A Handbook on
Computer Science

2

Digital Logic

CONTENTS

1. Logic Functions 72
2. Minimization 78
3. Combinational Circuits 82
4. Sequential Circuits 94
5. Number Representation and Computer Arithmetic 107

1 Logic Functions

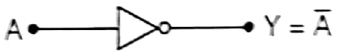
Logic Gates

- OR, AND, NOT are Basic Gates
- NAND, NOR are Universal Gates
- EX-OR, EX-NOR are Arithmetic Gates

NOT Gate

- Also referred to as "Inversion" or "Complementation".

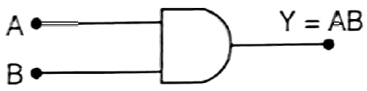
Symbol and Truth table:



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

AND Gate

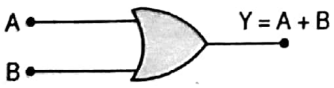
Symbol and Truth table:



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

OR Gate

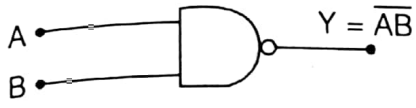
Symbol and Truth table:



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

NAND Gate

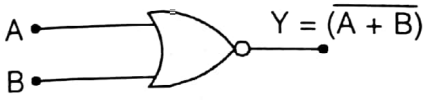
Symbol and Truth table:



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

NOR Gate

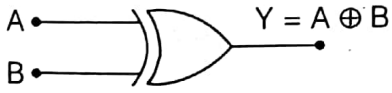
Symbol and Truth table:



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

EXOR Gate

- It is also called "stair case switch".
- **Symbol and Truth table :**



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

- **Boolean function of 2-input EXOR operation :**

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

(*i*) It acts like as an "odd number of 1's detector in the input".

(*ii*) It is mostly used in "parity generation and detection".

(*iii*) When both the inputs are same, then output becomes LOW or Logic '0'.

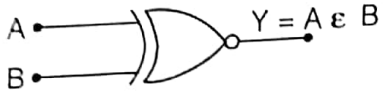
(*iv*) When both the inputs are different, then output becomes HIGH or Logic '1'.

(v) $\boxed{\begin{array}{l} A \oplus A = 0, \quad A \oplus 0 = A \\ A \oplus \overline{A} = 1, \quad A \oplus 1 = \overline{A} \end{array}}$

- $A \oplus A \oplus A \oplus \ldots$ upto n terms = 0, when n = even
- $A \oplus A \oplus A \oplus \ldots$ upto n terms = A, when n = odd

EXNOR Gate

- It acts like as an "even number of 1's detector".
- Symbol and Truth table :



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

- Boolean function of 2-input EXNOR operation :

$$\boxed{Y = A \odot B = \overline{A \oplus B} = (\overline{\overline{A}B + A\overline{B}}) = AB + \overline{A}\overline{B}}$$

(*i*) When both the inputs are same, then output becomes HIGH or Logic '1'.

(*ii*) When both the inputs are different, then output becomes LOW or Logic '0'.

(*iii*) $\boxed{\begin{array}{l} A \odot A = 1, \quad A \odot 1 = A \\ A \odot \overline{A} = 0, \quad A \odot 0 = \overline{A} \end{array}}$

- $A \odot A \odot A \odot \ldots$ upto n terms = 1, when n is even
- $A \odot A \odot A \odot \ldots$ upto n terms = A, when n is odd

Note:

- $\overline{A} \oplus B = A \oplus \overline{B} = A \odot \overline{B}$
- $\overline{A} \odot B = A \odot \overline{B} = A \oplus B$
- For odd no. of inputs EXOR and EXNOR are same and for even number of variables they are complements to each other.

  i.e. $A \oplus B \oplus C = A \odot B \odot C$

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

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

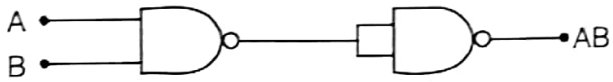
$$= (A \odot B)C + \overline{A \odot B}\,\bar{C}$$

$$= A \odot B \odot C$$

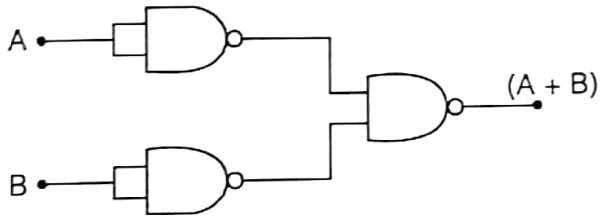
- $\overline{A \oplus B \oplus C} = A \odot B \oplus C = A \oplus B \odot C$
- $\overline{A \odot B \odot C} = A \oplus B \odot C = A \odot B \oplus C$
- $\boxed{A \oplus B \oplus AB = A + B}$
- EXOR and EXNOR are also called arithmetic gates as they are used in addition, subtraction, comparator circuits.

Implementation of Gates using NAND

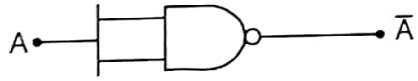
- AND gate :



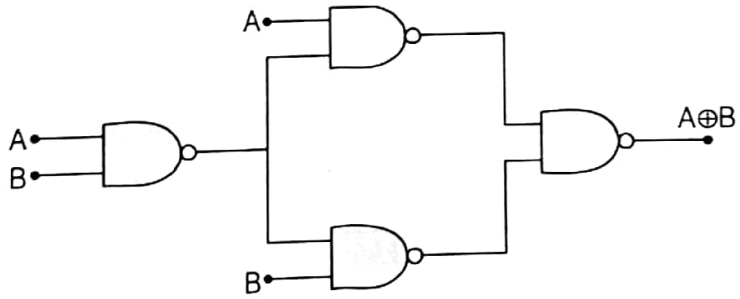
- OR gate :



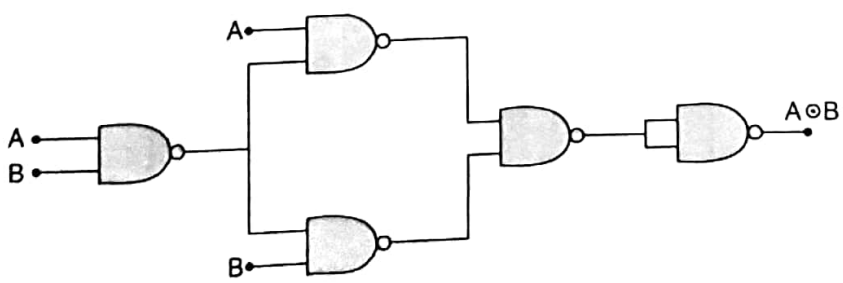
- NOT gate :



- EXOR gate :

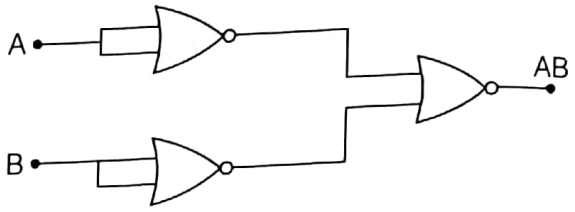


- EXNOR Gate :

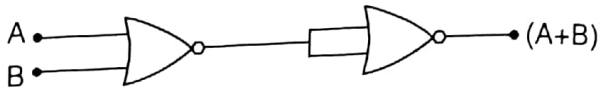


Implementation of Gates using NOR

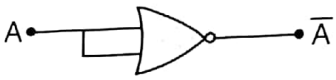
- AND gate :



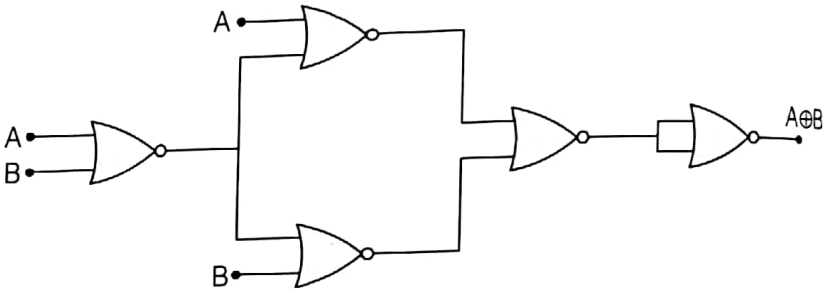
- OR gate :



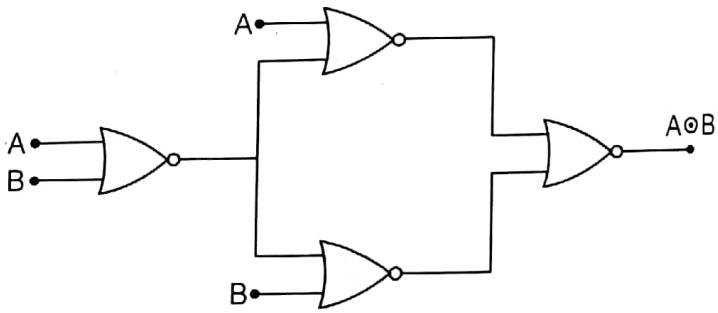
- NOT gate :



- EXOR gate :



- EXNOR gate :



Note:

- Number of NAND and NOR gates needed to implement other logic gates is shown in the following table.

Logic gate	No. of NAND gates	No. of NOR gates
NOT	1	1
AND	2	3
OR	3	2
EX OR	4	5
EX NOR	5	4

- **Alternative Symbols of Gates**

1. Bubbled – OR gate ≡ NAND gate
2. Bubbled – NAND gate ≡ OR gate
3. Bubble – NOR gate ≡ AND gate
4. Bubbled – AND gate ≡ NOR gate

■■■

Minimization

2

Boolean Algebra Laws

Identity Law

(*i*) $A + A = A$

(*ii*) $A \cdot A = A$

Commutative Law

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

(*ii*) $A \cdot B = B \cdot A$

Associative Law

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

(*ii*) $A \cdot (B \cdot C) = (A \cdot B)\, C = A\, B\, C$

Distributive Law

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

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

De Morgan's Law

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

(*ii*) $(AB)' = A' + B'$

Note:

- NAND and NOR gates do not follow associative law but follow commutative law.
- OR, AND, EXOR, EXNOR follow commutative and associative both laws.
- $A + AB = A$
- $A\,(A + B) = A$
- $A + A'B = A + B$
- $A\,(A' + B) = AB$

Boolean Algebraic Theorems

AND-operation Theorem

(*i*) $A \cdot A = A$

(*ii*) $A \cdot 0 = 0$

(*iii*) $A \cdot 1 = A$

(*iv*) $A \cdot \overline{A} = 0$

Involution Theorem

$$(A')' = \overline{\overline{A}} = A$$

OR-operation Theorem

(*i*) $A + A = A$

(*ii*) $A + 0 = A$

(*iii*) $A + 1 = 1$

(*iv*) $A + \overline{A} = 1$

De Morgan's Theorem

(*i*) $\overline{(A_1 \cdot A_2 \cdot A_3 \cdot \ldots A_n)} = \overline{A}_1 + \overline{A}_2 + \overline{A}_3 + \ldots + \overline{A}_n$

(*ii*) $\overline{(A_1 + A_2 + A_3 + \cdots + A_n)} = \overline{A}_1 \cdot \overline{A}_2 \cdot \overline{A}_3 \cdots \overline{A}_n$

Transposition Theorem

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

Distribution Theorem

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

Boolean Algebraic Theorems

Theorem No.	Theorem
1.	$(A+B)\cdot(A+\overline{B}) = A$
2.	$AB + \overline{A}C = (A+C)(\overline{A}+B)$
3.	$(A+B)(\overline{A}+C) = AC + \overline{A}B$
4.	$AB + \overline{A}C + BC = AB + \overline{A}C$
5.	$(A+B)(\overline{A}+C)(B+C) = (A+B)(\overline{A}+C)$
6.	$\overline{A \cdot B \cdot C \cdot \ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots$
7.	$\overline{A + B + C + \ldots} = \overline{A} \cdot \overline{B} \cdot \overline{C} \ldots$

Duality Theorem

- "Dual expression" is equivalent to write a negative logic of the given boolean relation. For this we :
  (*i*) Change each OR sign by an AND sign and vice-versa.
  (*ii*) Complement any '0' or '1' appearing in expression.
  (*iii*) Keep literals as it is.
- For 1-time Dual, it is called "Self Dual Expression".
- For n-variables, maximum possible Self-Dual Function = $(2)^{2^{n-1}} = 2^{(2^n/2)}$
- Number of Boolean functions formed over
  (*i*) *n*-boolean variables = 2^{2^n}
  (*ii*) *n*-*n* valued variables = n^{n^n}
- Number of *n*-valued functions formed over
  (*i*) *n*-boolean variables = n^{2^n}
  (*ii*) *n*-*n* valued varibles = n^{n^n}

Complementary Theorem

For obtaining complement expression we :

1. Change each OR sign by AND sign and vice-versa.
2. Complement any '0' or '1' appearing in expression.
3. Complement the individual literals.

Boolean Function Representation

Canonical Form

All the terms contain each literal.

Example: $F(A,B,C) = \overline{A}BC + ABC + \overline{A}B\overline{C}$

Standard Form

All the terms do not have each and every literal.

Example: $F(A,B,C) = \overline{A} + BC + A\overline{B}C$

Minterms and Maxterms

- *n*-binary variables have 2^n possible combinations and each of these possible combination is called "Minterm or Standard Product".

- "Maxterm" is the complement of corresponding "Minterm" i.e. $\boxed{M = \bar{m}}$.
- In an n-variable Karnaugh-map there are 2^n cells.

Complete Simplification Rules

- Construct the K-map and place 1's in those cells corresponding to the 1's in the truth table. Place 0's in the other cells.
- Examine the map for adjacent 1's and loop those 1's which are not adjacent to any other 1's. These are called isolated 1's.
- Next, look for those 1's which are adjacent to only one other 1. Loop any pair containing such a 1.
- Loop any octet even it contains some 1's that have already been looped.
- Loop any quad that contains one or more 1's which have not already been looped, making sure to use the minimum number of loops.
- Loop any pairs necessarily to include any 1's that have not yet been looped, making sure to use the minimum number of loops.
- Form the OR sum of all the terms generated by each loop.
- K-map will provide minimized expression but not necessarily unique.
- Maximum number of product terms for irreducible expression with n-variables are: 2^{n-1} and maximum number of literals = $n \times 2^{n-1}$.
- Let $f(x_1, x_2, \ldots, x_n)$ be equal to 1 iff k or more variables are equal to 1, then number of prime implicants in f are ${}^nC_k$.

■■■

Combinational Circuits

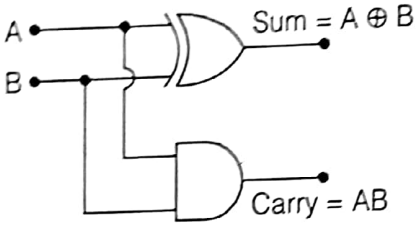
3

Combinational Circuits

- Output does not depend on previous value of input.
- No feedback is required.
- It consists of input variables, logic gates and output variables.
- No memory is required.

Half Adder

- A logic circuit for the addition of two one-bit numbers is known as half adder.
- Symbol and Truth Table:



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

- Logical expression :

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

  Carry = $C = AB$
- Total number of NAND/NOR gates required to implement half adder = 5.
- Minimum number of logic gates needed to implement half adder circuit (if we have all gates except EX-OR and EX-NOR) is "3".
- To implement half adder, three 2 : 1 MUXs are required.

Full Adder

- It performs the arithmetic sum of the three input bits i.e. addend bit, augend bit and carry bit.
- Logical expression :

  Sum = $S = A \oplus B \oplus C$

  Carry = $C = AB + BC + CA = AB + C(A \oplus B)$
- A full adder can be implemented by two half adders and one OR gate or 1 full subtractor and 1 NOT gate.

- Total number of NAND gates/NOR gates required to implement a full adder is "9".

Half Subtractor

- **Logical expression :**

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

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

- Total number of NAND/NOR gates required to implement the half subtractor is "5".

Full Subtractor

- It is a circuit which performs a subtraction between two bits taking into account that a '1' may have been borrowed by a lower significant stage.
- **Logical expression :**

  Difference = $D = A \oplus B \oplus C$

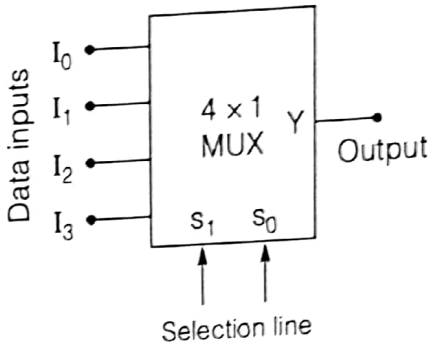
  Borrow = $B = \bar{A}B + \bar{A}C + BC = \bar{A}B + \left(\overline{A \oplus B}\right) \cdot C$

- A full subtractor can be implemented with two half subtractor and one OR gate or 1 full adder and 1 NOT gate.
- Number of NAND/NOR gates required to implement the full subtractor is "9".
- In parallel adder n-full adders or {$(n-1)$ full adders and 1 half adder} or {$(2n-1)$ half adders and $(n-1)$ OR-gates} are required to add two n-bit numbers.

Multiplexers (MUX)

- It selects binary information from one of 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.
- There are 2^n input lines where 'n' is the select line.
- MUX is also called data selector or many to one circuit or universal logic circuit or parallel to serial converter.
- If there are M number of data inputs and n is number of select lines then $n = \log_2 M$.

4 : 1 MUX

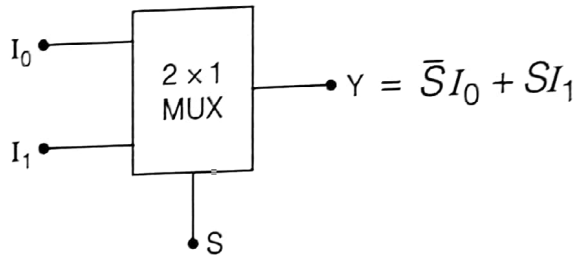


S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Y = \text{output} = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$$

- The size of the MUX is specified by the 2^n input lines and the single output line for n selection lines.

2 : 1 MUX



Implementation of Higher order MUX using Lower order MUX

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

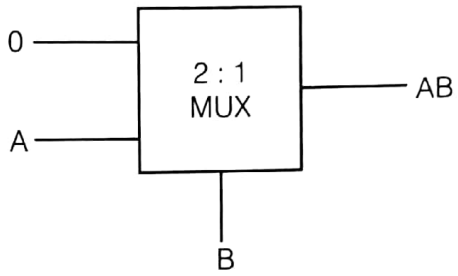
Note:

- To implement 2^n : 1 MUX by using 2 : 1 MUX, the total number of 2 : 1 MUXs required is $(2^n - 1)$.
- MUX is an Universal Logic circuit.
- Number of $(n \times 1)$ MUXs required to implement $(m \times 1)$ MUXs = $\left\lceil \frac{m-1}{n-1} \right\rceil$ and number of levels required = $\log_n m$.

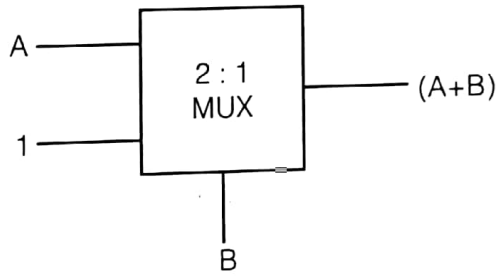
- Any two variable function can be implemented using one 4 : 1 MUX. (some of three variable functions can be implemented)
- With one 4 : 1 MUX and 1 NOT Gate
  - (*i*) All 2 variables logic functions can be implemented.
  - (*ii*) All 3 variables logic functions can be implemented.
- With one 8 : 1 MUX
  - (*i*) All three variable logic functions can be implemented.
  - (*ii*) Some of four variable logic functions can be implemented .
- With one 8 : 1 MUX and 1 NOT gate
  - (*i*) All three variable functions can be implemented.
  - (*ii*) All four variable functions can be implemented.

MUX as Universal Logic Circuit

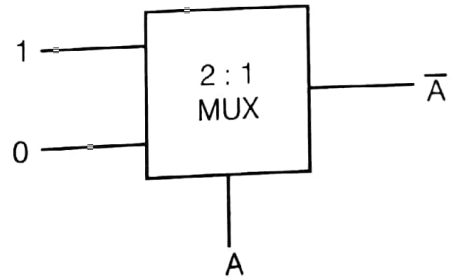
- **AND gate using MUX :**



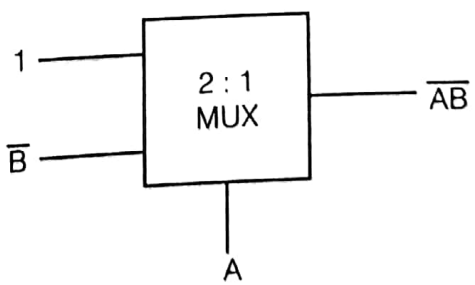
- **OR gate using MUX :**



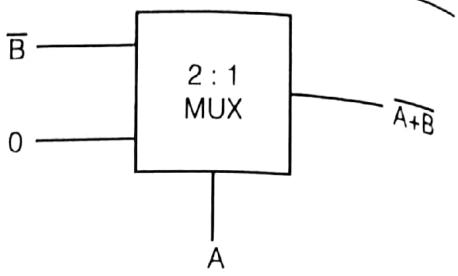
- **NOT gate using MUX :**



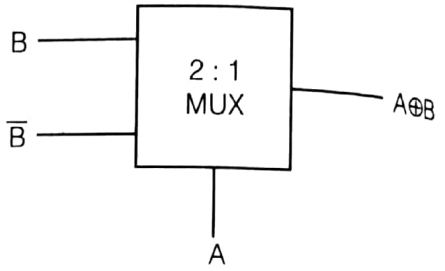
- **NAND gate using two 2 : 1 MUXs :**



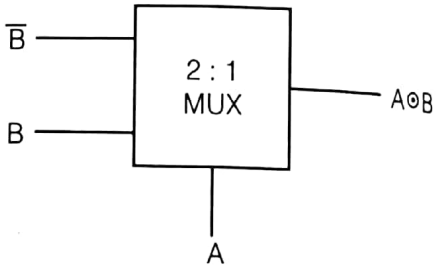
- NOR gate using two 2 : 1 MUXs :



- EXOR gate using two 2 : 1 MUXs :

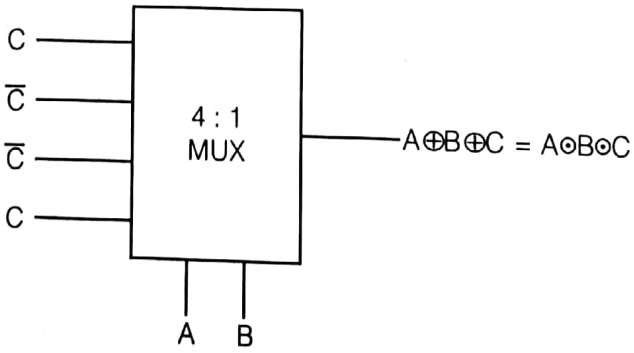


- EXNOR gate using two 2 : 1 MUXs :

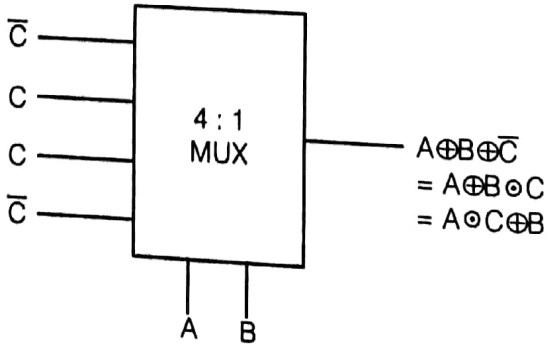


Three Input EXOR and EXNOR Gate Using MUX

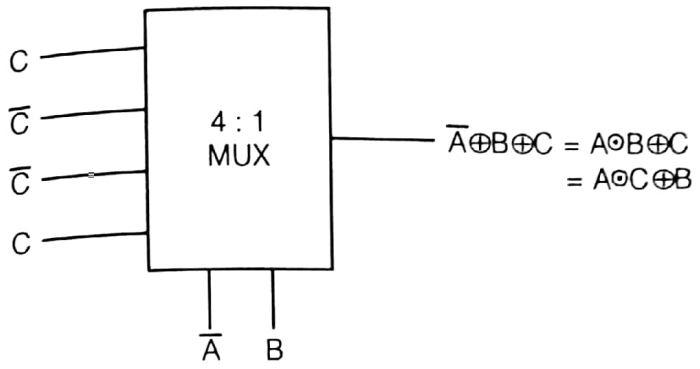
- $A \oplus B \oplus C$



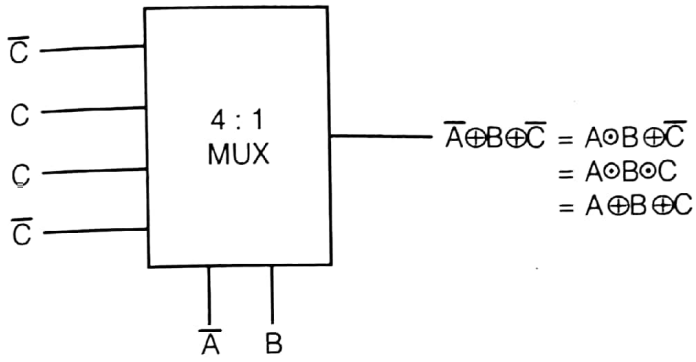
- $A \oplus B \oplus \overline{C}$



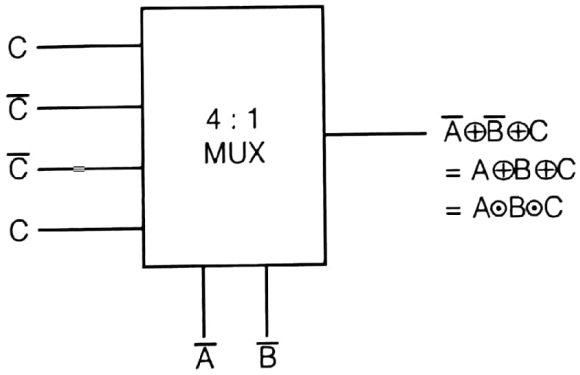
- $\bar{A} \oplus B \oplus C$



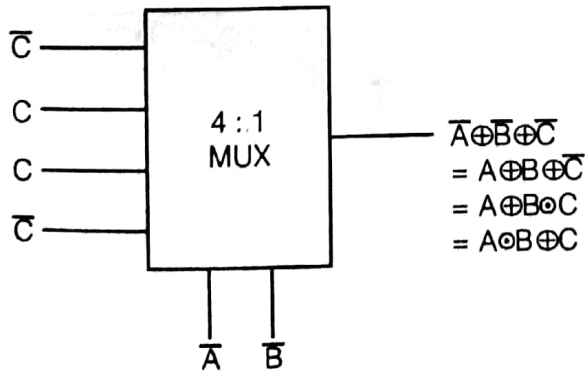
- $\bar{A} \oplus B \oplus \bar{C}$



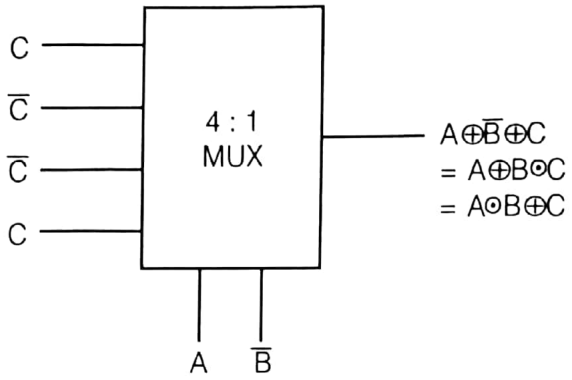
- $\bar{A} \oplus \bar{B} \oplus C$



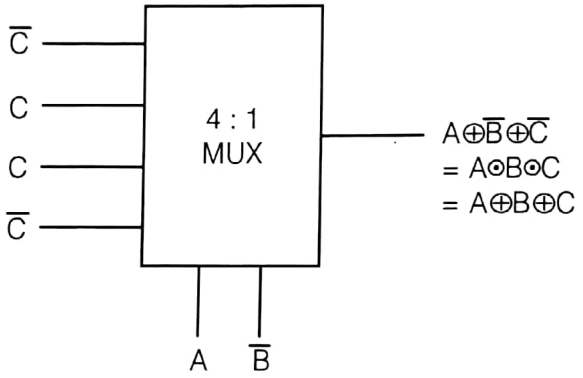
- $\bar{A} \oplus \bar{B} \oplus \bar{C}$



- $A \oplus \overline{B} \oplus C$



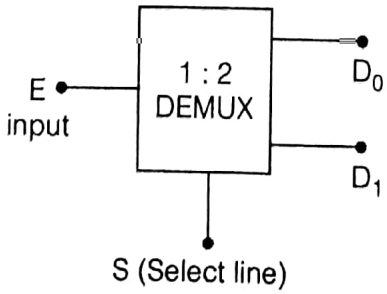
- $A \oplus \overline{B} \oplus \overline{C}$



Demultiplexer (DEMUX)

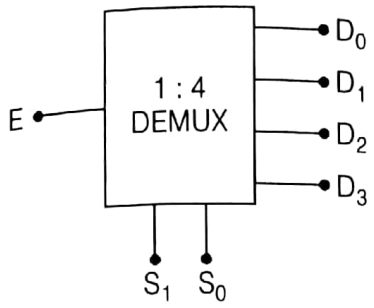
- It receives information on a single line and transmits this on one of 2^n possible output lines.
- A "Decoder" with an enable input can function as a "Demultiplexer".
- Demux is also called data distributor or one to many circuit.
- The number of select lines required in a single input and '*n*' output DEMUX is $\log_2 n$.

1 : 2 DEMUX



$$D_0 = \overline{S}\,E \text{ and } D_1 = SE$$

1 : 4 DEMUX



Select lines		Outputs			
S_1	S_0	D_3	D_2	D_1	D_0
0	0	0	0	0	E
0	1	0	0	E	0
1	0	0	E	0	0
1	1	E	0	0	0

$$D_0 = \bar{S}_1 \bar{S}_0 E, D_1 = \bar{S}_1 S_0 E, D_2 = S_1 \bar{S}_0 E, D_3 = S_1 S_0 E$$

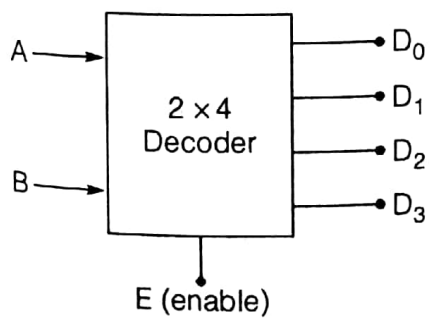
Implementation of Higher order DEMUX using Lower order DEMUX

Given DEMUX	To be Implemented DEMUX	Required Number of DEMUX
1 : 2	1 : 4	1 + 2 = 3
1 : 2	1 : 8	1 + 2 + 4 = 7
1 : 2	1 : 16	1 + 2 + 4 + 8 = 15
1 : 2	1 : 64	1 + 2 + 4 + 8 + 16 + 32 = 63
1 : 4	1 : 16	1 + 4 = 5

Decoders

- A "Decoder" have many inputs and many output lines.
- It is a combinational circuit that converts binary information from n input lines to a maximum 2^n unique output lines.
- If the n-bit decoded information has unused or don't care combinations, the decoder output will have less than 2^n outputs.
- Decoder is used to convert binary data into other codes like binary to octal (3 : 8 decoder) binary to hexadecimal (4 : 16 decoder).
- Total number of output lines : $m \leq 2^n$, where n is total number of input lines.

2 × 4 Decoder



E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

- Number of $n \times 2^n$ decoders required to realize $m \times 2^m$ decoders :

  $x + y + z + \ldots + (I < 2^n)$ where $x = \left\lceil \frac{2^m}{2^n} \right\rceil$, $y = \left\lceil \frac{x}{2^n} \right\rceil$, $z = \left\lceil \frac{y}{2^n} \right\rceil$, ... and

  Number of levels = 1+Number of times dividing with 2^n.
- 2×4 Decoder may act like a 1 : 4 DEMUX and Vice-versa.
- Decoder and Demux circuits are almost same.
- Decoder contains AND gates or NAND gates.
- To implement *n*-functions, we require:
  (a) *n*-muxs or
  (b) One $n \times 2n$ decoder and *n* OR gates.

Implementation of Decoders using Decoder

Given Decoder	To be Implemented Decoder	Required Number of Decoder
2 : 4	4 : 16	1 + 4 = 5
2 : 4	3 : 8	2 + 1 NOT Gate
4 : 16	8 : 256	1 + 16 = 17

Encoder

- Encoder is a combinational circuit which has many inputs and many outputs.
- It is used to convert other codes to binary such as octal to binary, hexadecimal to binary etc.
- In normal encoder only one of input line is high at a time and corresponding binary is available at outputs.
- In priority encoder more than one input line can be high but only for highest priority input line, binary output is available at output.

Code Converters

BCD to Excess-3 Code

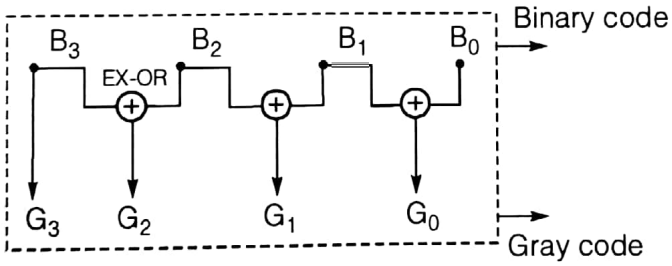
- Let input variables of **BCD code** is A, B, C, D and output variables of excess-3 is W, X, Y, Z.
- Required truth table: (BCD + 0011 = excess-3-code)

Inputs (BCD)				Output (Excess-3 code)			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

- Minimised boolean function :

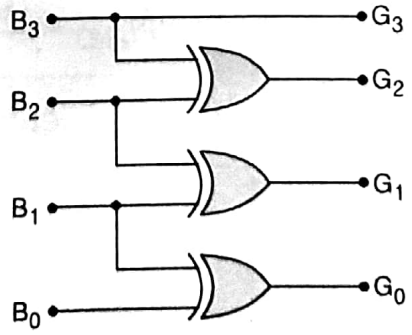
$$Z = \bar{D}$$
$$Y = CD + \bar{C}\bar{D}$$
$$X = \bar{B}C + \bar{B}D + B\bar{C}\bar{D}$$
$$W = A + BC + B\bar{D}$$

Binary-to Gray Code Converter

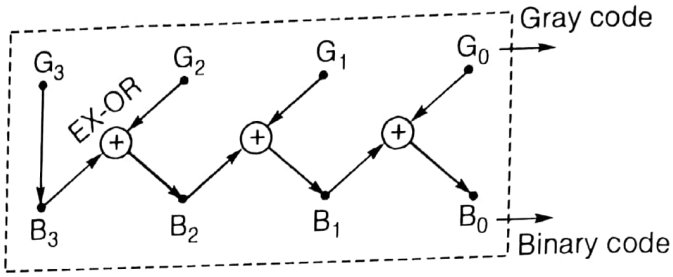


- Equivalent logical gate diagram :

$$G_3 = B_3$$
$$G_2 = B_3 \oplus B_2$$
$$G_1 = B_2 \oplus B_1$$
$$G_0 = B_1 \oplus B_0$$

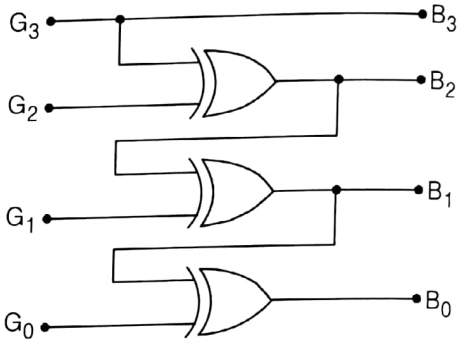


Gray to Binary Converter



- Equivalent logical gate diagram :

$$B_3 = G_3$$
$$B_2 = B_3 \oplus G_2$$
$$B_1 = B_2 \oplus G_1$$
$$B_0 = B_1 \oplus G_0$$



Look Ahead Carry Adder

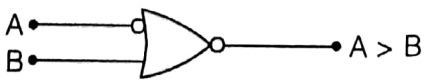
- In parallel adder for n bits there is $2n\ t_{pd}$ delay for providing result where t_{pd} is delay provided by each logic gate.
- Look ahead carry adder is used to make the addition faster.
- In look ahead carry adder to provide carry output it requires 3 logic gate delay.
- To provide sum it requires 4 logic gate array.
- Carry circuit is implemented using two level AND-OR gate circuit.
- Number of AND gates required in carry circuit for n bit look ahead carry adder is :

$$N_{AND} = \frac{n(n+1)}{2}$$

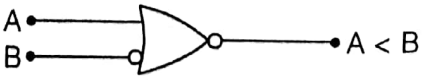
- Number of OR gates require in carry circuit for n-bit look ahead carry adder is n.

Comparator

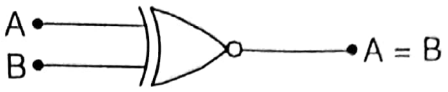
- $(A > B) = A\bar{B}$



- $(A < B) = \bar{A}B$



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



- **Example:**
  Suppose we have

$$P = A_3 A_2 A_1 A_0$$
$$Q = B_3 B_2 B_1 B_0$$

Logic expression for $P = Q$, $P > Q$, $P < Q$ is

$$Y_1 \text{[for } P = Q\text{]} = (A_2 \odot B_3) \cdot (A_2 \odot B_2) \cdot (A_1 \odot B_1) \cdot (A_0 \odot B_0)$$

$$Y_2 \text{[for } P > Q\text{]} = A_3 \bar{B}_3 + (A_3 \odot B_3)(A_2 \bar{B}_2) + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \bar{B}_1) + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1).(A_0 \bar{B}_0)$$

$$Y_3 \text{[for } P < Q\text{]} = \bar{A}_3 B_3 + (A_3 \odot B_3)(\bar{A}_2 B_2) + (A_3 \odot B_3)(A_2 \odot B_2)(\bar{A}_1 B_1) + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(\bar{A}_0 B_0)$$

■■■

Sequential Circuits

4

Sequential Circuit

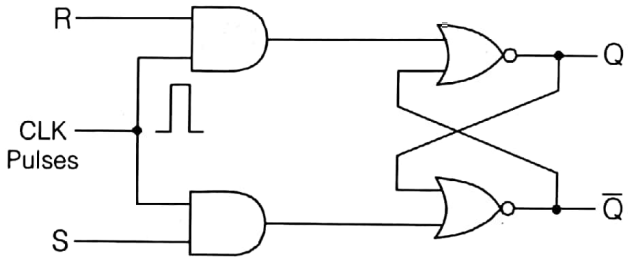
- Output depends on the present as well previous value of inputs.
- It consists input variables, sequential circuit and output.
- Memory is required.

Flip-Flops

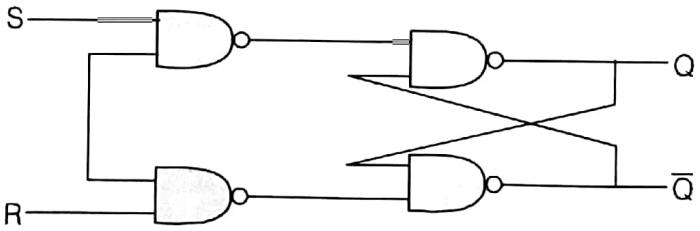
- These can store "one-bit of information".
- Flip-flop circuit is also known as "Bistable Multivibrator" or Latch".
- A flip-flop circuit can be constructed from two NAND-gates or two NOR-gates.

Clocked S-R Flip-flop

- Logic diagram 1 :



- Logic diagram 2 :



- Truth table of S-R FF :

Clock	S_n	R_n	Q_{n+1}	
0	X	X	Q_n	
1	0	0	Q_n	→ HOLD state
1	0	1	0	→ RESET state
1	1	0	1	→ SET state
1	1	1	Invalid	→ FORBIDDEN state

- S_n and R_n denotes the inputs and Q_n the output during the bit time 'n'.
- 'Q_{n+1}' denotes the output Q after CLK passes, i.e. in bit time $(n + 1)$.
- Characteristics table of SR-FF :

S	R	Q_n	Q_{n+1}	
0	0	0	0	→ Q_n
0	0	1	1	
0	1	0	0	→ 0
0	1	1	0	
1	0	0	1	→ 1
1	0	1	1	
1	1	0	X	→ Invalid
1	1	1	X	

- Excitation table of S-R flip flop :

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

- Characteristic equation of SR-FF :

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

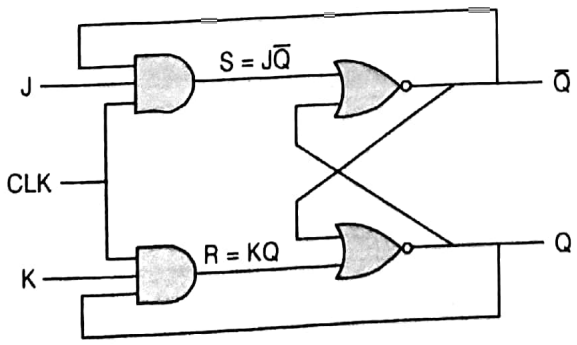
- Invalid states are present when both the inputs (S and R) are HIGH.
- Characteristic equation is valid only for $S.R = 0$.

J-K Flip-Flop

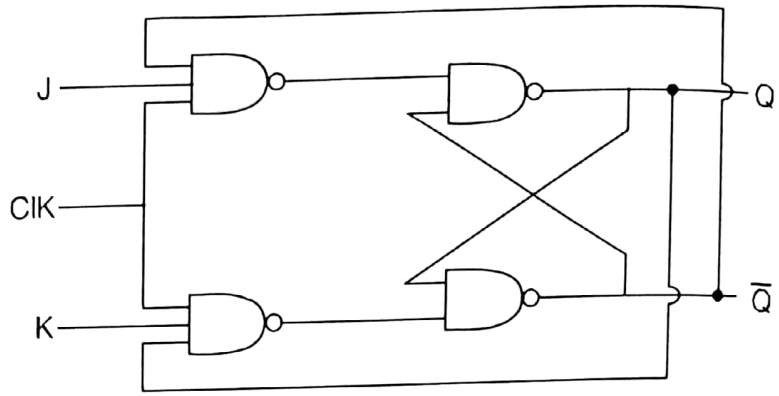
- JK-flip-flop is universal flip-flop because the flip-flops like D-flip-flop, SR-flip-flop and T-flip-flop can be derived from it.

$$S = J\bar{Q} \text{ and } R = KQ$$

- Logic diagram 1 :



- Logic diagram 2 :



- Truth Table of JK-FF :

Clock	J	K	Q_{n+1}	
0	X	X	Q_n	
1	0	0	Q_n	→ HOLD state
1	0	1	0	→ RESET state
1	1	0	1	→ SET state
1	1	1	$\overline{Q}_n$	→ TOGGLE state

- Characteristic table :

J	K	Q_n	Q_{n+1}	
0	0	0	0	→ Q_n
0	0	1	1	
0	1	0	0	→ 0
0	1	1	0	
1	0	0	1	→ 1
1	0	1	1	
1	1	0	1	→ $\overline{Q}_n$
1	1	1	0	

- **Excitation table of J-k flip flop :**

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

- Characteristic equation of JK Flip-flop :

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

- The "Race-around condition" will occur when :

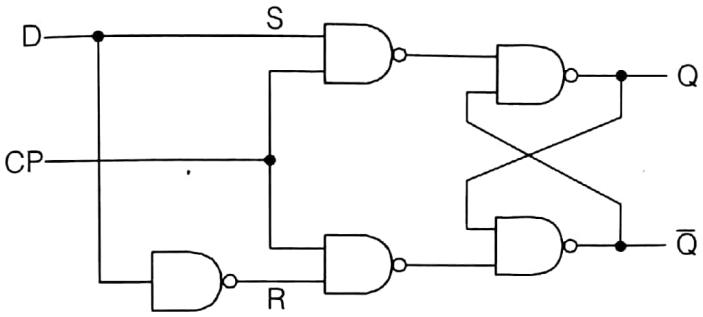
$$J = K = 1 \text{ and } t_{pd(FF)} < t_{pw}$$

- To avoid the "Race-around condition", we should maintain :

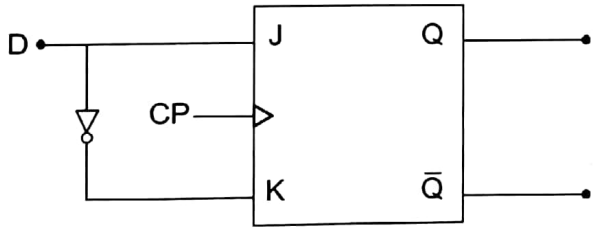
$$t_{pw} < t_{pd(FF)} < T$$

D-Flip-flop

- It is a FF with a delay equal to exactly one cycle of CLK :



- Graphical diagram :



$$J = D \text{ and } K = \bar{D}$$

- Truth table and Characteristic table :

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

(Truth table)

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

(Characteristic table)

- Characteristic equation of D-Flip-flop :

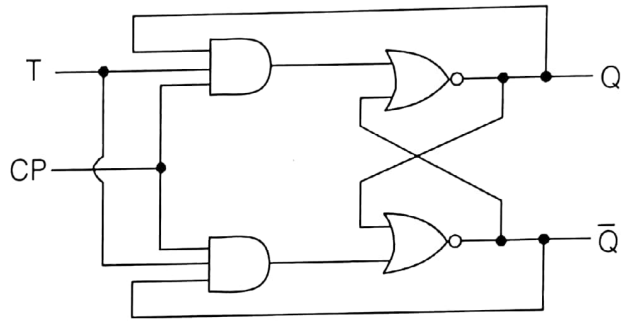
$$Q_{n+1} = D$$

- Excitation table of D-flip flop :

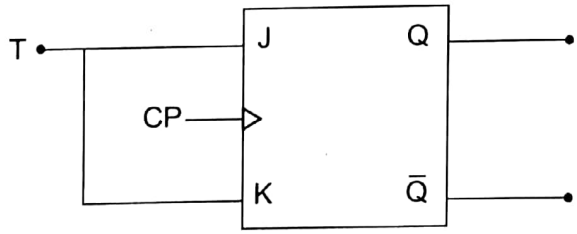
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Toggle Flip-Flop (T-FF)

- Logic diagram of T-FF :



- Graphical symbol of T-FF :



$$J = K = T$$

- Truth table and Characteristic table :

CLK	T	Q_{n+1}	
0	X	Q_n	
1	0	Q_n	→ HOLD state
1	1	$\overline{Q}_n$	→ RESET state

(Truth table)

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

(Characteristic table)

- Characteristic equation of T-Flip-flop :

$$Q_{n+1} = \overline{T}\, Q_n + T\, \overline{Q}_n = T \oplus Q_n$$

- Excitation table of T-flip flop :

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

Conversion from one Flip-flop to another Flip-flop

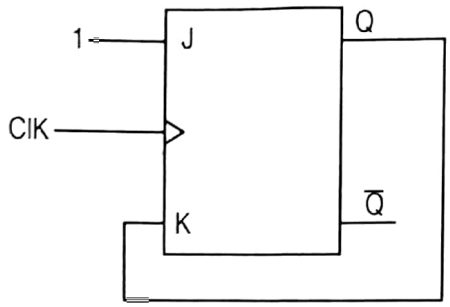
- SR-FF to JK-FF : $S = J\bar{Q}_n$ and $R = KQ_n$
- SR-FF to D-FF : $S = D$ and $R = \bar{D}$
- SR-FF to T-FF : $S = T\bar{Q}$ and $R = TQ$
- JK-FF to SR-FF : $J = S$ and $K = R$
- JK-FF to D-FF : $J = D$ and $K = \bar{D}$
- JK-FF to T-FF : $J = K = T$
- D-FF to SR-FF : $D = S + \bar{R}Q_n$
- D-FF to JK-FF : $D = J\bar{Q} + \bar{K}Q$
- D-FF to T-FF : $D = T \oplus Q$
- T-FF to JK-FF : $T = J\bar{Q} + KQ$
- T-FF to SR-FF : $T = S\bar{Q} + RQ$
- T-FF to D-FF : $T = D \oplus Q$

Note:

- Race-around condition occurs in JK-FF and T Flip-Flop to store 2-bits of information.
- Race-around condition always arises in "Asynchronous circuits".
- A Master-slave FF consists of an SR-FF followed by a T-FF.
- The frequency is always halved at the output of any FFs whose behaviour is same as TOGGLED-FF.

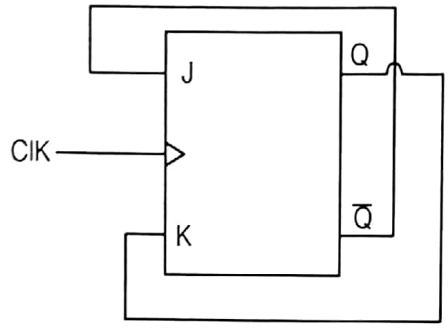
Toggle Mode of Operation

1.



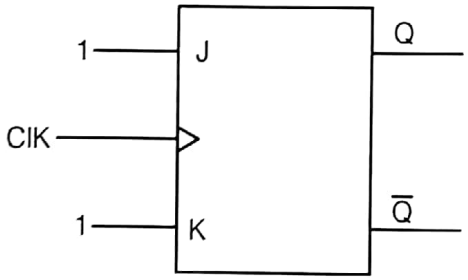
$Q(n+1) = \overline{Q}(n)$

2.



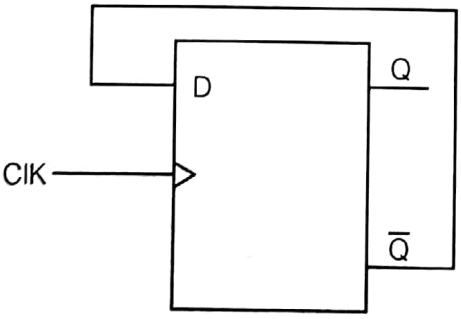
$Q(n+1) = \overline{Q}(n)$

3.



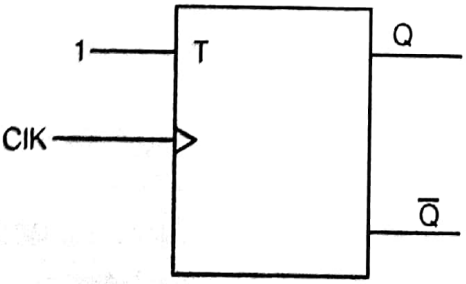
$Q(n+1) = \overline{Q}(n)$

4.



$Q(n+1) = \overline{Q}(n)$

5.



$Q(n+1) = \overline{Q}(n)$

Shift Registers

- In shift register each CLK PULSE shifts the content of register by one-bit to the RIGHT or LEFT.
- The "serial input" determines what goes into the left most flip-flop during the shift.

Serial-In Serial-Out (SISO)

4-bit Right-shift SISO Register

- In right shift SISO register, LSB data is applied at the MSB FF (D-FF).
- In 'n' bit register, to enter 'n' bit data, it requires 'n' clock pulses in serial form.
- If 'n' bit data is to be stored in SISO register then output to take serially for $(n-1)$ clock pulses are required.
- SISO register is used to provide 'n' clock pulse delay to the input data.
- If 'T' is the time period of clock pulse, then delay provided by SISO is nT.

4-bit Left-shift SISO Register

- In this above SISO register MSB data is applied to the LSB FF(D-FF).
- To enter the 'n' bit data in serial form we require 'n' clock pulses.
- To exit or getting output of 'n' bit data as serially we require $(n-1)$ clock pulses.

Serial-In Parallel-Out (SIPO)

- For n-bit serial input data to be stored the number of CLK pulses required = n.
- For n-bit-parallel output data to be stored the number of CLK pulses required = 0 (there is no need of CLK pulse).

Parallel-In Serial-Out (PISO)

- It stores parallel data. To store n bit number of CLK pulses required = 1 CLK pulse.
- To give serial out data number of CLK pulse required = $(n-1)$.

Parallel-In Parallel-Out (PIPO)

- For parallel in data the number of CLK pulses required = 1 CLK pulse.
- For parallel out data the number of CLK pulses required = 0 CLK pulse.
- **Time delay :** A SISO SRs may be used to introduce time delay "Δt" in digital signals

$$\Delta t = N \times T = N \times \frac{1}{f_c}$$

where, N = Number of FFs
T = Time period of CLK pulse
f_c = CLK frequency

- The amount of delay can be controlled by the "f_c" or number of FFs in the SR.

Note:

- All shift registers are JK-FFs.
- "PIPO" register is a storage register made up with D-FFs.
- "PIPO" register is not a shift register.
- Each Shift Left operation multiply the data by 2 and each Shift Right operation divides the data by 2.
- To convert temporal code into spacial code, we use SIPO register. While to convert spacial code into temporal code we use PISO register.

Counters

- Depending on clock pulse applied counters are of two types :

Asynchronous Counters	Synchronous Counters
• Different FF's are applied with different clocks. • Slower • Fixed count sequence either up or down. • Decoding error.	• All FF's are applied with same clock. • Faster • Any count sequence is possible. • No decoding error.

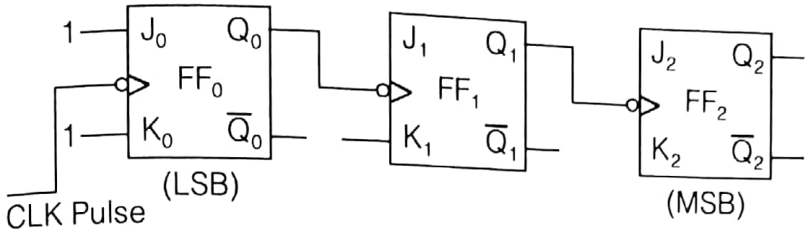
MOD Counter

- The "MOD-number" indicates the number of states in counting sequence.
- For n-FFs, counter will have 2^n different states and then this Counter is said to be "MOD-2^n Counter".
- MOD number indicates the frequency division obtained from the Last FF.
- It would be capable of counting upto $(2^n - 1)$ before returning to zero state.

Up/Down counter

- An "Up/Down Counter" can count in any direction depending upon the control input.

Ripple Counter



- In ripple counter with n-FFs there are 2^n possible states.
- With n-FFs the maximum count that can be counted by this counter is $2^n - 1$.

Disadvantage of Ripple Counter

- Decoding error is present due to propagation delay of FFs i.e. $t_{pd(FF)}$.
- For proper operation of the ripple counter :

$$T_{CLK} \geq nt_{pd(FF)} \quad ; \quad f_{CLK} \leq \frac{1}{nt_{pd(FF)}}$$

- Maximum CLK frequency : $f_{CLK,m} = \dfrac{1}{nt_{pd(FF)}}$

- For determination of Up/Down Counter :

Triggering with	CLK connection in	Access as
(–ve) edge	Q	UP Counter
(–ve) edge	$\overline{Q}$	Down Counter
(+ve) edge	Q	Down Counter
(+ve) edge	$\overline{Q}$	UP Counter

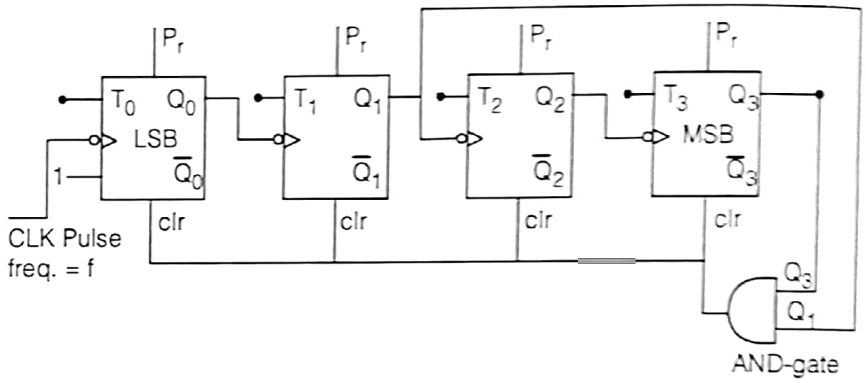
Note:

- In ripple counter flip flop applied with external clock will act as LSB bit.
- Clear and preset are known as asynchronous input to flip flop.
- If N = total number of states and n = number of FFs then $N \leq 2^n$ and $n \geq 3.32 \log_{10} N$.
- If $N = 2^n$, then we get BINARY COUNTER.
- If $N < 2^n$, then we get NON-BINARY COUNTER.

- In "MOD-N Counter", if applied input frequency is '*f*', then output frequency is *f*/*N*.
- If two counters are cascaded with MOD-M followed by MOD-N, then number of overall states of cascaded counter is (M × N) and counter is called "MOD-MN" counter.

Non-Binary Ripple Counter

Decade Counter or Mod-10 Counter

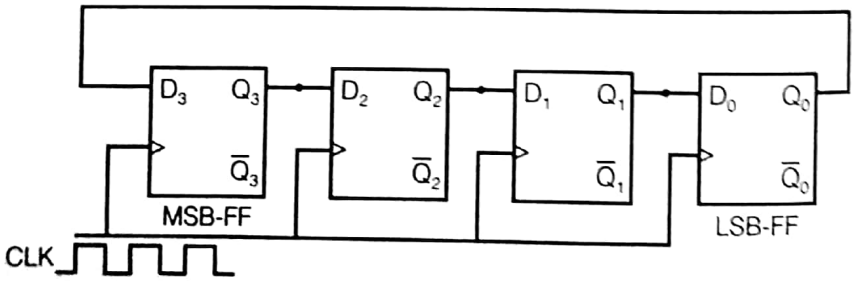


- Used states = 10 and unused states = 6
- For down counter, Mod Number = $2^n - N$.
- Output frequency of MOD-10 counter = *f*/10.

Synchronous (Parallel) Counters

NOT-Self Starting Ring Counter

- It is SISO shift register.



In 4-bit ring counter :

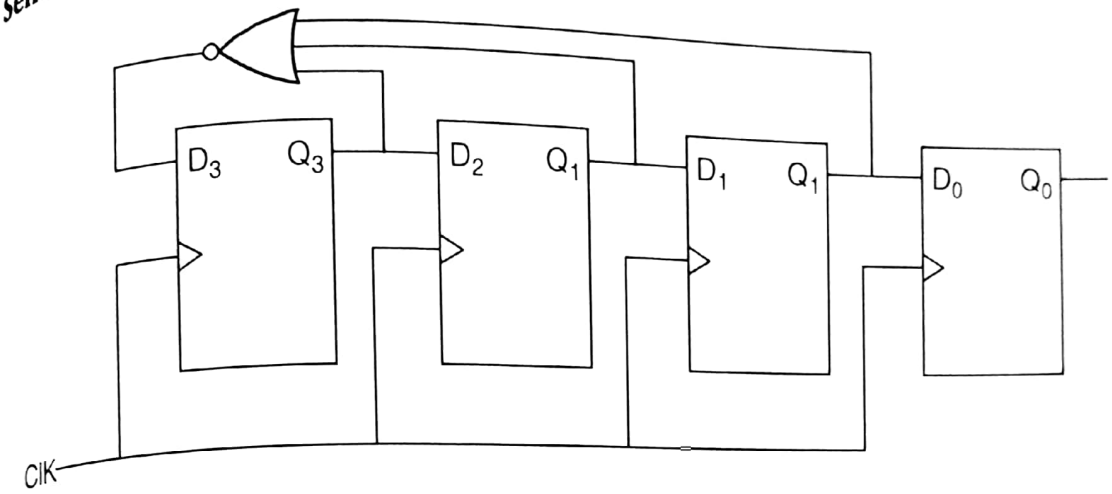
Used states = 4

Unused states = $2^4 - 4 = 12$

- In any counter if CLK frequency is '*f*' the FFs output frequency is "*f*/*N*" (where *N* = Number of states).
- With *n*-FFs, there are n-states present in ring counter.

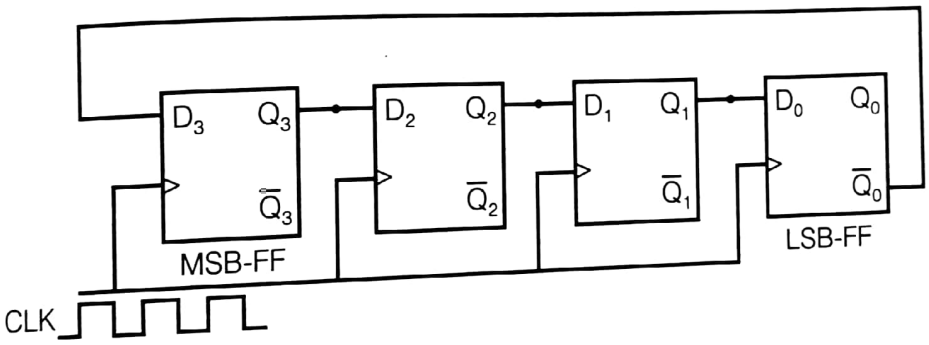
- With *n*-FFs, maximum count possible in ring counter is (2^{n-1}).
- Decoding is very easy in Ring counter, because there is no aid of extra circuit.

Self Starting Ring Counter



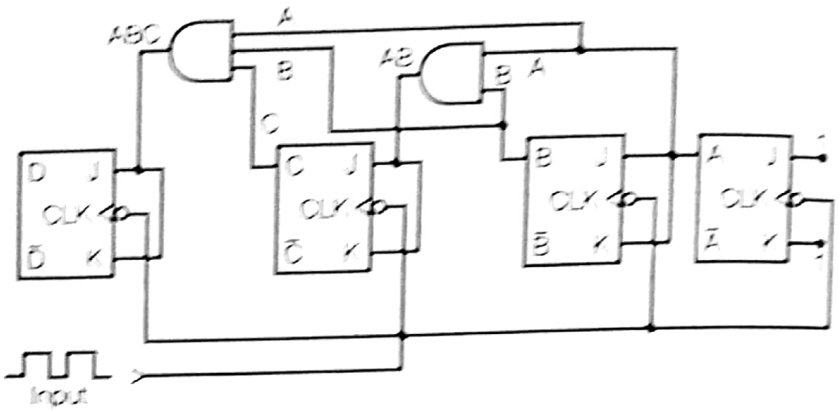
Twisted-Ring Counter

- Also known as Johnson Counter or Switch Tail Ring Counter



- With *n*-FFs there are *2n* states in this counter.
- With n FFs the maximum count by this counter is $(2^n - 1)$.
- In normal "Johnson Counter" with n FFs and the input frequency is '*f*' then output frequency of FFs is "*f*/2*n*"
- In a "Counter" if a feedback connection is used the number of possible states will decrease.
- In normal Johnson counter frequency at output of each flip is *f*/2*n* and duty cycle is 50%.
- In Johnson counter to decode each state one two input AND or NOR gate is used.
- Lockout may occur when counter enter into unused state.

Synchronous-Series Carry Counter



- Clock frequency : $f_{CLK} \leq \dfrac{1}{t_{pd(FF)} + (n-2)\, t_{pd(AND\text{-}gate)}}$
- Total delay of this counter is much lower than an asynchronous counter with same number of FFs.

$$\text{Total delay} = FF\, t_{pd(FF)} + t_{pd(AND\ gate)}$$

Synchronous-Parallel Carry Counter

- It is the "Fastest Counter".
- Clock frequency : $f_{CLK} = \dfrac{1}{t_{pd(FF)} + t_{pd(AND\ gate)}}$

■■■

Number Representation and Computer Arithmetic

5

Digital Number System

- A number system with base 'b' will have 'b' different digits from 0 to $(b-1)$.
- Number representation :

$$(N)_b = d_{n-1}\, d_{n-2} \cdots d_i \cdots d_1 d_0 . d_{-1}\, d_{-2} \cdots d_{-f} \cdots d_{-m}$$

where N = number,

b = base or radix,

$d_{n-1}\, d_{n-2} \ldots d_i \ldots d_1\, d_0$ represents integer portion of number $(N)_b$, $d_{-1}\, d_{-2} \ldots d_{-f} \ldots d_{-m}$ represents fraction portion of number and between these two there is a radical sign.

Weighted Number System

- Binary, octal, hexa decimal, BCD, 2421 etc.

Unweighted Number System

- Gray code, Excess 3-code

Number system	Base (b)	Digits
Binary	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexa decimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
		A, B, C, D, E, F

- In binary number system, a group of Four bits is known as "Nibble" and group of Eight bits is known as "Byte". (4 bits = 1 Nibble, 8 bits = 1 byte).

Codes

Binary Coded Decimal Code (BCD)

- Each digit of a decimal number is represented by binary equivalent.
- In **4-bit binary formats :** Total number of possible representation = 2^4 = 16, Valid BCD codes = 10 and Invalid BCD codes = 6.

- **In 8-bit binary formats :** Valid BCD codes = 100, Invalid BCD codes = 256 – 100 = 156
- BCD is also called 8421 code.
- During arithmetic operation if invalid BCD is present then add $(0110)_{to}$ get correct result.
- It is a not a self complementing code.

Excess-3 Code

- It is a 4-bit code.
- It can be derived from BCD code by adding "3" to each coded number.
- It is a "self-complementing code".
- It is the only code which is unweighted and self complementing.
- 2421, 3321, 4311 and 5211 all are self complementing codes where sum of weight is 9.

Gray Code

- Also called "minimum change codes" in which only one bit in the code group changes when going from one step to the next.
- Gray code is also called cyclic code or reflective code. Since error is minimum so also called **minimum error code**.

Binary-to-Gray Conversion

- 'MSB' in the gray code is same as corresponding digit in binary number.
- Starting from "Left to Right", EXOR each adjacent pair of binary digits to get next gray code digit.

Gray-to-Binary Conversion

- "MSB" of Binary is same as that of gray code.
- If a number system has base b then we can find its b's complement and $(b-1)$'s complement.
- To determine $(b-1)$'s complement subtract given number from maximum number possible to the given base i.e. $(r^n - 1)$.
- To determine b's complement, first determine $(b-1)$'s complement then add 1 to get b's complement of number.
- To convert decimal number into any other base b divide integer part with b and multiply fractional part with b.
- If $(X_3 X_2 X_1 X_0)_b = (A)_{10}$ then $A = X_0 + X_1 b^1 + X_2 b^2 + X_3 b^3$

Digital Number Representation

- In unsigned magnitude representation with 'n' bits, the possible integer values are [0 to $(2^n - 1)$].
- Extra bit is sign bit (MSB)
  if MSB = 0 → positive number
  if MSB = 1 → negative number
- In a sign magnitude representation the range of number.

$$\boxed{-(2^{n-1} - 1) \text{ to } +(2^{n-1} - 1)}$$

- In 1's complement representation, the positive number are represented similar to positive number in sign magnitude, but for representing negative number, first consider positive number and then take 1's complement of that.
- Range of 1's complement number

$$\boxed{-(2^{n-1} - 1) \text{ to } +(2^{n-1} - 1)}$$

- In 2's complement representation of a number positive numbers are represented similar to positive number in sign magnitude, but representing negative number first to write positive number then take 2's complement of it.
- Range of 2's complement representation number : $\boxed{-2^{n-1} \text{ to } +(2^{n-1} - 1)}$

Binary Arithmetic

- When both numbers have same sign then we add only magnitudes and use the sign as MSB.

1's Complement Addition

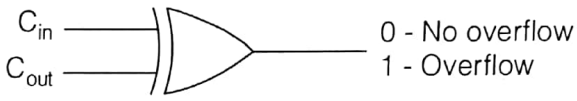
- When the numbers have different sign, keep one number as it is and 1's complement the other then we add magnitude only.
- If carry is present :
  (*i*) Add carry to LSB
  (*ii*) Sign of the result is sign of the uncomplemented number.
- If carry is not present :
  (*i*) Take 1's complement as the result.
  (*ii*) Sign of the result is sign of the complemented number.

2's Complement Addition

- In 2's complement method of representation if the number are of same sign then add the numbers and if carry is generated discard it and if carry is not generated take the 2's complement of results.

Over Flow

- Overflow occurs when two same signed numbers are added in signed representation. To detect overflow we use XOR gate whose inputs are C_{in} and C_{out} where C_{in} is input carry to MSB and C_{out} is output carry from MSB. If output of XOR gate is 1 then overflow, if it is zero then no overflow.



- In 2's complement representation there is only one representation for zero while in 1's complement representation there are two zeros + 0 and – 0.

Fixed Point and Floating Point Representation

- **Fixed Point :** Used for small range of values [Integers].
- **Floating Point :** Used for large range of values [Float/Real].

Fixed Point Representation

- **Fixed point unsigned integers:**

  Let the following n-bit number represent fixed point unsigned integer $(a_{n-1}\, a_{n-2}\, a_{n-3} \ldots a_1\, a_0)_2$

$$V = \sum_{i=0}^{n-1} a_i * 2^i = a_0 * 2^0 + a_1 * 2^1 + \ldots + a_{n-1} * 2^{n-1}; \quad a_i \in (0,\ 1)$$

  (*i*) $V \to V_{min}$, Iff $\forall i, a_i = 0 \Rightarrow V_{min} = 0$

  (*ii*) $V \to V_{max}$, Iff $\forall i, a_i = 1 \Rightarrow V_{max} = 2^0 + 2^1 + \ldots + 2^{n-1}$

  (*iii*) Range: (0 to $2^n - 1$)

- **Fixed Point Unsigned Fraction :**

  Let the following n-bit number represent unsigned fixed point fraction $(\cdot a_1\, a_2\, a_3 \ldots a_n)_2$

$$V = \sum_{i=1}^{n} a_i * 2^{-i}; \quad a_i \in (0,\ 1)$$

(i) $V \to V_{min}$, Iff $\forall i, a_i = 0 \Rightarrow V_{min} = 0$

(ii) $V \to V_{max}$, Iff $\forall i, a_i = 1$

(iii) Error $= \frac{1}{2^{n+1}} + \frac{1}{2^{n+2}} + \ldots\infty = \frac{1}{2^n}\underbrace{\left[\frac{1}{2} + \frac{1}{2^2} + \ldots\infty\right]}$

(iv) Error $= 2^{-n} \equiv$ Weight of least significant bit of a fraction.

(v) Range: (0 to $1 - 2^{-n}$)

(vi) The more the bits in the fraction, less the error, and more the accuracy (precision).

- **Fixed Point Signed Numbers:** The fixed point signed numbers are denoted using sign magnitude notation, 1's and 2's complement notation.

(i) Signed Magnitude Form :

- Sign bit is considered explicitly, hence additional hardware is required for resultant sign of arithmetic.
- Addition and substraction are performed on seperate hardware.
- 0 has 2 representations i.e., +0 : 0000 0000
  −0 : 1000 0000

(ii) 1's Complement Form :

- Sign bit is not considered explicitly hence no additional hardware required.
- 0 has 2 representations i.e., +0 : 0000 0000
  −0 : 1111 1111
- Non weighted system.

(iii) 2's Complement Form :

- Sign bit is not considered explicitly.
- Addition and subtractions are performed by using adder only.

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

- 0 has only 1 representation.
- Only code that assigns negative weight to the sign bit.

Floating Point Representation

- The floating numbers are stored in mantissa exponent form. Any floating point number is represented in the form $\boxed{\pm m \times r^{\pm e}}$;

where, m = mantissa/significant
e = exponent
r = radix/base

- Any floating point number is represented in "normalised" form.
- Zero cannot be normalized.
- In a normalised floating point number the MSB of mantissa must be non-zero.
- The value of expression is given by :
  (*i*) Explicit Normalization : $V = (-1)^{sign}\,[0.M]_2 \times 2^{E-Bias}$
  (*ii*) Implicit Normalization : $V = (-1)^{sign}\,[1.M]_2 \times 2^{E-Bias}$
- The biased exponent is an unsigned number which can represent signed exponent of original number.
- True Exponent = Biased Exponent – Bias

Sign	$E(K)$ Biased Exponent	Mantissa

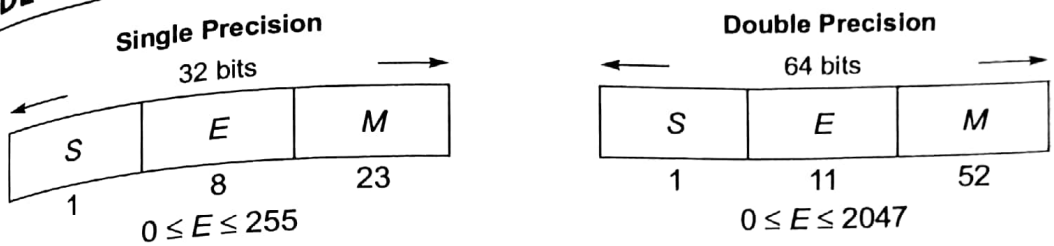
Bias = 2^{K-1}

$0 \le E \le 2^K - 1$, Bias = 2^{K-1}

- The IEEE 754 is concerned with floating point standard. Some of its features are:
  (*i*) The base of the system is 2.
  (*ii*) There is a provision for the values ±0 and ±∞.
  (*iii*) The floating point number is stored either with single precision or with double precision.
  (*iv*) Certain mantissa exponent combinations does not represent any number representing NAN.
  (*v*) The value represented in single precision

(*vi*)

$S(1)$	$E(8)$	$M(23)$	Value
0/1	000....0 $E = 0$	000....0 $M = 0$	± 0
0/1	11....1 $E = 255$	00....0 $M = 0$	± ∞
0/1	11....1 $E = 255$	Not all zeros	NAN (Not A Number)
0/1	000....0 $E = 0$	$M \neq 0$	$(-1)^S (0.M)_2 \times 2^{-126}$
0/1	$1 \le E \le 254$	xxx.....x	$(-1)^S (1.M)_2 \times 2^{E-127}$



Memory

- **Primary memories** : RAM and ROM.
- **Secondary memories** : Semi random and Serial access.

Semi Random Memory

Example : All disc memories (CD's DVD's).

Serial Access

Example : Magnetic tape, Magnetic bubble ferrite core, CCD (Charge coupled device) .

Difference between RAM and ROM

RAM	ROM
• It is read/write memory. • Random access (time required to access any memory is same) • Volatile memory (temporary data storage)	• It read only memory. with decoders and encoders • Random access. • Permanent data storage.

Random Access Memory (RAM)

There are two types of RAM : **Static RAM** and **Dynamic RAM.**

Static RAM	Dynamic RAM
• Data is stored like flip-flop. • BJT, MOSFET is used. • Faster • Power dissipation is more. • Density is less. • Used as cache memory. • No refreshing is required.	• Data is stored in MOS capacitor. • MOSFET is used. • Slower • Power dissipation is low. • Density is high. • Used as main memory. • Refreshing is required.

Difference between Static RAM and Dynamic RAM

Note:

- Size of ROM, m-function with n-variables implementation $= m \times 2^n$.
- Size of PLA, m-functions with n-variables each and p-product terms $= m \times n \times p$.
- Number of links required to implement PLA with x-inputs, p-product terms and y-outputs $= 2xp + yp + y$.

■■■