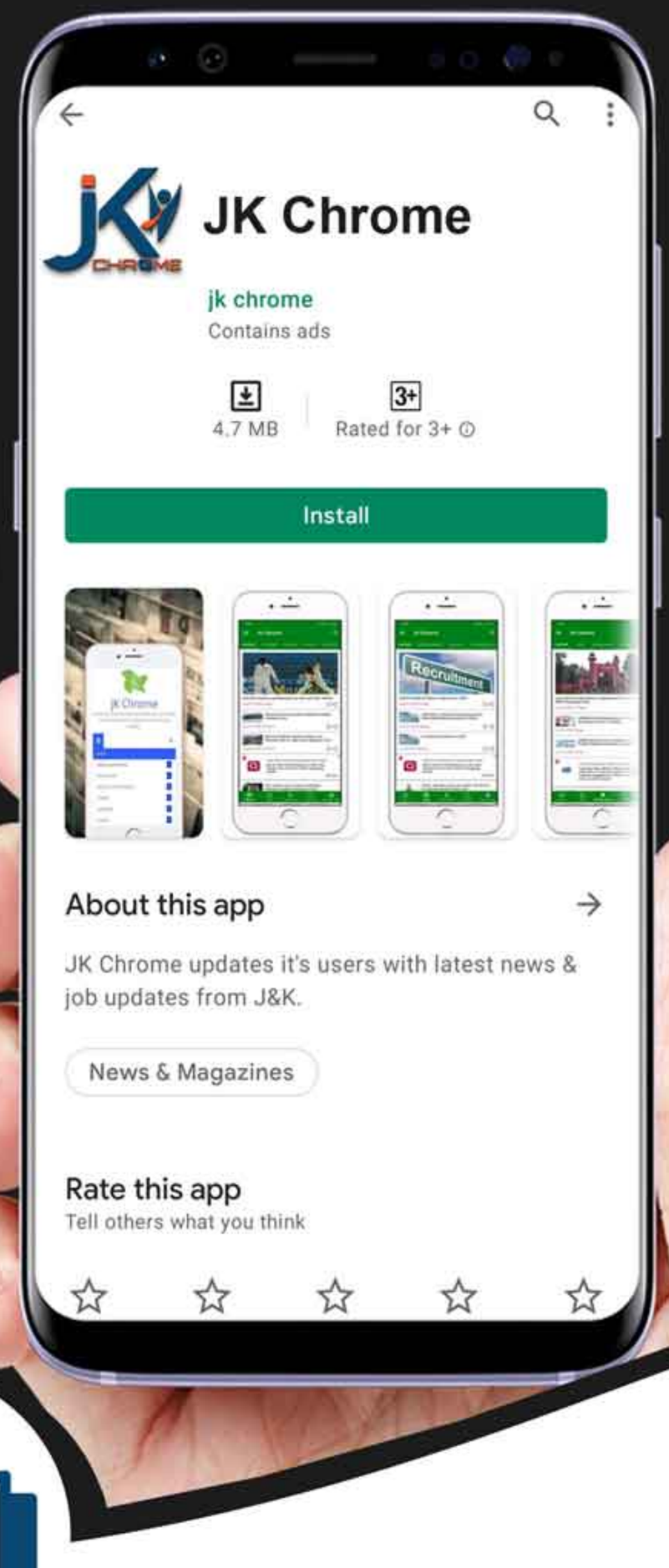
JK Chrome

JK Chrome | Employment Portal



Rated No.1 Job Application of India

Sarkari Naukri
Private Jobs
Employment News
Study Material
Notifications

JOBS

NOTIFICATIONS

G.K

STUDY MATERIAL

GET IT ON Google play

JK Chrome
jk chrome
Contains ads

Click here to DOWNLOAD

www.jkchrome.com | Email : contact@jkchrome.com

Digital Electronics

Index

Number System

Binary Codes

- Binary codes are codes that are represented in a binary system with modification from original ones.
- In general, *N* bits can represent up to 2^N distinct values.
- Conversely, to represent a range of M values, the number of bits required is.

1 bit	→	represents up to 2 values (0, 1)
2 bits	→	represents up to 4 values (00, 01, 10, 11)
3 bits	→	represents up to 8 values (000, 001, 010, 011, 100, 101, 110, 111)
4 bits	→	represents up to 16 values (0000, 0001, 0010, …, 1110, 1111)
32 values	→	requires 5 bits
40 values	→	requires 6 bits
64 values	→	requires 6 bits
100 values	→	requires 7 bits
1024 values	→	requires 10 bits

Logic Family

- The *base* or *radix* of a number system is the number of digits present. The decimal numeral system has a base or radix of 10, where the set of 10 symbols (digits) is {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. The weights are in powers of ten.
- In general, a bases-*b* number $(a_n\ a_{n-1} \ldots a_0\ .\ f_1\ f_2 \ldots f_m)_b$ has the value
  - $(a_n \times b^n) + (a_{n-1} \times b^{n-1}) + \ldots + (a_0 \times b^0) + (f_1 \times b^{-1}) + (f_2 \times b^{-2}) + \ldots + (f_m \times b^{-m})$
- **Weighted Binary System:** Weighted binary codes are those which obey the positional weighting principles, each position of a number represents a specific weight.

  e.g., 8421, 2421, 5211

- **Sequential Code:** A code is said to be sequential when two subsequent codes, seen as numbers in the binary representation, differ by one. The 8421 and excess-3 codes are sequential, whereas the 2421 and 5211 codes are not.
- **Non-weighted Codes:** non-weighted codes are codes that are not positionally weighted. That is each position within the binary number is not assigned a fixed value.

- **Reflective Code:** A code is said to be reflective when the code for 9 is a complement for the code for 0 and so is for 8 and 1 codes, 7 and 2, 6 and 3, 5 and 4. Codes 2421, 5211, and excess-3 are reflective, whereas the 8421 code is not.

BCD (Binary Coded Decimal)

- It is a straight assignment of the binary equivalent. To encode a decimal number using the common BCD encoding. Each decimal digit is stored in a 4-bit number.

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

- BCD encoding for number 127 would be

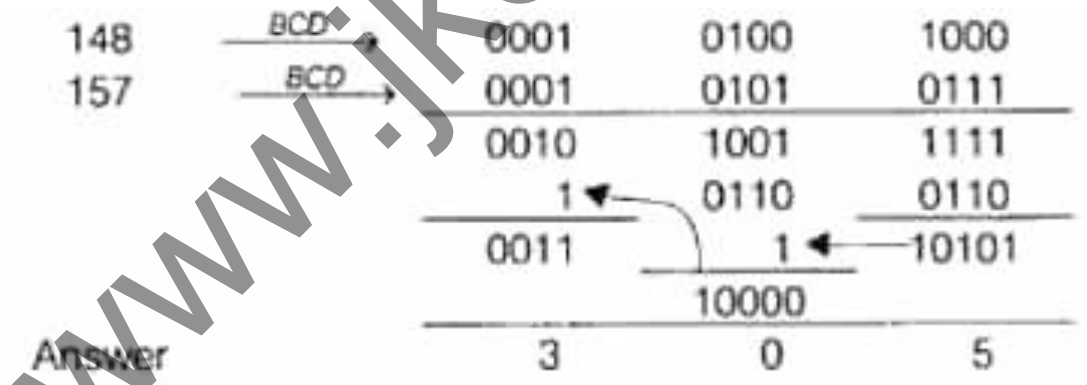
  1 2 7

  (0001 0010 0111) → BCD equivalent of 127

  whereas the pure binary number would be $(01111111)_2$

- **BCD**

  Add (148 + 157) = ?



When the sum of 2 digits is greater than or equal to 9, then we need to add 6 i.e., 0110.

- **2421 Code**

This is a weighted code, its weights are 2, 4, 2, and 1. A decimal number is represented in 4-bit form and the total 4 bits weight is

2 + 4 + 2 + 1 = 9.

Hence, the 2421 code represents the decimal numbers from 0 to 9.

Decimal	0	1	2	3	4	5	6	7	8	9
2421	0	1	10	11	100	1011	1100	1101	1110	1111

- **Excess-3 Code**

Excess-3 is a non-weighted code used to represent decimal numbers. The code derives its name from the fact that each binary code is the corresponding 8421 code plus 0011 (3).

e.g.,

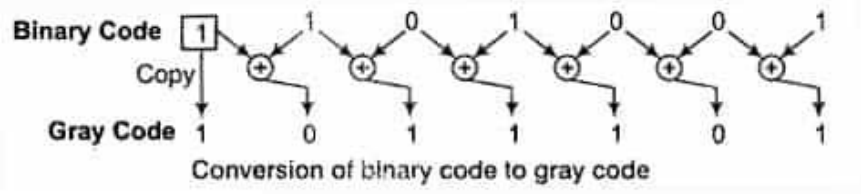
Decimal	8421	Excess-3
8	1000	1000 + 0011 = 1011
6	110	0110 + 0011 = 1001

- **Gray Code**

This is a variable weighted code and is cyclic. This means that it is arranged so that every transition from one value to the next value involves only one-bit Change.

- **Binary to Gray Code Conversion**

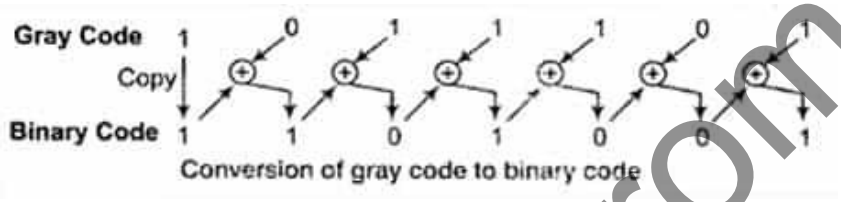
1. Write down the number in binary codes.
2. The Most Significant Bit (MSB) of the gray code will be the same as the MSB of binary code.
3. Perform XOR operation on MSB and next bit to the MSB in a binary number.
4. Repeat step 3 till all bits of the binary number have been XO Red, the resultant code is the gray code equivalent to the binary code.



Conversion of binary code to gray code

- **Gray Code to Binary Conversion**

1. Start with the MSB of gray coded numbers.
2. Copy this bit as the MSB of the binary number.
3. Now, perform the Ex-OR operation of this bit with the next bit of the binary number.
4. Repeat step 3 till all bits of gray coded numbers have been used in the XOR operation. The resultant number is the binary equivalent of the gray number.



Conversion of gray code to binary code

Complements

- Complements are used in a digital computer system for simplifying the subtraction operation and for logical manipulation.
- There are two types of complements for each baser system,

1. The r' s complement
2. The (r' – 1)'s complement

- **The r's Complement**

Given a positive number N with baser with an integer part of n digits. The r's complement of N is defined as r^n – N for N ≠ 0 and 0 for N = 0.

e.g., 10's complement of $(25.639)_{10}$ is $(10^2 - 25.639)$

100 - 25.639 = 74 – 361, here the number of digits in integer part is 2

means n = 2

- **The (r – 1)'s Complement**

Given, a positive number N in baser with an integer part of n digits and a fraction part of m digits, the (r – 1)'s complement of N is defined as $r^n - r^{-m} - N$.

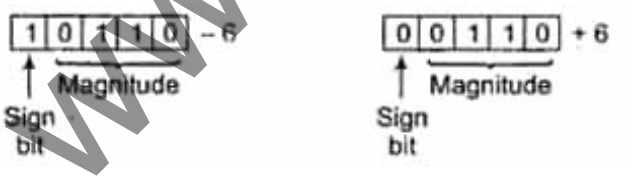
- 1's complement of $(52520)_{10}$ is $(10^5 - 1 - 52520)$
  $= 99999 - 52520 = 47479$
  Because number of integer part is 5, so $r^n = 10^5$ and no fractional part is Present' so $r^{-m} = 10^{-0} = 1$
- 1's complement of $(0.3267)_{10}$ is $(10^0 - 10^{-4} - 0.3267)$
  $= 1 - 0.0001 - 0.3267$
  $= 0.9999 - 0.3267 = 0.6732$
  No integer part, so $10^n = 10^0 = 1$
- 1's complement of $(101100)_2$ is $(2^6 - 2^0)_{10} - (101100)_2$
  $= (64 - 1)_{10} - (101100)_2$
  $= (63)_{10} = (101100)_2$
  $= 111111 - 101100 = 010011$
- **Key Points**
- 520 → Here, n = 3, but (052) → here n = 2.
- In the latter example, 0 is of no significance.

Representation of Integers

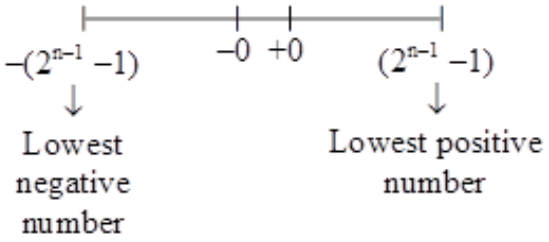
- The are three possible ways to represent a number

1. Signed magnitude method
2. 1's complement method
3. 2's complement method

- **Signed Magnitude Method**
- The number is divided into two parts, one is the sign bit and another part for magnitude, In the example we are using the 5-bit register to represent –6 and +6.



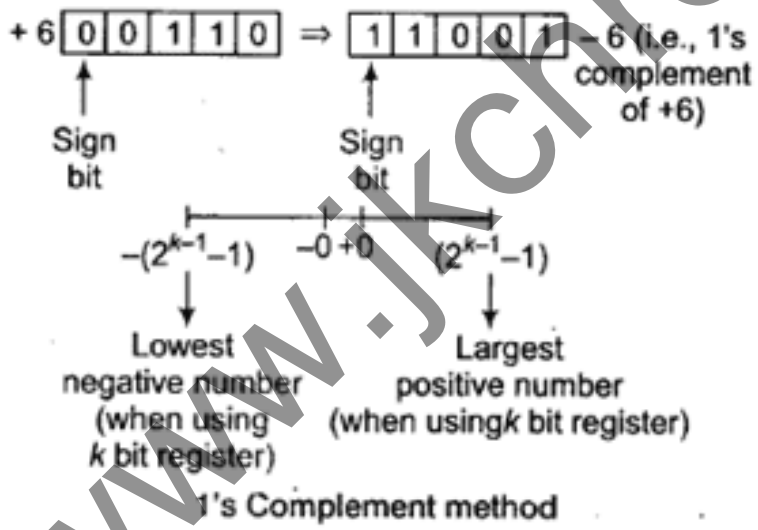
- **The range of Number** For n bit register, MSB will be a sign bit and (n – 1) bits will be the magnitude.



- **Key Points**
- The drawback of the signed magnitude method is that 0 will be having 2 different representations one will be 10000 i.e., –0 and the other one will be 00000 + 0.

1's Complement Method

- Positive numbers are represented in the same way as in the sign-magnitude method. If the number is negative, then it is represented using 1's complement method. For this, we first need to represent the number with a positive sign and then take 1's complement of this number.
- e.g., Suppose we are using a 5-bit register. The representation of –6 will be as below.

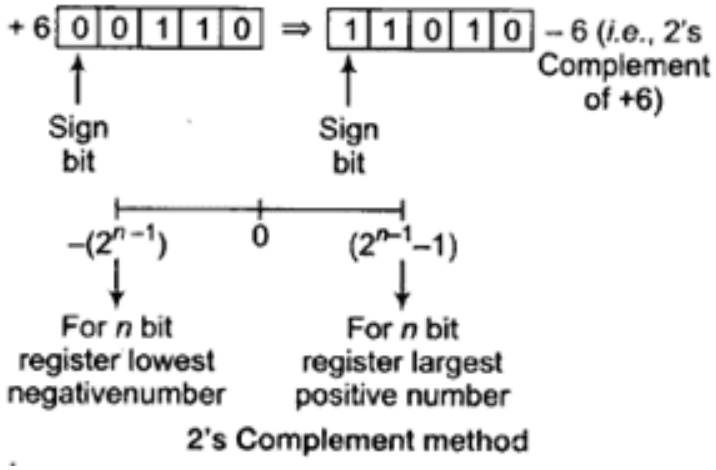


1's Complement method

- **Key Points**
- The only drawback of 1's complement method is that there are two different representations for zero, one is –0, and the other is +0.

2's Complement Method

- Positive numbers are represented in the same way as in sign-magnitude. For representing a negative number, we take 2's complement of the corresponding positive number.



Properties of 2's Complement

- 2's complement representation allows the use of binary arithmetic operations on signed integers, yielding the current 2's complement result.
- **Positive Numbers** Positive 2's complement numbers are represented as the simple binary.
- **Negative Numbers** Negative 2's complement numbers are represented as the binary number that when added to a positive number of the same magnitude equals zero.

Logic Gates

A logic gate is an idealised or physical device implementing a Boolean function, that is, it performs a logical operation in one or more logical inputs and produces a single logical output.

The logic gates can be classified as

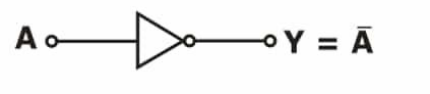
- NOT, AND, OR are basic gates.
- NAND, NOR are universal gates.
- EXOR, EXNOR are an arithmetic circuit or code converter or comparators.

NOT Gate (Inverter)

Truth Table for NOT Gate:

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

Circuit Symbol for NOT Gate:



AND Gate:

Truth Table for AND Gate:

Inputs		Output
A	B	Y = AB
0	0	0
0	1	0
1	0	0
1	1	1

Circuit Symbol for AND Gate:

Properties of AND logic:

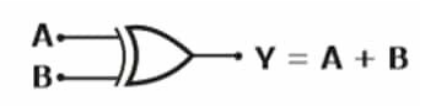
1. Commutative Law: AB = BA
2. Associative Law: ABC = (AB) C = (AC)B = A(BC)

OR Gate:

Truth Table:

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

Circuit Symbol for OR Gate:



Properties of OR logic:

1. Commutative Law: A + B = B + A
2. Associative Law: (A + B + C) = (A + B) + C = A + (B+ C)

NAND Gate:

Truth Table:

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

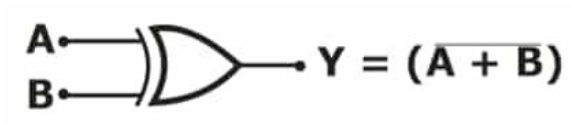
Circuit Symbol for NAND Gate:

NOR Gate:

Truth Table:

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

Circuit Symbol for NOR Gate:



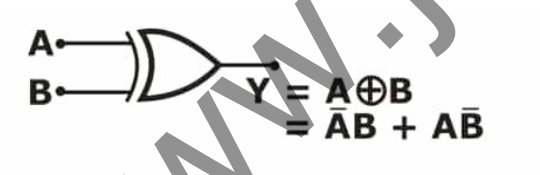
- NOR gate follows commutative law but not follow associative law

EXOR Gate:

Truth Table:

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

Circuit Symbol for EXOR Gate:



Properties of EXOR Logic:

- Enable input = 0
- Disable input = 1
- It is also called stair case switch.
- It is widely used in parity generation and detection.
- When both the inputs are different, then output becomes high or logic 1.
- When both the inputs are same, then output becomes low or logic 0.

Note:

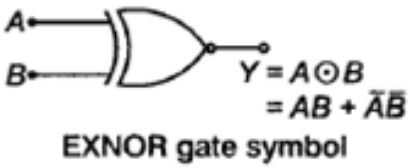
$A \oplus A = 0; A \oplus \bar{A} = 1; A \oplus 0 = A; A \oplus 1 = \bar{A}$

EX-NOR Gate:

Truth Table:

Inputs		Output
A	**B**	**Y = $A \odot B$**
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Circuit Symbol for EX-NOR Gate:



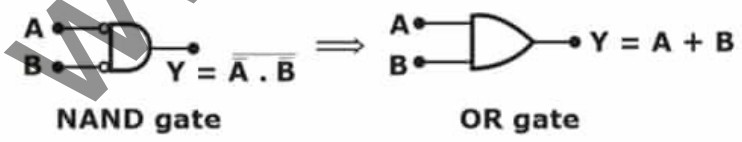
EXNOR gate symbol

Properties of EXNOR Gate:

- Enable input = 1
- Disable input = 0
- When both the inputs are same, then output .becomes high or logic 1.
- When both the inputs are different, then output becomes low or logic 0.

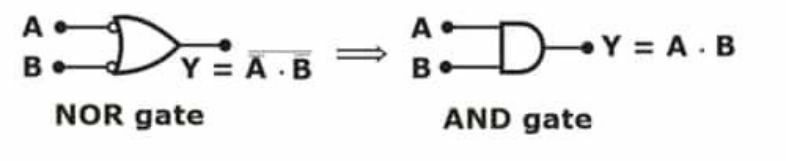
Logic Gate Conversions

- **OR Gate using NAND Gate:**

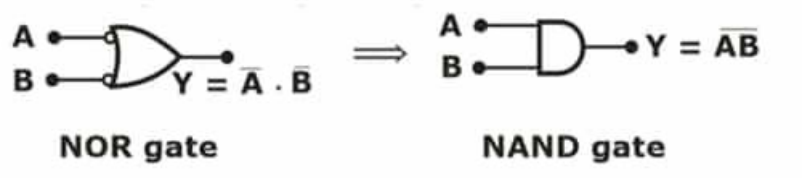


NAND gate OR gate

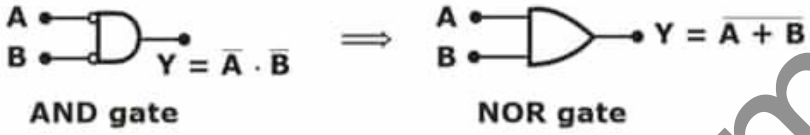
- **AND Gate using NOR Gate:**



- **NAND Gate using NOR Gate**



- **NOR Gate using AND Gate**



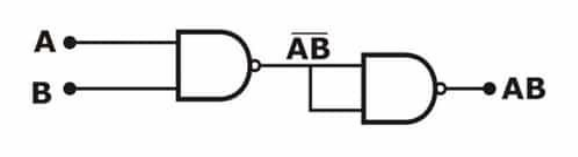
NAND and NOR Gate as Universal Gate

Logic gates	Required number of NAND gate	Required number of NOR gate
Not	1	1
AND	2	3
OR	3	2
EXOR	4	5
EXNOR	5	4

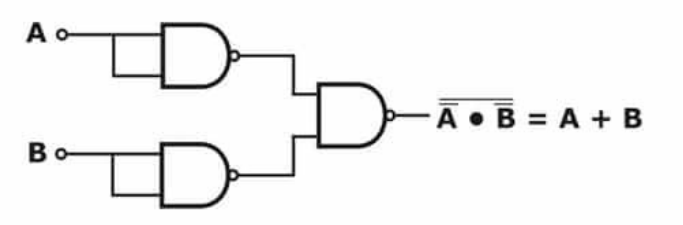
Designing different gates using NAND Gate as Universal Gate:

(i) NOT Gate:

(ii) AND Gate:

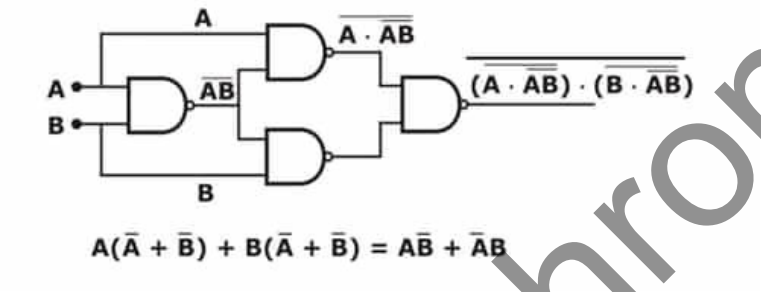


(iii) OR Gate:



h

(iv) EXOR Gate:



Minimization of Boolean Expression

1. Introduction to Boolean Algebra

- Boolean algebra is an algebraic structure defined on a set of elements together with two binary operators (+) and (.)
- A **variable** is a symbol, for example, 'A' used to represent a logical quantity, whose value can be **0** or **1**.
- The **complement of a variable** is the inverse of a variable and is represented by the variable over bar.
- A **literal** is a variable or the complement of a variable.
- **Boolean Value**
  - The value of Boolean variable can be either 1 or 0.
- **Boolean Operators:** There are three basic Boolean operators
  - AND (·) operator
  - OR (+) operator
  - NOT operator

2. Duality

- If an expression contains only the operations AND, OR and NOT. Then, the dual of that expression is obtained by replacing

each AND by OR,

each OR by AND,

all occurrences of 1 by 0,

all occurrences of 0 by 1.

The principle of duality is useful in determining the complement of a function.

- Logic expression: $(x \cdot y' \cdot z) + (x \cdot y \cdot z') + (y \cdot z) + 0$,
- Duality of above logic expression is: $(x + y' + z) \cdot (x + y + z') \cdot (y + z) \cdot 1$

Boolean Function:

- Any Boolean functions can be formed from binary variables and the Boolean operators.
- For a given value of the variable, the function can take only one value either 0 or 1.
- A Boolean function can be shown by a truth table. To show a function in a truth table we need a list of the 2^n combinations of 1's and 0's of the 'n' binary variables and a column showing the combinations for which the function is equal to 1 or 0. So, the table will have 2^n rows and columns for each input variable and tile final output.
- A function can be specified or represented in any of the following ways:
  - A truth table
  - A circuit
  - A Boolean expression
  - SOP (Sum Of Products)
  - POS (Product of Sums)
  - Canonical SOP
  - Canonical POS
- **Important Boolean operations over Boolean values:**

$0 \bullet 0 = 0$
$1 \bullet 1 = 1$
$0 \bullet 1 = 1 \bullet 0 = 0$
$0' = 1$
$1 + 1 = 1$
$0 + 0 = 0$
$1 + 0 = 0 + 1 = 1$
$1' = 0$

3. Basic Theorems

Law/Theorem	Law of Addition	Law of Multiplication
Identity Law	$x + 0 = x$	$x \cdot 1 = x$
Complement Law	$x + x' = 1$	$x \cdot x' = 0$
Idempotent Law	$x + x = x$	$x \cdot x = x$
Dominant Law	$x + 1 = 1$	$x \cdot 0 = 0$
Involution Law	$(x')' = x$	
Commutative Law	$x + y = y + x$	$x \cdot y = y \cdot x$
Associative Law	$x+(y+z) = (x+y)+z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Distributive Law	$x \cdot (y+z) = x \cdot y+x \cdot z$	$x+y \cdot z = (x+y) \cdot (x+z)$
Demorgan's Law	$(x+y)' = x' \cdot y'$	$(x \cdot y)' = x' + y'$
Absorption Law	$x + (x \cdot y) = x$	$x \cdot (x + y) = x$

- **Important Theorems used in Simplification**

NOT-Operation theorem:

- $\bar{\bar{A}} = A$

AND-Operation theorem:

- $\begin{bmatrix} A \cdot A = A \\ A \cdot 1 = A \\ A \cdot 0 = 0 \\ A \cdot \bar{A} = 0 \end{bmatrix}$

OR-Operation theorem:

- $\begin{bmatrix} A + A = A \\ A + 0 = A \\ A + 1 = 1 \\ A + \bar{A} = 1 \end{bmatrix}$

Distribution theorem:

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

Others:

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

- Consensus Theorem: This theorem is used to eliminate redundant term. It is applicable only when a boolean function contains three variables. Each variable used two times. Only one variable is complemented or uncomplemented. Then the related terms so that complemented or uncomplemented variable is the answer.

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

4. MinTerm & MaxTerm

- **Minterm:**
  - Each product term is known as a minimum term that contains all the variables used in a function.
  - A minterm is also called a **canonical product term**.
  - A minterm is a product term, but a product term may or may not be a minterm.
- **Maxterm:**
  - Each sum term is known as a maximum term that contains all of the variables used in the function.
  - A **maxterm** is a sum term of all variables in which each variable is either in complemented form or in uncomplemented form.
  - A maxterm is also called a **canonical sum term**.
  - A maxterm is a sum term, but a sum term may or may not be a maxterm.
- The following are examples of product term, minterm, sum term, and maxterm for a function of three variables a, b, and c:

- product terms: a, ac, b'c, abc, a'bc, a'b'c', …
- minterms: ab'c, abc, a'b'c, a'b'c', …
- sum terms: a, (a+b), (b+c), (a'+b), (a'+b'), …
- maxterms: (a+b+c), (a+b'+c), (a'+b'+c'), …

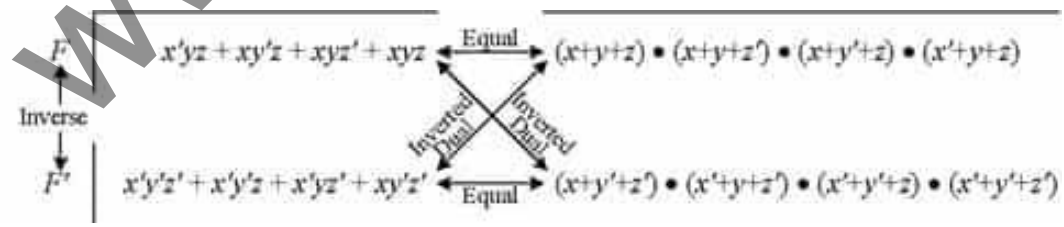
- **Representations of Minterm and Maxterm:**

x	y	z	Minterm	Minterm Notation	Maxterm	Maxterm Notation
0	0	0	$x'y'z'$	$m0$	$x + y + z$	$M0$
0	0	1	$x'y'z$	$m1$	$x + y + z'$	$M1$
0	1	0	$x'y z'$	$m2$	$x + y' + z$	$M2$
0	1	1	$x'y z$	$m3$	$x + y' + z'$	$M3$
1	0	0	$x y'z'$	$m4$	$x' + y + z$	$M4$
1	0	1	$x y'z$	$m5$	$x' + y + z'$	$M5$
1	1	0	$x y z'$	$m6$	$x' + y' + z$	$M6$
1	1	1	$x y z$	$m7$	$x' + y' + z'$	$M7$

- **Note:** With' *n'* variables maximum possible minimum and maximum terms = 2^n
- With' *n'* variables maximum possible logic expression = 2^{2^n}

5. SOP & POS

- **SOP (Sum of Product):** A sum of product expression is two or more OR functions of AND functions.
  - SOP expression is used when output becomes logic 1.
  - Example: $A\bar{B}C + \bar{A}\bar{B}C + A\bar{B}\bar{C}$
- **POS (Product of Sum):** It is the AND function of two or more OR function.
  - POS expression is used when output is logic '0'.
  - Example: $(A + B + C)\cdot(A + \bar{B} + C)\cdot(\bar{A} + \bar{B} + \bar{C})$
  -
- **Example:** SOP and POS Equivalences for function F and Its Inverse F'.



6. Duality Theorem

- To convert positive logic into negative logic and *vice-versa, a* dual function is used.
    - Change each AND sign by OR sign and *vice versa* (↔ +)
    - Complement any 0 or l appearing in expression.
    - Keep variable as it is.
    - Example:

$$A \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot C + AB\overline{C} \xleftrightarrow{\text{dual}} (A + \overline{B} + C) \cdot (\overline{A} + B + C) \cdot (A + B + \overline{C})$$

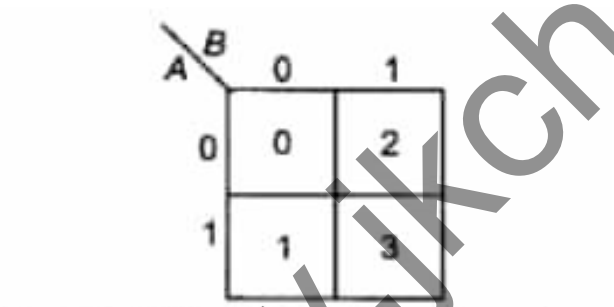
7. Minimization of Boolean Expressions

The following two approaches can be used for simplification of a Boolean expression:

- Algebraic method (using Boolean algebra rules)
- Karnaugh map method

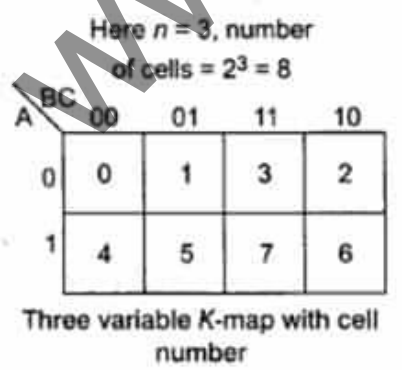
Representation of K-map: With n-variable Karnaugh-map, there are 2^n cells

- **2 –variable K Map:**



Two variable *K*-map with cell number

- **3 –variable K Map:**



Three variable *K*-map with cell number

- **4 –variable K Map:**

Here n = 4, number of cells = $2^n = 16$

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

Four variable *K*-map with cell number

- **NOTE:** Once the Karnaugh map has been populated with **1**s, **0**s and **X**s as specified the only task that remains is to **group adjacent terms of the same state** (usually **1**) in groups of 2 raised to any rational power, i.e. **1, 2, 4, 8, 16, 32, 64** and so on. The larger the group the simpler the final expression. It is also possible for groups to overlap. This is often done to achieve a larger group size, hence simplifying the final expression.

Minimization Procedure of Boolean Expression using K-map

- Construct a K-map.
- Find all groups of horizontal or vertically adjacent cells that contain 1.
    - Each group must be either rectangular or square with 1, 2, 4, 8, or 16 cells.
    - Each group should be as large as possible.
    - Each cell with 1 on the K-map must be covered at least once. The same
    - the cell can be included in several groups if necessary.
    - Select the least number of groups so as to cover all the 1's.
    - Adjacency applies to both vertical and horizontal borders.
- Translate each group into a product term. (Any variable whose value changes from cell to cell drops out from the term)
- Sum all the product terms.
- **Note:** Don't care conditions can be used to provide further simplification of a Boolean Expression.

Combinational Logic Circuits

1. Designing Combinational Circuits

The steps to design combinational circuits are as the following

- Understand the problem
- Find the required number of input and output variables
- Construct a truth table using the relationship between the input and output
- Obtain the Boolean function or the logical expression from the truth table using Karnaugh Map.
- Draw a logic circuit based on the obtained logical expression.

2. Arithmetic Circuits

Arithmetic circuits are used to perform addition and subtraction. Binary adder performs binary addition and binary subtractor performs binary subtraction.

Classification of Adder:

- Half Adder
- Full Adder

Classification of Subtractor:

- Half Subtractor
- Full Subtractor

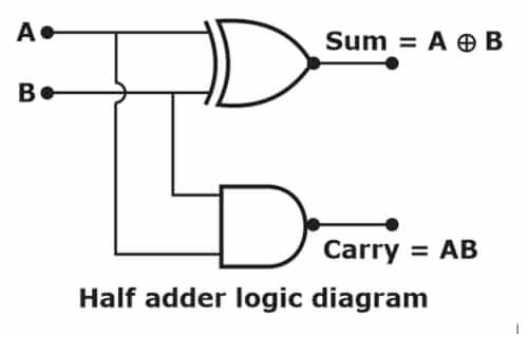
Half Adder

This circuit is used for addition of two one bit numbers.

- **The truth table of Half Adder:**

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

- **Half adder circuit:**

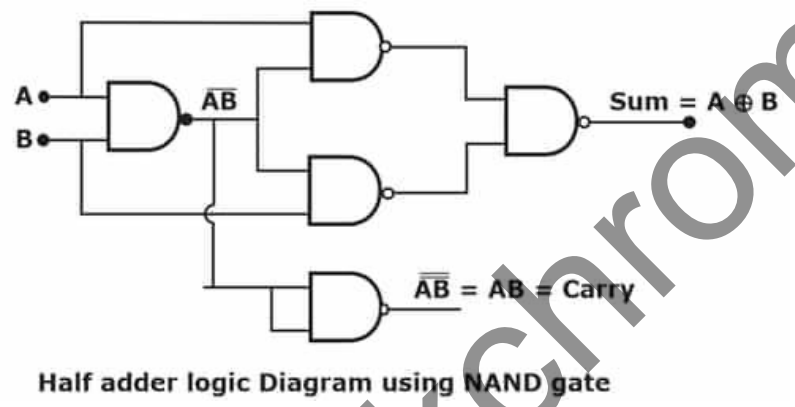


Half adder logic diagram

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

Carry (C) = AB

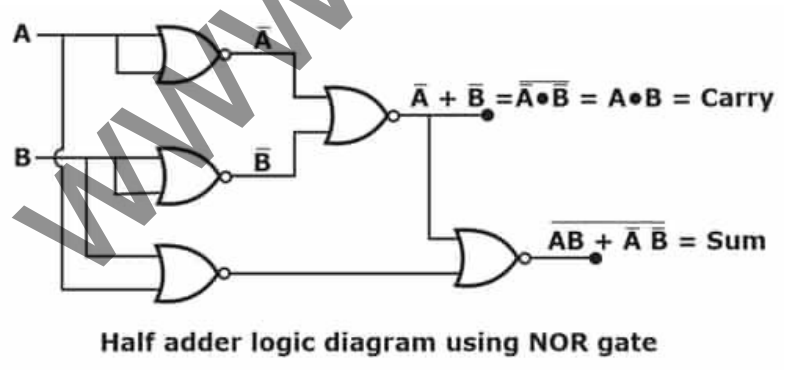
- **Implement of Half Adder Using NAND Gate:**



Half adder logic Diagram using NAND gate

Note: Required number of NAND Gates to implement Half Adder = 5

- **Implement of Half Adder Using NOR Gate:**

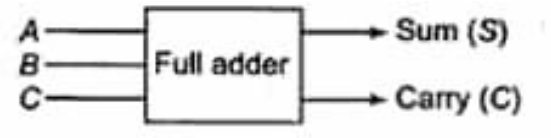


Half adder logic diagram using NOR gate

Note: Required number of NOR Gates to implement Half Adder = 5

Full Adder

A full adder is a combinational logic circuit that performs the arithmetic sum of three input bits. It consists of three inputs and two outputs.



- **The truth table for Full Adder:**

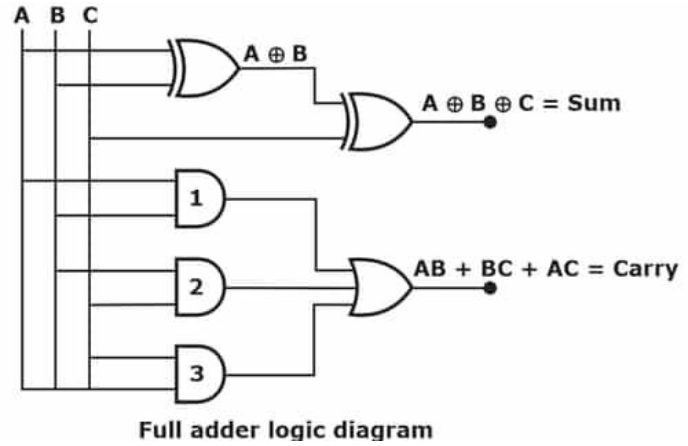
INPUT			OUTPUT	
A	**B**	**C(i)**	**S**	**C(o)**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adder Truth Table

- **The logic diagram of Full Adder:**

Sum (S) = $A \ominus B \ominus C = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

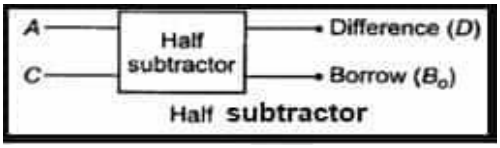
Carry (C_0) = AB + BC + AC



Full adder logic diagram

- A full adder = 2 Half adder + 1 OR Gate
- Required minimum number of NAND gate to implement FA = 9
- Required minimum number of NOR gate to implement FA = 9

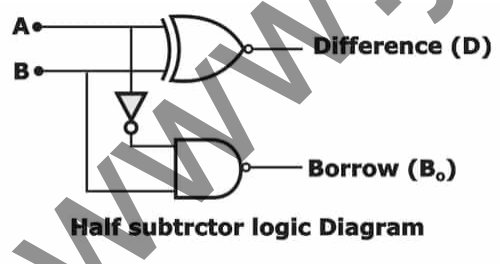
Half Subtractor



Half subtractor

- **Logic Diagram of Half Subtractor:**

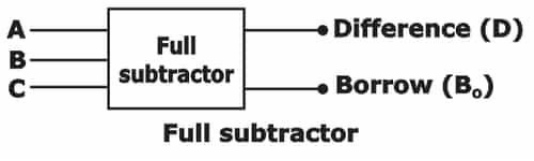
- Difference (D)
- Borrow (B_0) =
- To implement half subtractor the total number of NAND/NOR are required = 5



Half subtrctor logic Diagram

Full Subtractor

It is a combinational logic circuit that performs subtraction involving three bit namely minuend bit, subtrahend bit and borrows from the previous stage



Full subtractor

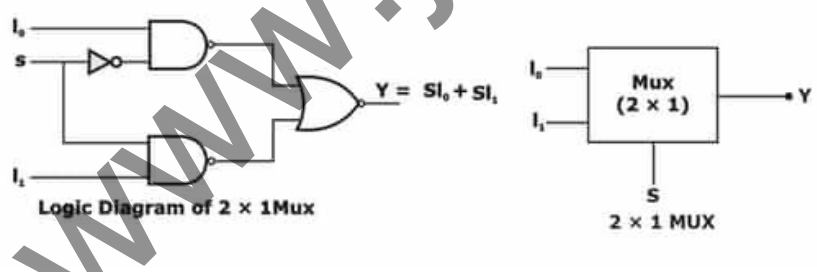
- Difference (D) $= A \oplus B \oplus C$
  Borrow $(B_0) = \bar{A}B + \bar{A}C + BC = \bar{A}B + \left(\overline{A \oplus B}\right) \cdot C$
- A full subtractor = 2 half subtractor + 1 OR gate
- To implement full subtractor of NAND/NOR gates are required = 9

3. Multiplexer (MUX)

- It is a combinational circuit that selects binary information from one of the many input lines and directs it to a single output line.
- The selection of a particular input line is controlled by a set of selection lines.
- MUX is also called: Many to one, Data selector, Universal circuit, or Parallel data serial.
- Multiplexing means transmitting a large number of information units over a smaller number of channels or lines. It is abbreviated as MUX.
- There are 2^n input lines and *n* selection lines whose bit combinations determine which input is selected.

m = 2^n implies n = log m where m = Number of data inputs, and n = Number of select lines.

2 × 1 MUX :



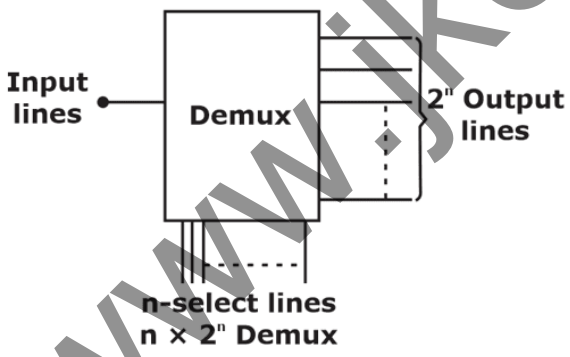
- **Implementation of one MUX using another MUX:**

Given MUX	To be Implemented MUX	Required Number of MUX
2×1	4×1	3
2×1	8×1	7
2×1	16×1	15
2×1	64×1	63
2×1	256×1	255
2×1	$2^n \times 1$	$(2^n - 1)$

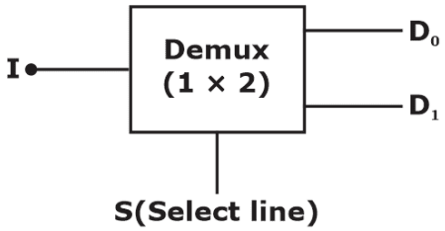
Given MUX	To be Implemented MUX	Required Number of MUX
2×1	4×1	3
4×1	16×1	$4 + 1 = 5$
4×1	64×1	$16 + 4 + 1 = 21$
8×1	64×1	$8 + 1 = 9$
8×1	256×1	$32 + 4 + 1 = 37$

4. Demultiplexer (DEMUX)

- It is a circuit that receives information on a single line and transmits this information on one of 2^n possible output lines.
- The selection of a specific output line is controlled by the bit values of *n* selected lines.



1 × 2 Demux:



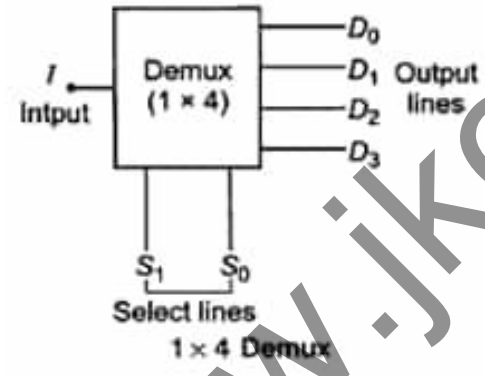
$D_0 = S'I$

$D_1 = SI$

- **The truth table of 1 × 2 Demux:**

S	D_0	D_1
0	0	1
1	1	0

1 × 4 Demux:

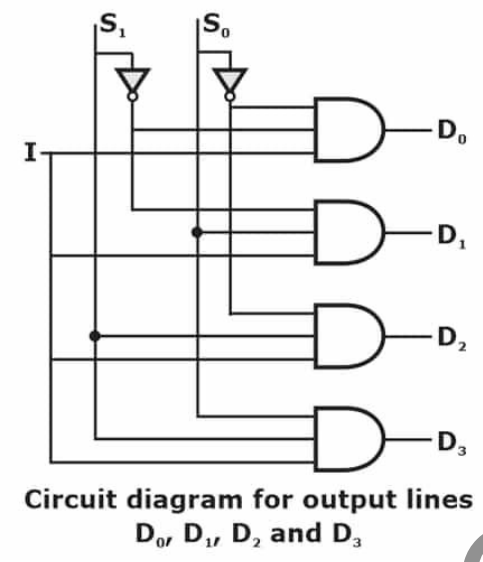


- $D_0 = \overline{S}_1\overline{S}_0 I$
- $D_1 = \overline{S}_1 S_0 I$
- $D_2 = S_1\overline{S}_0 I$
- $D_3 = S_1 S_0 I$

- **The Truth table of 1 × 4 Demux:**

Input	Select Lines	Output Lines
I	S_1 S_0	D_0 D_1 D_2 D_3
I	0 0	1 0 0 0
I	0 1	0 1 0 0
I	1 0	0 0 1 0
I	1 1	0 0 0 1

- **Circuit Diagram of 1 × 4 Demux:**



Circuit diagram for output lines D_0, D_1, D_2 and D_3

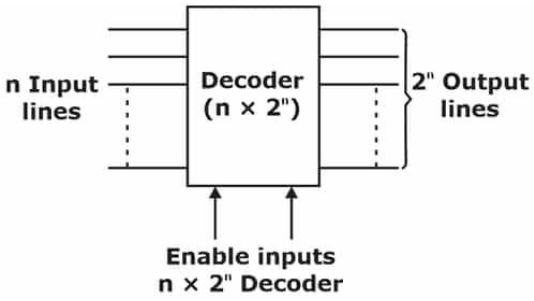
- **DEMUX Implementation using another DEMUX:**

Given DEMUX	To be Implemented DEMUX	Required Number of DEMUX
1 × 2	1 × 4	3
1 × 2	1 × 8	7
1 × 2	1 × 16	15
1 × 2	1 × 64	63
1 × 2	1×2^n	$(2^n - 1)$
1 × 4	1 × 6	5

5. Decoders

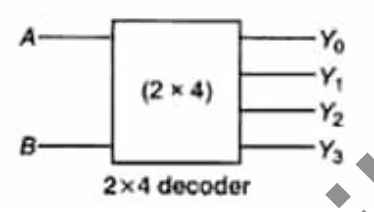
- A *decoder* is a combinational circuit that converts binary information from *n* input lines to maximum 2n unique output lines.

- If the *n*-bit decoded information has unused or don't-care combinations, the decoder output will have fewer than 2^n outputs.
- The decoders presented here are *n*-to-*m*-line decoders, where m ≤ 2^n. Their purpose is to generate the 2^n (or fewer) minterms of *n* input variables.



2 × 4 Decoder:

- $Y_0 = \overline{A}\overline{B}$
- $Y_1 = \overline{A}B$
- $Y_2 = A\overline{B}$
- $Y_3 = AB$

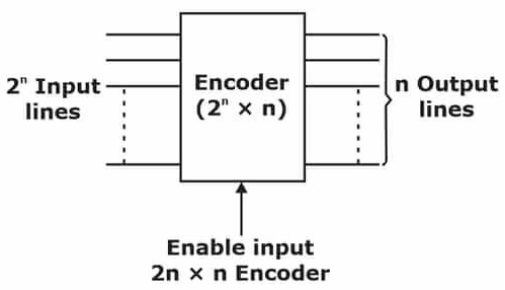


The Truth table of 2 × 4 Decoder:

A	B	Y_0	Y_1	Y_2	Y_3
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

6. Encoders

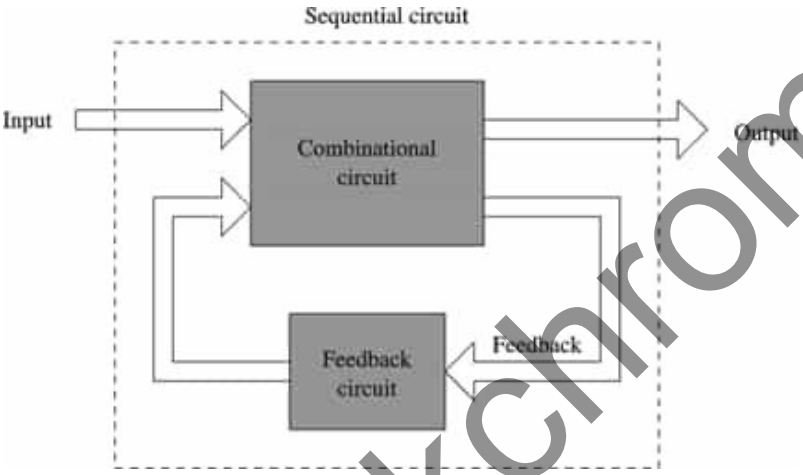
- It is a combinational circuit that converts information into the coded form (binary).
- It is a digital circuit that performs the *inverse* operation of a decoder.
- An encoder has 2^n (or fewer) input lines and *n* output lines.
- The output lines generate the binary code corresponding to the input value.



Y

Sequential Logic Circuits

1. Introduction



The sequential circuit is of two types.

- **Synchronous Sequential Circuit:** Change in input signals can affect memory elements only upon activation of clock signals.
- **Asynchronous Sequential Circuit:** Change in input signals can affect memory elements at any instant of time. These are faster than the synchronous circuit.

2. Flip Flops

- It is a one-bit memory cell which stores the 1-bit logical data (logic 0 or logic 1).
- It is a basic memory element.
- The most commonly used application of flip flops is in the implementation of a feedback circuit.

- As a memory relies on the feedback concept, flip flops can be used to design it.
- In the synchronous sequential circuit, Memory elements are clocked flip flops and generally edge triggered.
- In the asynchronous sequential circuit, Memory elements are unclocked flip flops/time delay elements which are generally level triggered.
- Flip flop circuit is also known as bistable multivibrator or latch because it has two stable states (1 state, 0 states).

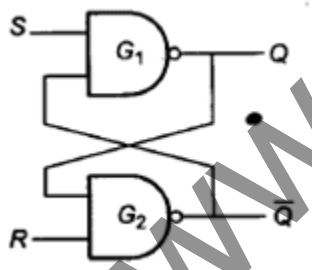
For the electronic circuits, there are mainly four types of flip flops present.

- **S-R Flip Flop (Basic Flip Flop)**
- **Delay Flip Flop (D Flip Flop)**
- **J-K Flip Flop**
- **T Flip Flop**

Basic SR Flip Flop

- The Set-Reset (SR) flip flop is designed with the help of two NOR gates or two NAND gates.
- SR Flip Flop is also called as SR latch.

SR Latch Implementation Using NAND Gates:

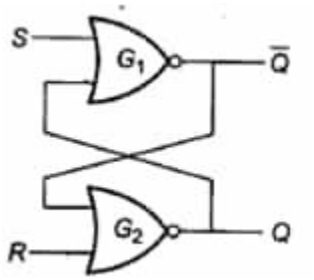


Logic diagram of SR latch using NAND gates

S	R	Q
0	0	Invalid
0	1	1
1	0	0
1	1	Previous state

Truth Table of Logic Diagram

SR Latch Using NOR Gates:



Logic diagram of SR latch using NOR gates

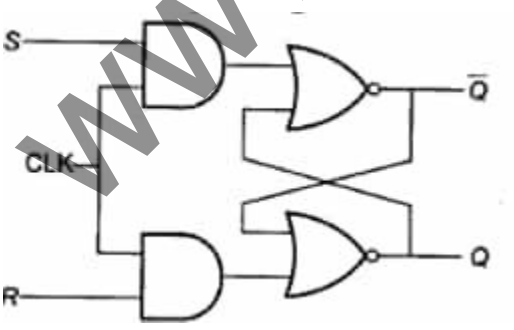
S	R	Q
0	0	Previous state
0	1	1
1	0	0
1	1	Invalid

Truth Table of Logic Diagram

Clocked SR Flip Flop Implementation using NAND Gates:

It is also called a Gated S-R flip flop. The main problem with S-R flip flops is using NOR and NAND gate in the invalid state. By using a bistable SR flip-flop this problem can be overcome. This can change outputs when certain invalid states are met, regardless of the condition of either the Set or the Reset inputs.

- **SR Flip Flop Using NOR Gates:**



SR flip flop using NOR gates

Clock	S	R	Q
0	×	×	Q_n
1	0	0	$Q_n \rightarrow$ Hold
1	0	1	0 → Hold
1	1	0	1 → Hold
1	1	1	Invalid

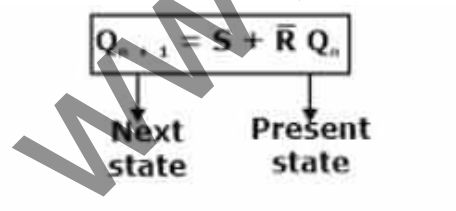
Truth Table of SR Flip Flop

With both S=1 and R=1, the occurrence of a clock pulse causes both outputs to momentarily go to 0. When the pulse is disabled (removed), the state of the flip-flop become indeterminate, depending on whether the set or reset input of the flip-flop remains at 1 longer than the transition to 0 at the end of the pulse.

Characteristic Table

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

Characteristic equation of SR flip flop



Excitation Table

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

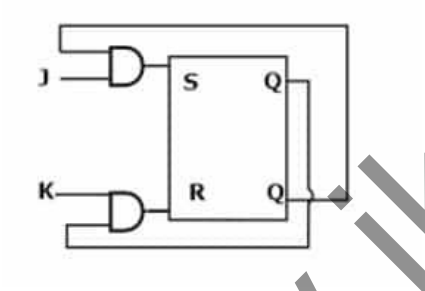
JK Flip Flop

A JK flip-flop eliminates the indeterminate state of the SR type. Inputs J and K is similar to the inputs S and R to set and clear the flip-flop (In JK flip-flop, the letter J is set and the letter K is for clear). When logic 1 are applied to both J and K inputs simultaneously, the flip-flop switches to its complement state. If Q=1 then it switches to Q=0 and vice versa.

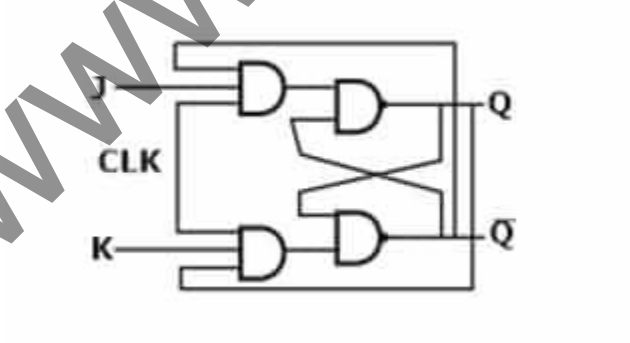
- **JK flip flop using SR flip flop:**

S = JQ'

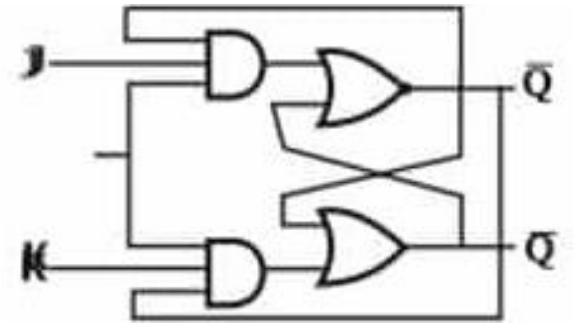
R = KQ



- **JK flip flop using NAND latch:**



- **JK flip flop using NOR latch:**



Characteristic Table

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

Excitation Table

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

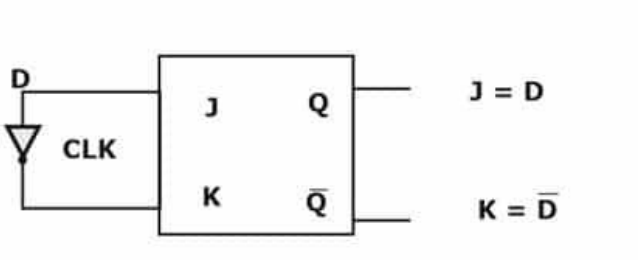
- **Characteristic equation for JK flip flop:**

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

D-Flip Flop

D flip flop is also known as a Transparent latch, Delay flip flop or data flip flop. The D input goes directly to the S (J) input and the complement of the D input goes to the R (K) input.

- The D-flipflop has only a single data input (D).
- If D = 1, the flip-flop is switched to the set state (unless it was already set).
- If D = 0, the flip-flop switches to the clear state.



Truth Table

Clock	D	Q_{n+1}
0	×	Q_n ← Memory
1	0	0 ← Reset
1	1	1 ← Set

Characteristic Table

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

Excitation Table

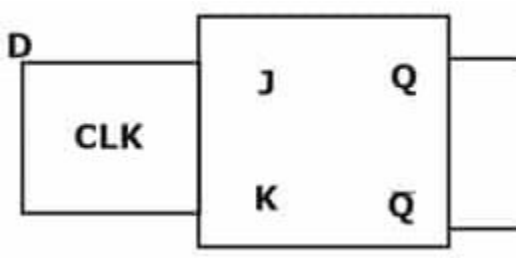
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

- **Characteristic equation for D-flop flop**

$Q_{n+1} = D$

T – Flip Flop

- The T flip-flop is a single input version of the JK flip-flop where T is connected to both J and K inputs directly.
- When T = 0, the flip flop enters into **Hold** mode, which means that the output, Q is kept the same as it was before the clock edge.
- When T = 1, the flip flop enters into **Toggle** mode, which means the output Q is negated after the clock edge, compared to the value before the clock edge.



Truth Table

Clock	T	Q_{n+1}
0	×	$Q_n \rightarrow$ Memory
1	0	$Q_n \rightarrow$ Hold
1	1	$Q_n \rightarrow$ Toggle

Characteristic Table

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

Excitation Table

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

- **The characteristic equation of T-Flip Flop:**

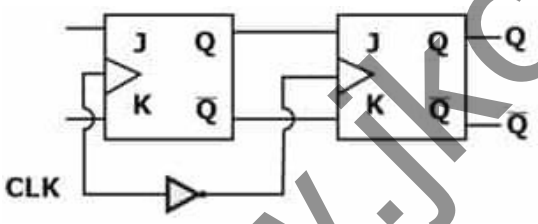
$Q_{n+1} = T \oplus Q_n$

- **Race Around Condition:**
  - The race around condition will occur in JK flip flop when J = *K* = 1 and $t_{pd\ (FF)} < t_{pw}$.
  - To avoid race around condition.

$t_{pw} < t_{pd\ (FF)} < T_{CLK}$

3. Master Slave (MS) Flip Flop

- A master-slave flip-flop is constructed from two separate flip-flops. One circuit serves as a **master** and the other as a **slave**. Input clock is applied to master and Inverted clock applied to slave.



- In Master Slave, flip flop output is changed only when slave output is changing.
- The master flip-flop is enabled on the positive edge of the clock pulse and the slave flip-flop is disabled by the inverter.
- The information at the external J and K inputs is transmitted to the master flip-flop.
- When the pulse returns to 0, the master flip-flop is disabled and the slave flip-flop is enabled. The slave flip-flop then goes to the same state as the master flip-flop.
- Master is level triggered, and Slave is edge triggered
- No race around condition occurs in Master Slave flip flop.
- It stores only one bit.

4. Flip Flop Conversions

The flip flop conversions are classified into different types which are:

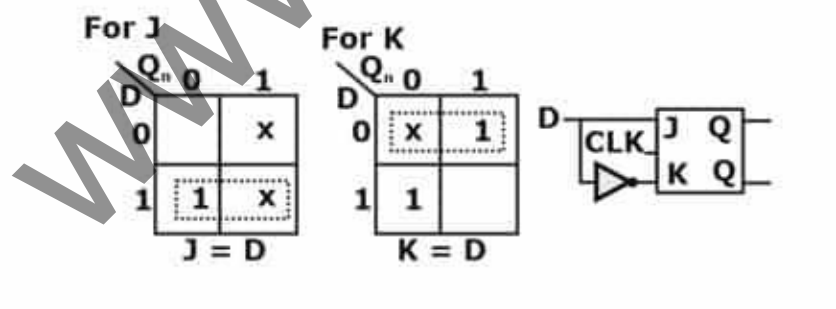
- SR-FF to JK-FF Conversion
- JK-FF to SR-FF Conversion
- SR-FF to D-FF Conversion
- D-FF to SR-FF Conversion
- JK-FF to T-FF Conversion
- JK-FF to D-FF Conversion
- D-FF to JK-FF Conversion

Procedure for Flip Flop conversion:

1. Conversion Table: Construct the characteristic table of required flip flop (unknown), and fill available (known) flip flop excitation.
2. Solve K map for given (known) flip flop as input and required flip flop as output.
3. Implement the required flip flop using the known flip flop.

Example: Conversion from JK flip flop to D flip flop is shown below.

D	Q_n	Q_{n+1}	J	K
0	0	0	0	×
0	1	0	×	1
1	0	1	1	×
1	1	1	×	0



Characteristic Table

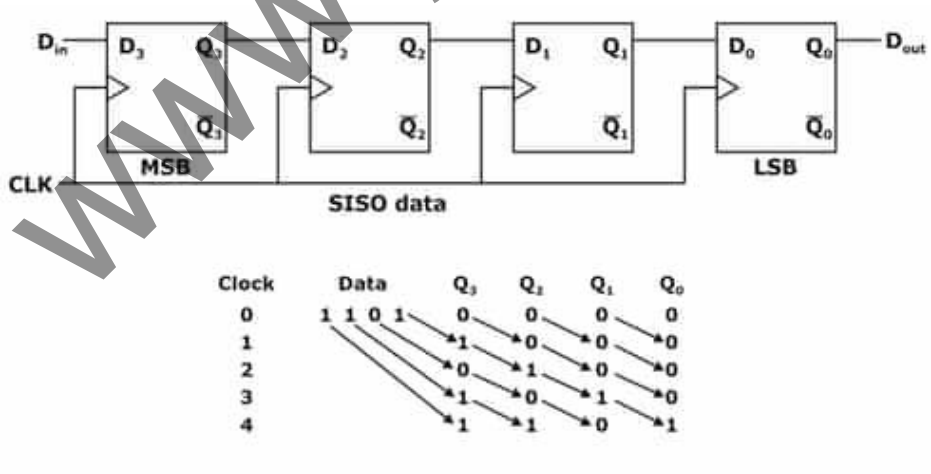
Other Conversion in Flip Flop circuits		
JK to D	JK to T	JK to SR
$J = D, K = \overline{D}$	$J = K = T$	$J = S$
		$K = R$
SR to JK	SR to D	SR to T
$S = J\overline{Q}$	$S = D$	$S = TQ$
$R = KQ$	$R = \overline{D}$	$R = TQ$
D to SR	D to JK	D to T
$D = S + \overline{R}Q$	$D = JQ + \overline{K}Q$	$D = T \oplus Q$
$T = JQ + KQ$	$T = SQ + RQ$	$T = D \oplus Q$

5. Registers

When a group of the flip flop is used to store a word (a group of bits) then it is called register. To store *n* bits, *n* flip flops are cascaded in the register. If in a register, the binary information can be moved from stage to stage, this type of registers is called shift registers. According to data movement in a register, *shift registers can be classified* as

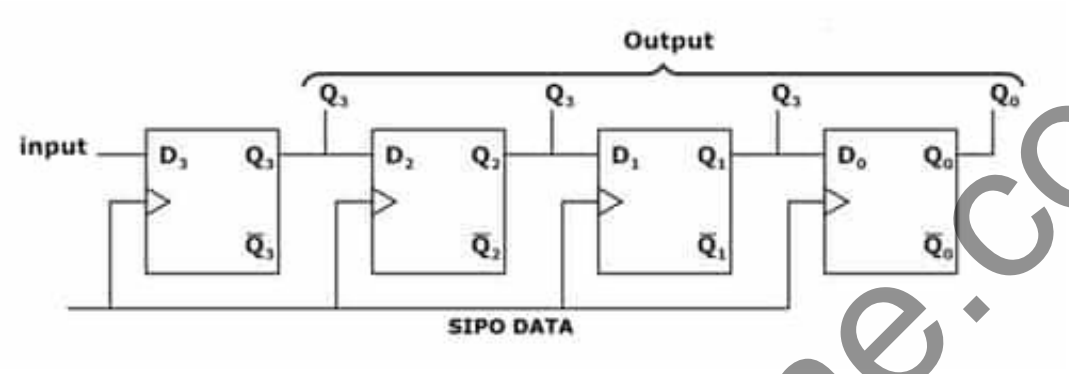
- Serial Input Serial Output (SISO)
- Serial Input Parallel Output (SIPO)
- Parallel Input Serial Output (PISO)
- Parallel Input Parallel Output (PIPO)

Serial Input Serial Output (SISO)



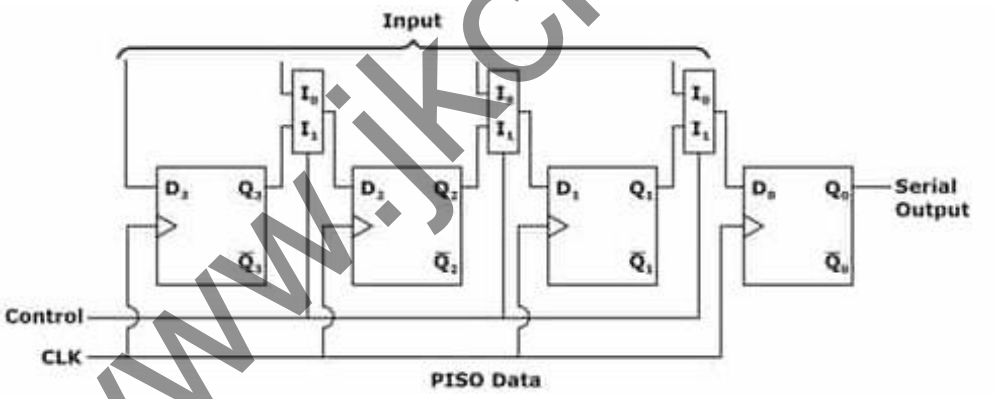
- In registers edge trigger circuit used to make circuit synchronous.
- If no clock is applied then get same data which is stored.
- In N bits SISO registers to provide N bits data, **Serially in** require N clock pulse, and **Serially out** require (N-1) clock pulse.

Serial Input Parallel Output (SIPO)



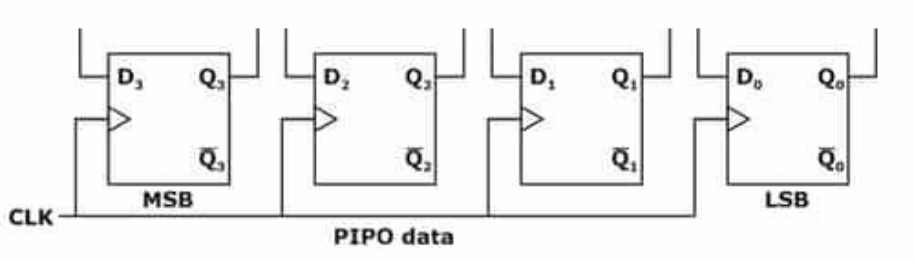
- To provide N-bit data: **Serial in** requires N clock pulse, and Parallel **out** requires no clock pulse.
- SIPO can provide n × t_{CIK} delay to the input.
- SIPO can convert serial data or temporal code to parallel or serial code.

Parallel Input Serial Output (PISO)



- If control = 0 then it acts as parallel input;
- If control = 1 then it acts as serial output;
- To provide parallel input, one clock pulse is required.
- To provide N bits serial output, it requires (N-1) clock pulse.
- PISO can convert special code to temporal code.

Parallel Input Parallel Output (PIPO)



- In PIPO register for parallel input number of pulse required is 1 clock pulse.
- In PIPO register for parallel output number of pulse required is 0 clock pulse.
- PIPO register cannot be used as a shift register.
- It is used for temporal storage of data in microcontroller, DSP, CPU etc.

Summary of Registers

Type of Register	Number of Pulses Required for Storage of n-bits Input	Number of Pulses Required to n-bit Output
SISO	n	n – 1
SIPO	n	0
PISO	1	n – 1
PIPO	1	0

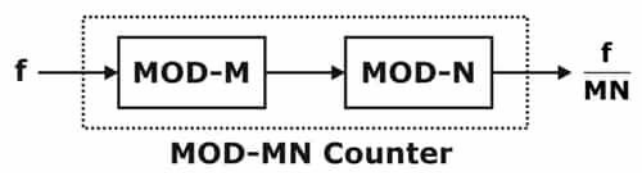
6. Counter

- A counter is a sequential logic circuit capable of counting the number of clock pulses arriving at its clock input.
- The sequence of count may be ascending, descending or non-sequence.
- For a counter circuit having *n* flip flops, Maximum possible states (N) = 2^n
- If N = 2^n, the counter acts as a binary counter.
- If N < 2^n, the counter the non-binary counter.
- It counter is capable to count from 0 to 2^n-1.
- MOD number is the Number of states present in a counter is known as modulus count or MOD number.
- For n-flip flops, the counter will have 2^n different states then this counter is said MOD- 2^n counter.

MOD-N Counter

- MOD number indicates frequency division obtained from the last flip flops.

- Cascaded two counters:



- MOD-MN counter:
  - Overall states of combined counter = MN
  - Input frequency = f
  - Output frequency f = f/(MN)

7. Classification of Counters

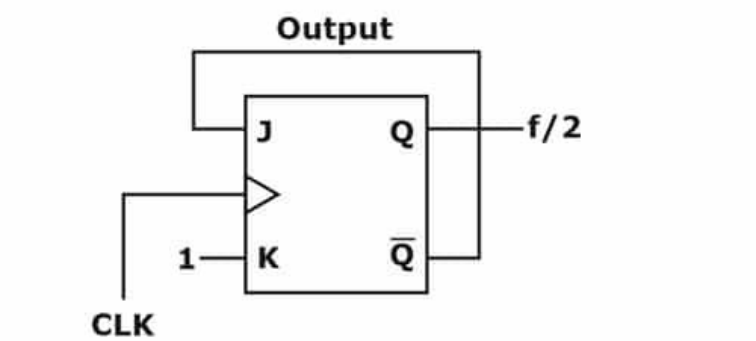
Based upon the applying clock pulse, counters are classified into two categories.

- Synchronous counter
- Asynchronous counter (ripple counter)

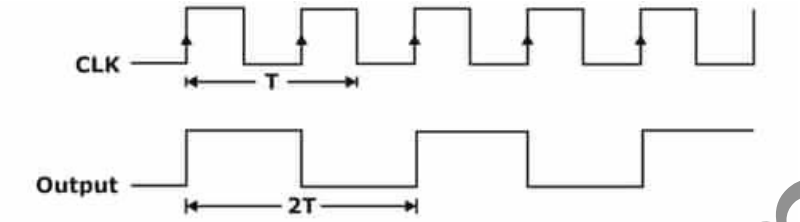
Synchronous Counter	Asynchronous Counter
All flip flops are triggered with same clock.	Different clock is applied to different flip flops.
It is faster.	It is lower
Design is complex.	I Design is relatively easy.
Decoding errors not present.	Decoding errors present.
Any required sequence can be designed.	Only fixed sequence can be designed.

8. Toggle Mode Circuit

These are frequency dividers circuit.

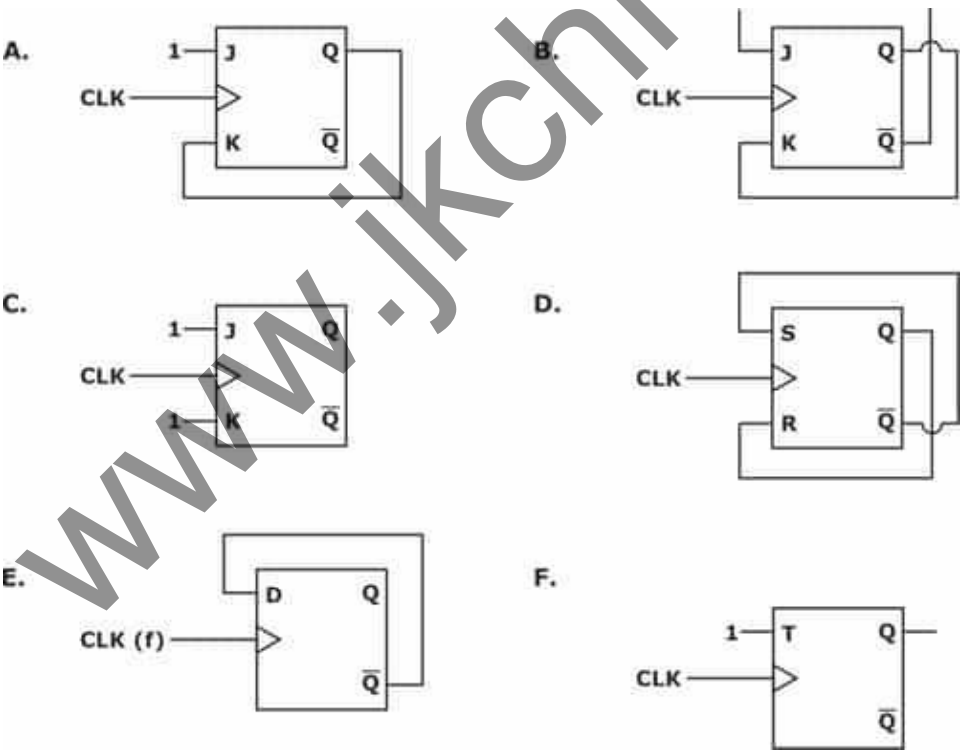


Logic symbol for toggle mode circuit



Clock and output waveforms of J-K flip-flop

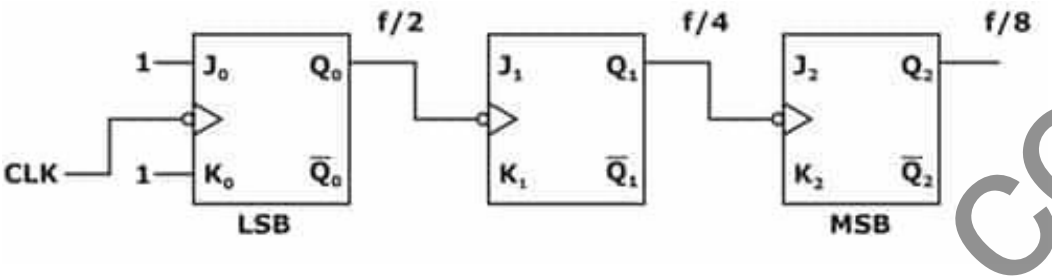
Other Toggle Mode Circuit



9. Asynchronous Counter (Ripple counter)

- A different clock pulse is applied to different flip flops.
- All flip flops are operating in toggle mode.
- In asynchronous counter flip flop applied with external clock acts as LSB bit.

3-bit Ripple Up Counter



- Input clock is applied at LSB bit.
- It n-bit ripple counter maximum possible states are 2^n.
- Bit ripple up counter counts from 0 to 2^{n-1}.
- If all states are used then with input frequency f, then output frequency will be $f/2^n$
- Calculation of Time Period of Flip Flop: In n-bit ripple counter if propagation delay of each flip flop is $t_{pd(FF)}$, then the time period of the clock is:

$$T_{clk} \geq nt_{pd(FF)}$$

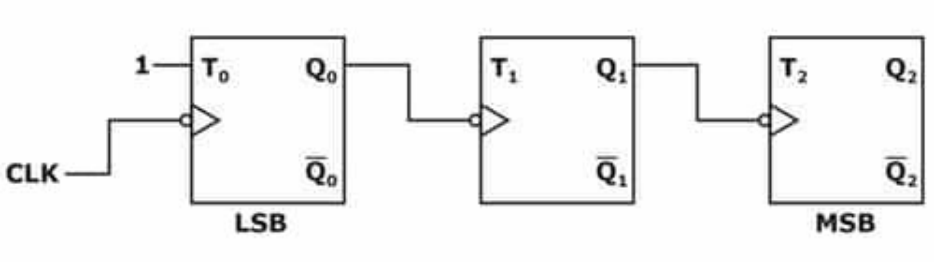
$$F_{clk} \leq \frac{1}{nt_{pd(FF)}}$$

- Maximum Clock Frequency:

$$f_{clk\ max} = \frac{1}{nt_{pd(FF)}}$$

- Due to propagation delays of flip flops decoding errors are present.
- Clear and preset are known as asynchronous input to flip flop.
- In any ripple counter, the following conditions will fulfil
  - Negative edge trigger and Q as clock ⇒ up counter
  - Positive edge trigger and Q as clock ⇒ up counter

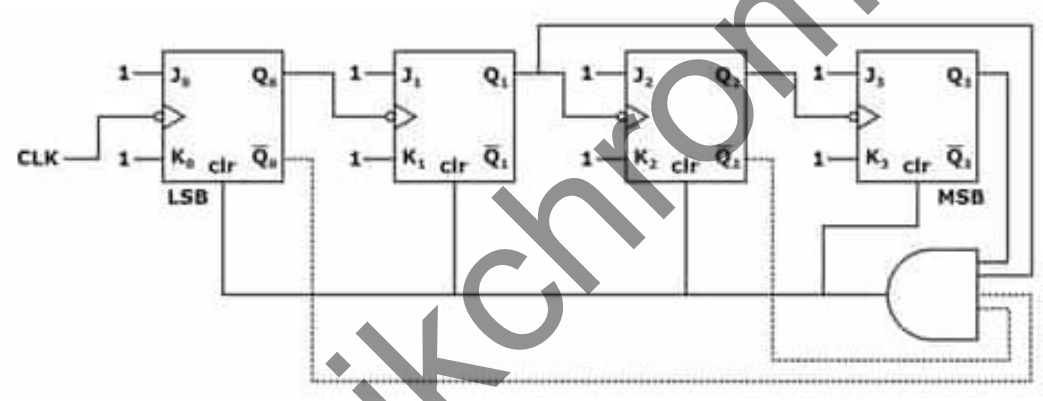
3-bit Ripple Down Counter



- Positive edge trigger and Q as clock ⇒ down counter
- Negative edge trigger and Q as clock ⇒ down counter

Non-binary Ripple Counter

Decode counter or BCD counter is an example of a non-binary counter. It requires 4 flip flops.

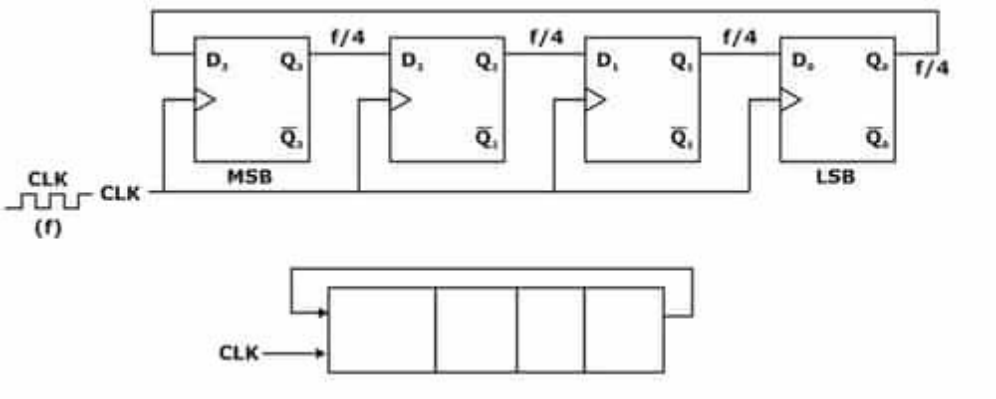


- Used state = 10 and unused states = 6 → (2^4 -10)
- Output frequency of BCD counter = f/10
- For making non-binary counter clear (clr) signal is used.
- c1r is active high, and (clr)' is active low.

10. Synchronous Counters

In this type of counter, there are no connections of the first flip flop output to a clock input of the next flip flop.

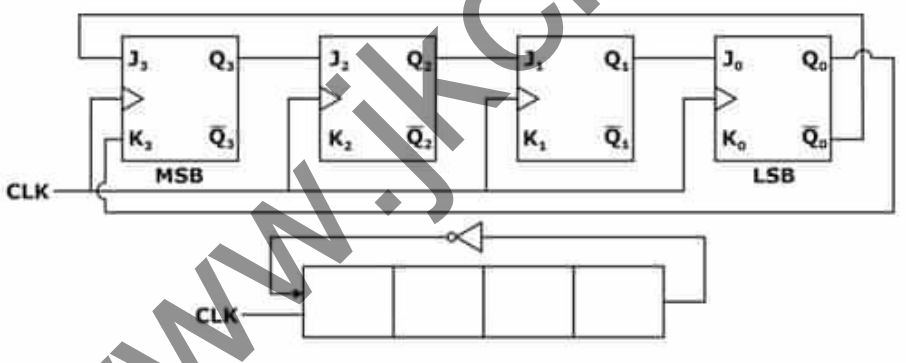
Ring Counter: It is a circular shift register with only flip flop being set at any particular time, all others are cleared. It is a shift register with feedback.



- In-ring counter, if the feedback is used the number of states is reduced.
- With n flip flops maximum states = n.
- Number of unused states in-ring counter = $2^n - n$
- Maximum Clock Frequency: If the input frequency is f, then at the output of every flip flop we get f/N frequency. In-ring counter, if the propagation delay of each flip flop is $t_{pd(FF)}$ then

$$T_{clk} \geq t_{pd(FF)}$$

Johonson Ring Counter: Jhonson ring counter is also called as a Twisted ring counter, Switch tail counter, Creeping counter, or Mobies counter.



- In n - bit Jhonson counter maximum used states = 2n, unused states = 2^n - 2n.
- If the input clock frequency is f, the output frequency of each flip flop is f /2n and the duty cycle is 50%.
- A disadvantage of Jhonson Ring Counter: Lockout may occur. To decode each state one, two-input AND or NOR gate is used.

Logic Family

1. Integrated Circuits

- Integrated circuits (ICs) are chips, pieces of semiconductor material, that contain all of the transistors, resistors, and capacitors necessary to create a digital circuit or system.
- The first ICs were fabricated using Ge BJTs in 1958.
    - Jack Kirby of Texas Instruments, Nobel Prize in 2000
    - Robert Noyes of Fairchild Semiconductors fabricated the first Si ICs in 1959.

Integration Levels:

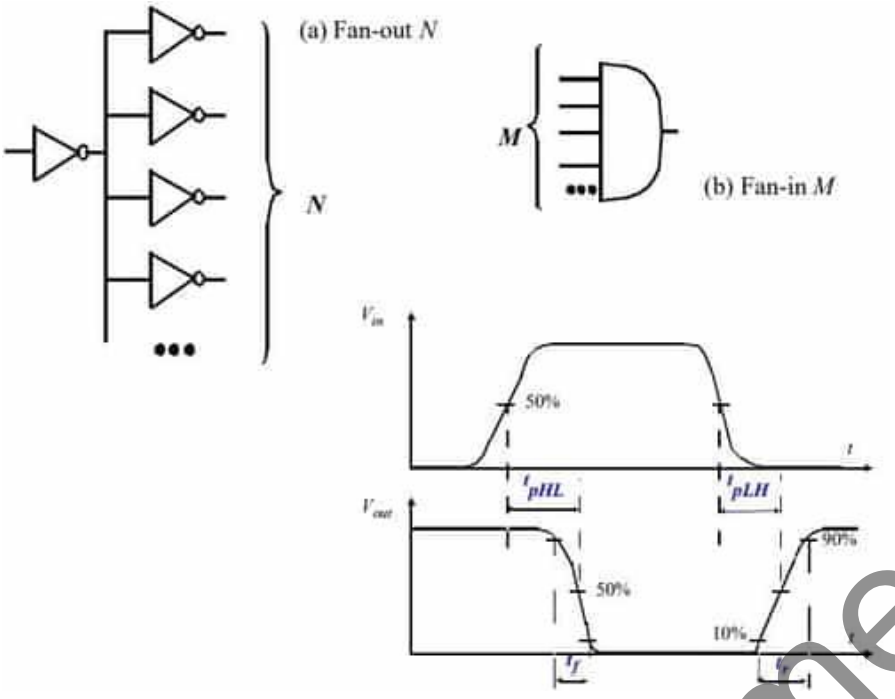
- SSI Small scale integration [12 gates/chip]
- MSI Medium scale integration [100 gates/chip]
- LSI Large scale integration [1K gates/chip]
- VLSI Very large scale integration [10K gates/chip]
- ULSI Ultra large scale integration [100K gates/chip]

Moore's Law:

- A prediction made by Moore (a co-founder of Intel) in 1965: "… a number of transistors to double every 2 years."

Characteristics of digital circuits

- **Fan in:**
    - Fan in is the number of inputs connected to the gate without any degradation in the voltage level.
- **Fan out:**
    - Fan out specifies the number of standard loads that the output of the gate can drive without impairment of its normal operation
- **Power dissipation:**
    - Power dissipation is a measure of power consumed by the gate when fully driven by all its inputs.
- **Propagation delay:**
    - Propagation delay is the average transition delay time for the signal to propagate from input to output when the signals change in value. It is expressed in ns.
- **Noise margin:**
    - It is the maximum noise voltage added to an input signal of a digital circuit that does not cause an undesirable change in the circuit output. It is expressed in volts.



2. Logic Families

Logic families are sets of chips that may implement different logical functions but use the same type of transistors and voltage levels for logical levels and for the power supplies. These families vary by speed, power consumption, cost, voltage & current levels. The most widely used families are:

- DL (Diode- logic)
- DTL （Diode-transistor logic）
- RTL （Resistor-transistor logic）
- TTL （Transistor -transistor logic）
- ECL （Emitter-coupled logic）
- MOS （Metal-oxide semiconductor）
- CMOS (Complementary Metal-oxide semiconductor）

3. Digital IC Terminology

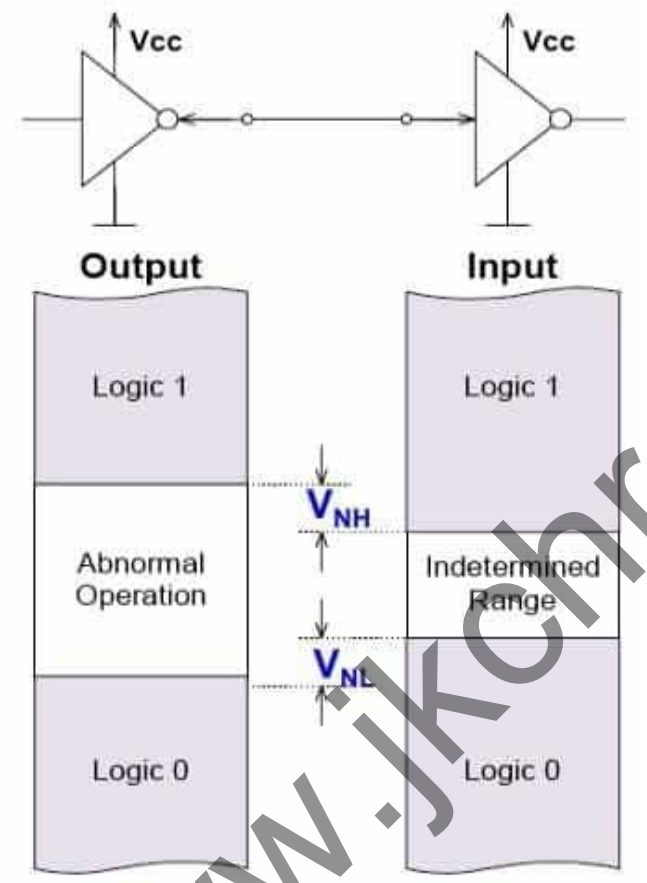
Voltage Parameters:

- V_{IH}(min): high-level input voltage, the minimum voltage level required for a logic 1 at an input.
- V_{IL}(max): low-level input voltage
- V_{OH}(min): high-level output voltage
- V_{OL}(max): low-level output voltage

- For proper operation, the input voltage levels to a logic must be kept outside the indeterminate range. Lower than VIL(max) and higher than VIH(min).

Noise Margin:

- The maximum noise voltage that can be tolerated by a circuit is termed its noise immunity (noise Margin)



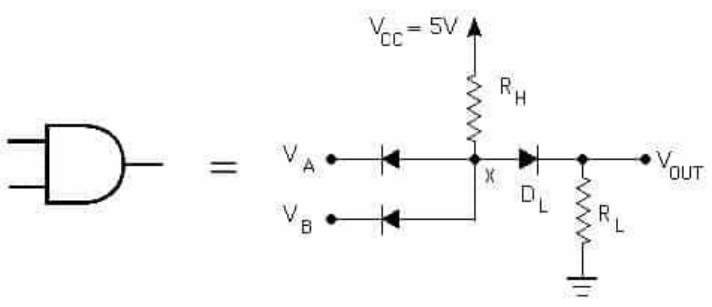
Current Parameters:

- I_{OH} – Current flowing into an output in the logical “1” state under specified load conditions
- I_{OL} – Current flowing into an output in the logical “0” state under specified load conditions
- I_{IH} – Current flowing into an input when a specified HI level is applied to that
  input

- I_{IL} – Current flowing into an input when a specified LO level is applied to that
  input

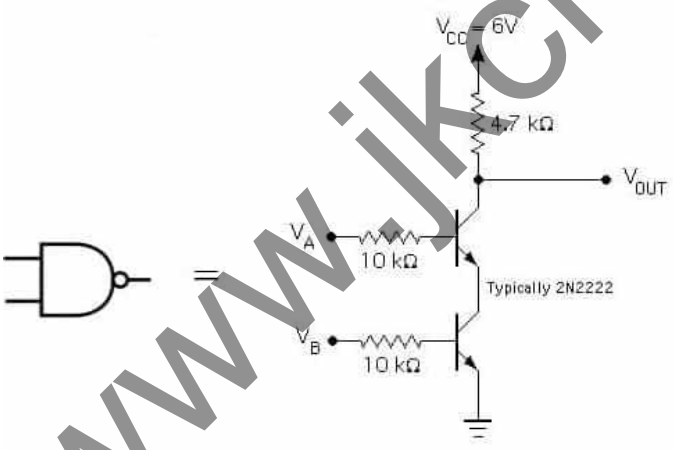
4. Diode Logic (DL)

- simplest; does not scale
- NOT not possible (need an active element)



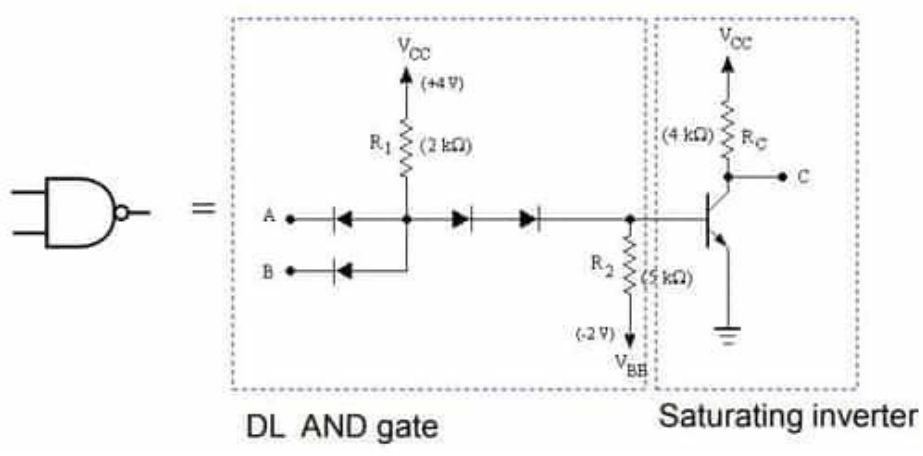
5. Resistor-Transistor Logic (RTL)

- replace diode switch with a transistor switch
- can be cascaded
- large power draw



6. Diode-Transistor Logic (DTL)

- essentially diode logic with transistor amplification
- reduced power consumption
- faster than RTL

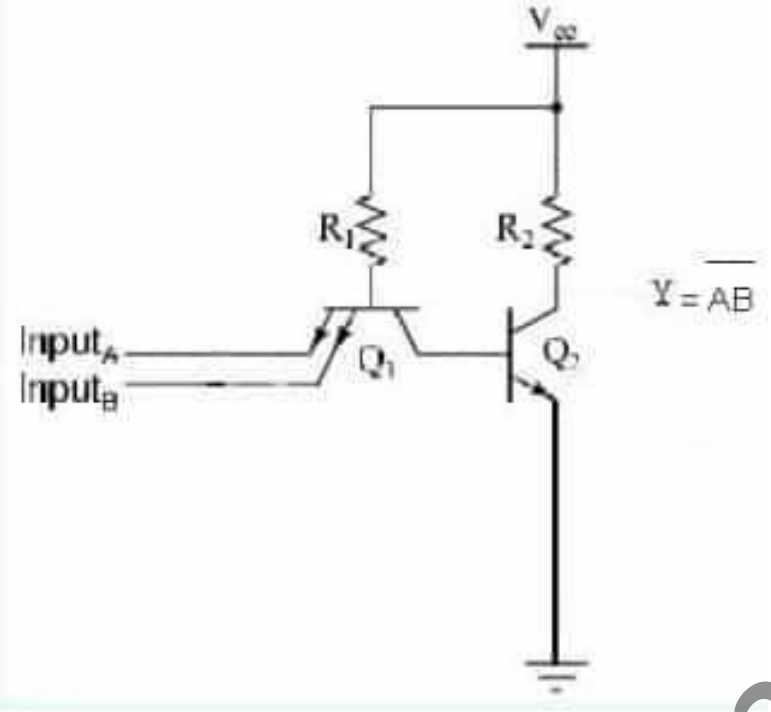


7. Transistor-transistor logic (TTL)

- based on bipolar transistors one of the most widely used families of small- and medium-scale devices – rarely used for VLSI
- typically operated from a 5V supply
- typical noise immunity about 1 – 1.6 V
- many forms, some optimised for speed, power, etc.
- High-speed versions comparable to CMOS (~ 1.5 ns)
- low-power versions down to about 1 mW/gate

TTL NAND Gate:

- Input terminals: The emitter of Q_1
- Output terminals: collector of Q_2
- When any input = logic '0'
    - Q_1 emitter junction is forward biased.
    - Also, its collector junction is FB,
    - so Q_1 goes in saturation.
    - Base of Q2 is at Low voltage
    - This causes base-emitter junction of Q2 to be RB, so Q2 goes in cut-off
    - Hence output is 5V or logic '1'
- When all inputs = logic '1'
    - Q1 emitter junction is RB.
    - so Q1 goes in cut-off.
    - Its collector voltage increases
    - This forward biases Q2,
    - so Q2 goes into saturation
    - Hence output is 0V

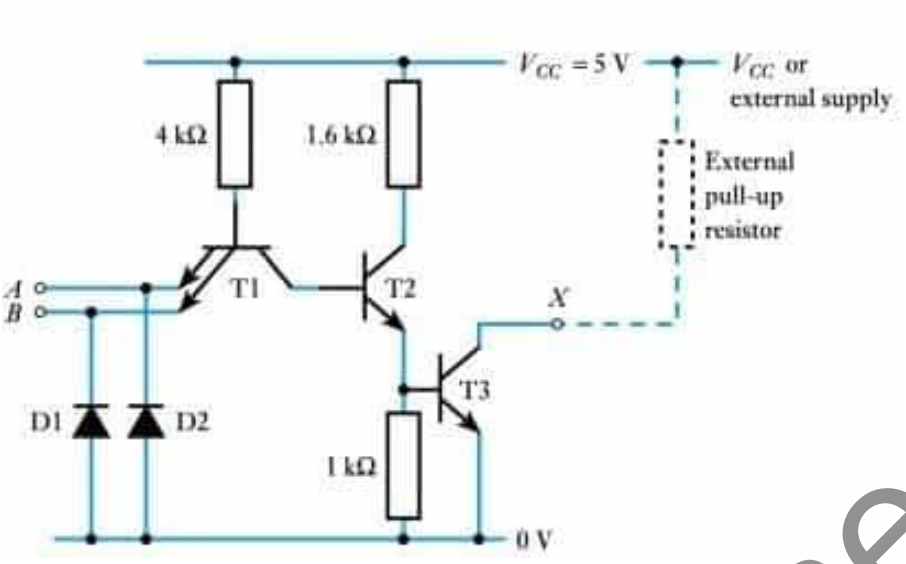


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

A TTL NAND gate with open collector output:

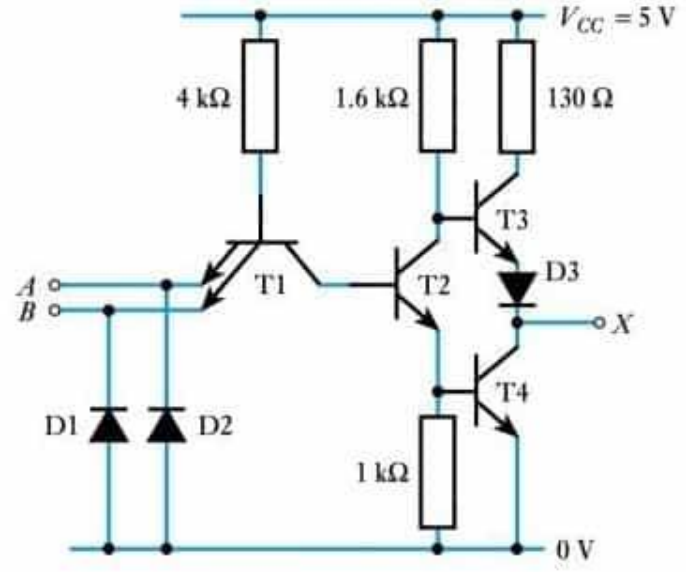
- Its similar to the previous circuit.
- Q2 is used as an emitter follower. The output of Q2 is fed to the input of Q3. Collector of Q2 and Q3 are in phase.
- This circuit needs an external 'Pull- up' resistor between output and power supply.
- The disadvantage of open- collector gate is their slow switching speed.
- The pull-up resistance is few kilo ohms. Gives a relatively long time constant, when multiplied by the stray output capacitance.

- Is worst when output goes from low to high.



TTL NAND gate with totem pole (active pull-up):

- In this circuit Q1 and the 4KΩ resistor act like a 2 input AND gate. The remaining circuit acts like an inverter. Transistors Q3 & Q4 form a totem-pole **i.e.one** NPN transistor in series with another.
- With a totem-pole output stage either Q3 or Q4 is on. When Q3 is 'on' output is high. When Q4 is 'on' output is low. If A or B is low, the Q1 conducts and the base voltage of Q2 is almost zero.
- Q2 cuts off, hence Q4 goes into cut off. Q3 base is high, Q3 acts as an emitter follower, the output Y' is high.
- If A and B are high, Q1 does not conduct (cut-off), Q2 base goes high (saturation). Q4 goes into saturation hence output is low.
- The drop across Diode D3 keeps the base emitter diode of Q3 reverse biased. Hence Q3 is off or else it conducts slightly when output is low.
- Now only Q4 conducts when output is low. Totem pole transistors produce a low output impedance. When Q3 is conducting the output impedance is approx 70 Ω. When Q4 is saturated the output impedance is only 12 Ω. Hence the output impedance of a totem pole circuit is low.
- Any stray output capacitance is rapidly charged or discharged through the low output impedance. Hence the output can change quickly from one state to the other.

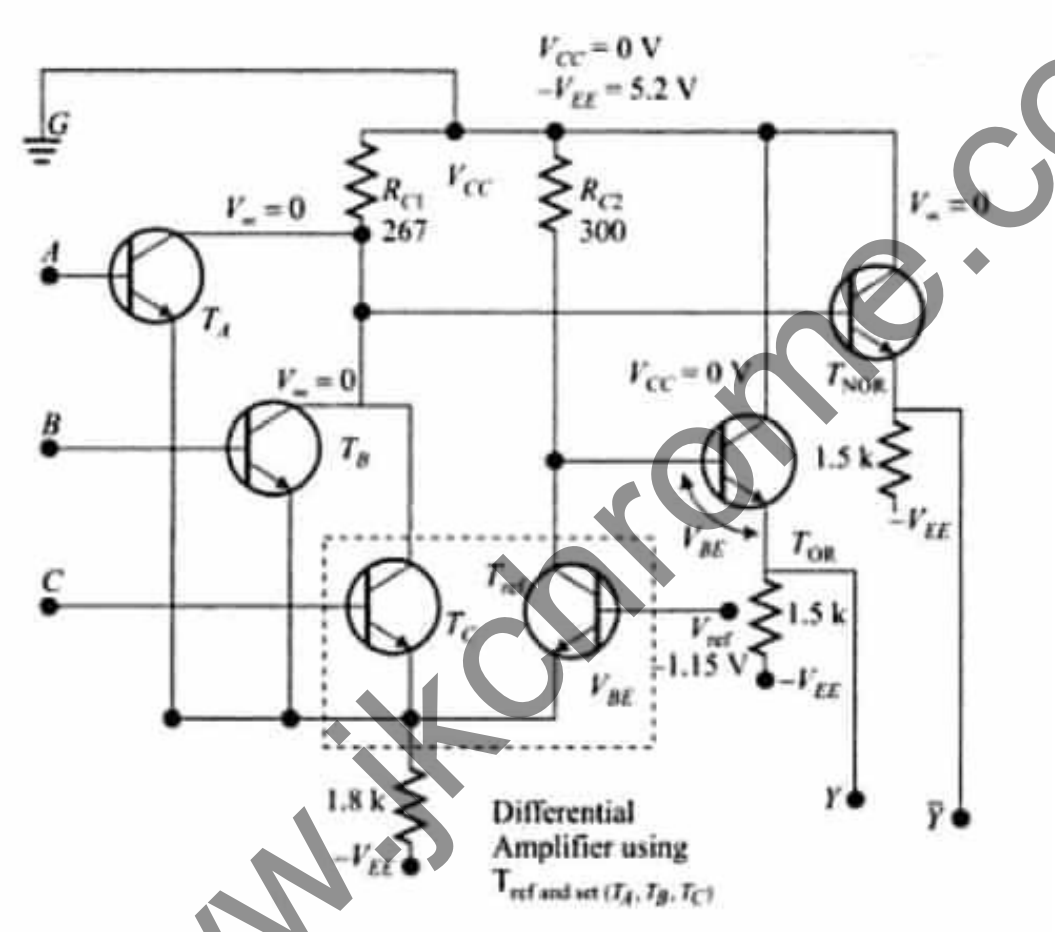


Types of TTL:

- *Standard TTL*
  - typical gate propagation delay of 10ns and a power dissipation of 10 mW per gate, for a power–delay product (PDP) or switching energy of about 100 pJ
- *Low-power TTL (L)*
  - slow switching speed (33ns)
  - reduction in power consumption (1 mW) (now essentially replaced by CMOS logic)
- *High-speed TTL (H)*
  - faster switching than standard TTL (6ns)
  - but significantly higher power dissipation (22 mW)
- *Schottky TTL (S)*
  - used Schottky diode clamps at gate inputs to prevent charge storage and improve switching time. A Schottky diode has a very low forward-voltage drop of 0.15–0.45V approx (silicon diode has a voltage drop of 0.6–1.7V). This lower voltage drop can provide higher switching speed.
  - Faster speed of (3ns) but had higher power dissipation (19 mW)
- *Low-power Schottky TTL (LS)*
  - used the higher resistance values of low-power TTL and the Schottky diodes to provide a good combination of speed (9.5ns) and reduced power consumption (2 mW), and PDP of about 20 PJ.

8. Emitter-coupled logic (ECL)

- based on bipolar transistors, but removes problems of storage time by preventing the transistors from saturating
- very fast operation - propagation delays of 1ns or less
- high power consumption, perhaps 60 mW/gate
- low noise immunity of about 0.2-0.25 V
- used in some high-speed specialist applications, but now largely replaced by high-speed CMOS



Input:

- Input is at the base of the transistor. The emitter of T_{ref} and input transistors couples together. [Hence the name]
- ECL basic gate is OR/NOR gate
- If any input is not connected, the transistor Ti base-emitter will be at cutoff. Therefore, it will be taken as low logic level

Output:

- The outputs (T_{OR} and T_{NOR}) are taken from the emitters of each transistor. The collector of T_{OR} and T_{NOR} connects to GND in the CC amplifier mode (also called emitter-follower mode).
- The emitter gives the output, which also connects to $-V_{EE}$ through a resistance R (~1.5kΩ)

Differential Amplifier:

- There is transistor T, which forms a differential amplifier pair between T and the parallel circuits of T_A, T_B, T_C. T gets the input reference voltage (V_R = -1.15V) from a reference supply circuit.
- The pairs amplify the difference of base voltage of T_A (or T_B or T_C) and V_{ref}.
- The emitters of the differential amplifier pairs connect through a common resistance R_E (~1.8kΩ) and to the $-V_{EE}$(~ -5V)

Emitter Follower (CC) amplifier:

- The collectors of (T_A, T_B, …) are also common.
- Common- collectors of the differential amplifier pairs connect through a resistance RC (~267Ω) to the GND

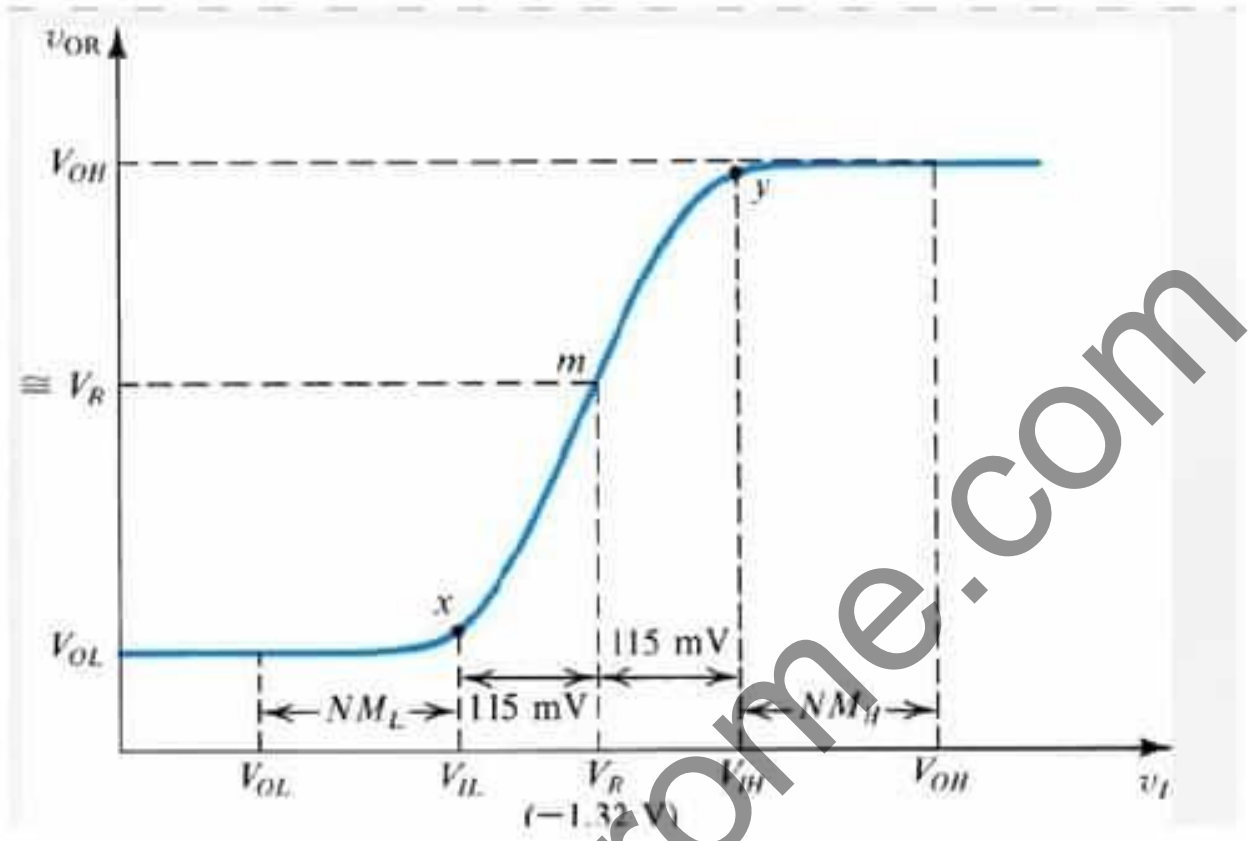
Working:

- Consider T_C and T_{ref}
  - Case 1: let all V_{in} = -1.6V. But V_{ref} = -1.15V, so V_{in} is low and V_{ref} logic high, So T_C is in cutoff and T_{ref} in normal inverting mode.
  - So T_{OR} gets -1.15V, i.e logic LOW, it is cut off and Y=-V_{EE} (LOW)
- Case 2: If V_{in} at T_C is -0.7V (HIGH), V_{ref} = -1.15V (LOW).
  - T_C is in normal inverting mode and T_{ref} is in the cutoff. $-V_{EE}$ is reflected at T_{NOR}.
  - So T_{NOR} is cut-off. Y' = -V_{EE} (i.e logic Low).
  - T_{OR} is ON, so Y=0v (LOW)

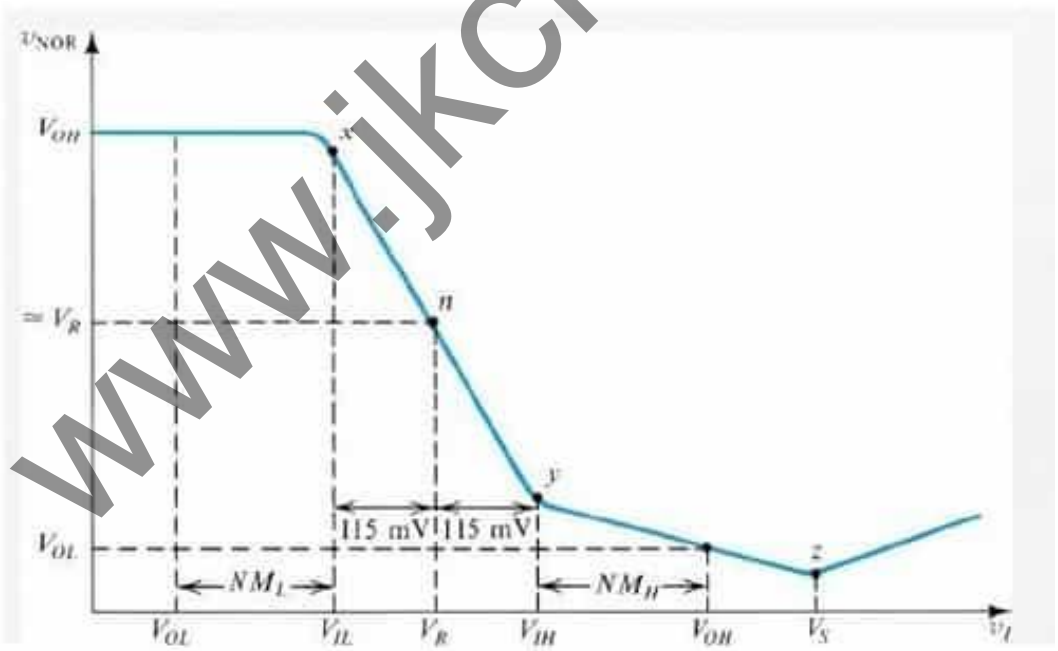
ECL features:

- Faster speed (2 ns propagation delay) of operation than TTL (10 ns), 74S TTL(3 ns)
- More power dissipation (50 mW/gate) than TTL (10 mW), 74S (19mW)
- Noise Margin at '1'or '0'output and input = 0.4V (– 1.7V and – 1.4V)

Transfer Characteristics of OR:

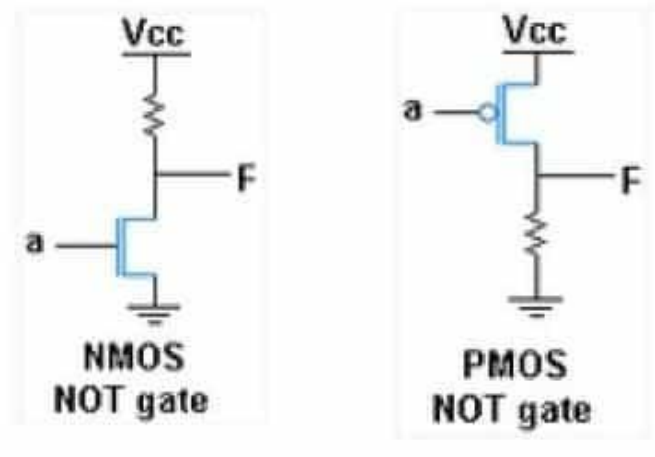


Transfer Characteristics of NOR:



9. MOS inverter

- *nMOS Inverter:*
  - when a='1', nMOS conducts, so F='0'
  - When a='0', nMOS is cut-off, so F=Vcc=logic '1'
- *pMOS Inverter:*
  - when a='1', pMOS is cut-off, so F='0'
  - When a='0', pMOS is on, so F=Vcc=logic '1'

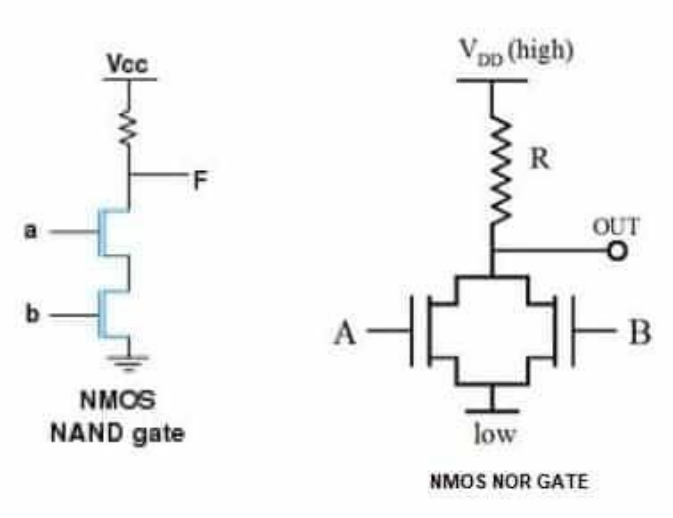


Advantages and Disadvantages of MOS inverter:

- **Advantage:**
  - only a single type of transistor, So, it can be fabricated at low cost.
- **Disadvantage:**
  - as current flows through the resistor in one of the two states, more power consumption is their processing speed is slow

NAND and NOR with nMOS

nMOS NAND	**nMOS NOR**
When any input is '0' • corresponding of MOS is off, So F=V_{cc}='1'	When any input is '1' • Corresponding MOS is on, So F=Gnd='0'
When both inputs are '1' • Both MOS is on. F = Gnd = '0'	When both inputs are '0' • Both MOS are off. Out = VDD = '1'

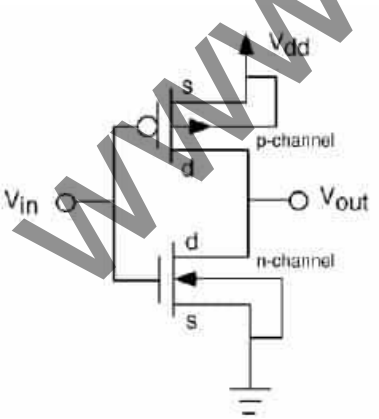


10. Complementary metal oxide semiconductor (CMOS)

- most widely used a family of large-scale devices combines high speed with low power consumption usually operates from a single supply of 5 – 15 V
- excellent noise immunity of about 30% of the supply voltage
- High fan-out: can be connected to a large number of gates (about 50)
- CMOS gates have equal no.of pMOS and nMOS
- CMOS inverter has a very high input resistance

CMOS inverter:

- Upper is pMOS, lower nMOS.
- When V_{in} = HIGH, Lower MOS on, V_{OUT} =LOW
- When V_{in} = LOW, Upper MOS on, V_{out} = Vd = HIGH

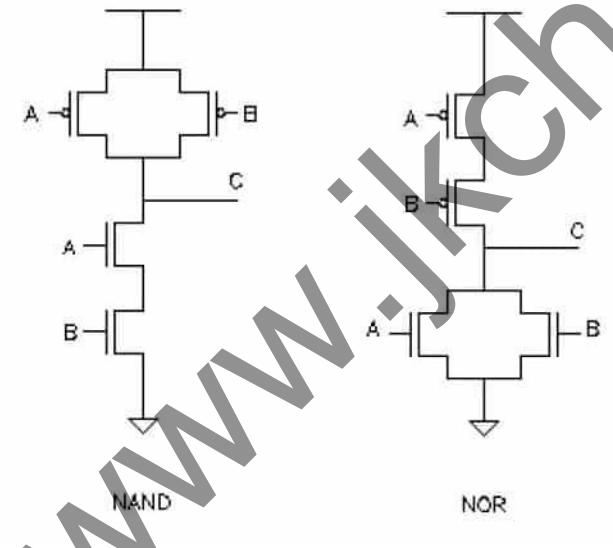


Advantages of CMOS:

- This configuration greatly reduces power consumption since one of the transistors is always off in both logic states.
- Processing speed can also be improved due to the relatively low resistance compared to the nMOS-only or pMOS-only type devices.
- High Fan-out (usually 50)
- excellent noise immunity

NAND and NOR with CMOS

CMOS NAND	CMOS NOR
If A='1' and B='1' • Upper parallel nMOS is off, lower series pMOS are on, so C=Gnd='0'	If A='0' and B='0' • Upper series nMOS is on, lower parallel pMOS are off, so C=V_{dd}='1'
If any A or B or both are '0' • Upper (any or both) parallel nMOS is on, lower series(any or both) pMOS are off, so C= V_{dd} ='1'	If any A or B or both are '1' • Upper (any or both) series nMOS is off, lower parallel (any or both) pMOS are on, so C= GND='0'



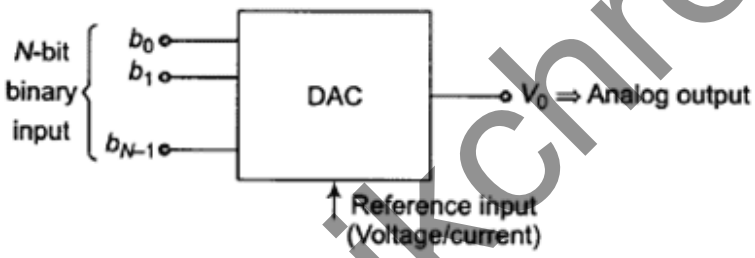
11. Logic families: Comparison

	TTL	ECL	CMOS
Base Gate	NAND	OR/NOR	NAND/NOR
Fan-in	12-14	>10	>10
Fan-out	10	25	50
Power dissipation (mW)	10	175	0.001
Noise Margin	0.5V	0.16V (lowest)	1.5V (Highest)
Propagation Delay (ns)	10	<3 lowest	15 Highest
Noise immunity	Very good	good	excellent

Data Converters

DAC and ADC: It is possible to convert the analog signal to digital and *vice-versa.* We can get analog from digital through DAC and can get digital from analog through ADC.

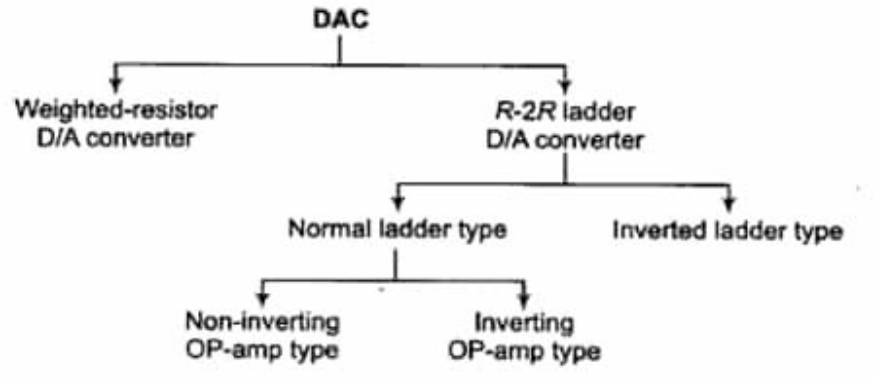
Digital to Analog Converter (DAC): D/A converter (also called a DAC) accepts an n-bit digital word and produces an analog sample.



$$V_0 = k\left[b_0 + 2^1 b_1 + 2^2 b_2 + \ldots 2^{N-2} b_{N-2} + 2^{N-1} b_{N-1}\right]$$

where, k = Proportionality factor, b_n = 1;if nth bit of digital input is 1, b_n = 0; if nth bit of digital input is 0.

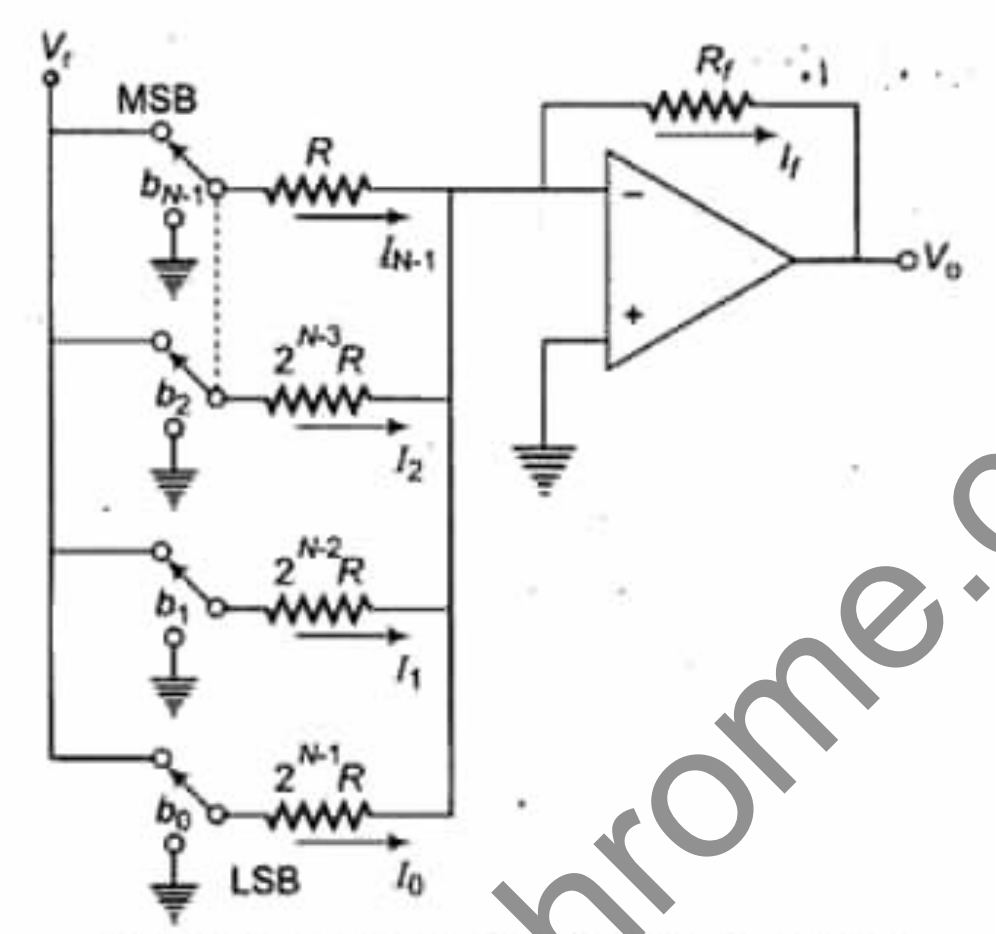
Classification of DAC:



Weighted Resistor DAC (N – bit):

- A DAC can be constructed by using a Summing Amplifier and a set of resistors R, 2R, 4R, 8R, etc as its inputs.
- The circuit consists of a reference voltage V_f, N binary-weighted resistors R, 2R, 4R, 8R, . . . , $2^{(N-1)}$ R, N single-pole double-throw switches, and an Op-amp together with its feedback resistance R_f = R/2.
- The switches are controlled by an N-bit digital input word D.

$$D = \frac{b_1}{2^1} + \frac{b_2}{2^2} + \cdots + \frac{b_N}{2^N}$$



Circuit diagram of *N*-bit weighted resistor DAC

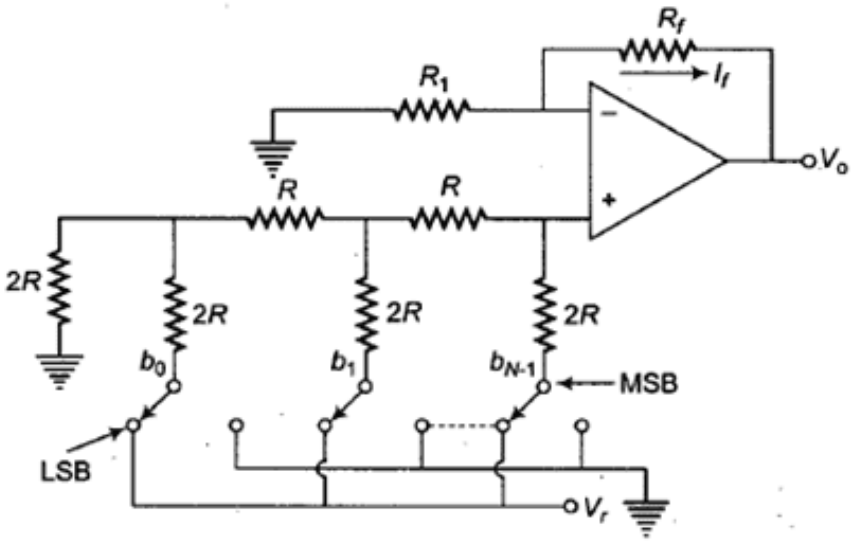
$V_0 = -R_f I_f = -V_f D$

$$V_0 = \frac{V_f R_f}{2^{N-1} R}\left[2^{N-1} b_{N-1} + 2^{N-2} b_{N-2} + \ldots 2b_1 + b_0\right]$$

LSB resistance = (2^{N-1}) MSB resistance.

- The accuracy of the DAC depends critically on the accuracy of the Reference voltage, the precision of the binary-weighted resistors, and the perfection of the switches.
- A disadvantage of the binary-weighted resistor network is that for a large number of bits (N > 4) the spread between the smallest and largest resistances becomes quite large. This implies difficulties in maintaining accuracy in resistor values.

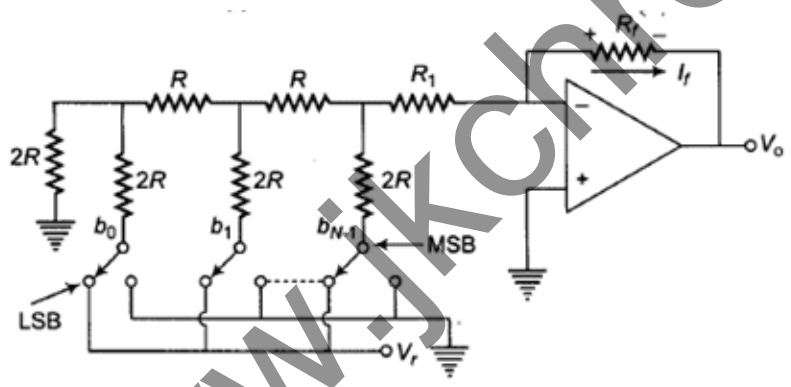
R – 2R Ladder DAC: Non-Inverting OP-amp type DAC



Circuit diagram of *R-2R* Ladder DAC

$$V_0 = \frac{V_r}{2^N}\sum_{i=0}^{N-1} 2^i b_i \left(1 + \frac{R_f}{R_1}\right)$$

Inverting Amplifier

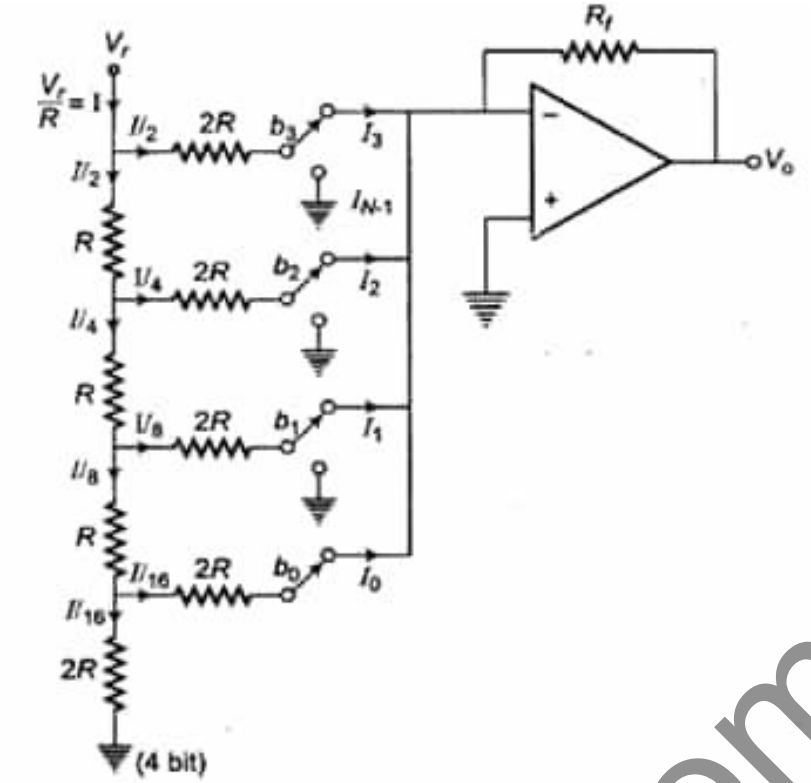


Circuit diagram of inverting amplifier using *R-2R* ladder

$$V_0 = \frac{-V_r}{2^N}\sum_{i=0}^{N-1} 2^i b_i \left(\frac{-R_f}{R_1 + R}\right) = (-R_f) I_f$$

$$I_f = \frac{V_r}{2^N}\sum_{i=0}^{N-1} 2^i b_i \left(\frac{1}{R + R_1}\right)$$

Inverted Ladder R – 2R Circuit



Circuit diagram of inverted ladder *R*-2*R* circuit

$$I_f = \frac{V_r}{2^N} \times \left(\sum_{i=0}^{N-1} 2^i b_i \right) \left(\frac{1}{R} \right)$$

$$V_0 = (-R_f) I_f$$

$$V_0 = \frac{V_r}{2^N} \times \left(\sum_{i=0}^{N-1} 2^i b_i \right) \left(\frac{-R_f}{R} \right)$$

Specifications for DAC:

- Resolution in DAC is changed in analog output with corresponding to 1 LSB bit increment at the input.
- Resolution = weight of LSB =
- V= Voltage corresponding to logic *N* = Number of bits.
- Analog Output Analog output = resolution x decimal equivalent of binary data

- **Maximum Analog Output Voltage (V_{FS})** V_{FS} is the maximum analog output voltage of DAC.

$$V_{FS} = \frac{V_r}{2^{N-1}} \times \left(2^{N-1}\right) = V_r$$

$V_{FS} = V_r$

- **Percentage Resolution:**

$$\%R = \frac{1}{2^N - 1} \times 100$$

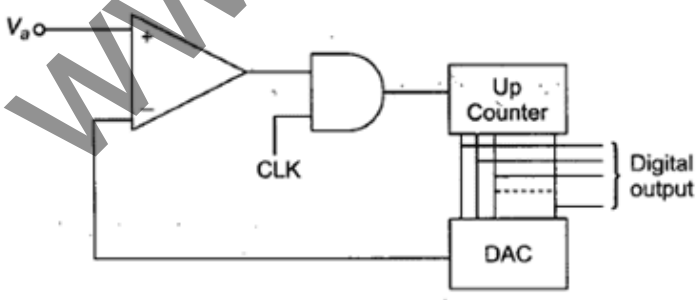
- Maximum Error Maximum error acceptable in ADC and DAC equals to resolution.
- Resolution (R-2R ladder type)

$$= \frac{V_r}{2^N}$$

Analog to Digital Converter:

- A/D converter (also called an ADC) accepts an analog sample V_A and produces an N-bit digital word.
- Examples of ADC usage are digital volt meters, cell phone, thermocouples, and a digital oscilloscope.
- Types of A/D Converters: Dual Slope A/D Converter, Successive Approximation A/D Converter, Flash A/D Converter, Delta-Sigma A/D Converter, etc.

Counter type ADC:

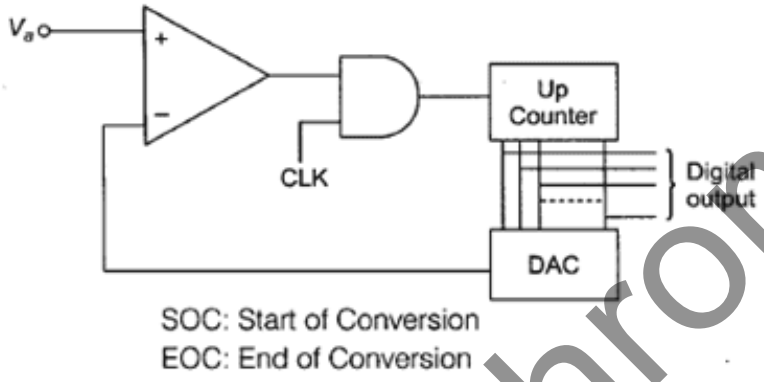


Circuit diagram of Analog to Digital Convertor (ADC)

- In N-bit counter type ADC:
  - Maximum number of clock pulses required for conversion = 2^N -1
  - Maximum time required for conversion = (2^N -1) T_{CLK}
  - Minimum number of clock pulses = 1
  - Average number of clock pulses = 2^{N-1}

Successive Approximation Type ADC:

- It is faster than digital ramp ADC.
- Conversion time (t_c) is independent of the value of the analog input voltage (V_a).
- It has fixed conversion time.

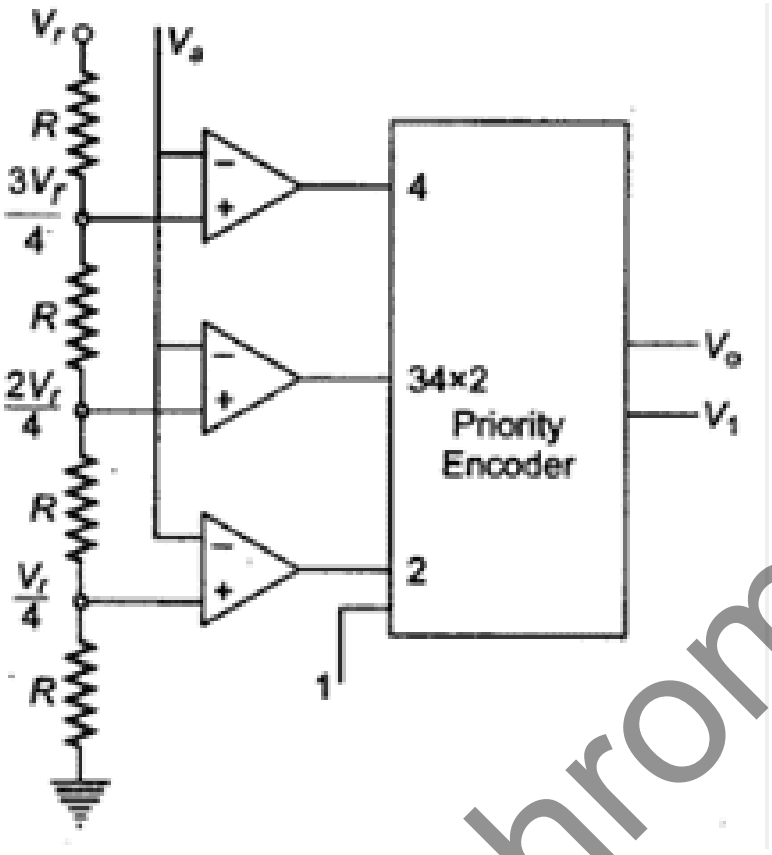


- Maximum number of clock pulses = N for conversion
- Maximum conversion time = N.T_{CLK}

Flash Type ADC:

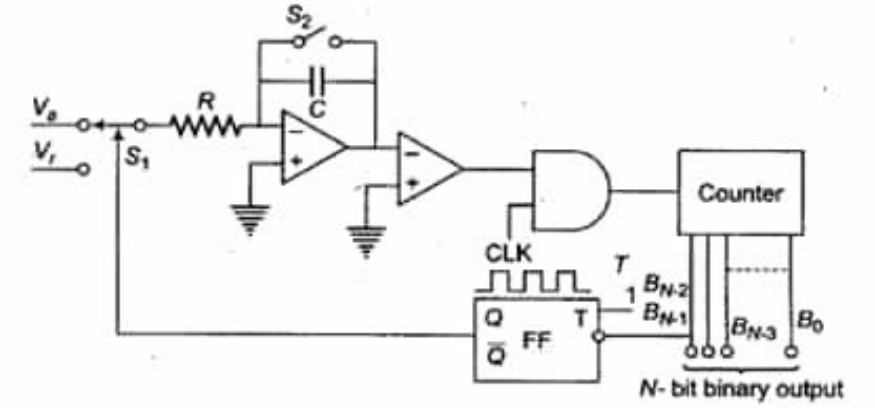
- It is also known as Parallel-comparator type ADC or Simultaneous converter.
- It is highest speed ADC (fastest ADC)
- Functional component
- It utilizes 2^N -1 comparators to compare the input signal level with each of the 2^N -1 possible quantization levels.
- The outputs of the comparators are processed by an encoding-logic block to provide the N bits of the output digital word.
- Complete conversion can be obtained within one clock cycle.
- For N-bit comparator:
  - Total number of comparators = 2^N -1,
  - Total number of resistors = 2^N,
  - Total number of priority encoders = 1 (2^N × N)

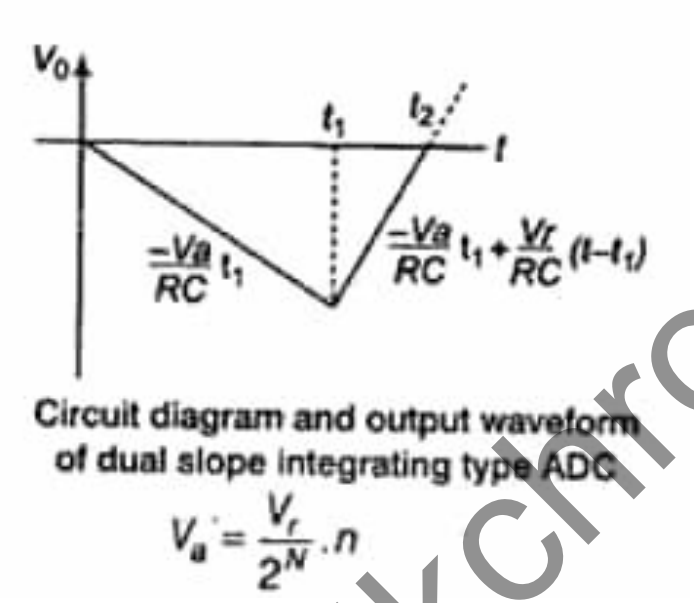
2-bit Flash Converter:



Dual Slope Integrating Type ADC:

- It has slowest conversion time but has relatively low cost.
- The following components are present in the Dual slope A/D converter:
  - Integrator
  - Electronically Controlled Switches
  - Counter
  - Clock
  - Control Logic
  - Comparator





Circuit diagram and output waveform of dual slope integrating type ADC

$$V_a = \frac{V_r}{2^N} . n$$

where, *n* = Count recorded in the counter.

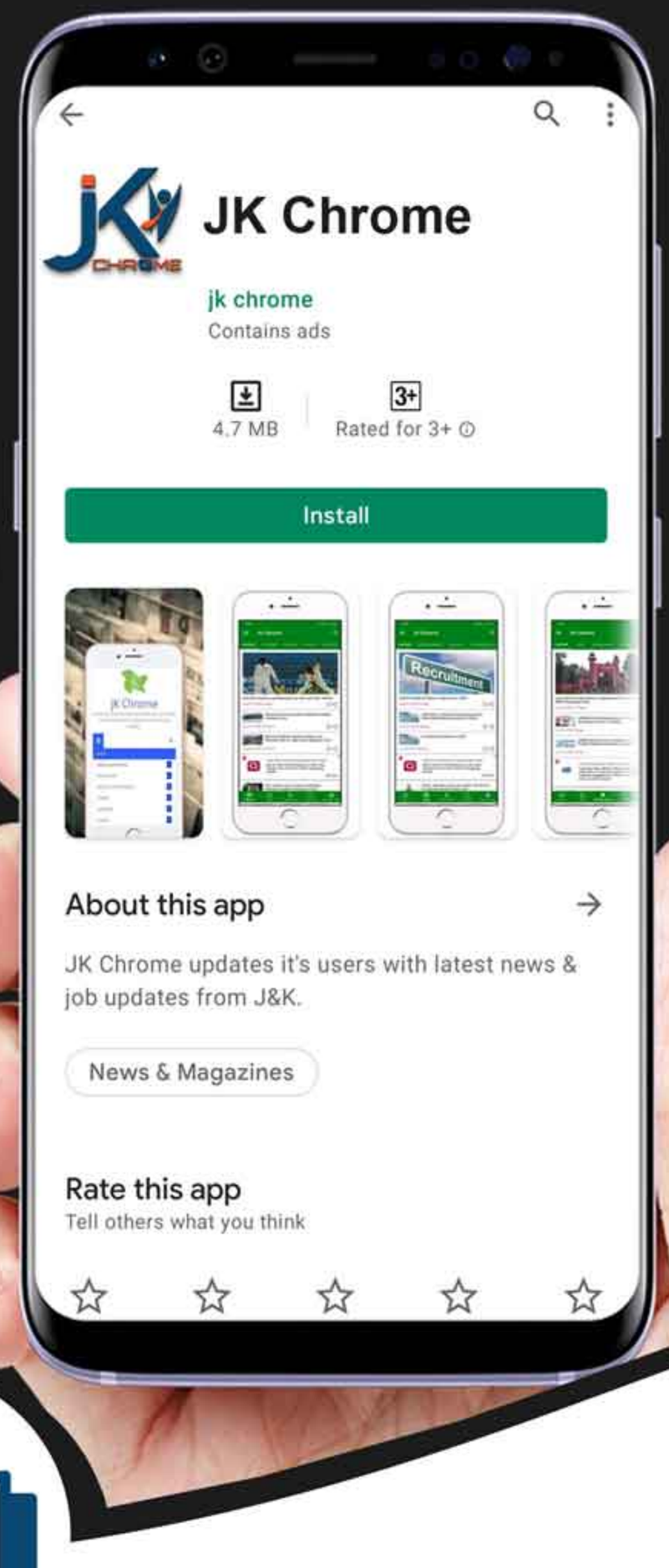
Dual Slope Integrating Type ADC:

- Total number of clock pulses = $2^N + n$
- Maximum number of clock pulses = $2^N + 2^N - 1 = 2^{N+1} - 1 == 2^{N+1}$

Type of ADC	Maximum Number of Clock Pulses	Feature
Counter type	$2^N - 1$	-
SAR	N	
Flash	1	Faster
Dual slope	$2^N + 1$	Most acurate

JK Chrome

JK Chrome | Employment Portal



Rated No.1 Job Application of India

Sarkari Naukri
Private Jobs
Employment News
Study Material
Notifications

JOBS

NOTIFICATIONS

G.K

STUDY MATERIAL

GET IT ON Google play

JK Chrome
jk chrome
Contains ads

Click here to DOWNLOAD

www.jkchrome.com | Email : contact@jkchrome.com