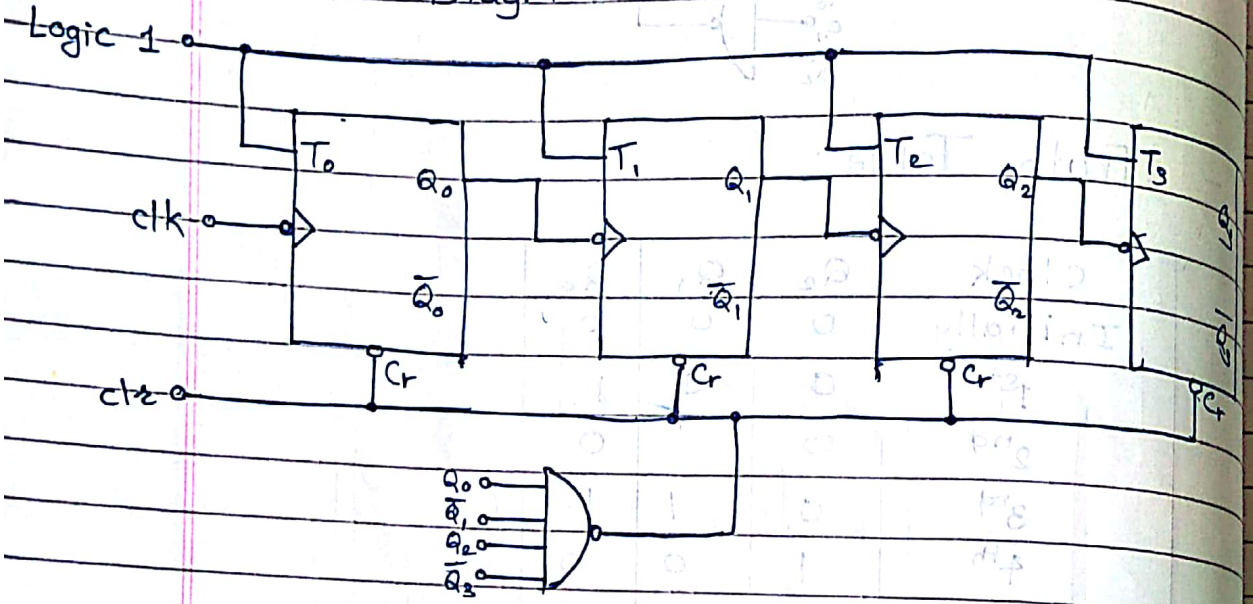
- IC 7490 is known as decade counter or MOD-10 Counter. It counts from 0 to 9.

- In this, two asynchronous counters are used: a MOD-2 counter and a MOD-5 counter connected to each other.

- Block Diagram:



- Circuit Diagram:



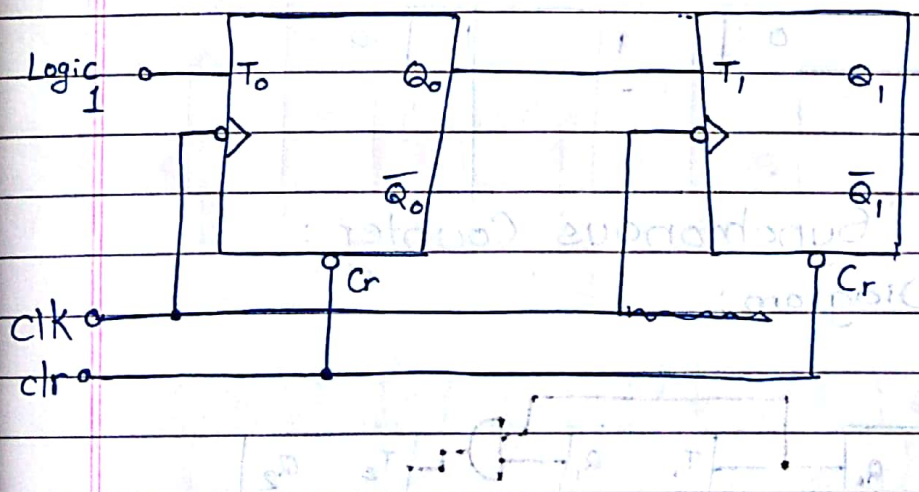
- Truth Table:

Clock	Q_D	Q_C	Q_B	Q_A
Initially	0	0	0	0
1st	0	0	0	1
2nd	0	0	1	0
3rd	0	0	1	1
4th	0	1	0	0
5th	0	1	0	1
6th	0	1	1	0
7th	0	1	1	1
8th	1	0	0	0
9th	1	0	0	1
10th	0	0	0	0

2] Synchronous Counter:

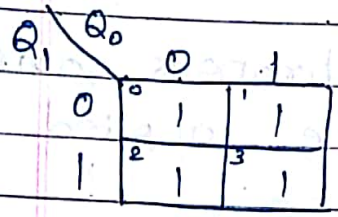
- In a synchronous counter, clock pulses are applied simultaneously to each T flip-flops.
- In a synchronous counter, the input is to be set according to the desired output.
- It is faster than asynchronous counter as the clock pulses are applied simultaneously.
- It is classified as:
1. Synchronous Up Counter
2. Synchronous Down Counter

1) 2-bit Synchronous Counter:

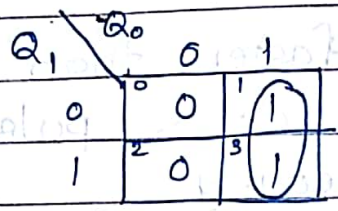


- Truth Table / Excitation Table:

Clock	Q_1	Q_0	T_1	T_0
Initially	0	0	0	1
1st	0	1	1	1
2nd	1	0	0	1
3rd	1	1	1	1
4th	0	0	0	1



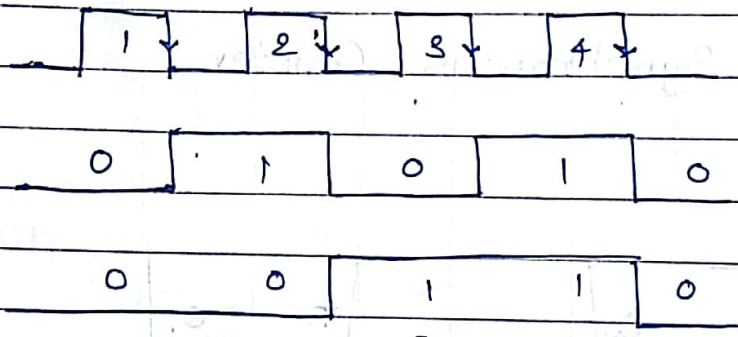
$T_0 = 1$



$T_1 = Q_0$

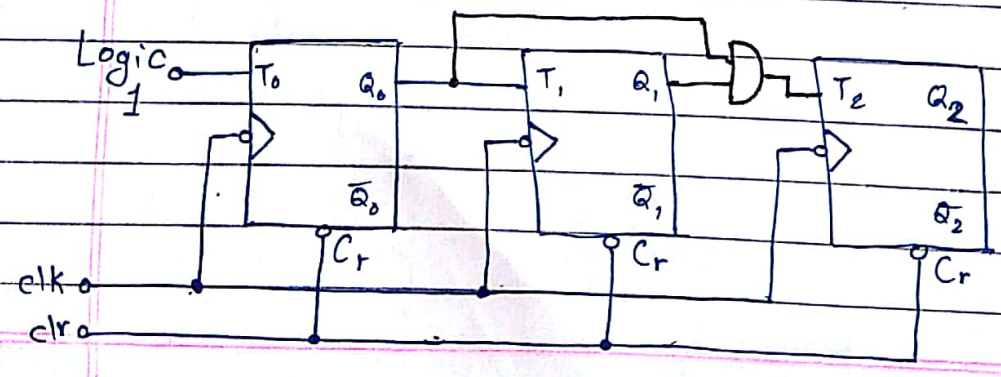
- Waveform:

2)



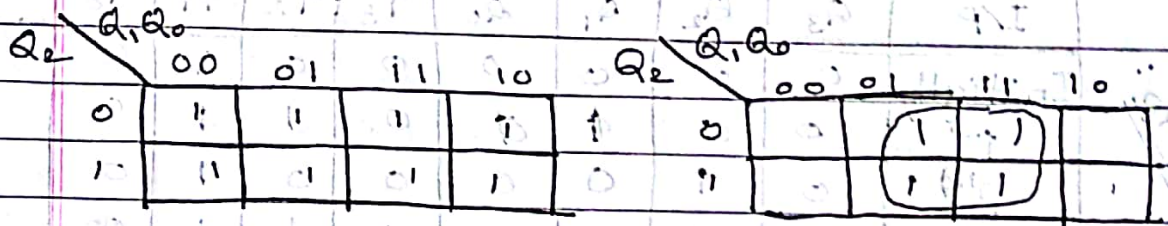
2) 3-bit Synchronous Counter:

- Block Diagram:



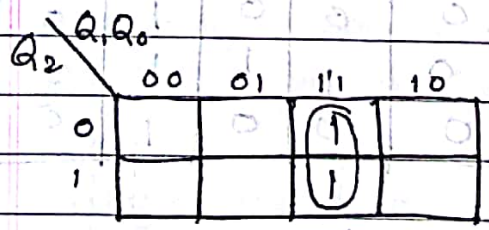
- Excitation Table:

Clock I/P	Output			Input		
clk	Q_2	Q_1	Q_0	T_2	T_1	T_0
Initially	0	0	0	0	0	1
1st	0	0	1	0	1	1
2nd	0	1	0	0	0	1
3rd	0	1	1	1	1	1
4th	1	0	0	0	0	1
5th	1	0	1	0	1	1
6th	1	1	0	0	0	1
7th	1	1	1	1	1	1
8th	0	0	0	0	0	1

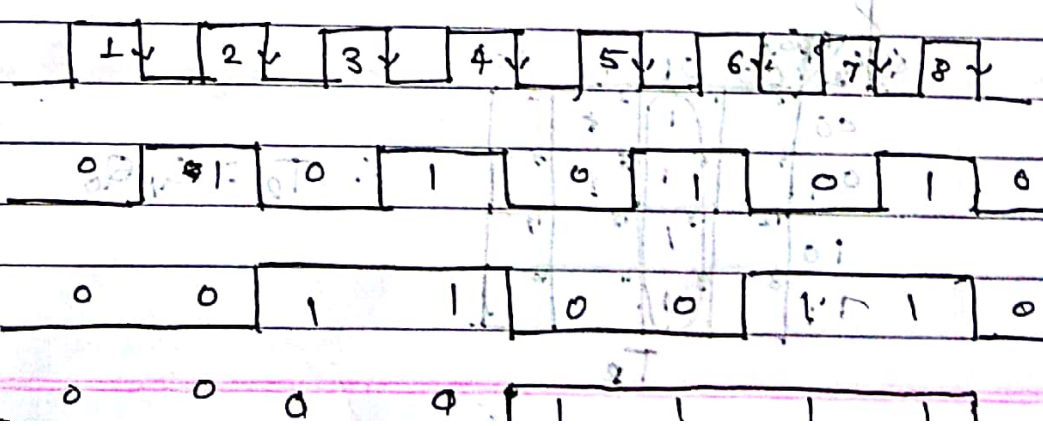


$\therefore T_0 = 1$

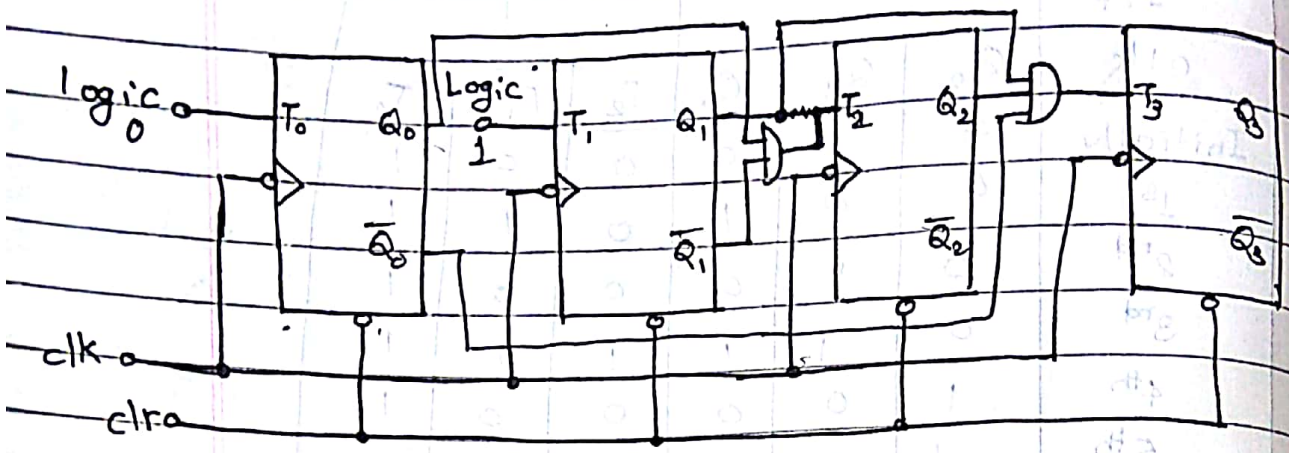
$\therefore T_1 = Q_0$



$\therefore T_2 = Q_1 \cdot Q_0$



4) Design 4-bit synchronous counter which count 0, 2, 4, 6, 8, 10, 12, 14, 0.

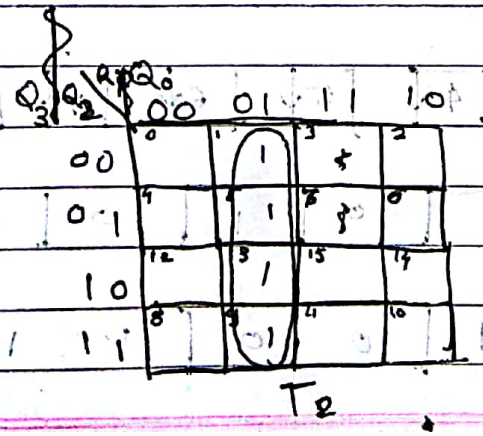


- Excitation Table :

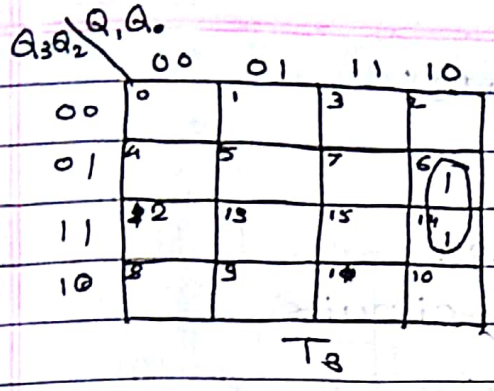
Clk	Output				Input			
I/P	Q_3	Q_2	Q_1	Q_0	T_3	T_2	T_1	T_0
Initially	0	0	0	0	0	0	1	0
1st	0	0	0	0	0	1	1	0
2nd	0	0	1	0	0	0	1	0
3rd	0	1	1	0	1	1	1	0
4th	1	0	0	0	0	0	1	0
5th	1	0	1	0	0	1	1	0
6th	1	1	0	0	0	0	1	0
7th	1	1	1	0	1	1	1	0
8th	0	0	0	0	0	0	1	0

• Waveform :

$\therefore T_0 = 0 \qquad T_1 = 1$



$\therefore T_2 = \bar{Q}_1 Q_0$



$\therefore T_B = Q_2 Q_1 \overline{Q_0}$

* DIFFERENCE BETWEEN SYNCHRONOUS & ASYNCHRONOUS COUNTER:

	Asynchronous Counter	Synchronous Counter
1	Output of the previous flip-flop is connected to the clock input of next stage.	The output of previous flip-flop is not connected to the clock input of next stage.
2	Simple logic implementation.	Complex logic implementation.
3	The flip-flops are not connected simultaneously.	The flip-flops are connected simultaneously
4	Large propagation delay	Low propagation delay
5	Low frequency of operation.	High frequency of operation

* Applications of Counters:
1) In A to D Converters
2) In digital clock
3) In frequency counters
4) In frequency divider circuits
5) In digital voltmeters

* REGISTERS:

- "A group of flip-flops used to store data is known as register."

- Thus, an n-bit register has a group of n-flipflops and can store any binary information or number containing n-bits.

★ Shift Register:

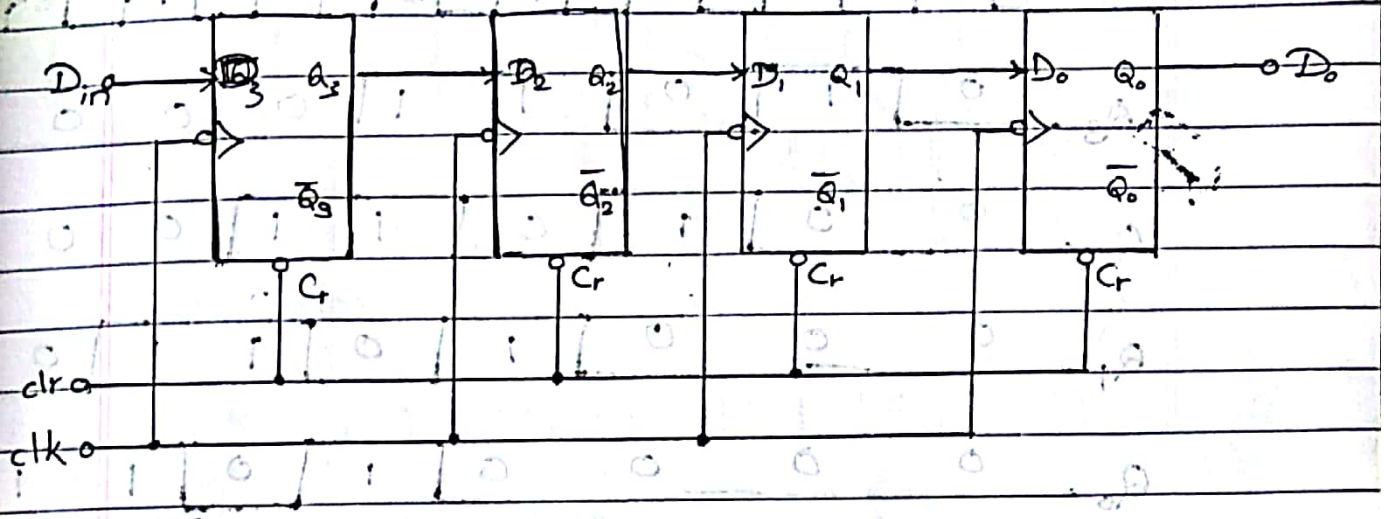
- In a register, the binary data can be shifted from one stage to another stage within the register or into or out of the register after application of clock pulses.

- Such shifting is essential for certain arithmetic & logical operations used in microprocessors.

- Types of Shift Registers:
1. SISO
2. SIPO
3. PIPO
4. PISO

1. Serial In Serial Out Shift Register (SISO):

- Block Diagram:



- Truth Table: I/P = 1101

Clock	Q_3	Q_2	Q_1	Q_0	
Initially	0	0	0	0	
1st	1	0	0	0	
2nd	0	1	0	0	
3rd	1	0	1	0	
4th	1	1	0	1	⇒ O/P
5th	0	1	1	0	
6th	0	0	1	1	
7th	0	0	0	1	
8th	0	0	0	0	

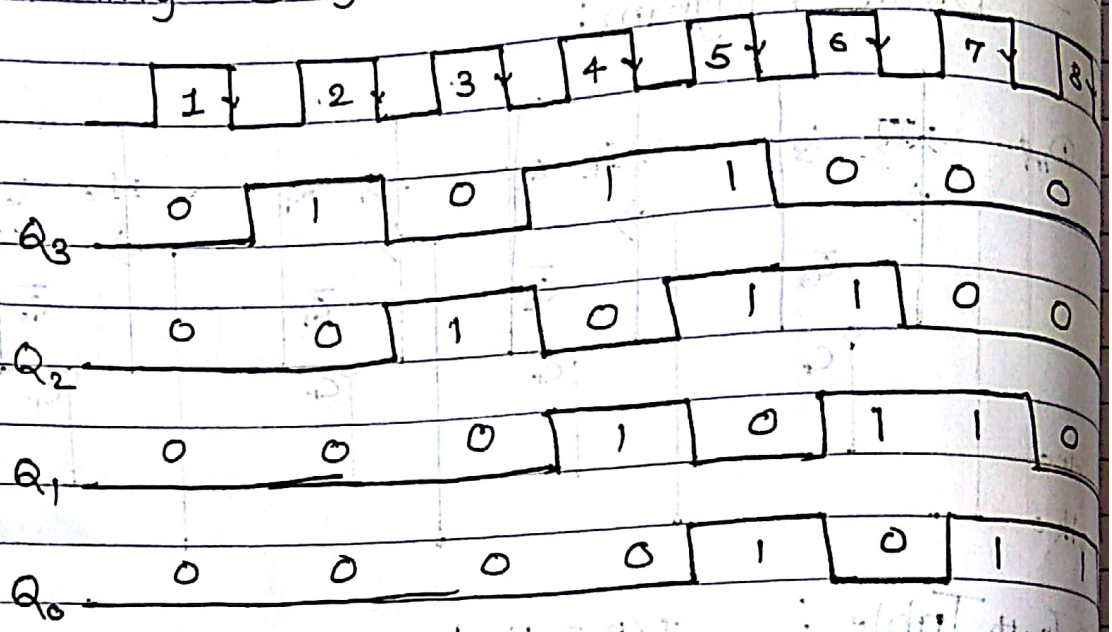
- Working:

- In SISO, the input is applied at one end and it is shifted towards next bit ~~Right~~ with every clock pulse applied to it. So, a 4-bit SISO shift register requires 4 clock pulses to store the data & 4 clock

pulses to retrieve the data.

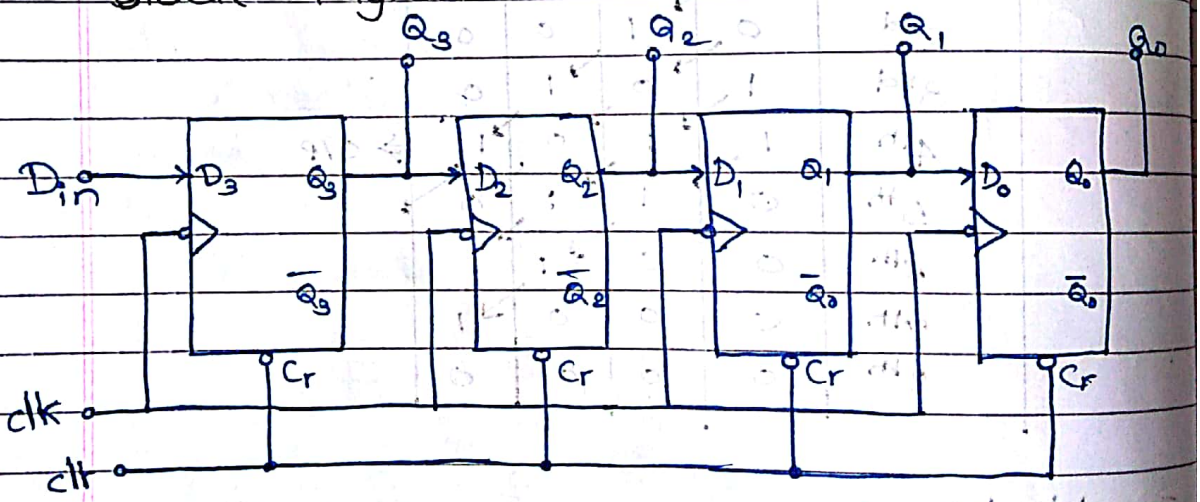
- Timing Diagram:

I/P = 1101



2. Serial In Parallel Out Shift Register (SIPO):

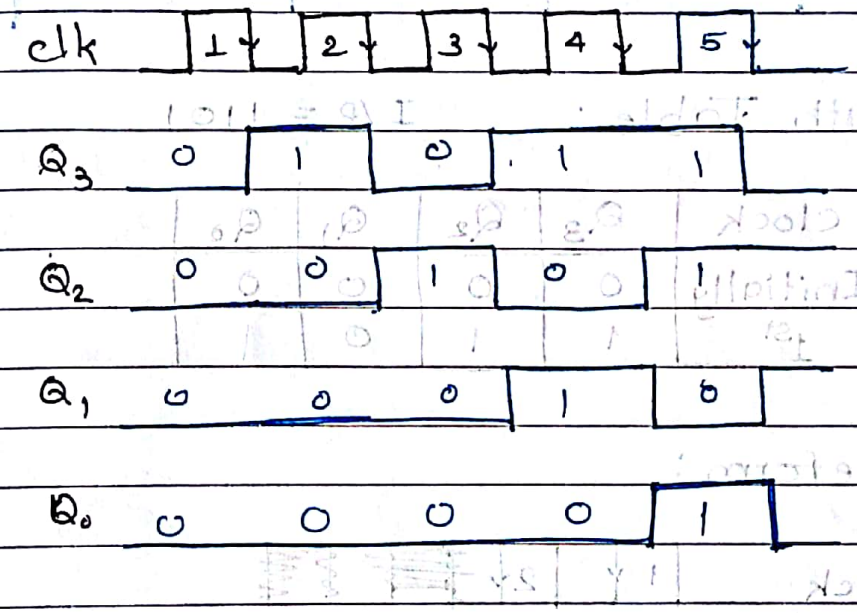
- Block diagram:



- Truth table: I/P = 1101

Clock	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1st	1	0	0	0
2nd	0	1	0	0
3rd	1	0	1	0
4th	1	1	0	1

- Waveform:



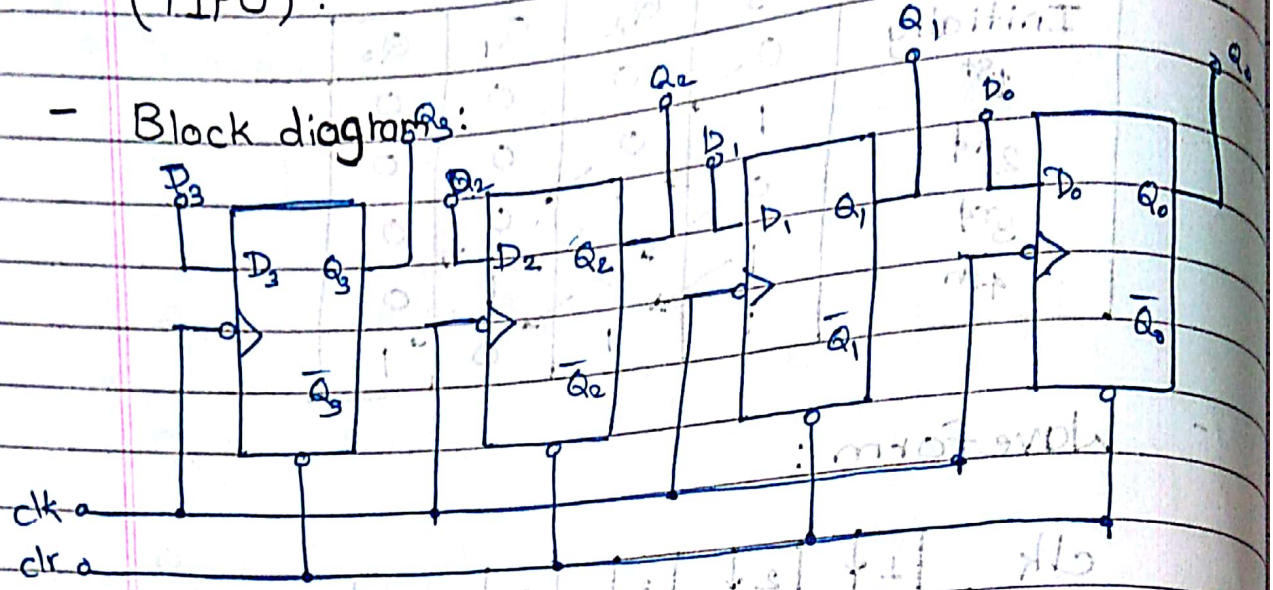
- Working:

- In this, the input is applied to the register serially whereas & the output is taken out parallelly.

- So, it needs total 4 clock pulses to store the data and as soon as we enter the data, we get the output.

3. Parallel In Parallel Out Shift Register (PIPO):

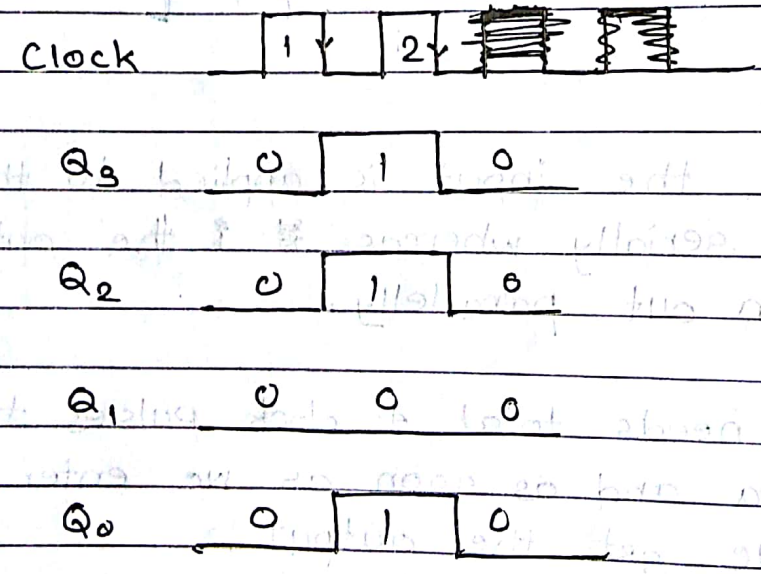
- Block diagram:



- Truth Table: I/P = 1101

Clock	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1st	1	1	0	1

- Waveform:



- Working:

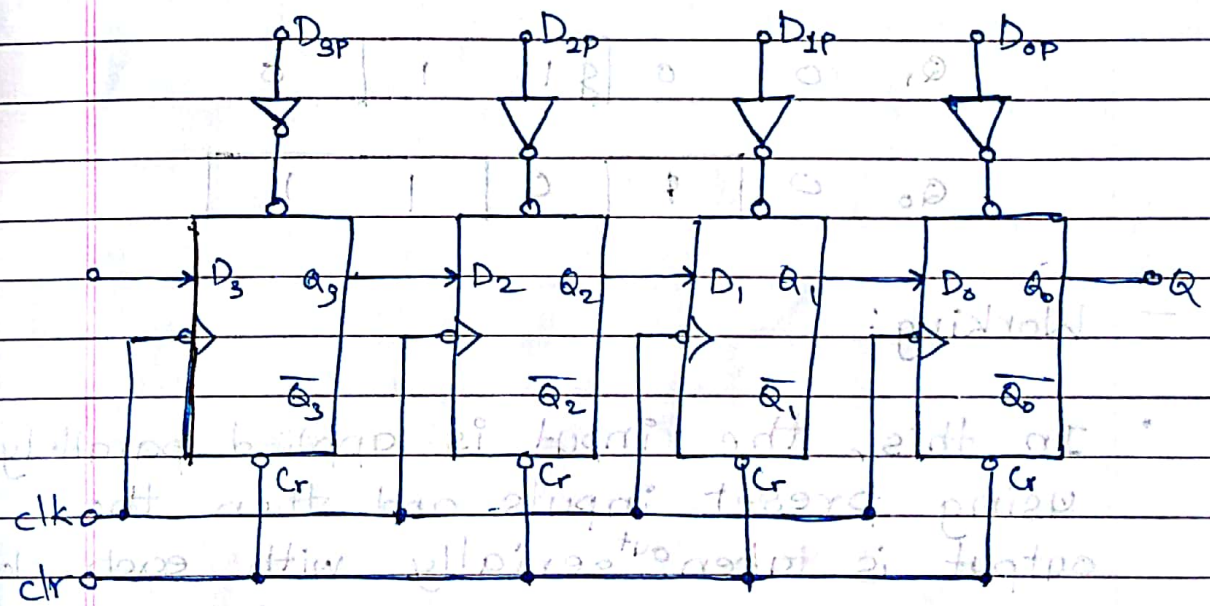
- In this, the input is applied to all the flip flop parallelly at the same time.

- When the clock is applied, the output is displayed at the same time.

- So, it requires only one clock pulse to store & display the output.

4. Parallel In ~~Parallel~~ Serial Out Shift Register (PISO):

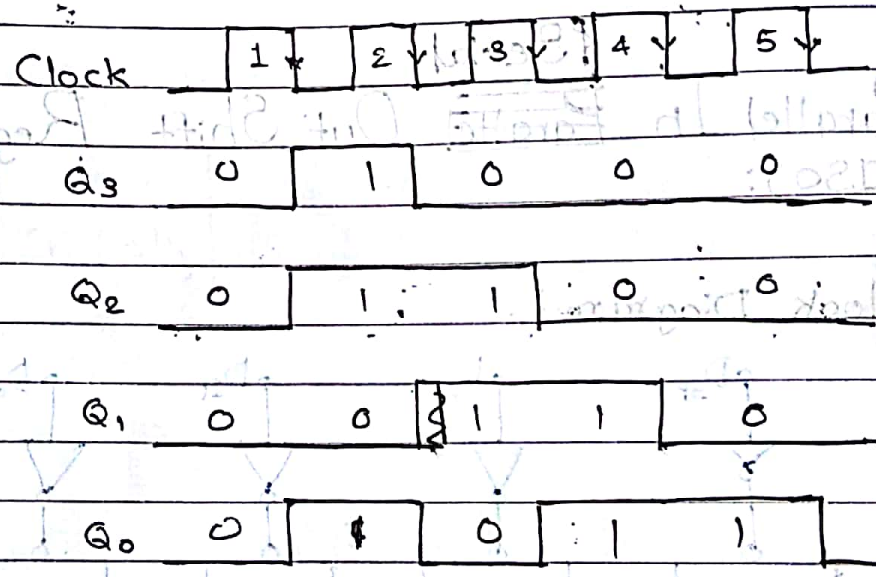
- Block Diagram:



- Truth Table: I/P = 1101

Clock	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1st	1	1	0	1
2nd	0	1	1	0
3rd	0	0	1	1
4th	0	0	0	1
5th	0	0	0	0

- Timing Diagram:

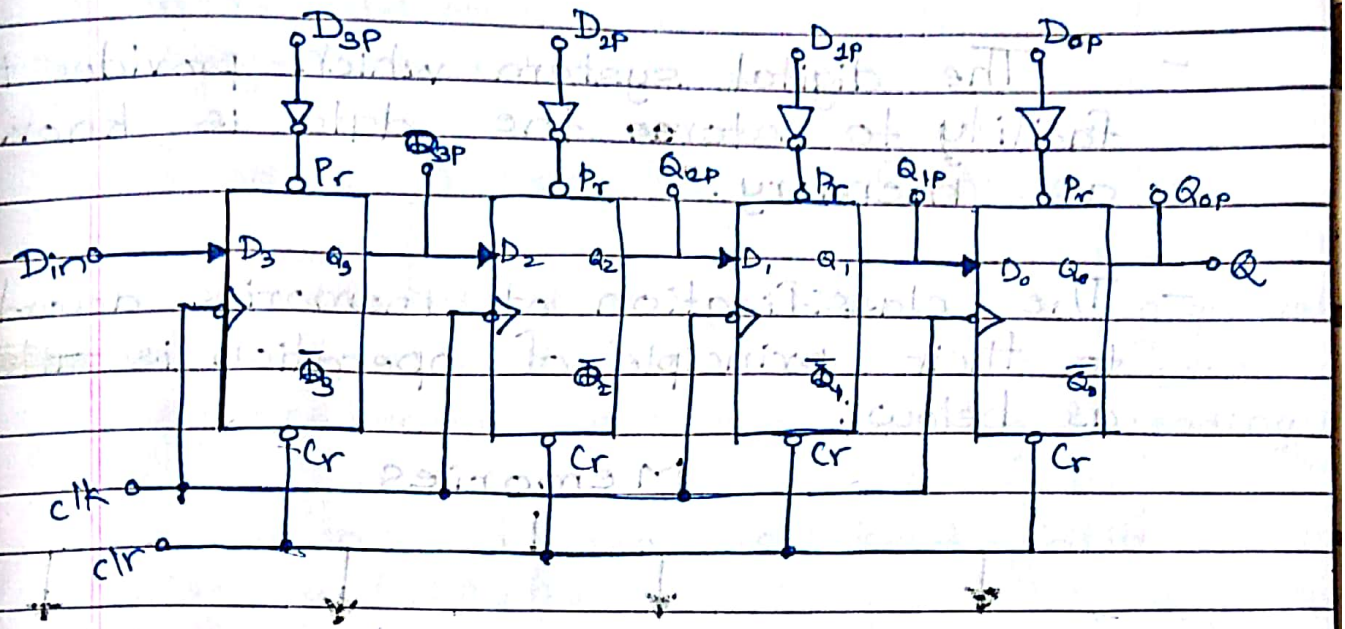


- Working:

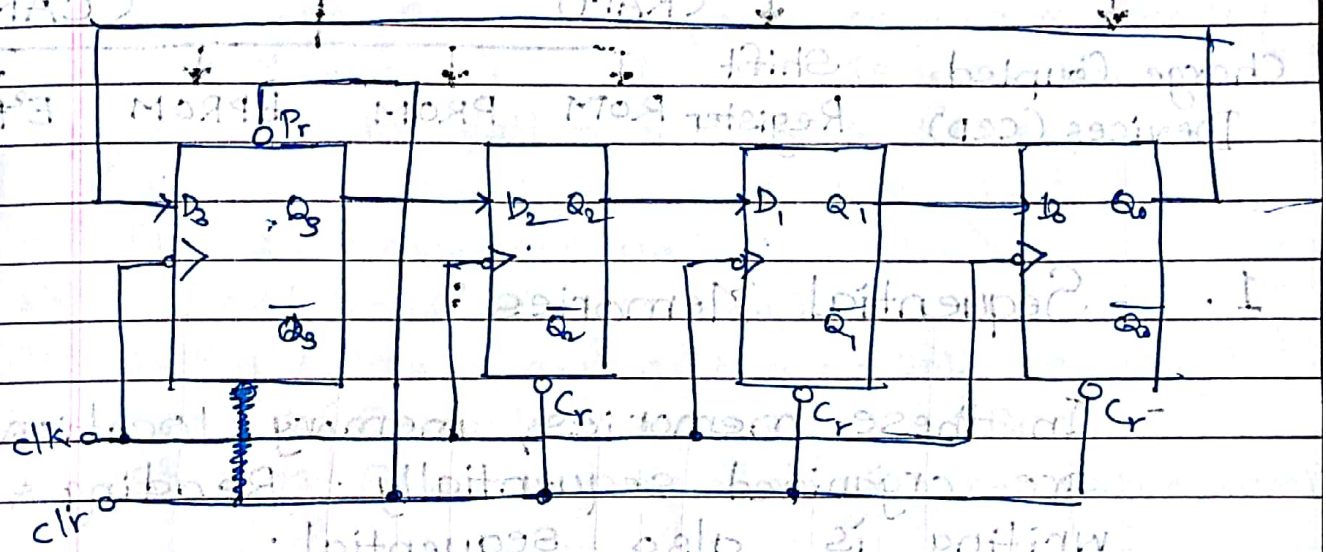
- In this, the input is applied parallely using preset inputs and then the output is taken out serially with each clo

- So, it requires no clocks to store the data as it is given via preset input a it requires 4 clocks to get output.

* Universal Shift Register :



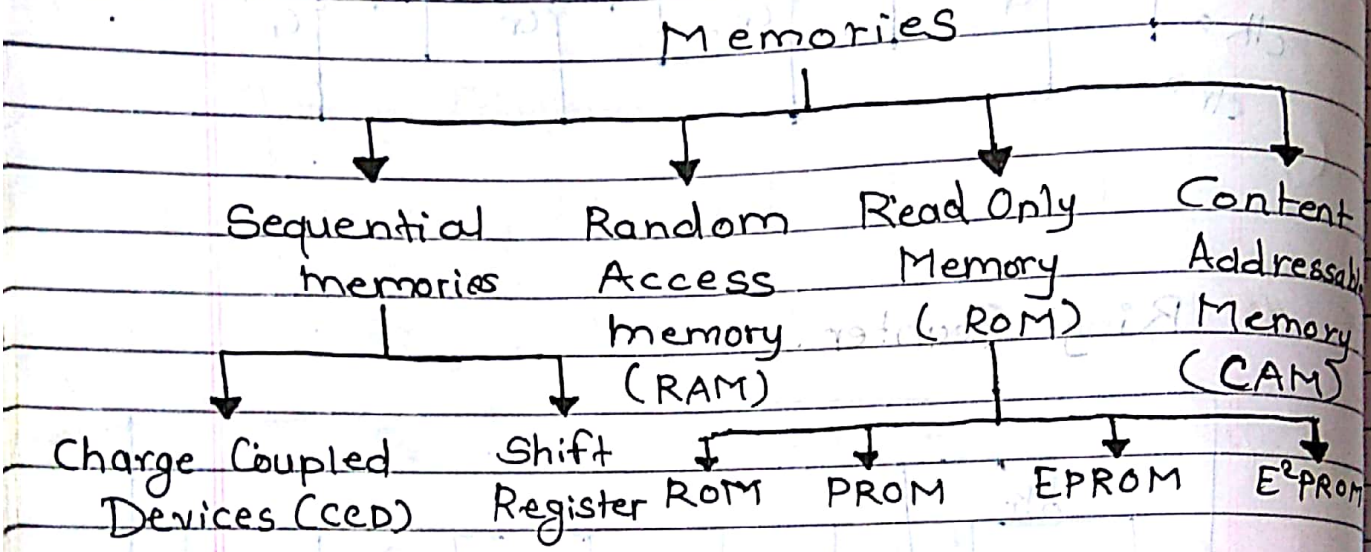
* Ring Counter :



* MEMORIES:

- "The digital system which provides the facility to stores the data is known as memory."

- The classification of memories according to their principle of operation is as below:



1. Sequential Memories:

- In these memories, memory locations are organised sequentially. Reading & writing is also sequential.

- So, the time required for accessing a memory location for reading or writing operation is different for different locations.

- It is classified as:
1) Shift Register
2) Charge Coupled Devices (CCD)

2. Random Access Memory (RAM):

- In this, the memory locations are organized in such a way that any memory location requires equal time for writing & reading.
- It is also known as Read & Write Memory (R/WM).

3. Read-Only Memory (ROM):

- This memory is designed only for reading information from it. It is used to store information which is fixed.
- It is classified as:
1) ROM (Read-Only Memory)
2) PROM (Programmable Reaad-Only Memory)
3) EPROM (Erasable Programmable ROM)
4) E^2PROM (Electrically Erasable Programmable ROM)

1) ROM:

It can be ~~mer~~ programmed only at the time of manufacturing. The data stored in it cannot be changed after fabrication.

2) PROM:

- It can be programmed using a special circuit - & PROM programmer.
- PROM can be programmed only once after which its constants are permanently fixed as in a ROM.

3) EPROM:

- It can be programmed using EPROM programmer.
- It can be erased by exposing to strong UV light for 20 minutes or longer.
- So, it can be programmed again & again.

4) E^2PROM:

- It is also known as EAPROM (Electrically Alterable PROM) or EEPROM (Electrically Erasable PROM).
- It can be erased by applying a voltage of proper polarity & amplitude.

* Comparison between RAM & ROM:

	RAM	ROM
1	RAM stands for Random Access Memory	ROM stands for Read-Only Memory
2	Both read & write operations can be performed.	Only read opration can be performed.
3	It is volatile	It is non volatile.

4	Types: Static RAM and Dynamic RAM	Types: PROM, EPROM, E^2PROM
5	Applications: Computer, calculator	Applications: Computer, microprocessor

* Comparison between EPROM & E^2PROM

	EPROM	E^2PROM
1	EPROM stands for Erasable Programmable Read-Only Memory.	E^2PROM stands for Electrically Erasable Programmable Read-Only Memory.
2	It requires UV rays to erase EPROM.	It can be programmed and erased electrically.
3	It requires 20-30 min. for erasing contents.	It requires only 10 ms. for erasing.
4	The chip must be removed for erasing & reprogramming.	It does not require the chip to be removed.
5	It has high density	It has low density
6	It is cheaper	It is expensive.