Boolean Algebra:

1. When no. of variable are less (1, 2, 3)
2. It is preferred when output is 0 or 1.

K-map:

1. When no. of variables are 2, 3, 4, 5 (upto 5 variable)
2. Output is 0, 1 or \( \overline{a} \).

Tabulation method:

1. It is used when no. of variables are more.

Boolean Algebra:

- A complement \( \overline{A} \) or \( A' \)
  \( \overline{A} = A \)

NOT:

- \( 0 = 1 \)
- \( 1 = 0 \)

AND:

- \( 0 \cdot 0 = 0 \) \( A \cdot A = A \)
- \( 0 \cdot 1 = 0 \) \( A \cdot 1 = A \)
- \( 1 \cdot 0 = 0 \) \( A \cdot 0 = 0 \)
- \( 1 \cdot 1 = 1 \) \( A \overline{A} = 0 \)

OR:

- \( 0 + 0 = 0 \) \( A + A = A \)
- \( 0 + 1 = 1 \) \( A + 1 = 1 \)
- \( 1 + 0 = 1 \) \( A + 0 = A \)
- \( 1 + 1 = 1 \) \( A + \overline{A} = 1 \)

Problem: \( AB + \overline{A} \overline{B} \)

Solution:

\( A (B + \overline{B}) \) \( (\because B + \overline{B} = 1) \)

\( = A \)
Problem: \( AB + A\overline{BC} + A\overline{B}C \), find the min. no. of NAND Gates.

Option: (a) 0 \( \checkmark \) (b) 1 \( \checkmark \) (c) 2 (d) 3

Solution:
\[
AB + A\overline{BC} + A\overline{B}C = AB + A\overline{B} (C + \overline{C}) = AB + A\overline{B} (\cdot \cdot \cdot C + \overline{C} = 1) = A \\
\]

No NAND gate required.

Advantage of Minimization:
- \( \Rightarrow \) No. of logic gate \( \downarrow \)
- \( \Rightarrow \) Speed \( \uparrow \)
- \( \Rightarrow \) Power dissipation \( \downarrow \)
- \( \Rightarrow \) Complexity of circuit less
- \( \Rightarrow \) Fan in \( \downarrow \) (no. of input \( \downarrow \))
- \( \Rightarrow \) Cost \( \downarrow \)

Problem: Simplify:
(a) \( \overline{A}B + \overline{A}BC + \overline{A}B\overline{C}D \)

Solution:
\[
\overline{A}BC + A\overline{B} (1 + \overline{C}D) = \overline{A}BC + A\overline{B} (\cdot \cdot \cdot 1 + \overline{C} = 1) = A(\overline{B} + BC) (\cdot \cdot \cdot \overline{B} + BC = \overline{B} + \overline{C}) = A(\overline{B} + \overline{C}) = \overline{A}B + AC
\]

(b) \( (A+B) (A+C) \)

Solution:
\[
A \cdot A + A \cdot C + AB + BC = A + A(1 + B) + BC = A(1 + B + C) + BC = A + BC
\]

Transposition Theorem:
\[
(A + B) (A + C) = A + BC
\]
Similarly:

\[(\bar{x} + y) (\bar{x} + z) = \bar{x} + yz\]

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

Sol:
- Take \(A + B = x\)
- \((x + C)(-A + \bar{B} + C)(x + \bar{C})\)
- \((x + C \bar{C})(A + \bar{B} + C)\)
- \(x(A + \bar{B} + C)\)
- \((A + B)(A + \bar{B} + C)\)
- \(A + B\bar{B} + BC\)
- \(A + BC\)

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

Sol:
- \((A + B)(A + \bar{B})(A + \bar{B})(A + \bar{B})\)
- \((A + B)(A + B)\)
- \((A) = 0\)


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

Sol:
- \((A + A)(A + B)\)
- \(1(A + B) = A + B\)

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

Sol:
- \((A + \bar{A})(A + \bar{B})\)
- \(1(A + \bar{B}) = A + \bar{B}\)

(G) \(AB + \bar{A}\bar{B} + A\bar{B}\)

Sol:
- \(A(B + \bar{B}) + \bar{A}B\)
- \(A + \bar{A}B\)
- \((A + \bar{A})(A + \bar{B})\)
- \(A + \bar{B}\) Ans.
\[ AB + \overline{AB} + A\overline{B} \]

**Solution:**

\[
B(A + \overline{A}) + AB
\]

\[
= \overline{B} + AB
\]

\[
= (\overline{B} + A)(\overline{B} + B)
\]

\[
= A + B \quad \text{Ans}_1
\]

\[ AB\overline{C} + ABC + \overline{A}BC \]

**Solution:**

\[
ABC + ABC + ABC + \overline{A}BC \quad (\because A + A = A)
\]

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

\[
= AB + BC
\]

\[
= B(A + C)
\]

\[ AB + \overline{AC} + \overline{BC} \rightarrow \text{redundant term.} \]

**Solution:**

\[
AB + \overline{AC} + BC(A + \overline{A})
\]

\[
= AB + \overline{A}B + BCA + \overline{ABC}
\]

\[
= AB((1 + \overline{C}) + \overline{AC}(1 + B))
\]

\[
= AB + \overline{A}C
\]

**Note:** In this case BC is known as redundant term, i.e., not used or not compulsory term.

\[ AB + \overline{AC} + BC = AB + \overline{AC} \quad \text{called consensus theorem or redundancy theorem.} \]

**Shortcut method:**

(a) Three variable.

(b) Each variable comes twice.

(c) One variable is complemented.

\[ AB + \overline{B}C + AC \]

**Solution:**

\[
BG + AC \quad \{ \text{The term which is complemented is taken.} \}
\]

\[ AB + BC + AC \]

**Solution:**

\[
\overline{AB} + BC
\]
(m) \((A+B)(\overline{A}+C)(B+C)\)

Sol: \((A+B)(\overline{A}+C)\), \(\therefore (B+C)\) is redundant term.

(n) \((A+B)(\overline{B}+C)(A+C)\)

Sol: \((A+B)(\overline{B}+C)\)

(ο) \(\overline{A}B + \overline{A}C + \overline{B}C\)

Sol: In this case all the variable are complemented only one are uncomplemented, then.

\[= \overline{A}B + \overline{A}C\]

\[\therefore \text{The term which is uncomplemented is taken}\]

(ρ) \(\overline{A}B + \overline{B}C + \overline{A}C\)

Sol: \(\overline{B}C + \overline{A}B \overline{A}C\)

(θ) \((\overline{A} + \overline{B})(\overline{B} + \overline{C})(\overline{A} + \overline{C})\)

Sol: \((\overline{B} + \overline{C})(\overline{A} + \overline{C})\)

\[\overline{A}B \overline{C} = \overline{A} + \overline{B} + \overline{C}\]

\[\overline{A}B \overline{C} = \overline{A} \cdot \overline{B} \cdot \overline{C}\]

Boolean Algebra:

Minimization

SOP \rightarrow \text{minimal}\n
→ \text{canonical}

POS \rightarrow \text{minimal}\n
→ \text{canonical}

Dual

Complement Expression

Truth table

Venn Diagram

Switching circuit

Statement
(A) Minimization :-

(a) \[ XY + X \bar{Y} \bar{W} Z \]

\[ A = XY \quad \text{and} \quad B = \bar{W} Z \]

Then,

\[ \bar{A} + \bar{A} B = (A + \bar{A})(A + B) = A + B = XY + \bar{W} Z \]

(b) let \( f(A \cdot B) = \bar{A} + B \) Then the value of

\[ f \left[ f \left( x + y, y \right), z \right] \]

is


(a) \( xy + z \) 

(b) \( \overline{xy} + z \) 

(c) \( \overline{xy} + y + z \) 

(d) \( x \)

\[ \text{Sol:} \quad f \left[ f \left\{ (x+y), y \right\}, z \right] \]

\[ = f \left[ x + \overline{y} + \overline{y}, z \right] \]

\[ = f \left[ \overline{x} \cdot \overline{y} + y, z \right] \]

\[ = \overline{x} \cdot \overline{y} + y + z \]

\[ = (\overline{x} + \overline{y} + y + z) \]

\[ = xy + y \overline{y} + z \]

\[ = xy + z \quad \text{Ans.} \]

(c) let \( x \cdot y = \overline{x} + y \) and \( z = x \cdot y \)

Then the value of \( z \cdot x \) is

(a) \( x \)

(b) \( 1 \)

(c) \( 0 \)

(d) \( \overline{x} \)

\[ \text{Ans.} \]
(B) **SOP (Sum of Product Form)**

\[ ABC + \overline{A}BC + \overline{A}BC \]

\[ \text{minterm} \]

\[ 9 \]

\[ \Rightarrow \text{In SOP form, each product term is known as minterm or Implicant.} \]

\[ \Rightarrow \text{SOP form is used when O/P of logical expression is 1.} \]

(means 1 \( \rightarrow \) A and 0 \( \rightarrow \overline{A} \))

Ex: 

5 \( \rightarrow \) 101 \( \rightarrow \overline{A}B\overline{C} \)

9 \( \rightarrow \) 1001 \( \rightarrow \overline{A}B\overline{C}D \)

**Ques:** For the given truth table, minimize SOP expression.

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \overline{A} \overline{B} )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( \overline{A} \overline{B} )</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Sol:** In SOP form only 1 taken.

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

\[ = \overline{B} (\overline{A} + A) \]

\[ = \overline{B} \]

\[ \Rightarrow \text{Y can written as :-} \]

\[ Y(A, B) = \Sigma m(0, 2) \]

**Ques:** Simplified the expression for

\[ Y(A, B) = \Sigma m(0, 2, 3) \]

**Sol:**

logical expression in SOP form:-

\[ Y = \overline{A}B + AB + AB \]

\[ = \overline{B} (\overline{A} + A) + AB \]

\[ = \overline{B} + AB \]

\[ = (\overline{B} + A) (\overline{B} + B) \]

\[ = A + \overline{B} \]
sop can be of two form,
(a) Minimal form 
(b) Canonical form

\[ A + \overline{A}B + \overline{A}C \]  
(\text{It is a minimal form})

In canonical form, each term must have all variable.

\[ \overline{A}(\overline{B} + \overline{C}) + \overline{A}B \]
\[ = \overline{A}B + \overline{A}C \oplus \overline{A}B \]

Thus each min-term will contain all variable.

Problem: In canonical sop form , no. of min term presenting the
logical expression \( A + \overline{B}C \) is.
(a) 4  
(b) 5  
(c) 6  
(d) 7

\text{Sol}:-  
\[ A + \overline{B}C \]
\[ = A(\overline{B} + \overline{C})(C + \overline{C}) + \overline{B}C (A + \overline{A}) \]
\[ = (AB + A\overline{B})(C + \overline{C}) + \overline{A}BC + \overline{A}\overline{B}C \]
\[ = ABC + A\overline{B}C + A\overline{B}C + A\overline{B}C + A\overline{B}C + \overline{A}\overline{B}C \]
\[ = ABC + A\overline{B}C + AB\overline{C} + \overline{A}BC + A\overline{B}C + \overline{A}\overline{B}C \]
\[ \text{i.e. 5 terms.} \]
(c) POS Form (Product of Sum):

\[(A + B + \bar{C})(\bar{A} + B + C)(A + B + C)\]

\[l, \text{max \_term}\]

\[\Rightarrow \text{POS form are used when o/p is logic '0'.}\]

\[0 \rightarrow A\]

\[1 \rightarrow \bar{A}\]

Ex: \[5 \rightarrow 101 \rightarrow \bar{A} \cdot B \cdot C\]

\[9 \rightarrow 1001 \rightarrow \bar{A} \cdot B \cdot \bar{C}\]

Ques: For a given truth table, minimize POS expression.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Sols: we take only that value at which o/p is '0'.

\[Y = (A + B)(\bar{A} + \bar{B})\]

\[= \bar{B} + AA\]

\[= \bar{B}\]

\[\Rightarrow Y \text{ can be written in POS form as,}\]

\[Y (A, B) = \Pi M (1, 3) = \bar{B}\]

and for Sop:

\[Y (A, B) = \Sigma m (0, 2) = \bar{B}\]

i.e.

\[\Sigma m(0, 2) = \Pi M (1, 3)\]

\[\Rightarrow \text{If } F (A, B, C) = \Sigma m (0, 1, 4, 7)\]

There are 3 variable then 8 combination then max term are, 2, 3, 5, 6.

\[F (A, B, C) = \Sigma m (0, 1, 4, 7) = \Pi M (2, 3, 5, 6)\]
with 'n' variable, maximum possible minterms or maxterms are \(2^n\). eq.

c) for, \(n = 2\), i.e. \((A, B)\)
Total no. of min or max terms are \(2^2 = 4\).

d) for, \(n = 3\), i.e. \((A, B, c)\)
Total no. of min or max terms are \(2^3 = 8\).

For \(n = 2\), \((A, B)\) total 16 logical expression i.e.

\[
\begin{array}{cccc}
1 & A & AB & AB \\
0 & A & AB & A+B \\
\bar{A}B + A\bar{B} & B & A+B & \bar{A} \bar{B} \\
AB + \bar{A}B & B & \bar{A}B & \bar{A} + \bar{B}
\end{array}
\]

Note: With \(n\) variable maximum possible logical expression are \(2^{2^n}\).

eq. for \(n = 2\), logical expression = \(2^2 = 16\)

for \(n = 3\), = \(2^3 = 256\)

Problem: For \(n = 4\), what is the total no. of logical expression.

Sol: logical expression = \(2^4\)

\[
= 2^4 = 35536.
\]
DUAL FORM:

+ive logic
⇒ +ive logic means higher voltage corresponds to logic '1'.
⇒ logic '0' → 0V
⇒ logic '1' → +5V

-ive logic
⇒ -ive logic means higher voltage corresponds to logic '0'.
⇒ logic '0' = +5V
⇒ logic '1' = 0V

Ques: logic 0 → -5V
logic 1 → 0V

Sol': Higher value of voltage (0V) for logic 1, then +ive logic.

Ques: ECL:
⇒ logic '0' → -1.7V
⇒ logic '1' → -0.8V

Sol': -0.8V is larger value than -1.7V then it is +ive logic.

+ive logic AND
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

-ive logic AND
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

+ive logic OR
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

-ive logic OR
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

⇒ For +ive logic or gate, convert 1 to 0 and 0 to 1.
⇒ we can say that +ive logic AND gate is equal to -ive logic OR gate and -ive logic AND gate is equal to +ive logic OR gate.
Dual expression is used to convert positive logic into negative logic or negative logic to positive logic.

\[ AB \xrightarrow{\text{Dual}} A + B \]
Dual is nothing but negative logic.

\[ \begin{align*}
\text{AND} & \quad \xrightarrow{\text{Dual}} \quad \text{OR} \\
\text{OR} & \quad \xrightarrow{\text{Dual}} \quad \text{AND}
\end{align*} \]

(-) \quad \text{AND} \quad \longleftrightarrow \quad \text{OR} \\
(-) \quad \ast \quad \longleftrightarrow \quad + \\
(-) \quad 1 \quad \longleftrightarrow \quad 0 \\
(-) \quad \text{keep variable as it is}

Find Dual.

\[ ABC + \overline{A}BC + ABC \]

Dual:

\[ (A + B + \overline{C})(\overline{A} + B + C)(A + B + C) \]

If we find again dual then,

\[ ABC + \overline{A}BC + ABC \]

For any logical expression, if two times dual is used resulting same expression.

Self Dual:

\[ AB + BC + AC \]

dual:

\[ \begin{align*}
&= (A + B)(B + C)(A + C) \\
&= (B + AC)(A + C) \\
&= BA + BC + AC + AC \\
&= AB + BC + AC \quad \text{(again same expression)}
\end{align*} \]

In some of the logical expression not all its dual gives the same expression.
In self-dual expression, if one time dual is used result in same expression.

\[ n \text{ variable} \rightarrow \text{self-dual} = 2^{2^{n-1}} \]

If there are \( n \) variables then total no. of self-dual expression is \( 2^{2^{n-1}} \).

**eq:**

(i) For \( n = 1 \) \( \Rightarrow 2^2 = 2 \).

Then 2 dual expression.

\[
\begin{align*}
A &\rightarrow \text{self-dual} \rightarrow A \\
\bar{A} &\rightarrow \bar{A}
\end{align*}
\]

Total self-dual expression are 2.

(ii) For \( n = 2 \) \( \Rightarrow 2^{2^1} = 4 \).

Then 4 dual expression.

\[
\begin{align*}
A &\rightarrow A, \quad B \rightarrow B \\
\bar{A} &\rightarrow \bar{A}, \quad \bar{B} \rightarrow \bar{B}
\end{align*}
\]

(iii) For \( n = 3 \) \( \Rightarrow 2^{2^2} = 16 \).

Then 16 dual expression.

\[
A, \bar{A}, B, \bar{B}, C, \bar{C}, \bar{A}B + BC + \bar{C}A, \quad AB + BC + CA, \quad ...
\]

(E) **Complement:**

if \( Y = ABC + \bar{A}BC + ABC \)

Complement is,

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

1. **AND \leftrightarrow \text{OR}**
2. **\cdot \leftrightarrow +**
3. **1 \leftrightarrow 0**
4. **complement of each variable**
Venn Diagram:

For two variables \((A, B)\):

\[
\begin{array}{c}
A \\
\hline \\
\hline \\
B \\
\end{array}
\]

\[
\begin{array}{c}
\text{AB} \\
\hline \\
\hline \\
\text{A\overline{B}} \\
\end{array}
\]

\[
\begin{array}{c}
\overline{A}B \\
\hline \\
\hline \\
\overline{AB} \\
\end{array}
\]

Guess: For a given venn diagram, minimize the SOP expression for shaded region.

Solution:

\[
Y = \overline{A}B + A\overline{B} + AB
\]

\[
= \overline{B}(\overline{A} + A) + AB
\]

\[
= \overline{B} + AB
\]

\[
= (\overline{B} + A)(\overline{B} + B)
\]

\[
= A + \overline{B}
\]

\[
\text{for POS form}
\]

Guess: SOP expression for shaded region.

Solution:

\[
Y = AB + A\overline{B} + \overline{A}B
\]

\[
= A(\overline{B} + \overline{A}) + \overline{A}B
\]

\[
= A + \overline{A}B
\]

\[
= (A + \overline{A})(A + B)
\]

\[
= A + B
\]

\[
\text{(in POS form)}
\]

Guess: SOP expression
Sol:-
\[ \overline{A}B + A\overline{B} + AB + \overline{A}B \]
\[ = B(A + \overline{A}) + \overline{B}(A + \overline{A}) \]
\[ = B + \overline{B} \]
\[ = 1. \]

\[ \Rightarrow \text{For 3-variable:-} \]

SOP form for shaded portion

\[ = ABC + \overline{A}\overline{B}C + A\overline{B}C + \overline{A}\overline{B}C + ABC + A\overline{BC} \]
\[ = BC(A + \overline{A}) + AB(\overline{C} + C) + AC(B + \overline{B}) \]
\[ \Rightarrow \text{extracted.} \]

\[ = AB + BC + CA \]
(6) Switching Circuit :-

For Series :-

Truth table :-

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

For Parallel :-

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

⇒ In place of bulb if there is resistor then answer remains the same but some drop.

Truth table :-

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

⇒ In place of switch if there is a transistor.

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(i) For A = L, transistor becomes short circuit.

For two switch A and B :-

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
NAND \( \Rightarrow \)

\[ y = \overline{AB} \]

\[ \begin{array}{ccc}
A & B & y \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array} \]

OR \( \Rightarrow \)

\[ y = A + B \]

\[ \begin{array}{ccc}
A & B & y \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array} \]

NOR \( \Rightarrow \)

\[ y = A + B \]

\[ \begin{array}{ccc}
A & B & y \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
\end{array} \]

Question

\[ y = A \cdot (B + C) \cdot \overline{D} \]

Solution

\[ y = (AB + AC) \cdot \overline{D} \]

\[ = ABD + ACD \]
Statement:

Guess: A logic circuit have 3 input A, B, C and output Y is 1. for the following combination.

(i) B and C are true = \overline{B} \overline{C}
(ii) A and C are false = \overline{A} \overline{C}
(iii) A, B and C are true = ABC
(iv) A, B and C are false = \overline{A} \overline{B} \overline{C}

Then minimize the output for Y.

Solution:

\[ Y = BC + \overline{A} \overline{C} + ABC + \overline{A} \overline{B} \overline{C} \]
\[ = BC(1+\overline{A}) + \overline{A} \overline{C}(1+\overline{B}) \]
\[ = BC + \overline{A} \overline{C} \]

If Y = 0, then take max term (POS form).

Guess: A logic circuit have 3 input A, B, C and output Y is 1. when majority no. of inputs are logic 1.

(a) Minimizing expression F
(b) Implement logic circuit

Solution:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ F = \overline{A} \overline{B} \overline{C} + A \overline{B} C + A B \overline{C} + A B C \]
\[ = \overline{A} \overline{B} \overline{C} + A B C + AB \overline{C} + ABC \]
\[ = BC(A+\overline{A}) + AC(B+\overline{B}) + AB(\overline{C}+\overline{C}) \]
\[ = AB + BC + CA. \]
LOGIC GATES:

Basic Building Blocks:

- **NOT**
- **AND**
- **OR**
- **NAND** - universal gate
- **NOR**
- **EXOR** - Arithmetic ckt.
- **EXNOR** - comparator, parity generator/checker, code converter (Binary to Gray, Gray to Binary)

**NOT:**

\[
\begin{array}{c|c}
A & \bar{A} = \overline{Y} \\
0 & 1 \\
1 & 0 \\
\end{array}
\]

Gates: Circuit shown in the fig are:

- (a) Buffer
- (b) Asstable MV
- (c) Bistable MV
- (d) Square wave generator

Solution:

If there is no feedback then it is buffer. In buffer if we apply 0 then get 0

\[
\begin{array}{c|c|c}
& 0 & \text{no} I/P \rightarrow 0 \\
& 1 & \text{no} I/P \rightarrow 1 \\
\end{array}
\]

Buffer means whatever the I/P ie. the O/P.

\[=\] But there is a feedback and the O/P is stable if we give 1 as V/P, O/P is also 1 and it gives 0 then O/P is 0 then two stable state.

\[=\] Hence it is *Bistable multivibrator.*
**Quest**: The circuit shown is

```
[Diagram]
```

**Sol**:

**a)** for: $t_{pd}$

$= 0 \rightarrow S_{pd}$

$= 1 \rightarrow S_{pd}$

It is called

- Square wave generator.
- As output is not stable sometime 1 and sometime 0, hence it is also called an unstable multivibrator.
- Clock generator
- Ring oscillator.

Total time period ($T$) = $6t_{pd}$

then,

$$T = 2N\times t_{pd}$$

$N$ = no. of inverters in feedback

**Quest**: In a circuit shown in fig., the propagation delay of each NOT gate is 100 pSec. Then frequency of generator square wave is

```
[Diagram]
```

(a) 10 GHz
(b) 1 GHz
(c) 100 MHz
(d) 10 MHz

**Sol**:

$T = 2N \times t_{pd}$

$= 2 \times 5 \times 100 \text{pSec} = 1000 \text{pSec}$

$\therefore \frac{1}{f} = \frac{1}{T}$

$= \frac{1}{1000 \times 10^{12} \text{Sec}}$

$= 10^9 \text{Hz}$

$$f = 10^9 \text{Hz}$$

```
The circuit shown in the figure has the propagation delay of each NOT gate as 2nsec. Then time period of generated square wave is

\[
T = 2 \times \text{propagation delay} = 2 \times 3 \times 2\text{nsec} = 12\text{nsec}
\]

Thus time period at x and y is same.

AND gate:

\[
Y = A \cdot B
\]

\[
\begin{array}{c|c|c|c}
A & B & Y \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

\(\Rightarrow\) AND gate follows both commutative law and associative law.

\((\cdot)\) \(AB \equiv BA\)

i.e.

\[
\begin{align*}
A & \quad \text{and} \quad B \\
\quad \text{and} \quad \text{AB} \\
\end{align*}
\]

or,

\[
\begin{align*}
A & \quad \text{or} \quad B \\
\quad \text{or} \quad \text{AB} \\
\end{align*}
\]
1. Disable & Enables:

⇒ Thus o/p remains in '0' due to control if disable. AND gate is not in working state.

⇒ AND gate is in working state o/p is changing in Enabled state.

⇒ In TTL logic family, If any I/P is open and float then it will act as '1'.

⇒ In ECL logic family, floating input will act as logic '0'.

⇒ Question occurs mostly from ECL and TTL in Exam.

Unused I/P's:

1. In Multipin (I/P) AND gate unused I/P can be connected to logic 1. or "pull up".

⇒ unused I/P can be connected to logic '0' or "pull down".
2. \( \begin{array}{c}
A \\
B
\end{array} \quad \begin{array}{c}
A \cdot B \\
= AB
\end{array} \quad \begin{array}{c}
A \\
B
\end{array} \quad \begin{array}{c}
A \cdot A \cdot B \\
= AB
\end{array} \)

- unused I/P can be connected to one of the used I/P.

3. \( \begin{array}{c}
A \\
B
\end{array} \rightarrow Y = AB \) (only for TTL)

- If it is TTL logic family, then unused I/P can be open or floated (unconnected)

Note: Because of unnecessary I/P attached to B, fan in will be down.

- Best way to connecting unused pin (I/P) in AND gate is connecting to logic '1'.

\( \begin{array}{c}
A \\
B
\end{array} \rightarrow Y = AB \)
OR Gate: (Inclusive OR)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

When any of the I/P is High in OR gate then O/P is High.

OR gate follows both commutative and associative law.

I) Commutative law:

\[ A + B = B + A \]

\[ A + B = A + B \]

II) Associative law:

\[ A + B + C = (A + B) + C = A + (B + C) \]

Enable and Disable:

I/P is changing as O/P is changing or we say the gate is enabled.
Unused I/P's:

1. In OR gate, unused I/P is connected to logic '0'—"pull down."

2. Connect to one of the used I/P.

3. If it is ECL then unused I/P can be open or floated.

4. In OR gate, Best way of connecting the unused I/P is 0 to connect to logic '0'.

\[ Y = A + B \]

**Problem:** In the circuit shown in fig. in TTL, AND, OR, INVERTER circuit for the given I/P 0 I/P is

\[ \begin{array}{c}
\text{A} \\
\text{B} \\
\text{AB} \\
\text{1+AB} \\
\text{0} \\
\text{Y} \\
\end{array} \]

\[ \begin{array}{c}
\text{1} \\
\text{1} \\
\text{0} \\
\text{0} \\
\text{0} \\
\text{0} \\
\end{array} \]

(a) 0
(b) 1
(c) AB
(d) AB

**Solution:** In TTL, all I/P's are float then it is logic 1.
Problem. For ECL AND, OR, INVERTER

\[ \text{Diagram of ECL AND, OR, INVERTER} \]

(A) 0
(B) 1
(C) AB
(D) \( \overline{AB} \)

Sol.: If all \( \text{i/p} \) are floating in ECL then it is 0,
and \( \text{o/p} y = \overline{AB} \) Ans.
**NAND GATE:** (Bubbled OR)

\[ \bar{A} \bar{B} = \bar{A} \lor \bar{B} \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \bar{A} \lor \bar{B} \]

\( \Rightarrow \) when both I/p high the o/p is low.

[Diagram of NAND gate]

0 disable (not changing if one 1 enable.
I/p is zero)

\( \Rightarrow \) NAND gate follow commutative law but not follow associative law.

\[ \overline{A \land B} \neq \overline{A \land C} \land \overline{B \land C} \]

\[ (A \land B) \land C = A \land B \lor C \]

\( \Rightarrow \) The only two gate not follow associative law i.e. universal gate NAND or NOR gate.

\( \Rightarrow \) unused I/p in NAND gate can be connected similar to unused I/p in AND gate.

\[ A \land B \]

**NOR GATE:** (Bubbled AND)

\( \Rightarrow \) OR gate followed by NOT gate.

\[ \bar{A} \lor \bar{B} = \bar{A} \cdot \bar{B} \]

\[ A \cdot B = \bar{A} \cdot \bar{B} \]
A  B  Y
0  0  1  \[\Rightarrow\text{when both I/p is low the o/p is High.}\]
0  1  0
1  0  0
1  1  0

\[\begin{array}{c}
A \\
\uparrow \\
\downarrow \\
\rightarrow \\
enable
\end{array}\quad \begin{array}{c}
A \\
\uparrow \\
\downarrow \\
\rightarrow \\
disable
\end{array}\]

1. Enable and disable both are same as OR gate.
2. NOR gate follows commutative law and not follow associative law.
   
i.e. \( A+B = B+A \)
   
\( A+B+C \neq A+B+C \)

\[\Rightarrow\text{unused I/p in NOR gate can be connected similar to OR gate.}\]

EXOR \(\oplus\), XOR :-

\[\Rightarrow\text{Exclusive OR gate.}\]

OR gate is also called as inclusive OR gate.

\[Y = A \oplus B\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
</table>
| 0  | 0  | 0  | \[\Rightarrow\text{when } A = B , o/p \text{ is low i.e '0'.}\]
| 0  | 1  | 1  | \[\Rightarrow\text{when } A \neq B , o/p \text{ is High i.e. logic '1'.}\]
| 1  | 0  | 1  |
| 1  | 1  | 0  |
It is also called controlled inverter.

**Note:**
- \( A \oplus A = 0 \)
- \( A \oplus 0 = A \)
- \( A \oplus 1 = \bar{A} \)
- \( A \oplus B = C \) then,
  1. \( A \oplus C = B \)
  2. \( B \oplus C = A \)
  3. \( A \oplus B \oplus C = 0 \)

Since, \( A \oplus A = 0 \) \( A \oplus A \oplus A = A \) \( A \oplus A \oplus A \oplus A = 0 \) Then we say:
- Odd no. of same I/P gives same O/P and even no. of same I/P gives same as O/P.

\( B \oplus B \oplus B \oplus \ldots n = B \), if \( n \) is odd
\( = 0 \), if \( n \) is even.

**Problem:** Theckt shown in fig. contains cascading of 20 EXOR gate. If \( x \) is the I/P then O/P is.

(a) 0
(b) 1
(c) \( x \)
(d) \( \bar{x} \)

**Sol:** \( O/P \) of even EXOR gate have same O/P.

Ans: \( x \)
Internal diagram of EXOR gate:

\[ Y = \overline{A}B + \overline{B}A \]

- EXOR gate follows both commutative and associative law.
- EXOR gate is available with two I/P's only.

Truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Y (A ⊕ B ⊕ C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- The output of EXOR gate is 1 when the number of 1's at the I/P is odd.

Logical expression:

\[ Y = \overline{A}BC + \overline{A}BC + ABC + ABC \]

\[ = 001 + 010 + 100 + 111 \rightarrow \text{odd no. of 1's} \]

\[ Y = \Sigma m(1,2,4,7) \]

- The reduced form of this expression is,

\[ A \oplus B \oplus C \]
**EXNOR or XNOR:**

\[
\begin{array}{cccc}
A & B & Y \\
0 & 0 & 1 & \Rightarrow \text{whenever the I/p is same o/p is high.} \\
0 & 1 & 0 & \\
1 & 0 & 0 & \\
1 & 1 & 1 & \\
\end{array}
\]

**SOP expression** = \( \overline{A}B + AB \)

**POS expression** = \( (A + \overline{B})(\overline{A} + B) \)

\[
\begin{array}{|c|c|}
\hline
\text{EXOR} & \overline{A}B + AB \leftrightarrow \text{SOP} \quad \text{EXNOR} & \overline{A}B + AB \\
& (A + \overline{B})(\overline{A} + B) \leftrightarrow \text{POS} & (A + \overline{B})(\overline{A} + B) \\
\hline
\end{array}
\]

\( \Rightarrow \) when \( A = B \), then o/p is high. Therefore, coincidence logic ckt and also called as equivalent detector.

\( \Rightarrow \) when \( A \neq B \), the o/p is low.

\( \Rightarrow \) Enable and Disable:

\[
\begin{array}{ccc}
\text{control} & \text{(inverter)} & 0 \\
\hline
A & \overline{A} & 1 \\
\hline
\end{array}
\]

\[
\begin{array}{ccc}
\text{control} & \text{(Buffer)} & 1 \\
\hline
A & \overline{A} & 0 \\
\hline
\end{array}
\]
\[
\begin{align*}
A \oplus A &= 1 \\
A \oplus \bar{A} &= 0 \\
A \oplus 0 &= \bar{A} \\
A \oplus 1 &= A \\
\end{align*}
\]

Since, \( A \oplus A = 1 \)
\[
A \oplus A \oplus A = A \\
A \oplus A \oplus A \oplus A = 1.
\]

and so on.

\[
B \oplus B \oplus B \oplus B \ldots n = 1, \quad \text{if}\ n = \text{even} \\
B \quad \text{if}\ n = \text{odd}
\]

\( \Rightarrow \) \ EXOR and EXNOR is not always complement, it is complement only when the no. of \( 1/0 \) is even and if \( 1/p \) is odd then \( \oplus \) OR \( \oplus \) EXNOR are same.

i.e. \( A \oplus B \oplus C = A \oplus B \oplus C \Rightarrow \text{same.} \)

and, \( A \oplus B \oplus C \oplus D = A \oplus B \oplus C \oplus D \Rightarrow \text{complement} \)

Guess: Find expression of \( A \oplus B \oplus C \).

Sol:-
\[
\begin{align*}
A \oplus B \oplus C \\
= (\bar{A}B + AB) \oplus C \\
= (\bar{A}B + AB)C + (\bar{A}B + AB)C \\
= (\bar{A}B \cdot AB)C + (\bar{A}B + AB)C \\
\text{Since,} \\
(\bar{A}B + AB) = (A \oplus B) = A \oplus B = \bar{A}B + AB \\
= (\bar{A}B + AB)C + (\bar{A}B + AB)C \\
= ABC + ABC + \bar{AB}C + \bar{ABC} + ABC \\
= A \oplus B \oplus C.
\end{align*}
\]
Minimize

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Y</th>
</tr>
</thead>
</table>
| 0 | 0 | 0 | 1 | (A) $A \oplus B \oplus C$
| 0 | 0 | 1 | 0 | (B) $A \oplus B \ominus C$
| 0 | 1 | 0 | 0 | (C) $A \ominus B \ominus C$
| 1 | 0 | 0 | 1 | (D) $A \oplus B + B \ominus C + A \ominus C$
| 1 | 0 | 1 | 1
| 1 | 1 | 0 | 1
| 1 | 1 | 1 | Q

Sol:- for EXOR $\Rightarrow$ O/P is 1 when odd no. of 1's at I/P.

In this case,

\[ Y = \overline{A \oplus B \ominus C} \]
\[ = \overline{A \oplus B \ominus C} \text{ Ans}_2 \]

⇒ EXOR and EXNOR are never always complemented, It is complement only when even variable occurs.

⇒ EXNOR gate is even no. of 1's detector when no. of I/P's are even.

⇒ EXNOR gate is odd no. of 1's detector when no. of I/P's are odd.

Problem:- $\overline{A \oplus B} = A \oplus B$

Sol:- put $x = \overline{A}$, $y = B$

\[ x \oplus y = A \bar{x} + \bar{A}x \]
\[ = \bar{A}B + AB = A \oplus B. \]

Problem :- $\bar{A} \oplus \bar{B}$

Sol:- put $x = A$, $y = \bar{B}$

\[ x \oplus y = x \bar{y} + y \bar{x} \]
\[ = AB + \bar{A}B \]
\[ = A \oplus B. \]
Problem: \( A \oplus B \oplus AB \)

Solution:
\[
\begin{align*}
(A\overline{B} + \overline{A}B) \oplus AB & = (A\overline{B} + \overline{A}B) \overline{A}B + (A\overline{B} + \overline{A}B) AB \\
& = A\overline{B}(A + \overline{B}) + AB(\overline{A} + \overline{B}) + (\overline{A}\overline{B} \cdot \overline{A}B) AB \\
& = A\overline{B} \oplus AB + [(\overline{A} + \overline{B})(A + \overline{B})] AB \\
& = A\overline{B} \oplus AB + [\overline{AB} + \overline{A}B] AB \\
& = A\overline{B} \oplus AB + AB \\
& = (A + \overline{A})(A + \overline{B}) = A + B \quad \text{Ans}.
\end{align*}
\]

\[ A \oplus B \oplus AB = A + B \]

Problem: \( A \oplus B \oplus AB \)

Diagram with logic gates:

\[ y = 1 \] for \( x = 0 \)

\[ y = 1 \] for \( x = 1 \)

\[ y = 1 \] for \( x = \overline{A} \overline{B} \)

\[ y = 1 \] for \( x = \overline{A} \overline{B} \)

Solution:
\[ y = 1 \quad \text{Ans} \]

Symbols:
- NAND \( \Rightarrow \)
- NOR \( \Rightarrow \)
- AND \( \Rightarrow \)
- OR \( \Rightarrow \)

Bubbled OR

Bubbled AND
NAND as universal:

1. NOT: \[ A \rightarrow \bar{A} \Rightarrow 1 \text{ gate required} \]

2. AND: \[ A \quad B \rightarrow \overline{A \cdot B} \Rightarrow 2 \text{ gate} \]

3. OR: \[ A \quad B \rightarrow \overline{-A + B} \Rightarrow 3 \text{ gate} \]

4. EXOR: \[ A \quad B \Rightarrow y = A \oplus B = \overline{A \cdot B} + \overline{A} \cdot B \]

\[ y = (A \cdot \overline{A \cdot B} \cdot B \cdot \overline{A \cdot B}) \]
\[ = (A \cdot \overline{A \cdot B} + B \cdot \overline{A \cdot B}) \]
\[ = (A \cdot \overline{A + B} + B \cdot \overline{A + B}) \]
\[ = A \overline{B} + B \overline{A} = A \oplus B \]

5. EXNOR: \[ A \quad B \Rightarrow 5 \text{ gate} \]

6. NOR: \[ A \quad B \Rightarrow 4 \text{ gate} \]
NOR AS Universal:

(i) NOT:

\[
\begin{array}{c}
\text{A} \\
\hline
\text{A}
\end{array}
\Rightarrow 1 \text{ gate}
\]

(ii) AND:

\[
\begin{array}{c}
\text{A} \\
\hline
\text{A} \\
\hline
\text{AB}
\end{array}
\Rightarrow 3 \text{ gate}
\]

(iii) OR:

\[
\begin{array}{c}
\text{A} \\
\hline
\text{A+B}
\end{array}
\Rightarrow 2 \text{ gate}
\]

(iv) EXNOR:

\[
\begin{array}{c}
\text{A} \\
\hline
\text{A+B}
\end{array}
\Rightarrow 4 \text{ gate}
\]

\[
Y = (\overline{A + A+B}) + (\overline{B + A+B}) = \overline{A} (A+B) + \overline{B} (A+B)
\]

\[
= (A \cdot A + B) (B + A + B) = \overline{A} \overline{B} + \overline{B} A = \overline{A} \overline{B}
\]

(v) EXOR:

\[
\begin{array}{c}
\text{A} \\
\hline
\text{EXOR} \\
\hline
\text{EXOR} \Rightarrow 5 \text{ gate}
\end{array}
\]

(vi) NAND:

\[
\begin{array}{c}
\text{A} \\
\hline
\text{AND} \\
\hline
\text{NAND} \Rightarrow 4 \text{ gate}
\end{array}
\]

\[
\begin{array}{c}
\text{B} \\
\hline
\text{3 gate}
\end{array}
\]

\[
\begin{array}{c}
\text{1 gate}
\end{array}
\]
Note:- Logic gate | No. of NAND | No. of NOR
---|---|---
NOT | 1 | 1
AND | 2 | 3
OR | 3 | 2
EXOR | 4 | 5
EXNOR | 5 | L

Problem: To implement $\bar{xyz}$. The min no. of two I/p NAND gate required.

Sol.- $\bar{xyz}$

Total no. of NAND gate = 2 + 2 + 1 = 5. Ans.

Problem: To implement $xy+wz$, the min no. of 2 input NAND gate required.

Sol.- $xy+wz$

⇒ 1st inverter cancelled 2nd and 3rd cancelled 4th
⇒ Now the total no. of NAND & gate is.
$= 2 +$ Bubbled OR (= NAND)
$= 2 + 1 = 3$
$= 3$ NAND gate required.

Note:-

Two level AND-OR = Two level NAND-NAND

AND-OR = NAND-NAND
To implement SOP form, only NAND gate alone.

To implement POS form, only NOR gate alone.

Given: if \((A+B)(C+D)\) then min no. of gate.

Solution:

\begin{align*}
\text{Twolevel OR\text{-}AND} & & \text{Twolevel NOR\text{-}NOR} \\
(A+B)(C+D) &= \left( \overline{\overline{A} + \overline{B} + \overline{C} + \overline{D}} \right) \\
&= \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}
\end{align*}

\[ \text{OR\text{-}AND = NOR\text{-}NOR} \]
Digital circuits

\[ \text{combination ckt} \quad \text{sequential ckt} \]

\[ \Rightarrow \text{Present o/p is only depend on present i/p.} \]
\[ \Rightarrow \text{present o/p } \begin{cases} \text{Present i/p} \\ \text{Previous o/p} \end{cases} \]

\[ \Rightarrow \text{No feedback} \]
\[ \Rightarrow \text{feedback.} \]

\[ \Rightarrow \text{No memory} \]
\[ \Rightarrow \text{Memory.} \]

\[ \Rightarrow \text{E.g. Half Adder (HA)} \]
\[ \Rightarrow \text{E.g.: Flip Flop (FF)} \]

- FA
- MUX
- DEMUX
- Register
- Counter
COMBINATIONAL CIRCUIT

Procedure to Design:
(i) Identify I/p and O/p.
(ii) Construct truth table
(iii) Write logical expression in SOP or POS form.
(iv) Minimize logical expression if possible
(v) Implement logic circuits.

(A) HALF ADDER (HA):

\[ A \quad \text{HA} \quad \text{SUM} \]
\[ B \quad \text{CARRY} \]

Truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>SUM</th>
<th>CARRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Logical expression:

\[ \text{SUM} = \overline{A}B + AB = A \oplus B \]
\[ \text{CARRY} = AB \]

Implement:

\[ \text{A} \quad \text{SUM} \]
\[ \text{B} \quad \text{CARRY} \]

Important Notes:

(i) Logical expression for SUM: \( A \oplus B \)  CARRY: \( AB \)
(ii) Min. no. of NAND Gate: 5
(iii) Min. no. of NOR Gate: 5
(iv) No. of MUX: 3
(v) No. of Decoder: 1, 2X4 decoder and 1 or gate
HA using NAND gate:

\[ A \oplus B = \text{SUM} \]
\[ AB = \text{CARRY} \]

Total no. of gate = 5.

HA using NOR gate:

\[ \overline{A + B} + \overline{A \cdot B} = A \oplus B = \text{SUM} \]
\[ AB = \text{CARRY} \]

3 No. of NOR gate

Total no. of gate = 2 + 3 = 5.

(B) HALF SUBTRACTOR:

\[
\begin{array}{c}
\text{A} \\
\text{B} \\
\text{HS} \\
\end{array}
\]

Truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Diff</th>
<th>Barrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Logical expression:

\[
\text{Difference} = \overline{A} \cdot B + A \cdot \overline{B}
\]

\[
\text{Barrow} = \overline{A} \cdot B
\]

Implement:

\[
\begin{array}{c}
\text{A} \\
\text{B} \\
\end{array}
\]

Difference

\[
\begin{array}{c}
\text{A} \\
\text{B} \\
\end{array}
\]

Barrow:
Note:

- If control = 0, then A \oplus 0 = A
- Then \( Y = A \overline{B} H.A. \)
- If control = 1, then, \( A \oplus 1 = A \)
- Then \( Y = \overline{A} \overline{B} \) and ckt is HS.

Important ques:-

No. of NAND gate = 5
No. of NOR gate = 5
No. of MUX = 3, \( (2 \times 1 \text{ MUX}) \)
No. of DECODER = 1, \( (2 \times 4 \text{ Decoder}) \) and 1 OR Gate.

Logical expression for difference = \( A \overline{B} + A \overline{B} \), Barrow \( \overline{A} \overline{B} \)

HS using NAND gate:-

\[ \begin{align*}
A \quad & - \quad \overline{A} \overline{B} = \text{Difference} \\
B \quad & - \quad \overline{B} \overline{A} = \text{Barrow}
\end{align*} \]

No. of NAND gate = 5

HS using NOR gate:-

\[ \begin{align*}
A \quad & - \quad (A + A \overline{B}) = \overline{A} \text{ Barrow} \\
B \quad & - \quad A \overline{B} \text{ Difference}
\end{align*} \]

Now, \( A + \overline{A} \overline{B} = \overline{A} (\overline{A} \overline{B}) = \overline{A} (\overline{A} + \overline{B}) = \overline{A} \overline{B} \)

classmate \( \neq \) No of NOR gate = 5
FULL ADDER:

```
\[
\begin{array}{ccc}
\text{A} & \text{B} & \text{SUM} \\
\text{C} & \text{CARRY} \\
\hline
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]
```

Truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>SUM</th>
<th>CARRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Logical expression:

\[
\text{SUM} = \overline{\text{A}} \overline{\text{B}} \overline{\text{C}} + \overline{\text{A}} \overline{\text{B}} \overline{\text{C}} + \overline{\text{A}} \overline{\text{B}} \overline{\text{C}} + \overline{\text{A}} \overline{\text{B}} \overline{\text{C}} = \Sigma m(1, 2, 4, 7) \\
\text{CARRY} = \overline{\text{A}} \overline{\text{B}} \overline{\text{C}} + \overline{\text{A}} \overline{\text{B}} \overline{\text{C}} + \overline{\text{A}} \overline{\text{B}} \overline{\text{C}} + \overline{\text{A}} \overline{\text{B}} \overline{\text{C}} = \Sigma m(3, 5, 6, 7)
\]

The truth table of carry shows the majority of 1's function.

In full adder, each logic gate have propagation delay of \( t_{PD} \), to provide sum or carry output, it requires to \( 2t_{PD} \) delay.
Now, \( \text{CARRY} = \overline{ABC} + \overline{A}BC + \overline{A}B\overline{C} + ABC \)
\[ = AB(\overline{C+C}) + C(\overline{A}B+\overline{A}B) \]
\[ = AB + C(\overline{A}B) \] \( \text{46} \)

\[ \text{CARRY} = AB + C(A\oplus B) \]

Implementation:

\[ \text{A} \]
\[ \text{B} \]
\[ \text{C} \]

\[ \text{CARRY} = AB + C(A\oplus B) \]

Important questions:

1. Logical expression for \( \text{SUM} = A\oplus B\oplus C \) \( \text{CARRY} = AB + BC + AC \)
2. No. of HA and OR gate = 2HA, 1-OR
3. Min. no. of NAND = 9
4. Min. no. of NOR = 9
5. No. of MULT.
6. No. of DECODER = 1, (3\times8) Decoder and 2-OR gate
Implementation of Full adder using NAND gate:

\[ A \oplus B \oplus C = A \overline{B} \overline{C} \overline{C} \]

\[ = A \overline{B} + C(A \overline{B}) \]

Implementation of Full adder using NOR gate:

Since \( A \oplus B \oplus C = A \overline{B} \overline{C} \overline{C} \), the circuit is same only NAND is replaced by NOR.

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

and,

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

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

\[ = \overline{A \overline{B} + A \overline{B} + A \overline{A} B + B A B + B A B} \]

\[ = A C + A B + B C + A B \]

\[ = C \cdot (A + B) + A B \]

\[ \text{CARRY} = A B + B C + C A \]
PARALLEL ADDER:

There are three types of adders:
1. Serial adder (we write with sequential adders).
2. Parallel adder.
3. Look ahead carry adder.

⇒ In serial adder, only one full adder (FA) is used to add groups of bits.
⇒ It is the slowest adder.

Parallel adder:

⇒ For 4-bit adder:
   - 3 FA and 1 HA required.
   - Or, 4 FA is required.
   1 1 0 0 0 0

⇒ Parallel adder is used to add groups of bits.
⇒ To add two N-bit no. it requires (N-1) full adder and 1 half adder, or,
   N full adders or,
   (2N-1) half adders and (N-1) OR gates required.

Now,

Diagram of Parallel adder:

\[
\begin{array}{c}
A_3 A_2 A_1 A_0 \\
1 1 0 1 \\
1 0 1 1 \\
B_3 B_2 B_1 B_0
\end{array}
\]

\[
\begin{array}{c}
C_4 \\
S_3 S_2 S_1 S_0
\end{array}
\]

\[
CARRY \quad SUM
\]

\[
\begin{array}{c}
FA \\
FA \\
FA \\
FA
\end{array}
\]

© Wiki Engineering

[Image with diagrams and equations]
Parallel adder is also called Ripple carry adder.

Propagation delay from I/P array to O/P array. Hence it is also known as Ripple carry adder.

In parallel adder each FA will provide 2 logic gate delay. In n bit parallel adder provide total delay of:

\[ \text{Delay} = 2n \text{ gate delay} \]

LOOK AHEAD CARRY CIRCUIT:

Disadvantage of parallel adder is carry propagation delay present. As no. of bit increases speed of operation reduced.

To avoid this look ahead carry adder is used.

\[
\begin{align*}
P_i &= \text{Propagation} \\
G_i &= \text{Generation term} \\
P_i &= A_i \oplus B_i \\
G_{ii} &= A_i \odot B_i = A_i'B_i \\
S_i &= P_i \oplus C_i \\
C_{in} &= P_iC_i + G_{ii}
\end{align*}
\]

For four (4) bit look ahead carry adder:

I/P: \( A_3 \ A_2 \ A_1 \ A_0 \) \( B_3 \ B_2 \ B_1 \ B_0 \)

Then \( P_i = A_i \oplus B_i \)
\( P_2 = A_2 \oplus B_2 \)
\( P_3 = A_3 \oplus B_3 \)
\( C_{in} = P_3C_3 + G_{ii} \)
\[ \begin{align*}
G_0 &= A_0 B_0 \\
G_1 &= A_1 B_1 \\
G_2 &= A_2 B_2 \\
G_3 &= A_3 B_3
\end{align*} \]

\[ S_0 = P_0 \oplus G_0, \quad S_1 = P_1 \oplus G_1, \quad S_2 = P_2 \oplus G_2, \quad S_3 = P_3 \oplus G_3 \]

\[ C_{i+1} = P_i C_i + G_i \]  
look ahead carry generator expression

\[ \begin{align*}
C_0 &= P_0 C_0 + G_0 \\
C_1 &= P_1 C_0 + G_1 = P_1 (P_0 C_0 + G_0) + G_1 = P_1 P_0 C_0 + P_1 G_0 + G_1 \\
C_2 &= P_2 C_1 + G_2 = P_2 P_1 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2 \\
C_3 &= P_3 C_2 + G_3 = P_3 P_2 C_1 + P_3 P_2 G_1 + P_3 G_2 + G_3
\end{align*} \]

\[ \Rightarrow \text{Total no. of AND Gate inside} = 1 + 2 + 3 + 4 = n(n+1) = \frac{4 \times 5}{2} = 10 \]

\[ \text{no. of AND Gate} = \frac{n(n+1)}{2} \]

\[ \text{no. of OR Gate} = n \]

\[ \Rightarrow \text{Total propagation delay} = 2t_{pd} \]

\[ \Rightarrow \text{This is faster than parallel adder.} \]
FULL SUBTRACTOR :

\[ \begin{array}{ccc}
A & \bar{A} & \bar{A} \\
\bar{C} & \bar{C} & \bar{C} \\
C & \bar{C} & \bar{C} \\
& \text{Diff} & \text{BARROW} \\
\end{array} \]

Truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Diff (A-B-C)</th>
<th>BARROW</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Logic expression:

\[ \text{Diff} = \sum m (1, 2, 4, 7) \]
\[ = A \oplus \bar{B} \oplus C \]
\[ \text{BARROW} = \bar{A} \bar{B} C + \bar{A} \bar{B} \bar{C} + \bar{A} B C + A B C \]
\[ = B C + \bar{A} \left( B C + B \bar{C} \right) \]
\[ = \bar{B} C + \bar{A} \left( B \oplus C \right) \]

add \( \bar{A} B C \) two more:

\[ = A B \left( \bar{C} + C \right) + \left( \bar{A} + A \right) B C + \bar{C} \bar{A} \left( B + \bar{B} \right) \]
\[ = \bar{A} B + \bar{A} C + B C \]
\[ = \sum m (1, 2, 3, 7) \]

Implementation:

\[ \begin{array}{ccc}
A & B & C \\
\text{Barrow} & \text{Difference} \\
\end{array} \]
Barrow expression:

\[
\overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC}
= \overline{AB}(C+\overline{C}) + C(\overline{AB} + A\overline{B})
= \overline{AB} + C(A\overline{B})
\]

\[A\oplus B = \text{SUM}\]
\[\overline{AB} + C(A\overline{B}) = \text{BARROW}\.
\]

\[\Rightarrow \text{Full subtractor will be implemented with 2-HS and 1-OR gate.}\]

**Important aues:**

- no. of NAND gate = 9
- no. of NOR gate = 9
- logical expression for Difference = \(A\oplus B\oplus C\)
- logical expression for Barrow = \(\overline{AB} + \overline{AC} + BC\) or, \(\overline{AB} + C(A\overline{B})\)
- no. of Mux =
- no. of Decoder = 1 (3x8) Decoder and 2 OR gate
**Comparator:**

\[ A > B \rightarrow x \]

<table>
<thead>
<tr>
<th></th>
<th>( A )</th>
<th>( B )</th>
<th>Comparator</th>
<th>( A = B \rightarrow ) ( y )</th>
<th>( A &lt; B \rightarrow ) ( z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Truth Table:**

\[
\begin{array}{ccc}
\text{X} & \text{Y} & \text{Z} \\
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

**Expression:**

\[ x = \overline{AB} \\
y = \overline{A} \oplus B = \overline{AB} + AB \\
z = \overline{AB} \]

**Implementation:**

\[ \Rightarrow \]

**Notes:** For equality condition, \( A \oplus B \), condition holds.

If \( A_3, A_2, A_1, A_0 \) are equal to \( B_3, B_2, B_1, B_0 \) Then the equality condition is.

\[
(A_3 \oplus B_3) (A_2 \oplus B_2) (A_1 \oplus B_1) (A_0 \oplus B_0)
\]
Then the ckt is:

K - MAP:

⇒ It is used when o/p is 0, 1, and x (don't care)
⇒ In K-MAP gray code representation is used
⇒ K-map is graphical representation

⇒ each successive term is changed by only one bit.

<table>
<thead>
<tr>
<th>Two variable: A</th>
<th>msb</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

(For Two variable)

For Three variable:

<table>
<thead>
<tr>
<th>msb A</th>
<th>msb Bc</th>
<th>msb 00</th>
<th>msb 01</th>
<th>msb 11</th>
<th>msb 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>11</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
four variable:

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>01</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>19</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>9</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

\[ \Rightarrow \text{Minimize: } \]

\[ f(A, B) = \Sigma m(0, 2, 3) \]

\[ \text{Sol: } \]

\[ f(A, B) = A + \overline{B} \quad ( + \text{ is put due to SOP form}) \]

\[ \text{G: } \]

\[ f(A, B) = \Sigma m(1, 2, 3) \]

\[ \text{Sol: } \]

\[ f(A, B) = A + B \]

\[ \text{E: } \]

\[ f(A, B) = \Sigma m(0, 1, 2, 3) \]

\[ \text{Sol: } \]

\[ f(A, B) = 1 \]

\[ \Rightarrow \text{In k-map if all are one means the function is 1.} \]

\[ \text{G: } \]

\[ f(A, B) \Sigma m(1, 3) + \Sigma d(2) \]

\[ \text{Sol: } \]

\[ f(A, B) = \overline{B} \quad (\text{no need of any gate}) \]
\[ f(A, B) = \Sigma m(0, 3) + \Sigma d(2) \]

So:
\[
\begin{array}{cccc}
A & B & 0 & 1 \\
0 & & & \\
1 & & & \\
\end{array}
\]
\[ f(A, B) = A + B \]

\[ f(A, B) = \Sigma (0, 3) + \Sigma d(1, 8, 11) \]

So:
\[
\begin{array}{ccc}
A & B & 0 & 1 \\
1 & & & \\
1 & & & \\
\end{array}
\]
\[ f(A, B) = 1 \]

3. In SOP form if all are 1's means o/p is 1.
3. All are don't care means don't care.

Three variable:

\[ f(A, B, C) = \Sigma m(1, 3, 5, 7) \]

So:
\[
\begin{array}{cccc}
A & B & C & 00 & 01 & 11 & 10 \\
0 & & & & & & \\
1 & & & & & & \\
\end{array}
\]
\[ f(A, B, C) = C \]

\[ f(A, B, C) = \Sigma m(0, 1, 3, 6) \]

So:
\[
\begin{array}{cccc}
A & B & C & \bar{B}C & \bar{B}C & BC & B \bar{C} \\
\end{array}
\]
\[ f(A, B, C) = \bar{A}B + \bar{A}C + ABC \]

\[ f(A, B, C) = \Sigma m(1, 3, 6, 7) \]

If we take BC then it is redundant term and it must be removed.

\[ f(A, B, C) = \bar{A}C + AB \]

\[ f(A, B, C) = AC + AB \]
Procedure:
1. Actets
2. Goods
3. Pairs
4. Single term
5. Remove redundant

\[ f(A, B, C) = \Sigma m(0, 1, 2, 4, 7) \]

| \( \bar{A} \) | \( B \bar{C} \) | \( 
\bar{B} \bar{C} \) | \( B \bar{C} \) | \( B \bar{C} \) |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bar{A} )</td>
<td>( \bar{B} \bar{C} )</td>
<td>( \bar{B} \bar{C} )</td>
<td>( \bar{B} \bar{C} )</td>
<td>( \bar{B} \bar{C} )</td>
</tr>
<tr>
<td>( A )</td>
<td>( B \bar{C} )</td>
<td>( B \bar{C} )</td>
<td>( B \bar{C} )</td>
<td>( B \bar{C} )</td>
</tr>
</tbody>
</table>

\[ = \bar{A} \bar{B} + \bar{A} \bar{C} + \bar{B} \bar{C} + ABC \]

\[ f(A, B, C) = \Sigma m(0, 1, 5, 6, 7) \]

<table>
<thead>
<tr>
<th>( \bar{A} )</th>
<th>( \bar{B} \bar{C} )</th>
<th>( \bar{B} \bar{C} )</th>
<th>( B \bar{C} )</th>
<th>( B \bar{C} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bar{A} )</td>
<td>( \bar{B} \bar{C} )</td>
<td>( \bar{B} \bar{C} )</td>
<td>( \bar{B} \bar{C} )</td>
<td>( \bar{B} \bar{C} )</td>
</tr>
<tr>
<td>( A )</td>
<td>( B \bar{C} )</td>
<td>( B \bar{C} )</td>
<td>( B \bar{C} )</td>
<td>( B \bar{C} )</td>
</tr>
</tbody>
</table>

\[ f(A, B, C) = \bar{A} \bar{B} + \bar{B} \bar{C} + AB \] two sol
\[ = \bar{A} \bar{B} + \bar{A} B + AC \]

\[ = k-map provide minimize expression but not necessarily unique i.e. two sol' also. \]

\[ f(A, B, C) = \Sigma m(0, 1, 2, 5, 7) + \Sigma d(3, 6) \]

<table>
<thead>
<tr>
<th>( \bar{A} )</th>
<th>( B \bar{C} )</th>
<th>( \bar{B} \bar{C} )</th>
<th>( B \bar{C} )</th>
<th>( B \bar{C} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bar{A} )</td>
<td>( \bar{B} \bar{C} )</td>
<td>( \bar{B} \bar{C} )</td>
<td>( \bar{B} \bar{C} )</td>
<td>( \bar{B} \bar{C} )</td>
</tr>
<tr>
<td>( A )</td>
<td>( B \bar{C} )</td>
<td>( B \bar{C} )</td>
<td>( B \bar{C} )</td>
<td>( B \bar{C} )</td>
</tr>
</tbody>
</table>

\[ f(A, B, C) = \bar{A} + C \]

\[ f(A, B, C) = \Sigma m(0, 1, 6, 7) + \Sigma d(3, 5) \]
\[ f(A, B, C) = A^c B^c + AB \]

\[ f(A, B, C) = \sum m(0, 1, 6, 7) + \sum d(3, 4, 5) \]

\[ f(A, B, C, D) = \sum m(0, 1, 3, 5, 7, 8, 9, 11, 13, 15) \]

\[ f(A, B, C, D) = D + BC \]

\[ f(A, B, C, D) = \sum m(0, 1, 4, 5, 8, 9, 13, 15) \]

\[ f(A, B, C, D) = AC + BC + ABD \]

\[ f(A, B, C, D) = \sum m(0, 2, 8, 10, 14) + \sum d(5, 15) \]
\( f(A, B, C, D) = \overline{B} \overline{D} + A \overline{C} \overline{D} \)

**POS (Product of Sums)**:

Simplify:

\( A: \quad f(A, B) = \Pi M(0, 2, 3) \)

\[
\begin{array}{ccc}
A & B & \overline{D}, \overline{E}, \overline{D} \\
0, 0 & 0 & 1 \\
1, 0 & 1 & 0 \\
\end{array}
\]

\( f(A, B) = B \cdot \overline{A} \)

\( B: \quad f(A, B) = \Pi M(0, 3) + \Pi d(1) \)

\[
\begin{array}{ccc}
A & B & \overline{D} \\
0 & 0 & 1 \\
\overline{A} & 0 & 0 \\
\end{array}
\]

\( f(A, B) = A + \overline{B} = A \overline{B} \)

\( B: \quad f(A, B, C) = \Pi M(0, 1, 3, 5, 7) \)

\[
\begin{array}{ccc}
A & B+C & B+C & B+C & B+C \\
0 & 1 & 1 & 1 & 1 \\
\overline{A} & 1 & 1 & 1 & 1 \\
\end{array}
\]

\( f(A, B, C) = \overline{C} + (A + B) \)

\( = \overline{C} (A + B) \)
for the \textit{k-map} minimize \textit{pos} expression is:

\[
\begin{array}{c|cc|cc|cc|cc}
A & \overline{B} & C & \overline{B+\overline{C}} & B+\overline{C} & \overline{B+\overline{C}} & B+C \\
\hline
A & 0 & x & x & 1 \cr
\overline{A} & 0 & 1 & 1 & 0 & x
\end{array}
\]

\[ (A, B, C) = (B+C)(\overline{B+C}) \]

\[ \Rightarrow \text{The two function are same if the position of 1's are 0's are same in k-map, and if the 1's place 0 are placed and at 0's place 1's are placed then the function is complemen to each other.} \]

\[ \Rightarrow \text{Problem - 26 - Page - 13} \]

\[ Q:\quad W = R + \overline{P}A + \overline{R}S \quad x = P\overline{A}R\overline{S} + P\overline{A}\overline{R}\overline{S} + P\overline{A}RS \]

\[ Y = RS + PR + \overline{P}A + \overline{P}A \quad Z = R + S + \overline{P}A + \overline{P}AR + \overline{P}AS \]

\[ \Rightarrow \text{Then} \quad W = Z = \overline{x} \]
If Truth table is:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>C</td>
</tr>
</tbody>
</table>

(The method for writing the expression from Truth table)

Then,

\[ Y = \overline{A}B \cdot 0 + \overline{A}B \cdot 1 + A\overline{B} \cdot 0 + AB \cdot C \]

\[ = \overline{A}B + ABC \]

\[ = B(\overline{A} + AC) \]

\[ = B(\overline{A} + C) \]
MULTIPLEXER

Many I/p and one O/p.

\[ m = 2^n \]
\[ n = \log_2 m \]

where \( m \) = no. of data I/p.
\( n \) = no. of select I/p. (control I/p)

2:1 MUX :-

\( \text{symbol of MUX} \)
Truth table:

<table>
<thead>
<tr>
<th>I₀</th>
<th>I₁</th>
<th>I₂</th>
<th>I₃</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Logical expression:

\[ y = S_1S_0I₀ + S_0S_1I₁ + S_1S₀I₂ + S_1S₀I₃ \]
I. Implementation of higher order MUX with lower order:

(a) Implement 4:1 MUX using 2:1 MUX:

In 4:1, no. of 2:1 MUX = 3

S_1S_0 \rightarrow Y
10 \rightarrow I_2

(b) Implement 8:1 MUX using 2:1 MUX:

no. of MUX = \frac{8}{2} + \frac{4}{2} + \frac{2}{2} = 4 + 2 + 1 = 7

In 8:1, no. of 2:1 MUX = 7
(C) 16x1  \[ \frac{8+4+2+1}{16} \xrightarrow{15 \text{ mux}} 2\times1 \text{ MUX} \]

(D) 64x1  \[ \frac{63 \text{ mux}}{32+16+8+4+2+1} \xrightarrow{2\times1 \text{ MUX}} 2\times1 \text{ MUX} \]

(E) 256x1  \[ \frac{255 \text{ mux}}{128+64+32+16+8+4+2+1} \xrightarrow{1 \text{ MUX}} 2\times1 \text{ MUX} \]

Therefore, for \( 2^n \times 1 \text{ MUX} \), the no. of \( 2^m \times 1 \text{ MUX} \) is \( 2^n - 1 \).

(F) 16x1 MUX from 4x1 MUX:

\[ \text{no. of MUX} = \frac{16}{4} + \frac{4}{4} = 4 + 1 = 5 \]

(G) 64x1 MUX  \[ \frac{63 \text{ mux}}{64 + 16 + 8 + 4 + 2 + 1} \xrightarrow{2\times1 \text{ MUX}} 4\times1 \text{ MUX} \]

(H) 64x1 MUX  \[ \frac{63 \text{ mux}}{64 + 8 + 4 + 2 + 1} \xrightarrow{8\times1 \text{ MUX}} 8\times1 \text{ MUX} \]
II MUX as universal :-

\[ Y = S_0 I_0 + S_1 I_1 \]
\[ A \times 1 + 0 \times A = A \]

\[ \begin{array}{c|c|c}
A & Y \\
0 & 1 & 0 \\
\end{array} \]

\[ \Rightarrow 1:1 \text{ MUX is required for NOT gate.} \]

AND :-

\[ Y = A \times 1 + AB \]
\[ AB \]

\[ \Rightarrow 1:1 \text{ MUX (2:1) is required for AND gate.} \]

OR :-

\[ Y = AB + A \times 1 \]
\[ A + B \]

\[ \Rightarrow 1:1 \text{ MUX (2:1) is required for OR gate.} \]

NAND :-

\[ Y = \overline{A} \times 1 + \overline{B} A \]
\[ \overline{A} + B \]
\[ \overline{A} B \]

\[ \Rightarrow \text{NAND gate.} \]

\[ \overline{B} \]

(classmate)
2 - MUX required for NAND gate.

\[ Y = \overline{A\overline{B}} + A \cdot \overline{0} = A + \overline{B} \]

For \( B \equiv 1 \):

\[ Y = \overline{B}A + \overline{B}A \]

\[ \Rightarrow 2\text{ MUX (2:1)} \text{ required for NOR gate.} \]

2 - MUX (2:1) required for EXOR gate.

\[ Y = \overline{A\overline{B}} + AB \]

\[ \Rightarrow 2\text{ MUX (2:1)} \text{ required for EXNOR gate.} \]

Gues: EXOR, AND gate required 2X1 MUX.

(a) 1, 1
(b) 2, 1
(c) 1, 2
(d) 2, 2

Sol: EXOR = 2, AND = 1
for HA - 3 Mux required (2:1)
for HS - 3 Mux required (2:1)

\[ A \quad H/S \quad A \oplus B \quad \text{sum Diff} \]
\[ B \quad \bar{A}B \quad \text{borrow} \]

\[ A \oplus B = 2 \text{ Mux required and} \]
\[ \bar{A}B = 1 \text{ Mux} \]
\[ 0 \]
\[ y = \bar{A}B \]
\[ A \]

\[ \star \]
4:1 MUX :-

AND :-
\[ 0 \]
\[ 0 \]
\[ 0 \]
\[ 0 \]
\[ A \]
\[ B \]
\[ s, s_0 \]
\[ 1 \]
\[ 1 \]
\[ 1 \]
\[ 1 \]
\[ 4:1 \]
\[ \rightarrow y \]

OR :-
\[ 0 \]
\[ 0 \]
\[ 0 \]
\[ 0 \]
\[ A \]
\[ B \]
\[ s_1, s_o \]
\[ 1 \]
\[ 1 \]
\[ 1 \]
\[ 1 \]
\[ 4:1 \]
\[ \rightarrow y \]

EXOR :-
\[ 0 \]
\[ 0 \]
\[ 0 \]
\[ 0 \]
\[ A \]
\[ B \]
\[ s_1, s_o \]
\[ 1 \]
\[ 1 \]
\[ 1 \]
\[ 1 \]
\[ 4:1 \]
\[ \rightarrow y \]

Any two variable function is implemented with 4:1 MUX.

EXNOR :-
\[ 1 \]
\[ 0 \]
\[ 0 \]
\[ 0 \]
\[ A \]
\[ B \]
\[ 1 \]
\[ 1 \]
\[ 1 \]
\[ 1 \]
\[ 4:1 \]
\[ \rightarrow y \]
III. Determine minimize a/p logical expression:-

Simplify:-

\[
\begin{align*}
\overline{C} & \quad \overline{I_0} \\
I_1 & \quad \overline{I_3} \\
I_2 & \quad \overline{I_5} \\
S_0 & = \overline{A} \quad S_1 = B
\end{align*}
\]

\[Y = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}\overline{G} + A\overline{B}\overline{C} + ABC\]

\[= \overline{A}\overline{C}(\overline{B} + \overline{G}) + AC(\overline{B} + G)\]

\[= \overline{A}\overline{C} + AC\]

\[= A\overline{C}\]

IV. Implementation of given logical expression:-

\[\Sigma f(A, B, C) = m(0, 1, 4, 6, 7)\]

\[
\begin{align*}
\text{Sol}:- \\
\overline{C} & \quad \overline{I_0} \\
I_1 & \quad \overline{I_3} \\
I_2 & \quad \overline{I_5} \\
A & \quad \overline{B}
\end{align*}
\]
Q:- Implement logical expression.

\[ \bar{f}(A, B, C) = (1, 2, 3, 5, 6, 7) \]

(i) \( AB \) as select line

(ii) \( AC \)

(iii) \( BC \)

\[
\begin{array}{c|cccc}
I_0 & I_1 & I_2 & I_3 \\
\hline
C & 0 & 0 & 4 & 6 \\
C & 1 & 1 & 5 & 7 \\
\end{array}
\]

\[ \Rightarrow \text{1-4:1 MUX required.} \]

\[
\begin{array}{c|cccc}
I_0 & I_1 & I_2 & I_3 \\
\hline
B & 0 & 0 & 4 & 5 \\
B & 1 & 0 & 6 & 7 \\
\end{array}
\]

\[ \Rightarrow \text{1-4:1 MUX required.} \]
Using one 4:1 MUX:

- Any two variable function implement
- Some of three variable implement

Using one 4x1 MUX and one NOT:

- All Two implement
- All Three

Using one 8x1 MUX:

- All Three
- Some four

Using one 8x1 MUX and one NOT:

- All Three are four implement
- All Four are implemented

Guess:
\[ f = \Pi M(0, 1, 4, 7) \]

Solution:
- First convert it into minterm expression:
  \[ f = \Sigma m(2, 3, 5, 6) \]
Hazard:
- Hazard occurs due to propagation delay of the logic gate.
- This is unwanted change at the o/p.

For the given circuit determine o/p waveform when no propagation delay:

\[ \text{A} \rightarrow \text{X} \rightarrow \text{Y} \]

\[ \begin{align*}
\text{A} & : 0 \quad 1 \\
\text{X} & : 1 \quad 0 \\
\text{Y} & : 0 \quad 0 \quad \text{static '0'}
\end{align*} \]

Case II:
- If there is propagation delay of 1ns in NOT gate and no delay in AND gate.

\[ \begin{align*}
\text{A} & : 0 \quad 1 \\
\text{X} & : 1 \quad 0 \\
\text{Y} & : 0 \quad 0 \quad \text{Static '0' Hazard}
\end{align*} \]

Case III:
- If \( t_{pd} \) (NOT) = 1ns, \( t_{pd} \) (AND) = 2ns.

\[ \begin{align*}
\text{A} & : 0 \quad 1 \\
\text{X} & : 1 \quad 0 \\
\text{Y} & : 0 \quad 0
\end{align*} \]
Hazard

\[ \downarrow \]

Static

\[ \downarrow \]

Dynamic

\[ \bigcirc \]

Essential

\[ \downarrow \]

To avoid static and dynamic Hazard redundant terms are added in combinational ckt.

\[ \Rightarrow \]

Essential Hazard: These Hazards can not be avoided but feels essential.
Memories

<table>
<thead>
<tr>
<th>ROM</th>
<th>Semi random</th>
<th>Serial access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read / write</td>
<td>Read only</td>
<td>AI disk</td>
</tr>
<tr>
<td>Random access</td>
<td>Random access</td>
<td>CD</td>
</tr>
<tr>
<td>Volatile</td>
<td>Non volatile</td>
<td>DVD</td>
</tr>
<tr>
<td>Temporary data</td>
<td>Permanent data</td>
<td>HD</td>
</tr>
</tbody>
</table>

Ferrite core → DRO → Discriminative read only out.

**RAM (Random access memory):**

1. Each memory location if m bits are stored then memory capacity = \(2^n \times m\)
2. with \(n\)-bit address = max no. of memory location required is = \(2^n\).

- \(4K \times 8\) memory.
  \[
  2^2 \times 2^{10} \times 8 = 2^{12} \times 8
  \]
  = 12-address line
  8-data lines.

**RAM**

1. Stored like FF
1. Data stored in MOS capacitors
Memories

<table>
<thead>
<tr>
<th>RAM</th>
<th>ROM</th>
<th>Serial access memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read / write</td>
<td>Read only</td>
<td>All disk</td>
</tr>
<tr>
<td>Random access</td>
<td>Random access</td>
<td>CD</td>
</tr>
<tr>
<td>Volatile</td>
<td>Non volatile</td>
<td>DVD</td>
</tr>
<tr>
<td>Temporary</td>
<td>Permanent</td>
<td>HD</td>
</tr>
<tr>
<td>data</td>
<td>data</td>
<td>(charged couple)</td>
</tr>
<tr>
<td>BIOS / System program</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ferrite core → DRO → Discrete read only out.

RAM (Random access memory):
- Each memory location if m bits are stored then memory capacity is \( 2^n \times m \).
- With n-bit address, max. no. of memory location required is \( 2^n \).

- \( 4^{12} \times 8 \) memory,
  \[ = 2^2 \times 2^{10} \times 8 = 2^{12} \times 8 \]
- 12-address line
- 8-data lines.

Static
- Stored like FF

Dynamic
- Data stored in MOS capacit.
DEMUX (DEMULTIPLEXER):-

- Single I/p and Many O/p.

DEMUX is combinatorial - cut which have one I/p and many O/p depending on select I/p. I/p is transferred to any of the O/p.

Also known as 1-to-many cut or data distributor.

Truth table: -

<table>
<thead>
<tr>
<th>S</th>
<th>Y₀</th>
<th>Y₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Expression: -

Y₀ = S₀S₁I
Y₁ = S₀I

Implementation:

4:1 & 1:4 DEMUX: -

- Y₀ = S₀S₁I
- Y₁ = S₀S₀I
- Y₂ = S₁S₀I
- Y₃ = S₀S₀I
Implementation of higher order DEMUX from lower order:

(i) $1 \times 4$ DEMUX $\leftrightarrow 3 \rightarrow 1 \times 2$ DEMUX

(ii) $1 \times 8$ DEMUX $\leftrightarrow 7 \rightarrow 1 \times 2$ DEMUX

(iii) $1 \times 16$ DEMUX $\leftrightarrow 5 \rightarrow 1 \times 4$ DEMUX

(iv) $1 \times 64$ DEMUX $\leftrightarrow 21 \rightarrow 1 \times 4$ DEMUX $\frac{64}{4} + 16 + \frac{4}{4} + 1$

(v) $1 \times 64$ DEMUX $\leftrightarrow 9 \rightarrow 1 \times 8$ DEMUX

(vi) $1 \times 256$ DEMUX $\leftrightarrow 7 \rightarrow 1 \times 16$ DEMUX
**DECODER**:

1. Decoder is a combinational ckt which have many i/p and many o/p.
2. It is used to convert binary data to other code (binary to
   - Binary to octal (3x8)
   - BCD to Decimal (4x10)
   - Binary to hexadecimal
   - BCD to seven segment
3. 2 to 4 decoder is minimum possible decoder.

**2x4 Decoder**:

![Diagram of 2x4 Decoder]

**Truth table**:

<table>
<thead>
<tr>
<th>E</th>
<th>A</th>
<th>B</th>
<th>Y3</th>
<th>Y2</th>
<th>Y1</th>
<th>Y0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Logical Expression**:

- \( Y_0 = \overline{A} \overline{B} E \)
- \( Y_1 = \overline{A} \overline{B} E \)
- \( Y_2 = AB \overline{E} \)
- \( Y_3 = ABE \)

- Decoder and DEMUX internal ckt remains same.
A: Implement Half adder using 2x4 decoder.

- **Sum**: \( \bar{A} \bar{B} + AB \)
- **Carry**: \( \bar{A} \bar{B} \)

2. We implement HA using 1-2x4 decoder and 1 or gate and same for MS.

**Binary to octal decoder:**

- Also called as 8x8 decoder.

**Solution:**

- **Sum**: \( \sum m(1, 2, 4, 7) = \bar{A} \bar{B} \bar{C} + \bar{A} \bar{B} \bar{C} + A \bar{B} \bar{C} + A \bar{B} \bar{C} \)
- **Carry**: \( \sum m(3, 5, 6, 7) \)
6: Implementation of Higher order decoder using lower order:

\[ 4 \times 16 \rightarrow 5 \rightarrow 2 \times 4 \rightarrow 1 \times 4 \rightarrow 1 \times 4 \times 4 \] (Since 2 \times 4 Decoder means 1 \times 4 MUX using 2 select line)

\[ 16 \times 1 \rightarrow 5 \rightarrow 4 \times 1 \]
Encoder :-

Encoder is the combinational ckt which have many I/p and many O/p.

Encoder is used to convert other code to Binary.

- Octal to Binary
- Decimal to BCD
- Hexadecimal to Binary

Octal to Binary Encoder :-

<table>
<thead>
<tr>
<th>I_0</th>
<th>I_1</th>
<th>I_2 8x3</th>
<th>I_3 OCTAL to</th>
<th>I_4 BINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>I_5</td>
<td>Y_0</td>
<td>Y_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- In normal encoder one of the I/p line is high and corresponding binary available at the O/p.
- In priority encoder no. of I/p is high, only highest priority no. corresponding binary is available at the O/p.

Truth table :-

<table>
<thead>
<tr>
<th>I_0</th>
<th>I_1</th>
<th>I_2</th>
<th>I_3</th>
<th>I_4</th>
<th>I_5</th>
<th>I_6</th>
<th>Y_2</th>
<th>Y_1</th>
<th>Y_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Classmate
Logical expression:

\[ Y_0 = I_1 + I_3 + I_5 + I_7 \]
\[ Y_1 = I_2 + I_3 + I_6 + I_7 \]
\[ Y_2 = I_4 + I_5 + I_6 + I_7 \]

Decoder contains AND Gate.

DEMUX contains AND Gate.

ENCODER contains OR Gate.
It is basic memory element.

It can store 1 bit.

FF have two o/p which have complemented to each other.

It have two stable state hence it is known as bistable multivibrator.

**Contents:**

0. SR latch  \[\rightarrow\] NAND  \[\rightarrow\] NOR

1. SR FF  \[\rightarrow\] GKTs
2. JK FF  \[\rightarrow\] Truth table
3. D FF  \[\rightarrow\] Characteristic table
4. T FF  \[\rightarrow\] Characteristic equation
   \[\rightarrow\] Excitation table
   \[\rightarrow\] Conversion from one to another
   \[\rightarrow\] Simple Ckt.

Using NOT gate the problem is, it have only one I/p then we use NAND or NOR gate instead of NOT gate.

FF is not only used for storing 1 bit but it also used for frequency divider.
SR latch using NAND:

\[ S \quad R \quad \overset{\text{NAND}}{\rightarrow} \quad \overset{\text{enable} = 1}{\rightarrow} \quad A \quad B \quad Y \]

\[
\begin{array}{c|c|c|c|c|}
S & R & A & B & Y \\
0 & 0 & & & 1 \\
0 & 1 & & & 1 \\
1 & 0 & & & 1 \\
1 & 1 & & & 0 \\
\end{array}
\]

Truth table:

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>A</th>
<th>Previous state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>Invalid ( \bar{A} = \bar{A} = 1 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Previous state (no change)</td>
</tr>
</tbody>
</table>

⇒ In SR latch if both gates are enabled o/p remains same previous state and both are disable then o/p remains same invalid state.

SR latch using NOR gate:

\[ S \quad \overset{\text{NAND}}{\rightarrow} \quad \overset{\text{disable is 1 and}}{\rightarrow} \quad \overset{\text{NAND enable is 1 and}}{\rightarrow} \quad \overset{\text{disable is 1 and}}{\rightarrow} \quad \overset{\text{NOR - E = 0}}{\rightarrow} \quad \overset{\text{Then we change x and}}{\rightarrow} \quad \overset{\text{position}}{\rightarrow} \quad \bar{A} \]

Truth table:

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>\bar{A}</th>
<th>Previous state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Invalid ( \bar{A} = \bar{A} = 0 )</td>
</tr>
</tbody>
</table>

⇒ SR latch is used to eliminate switch bouncing.

⇒ Bouncing means vibration of switches when on or off.
SR Flip Flop:

- **S**: Set
- **R**: Reset

Truth table:

<table>
<thead>
<tr>
<th>Clock</th>
<th>S</th>
<th>R</th>
<th>Q_{n+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Q_n</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Hold state</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 → Reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 → Set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Invalid → Unused</td>
</tr>
</tbody>
</table>

For NAND gate SR FF:

Truth table is same as for NAND gate SR FF.
Characteristic table:

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Qn</th>
<th>Qn+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>x</td>
</tr>
</tbody>
</table>

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

\[ Q_{n+1} = S + \bar{R} Q_n \] and \( SR = 0 \) — (i)

⇒ Since \( S = 1, R = 1 \), the o/p is invalid because \( S \cdot R = 1 \) not satisfy the above condition.

Excitation table:

<table>
<thead>
<tr>
<th>Qn</th>
<th>Qn+1</th>
<th>S</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>x</td>
<td>0</td>
</tr>
</tbody>
</table>

⇒ Disadvantage of \( SR \) FF is invalid state present when \( S = 1 \) and \( R = 1 \).

⇒ To avoid this JK FF is used.
**JK Flip Flop**

\[
\begin{align*}
S &= J\bar{Q} \\
R &= K\bar{Q}
\end{align*}
\]

**Truth Table**

<table>
<thead>
<tr>
<th>Clock</th>
<th>J</th>
<th>K</th>
<th>( Q_{n+1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>( Q_n )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( Q_n )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( \bar{Q}_n )</td>
</tr>
</tbody>
</table>

- **JK FF using NAND gate**

- **JK FF using NOR gate**
JK FF characteristic table:

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Qn</th>
<th>Qn+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

logical Expression:

minimization:

\[ Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n \]

\[ Q_{n+1} = JQ_n + K\overline{Q_n} \]

Excitation table:

<table>
<thead>
<tr>
<th>Qn</th>
<th>Qn+1</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>x</td>
<td>0</td>
</tr>
</tbody>
</table>

- Drawback in JK FF is race around condition which is eliminated in D flip-flop.
D-Flip Flop:

![D-Flip Flop Diagram]

Truth table:

<table>
<thead>
<tr>
<th>Clock (clk)</th>
<th>D</th>
<th>(Q_{n+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>*</td>
<td>(Q_n)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Characteristic table:

<table>
<thead>
<tr>
<th>D</th>
<th>(Q_n)</th>
<th>(Q_{n+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Excitation table:

<table>
<thead>
<tr>
<th>(Q_n)</th>
<th>(Q_{n+1})</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[Q_{n+1} = D\] Therefore it is also called transparent latch.
T Flip-Flop (Toggle):

\[ \begin{array}{ccc}
J & Q \\
\text{clk} & \text{Q} \\
K & \Theta \\
\end{array} \]

\[ J = K = T \]

Truth table:

\[
\begin{array}{ccc|ccc}
\text{clk} & T & \text{Q} & Q_{n+1} \\
0 & 0 & X & X & Q_n & 0 \\
1 & 0 & 0 & 0 & Q_n & 1 \\
1 & 1 & 1 & 1 & \overline{Q}_n & 1 \\
\end{array}
\]

Characteristic table:

\[
\begin{array}{cccc|c}
T & Q_n & Q_{n+1} & Q_{n+1} = TQ_n + T\overline{Q}_n = T \oplus Q_n \\
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
\end{array}
\]

Excitation table:

\[
\begin{array}{ccc|c}
Q_n & Q_{n+1} & T \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]
Important:

\[
\begin{array}{ccc|c}
J & K & Q_{n+1} \\
0 & 0 & Q_n \\
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 1 & Q_n \\
\end{array}
\]

SR-FF \rightarrow 0 \rightarrow D-FF \rightarrow T-FF

⇒ All tables are inside JK FF therefore it is also called as JK FF universal flip-flop.

Excitation table:

<table>
<thead>
<tr>
<th>( Q_n )</th>
<th>( Q_{n+1} )</th>
<th>( S )</th>
<th>( R )</th>
<th>( J )</th>
<th>( K )</th>
<th>( D )</th>
<th>( T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

FF \rightarrow Flip Flop - one bit storing element:

\[ Q_{n+1} = S + R \overline{Q_n} \Rightarrow SR = \text{Set Reset} \]

\[ Q_{n+1} = J \overline{Q_n} + \overline{K} Q_n \Rightarrow JK = \text{name of person who gave the IC} \]

\[ Q_{n+1} = D \Rightarrow D = \text{Delay element} \]

\[ Q_{n+1} = T \oplus Q_n \Rightarrow T = \text{Toggle} \]

Toggle mode of JK:

```
\[ \begin{array}{ccc}
J & Q & f/2 \\
K & \theta & \\ \\
\end{array}
\]
```

```
\[ \begin{array}{ccc}
T & \theta & f/2 \\
K & \theta & \\ \\
\end{array}
\]
```

```
\[ \begin{array}{ccc}
J & Q & f/2 \\
K & \theta & \\ \\
\end{array}
\]
```

```
\[ \begin{array}{ccc}
T & \theta & f/2 \\
K & \theta & \\ \\
\end{array}
\]
```

```
\[ \begin{array}{ccc}
S & Q & f/2 \\
R & \theta & \\ \\
\end{array}
\]
```

All 7 diagrams are in Toggle mode.

PAGE
In level trigger ckt, o/p may changes many time in single clock

In edge trigger, o/p may change only ones in single pulse.
Race Arrround condition:-

⇒ occurs in JK flip-flop (draw back)

\[ t_{PW} = 10 \text{ nsec} \]
\[ t_{PFF} = 10 \text{ nsec} \]

Then:

- Clock:
  \[ \longleftrightarrow 10 \text{ nsec} \]
- \( O/P \):-

Then, To remove race arrround condition:

\[ t_{PFF} \leq t_{PD\text{clock}} \leq t_{PDFF} \]

In JK FF, RAC occurs when \( J = K = 1 \), then \( t_{P} \) and \( t_{PDFF} \)
is less than that of \( t_{PD\text{clock}} \). And therefore the \( O/P \)
is changes several time in single clock pulse.

Condition to remove Race arrround condition:

1. \( t_{PD\text{clock}} < < t_{PDFF} \)
2. Use of Master-slave flipflop.
3. To increase the propagation delay of JK flip flop.
Since the input of slave never goes to (1,1) therefore in master-slave the race around condition is removed.

Since \( \text{up} \) of slave is \( J = Q \) and \( K = \overline{Q} \) therefore it is always (1,0) or (0,1).

Since race around condition occurs only when the \( I/p \) is (1,1).

In M-S FF, O/p is change only when slave O/p is changing.

In M-S FF, Master is level triggered and edge is slave is edge triggered.
Conversion of one FF to other FFs

Procedure:-

⇒ Required FF characteristic table.
⇒ Available FF excitation table.
⇒ Write logical expression for excitation.

(i)

JK-Flip Flop to D-Flip Flop

<table>
<thead>
<tr>
<th>D</th>
<th>Qn</th>
<th>Q_{n+1}</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>0</td>
</tr>
</tbody>
</table>

⇒ write the logical expression for \( J \) and \( K \):

\[
\begin{array}{c|c}
D & Q_n \\
\hline
0 & x \\
1 & x
\end{array}
\quad
\begin{array}{c|c|c}
D & Q_n \\
\hline
\hline
0 & x & 1 \\
1 & x & 0
\end{array}
\]

\( J = D \quad K = \overline{D} \quad \overline{D} = 1 \)

⇒ Implementation:-

![D Flip Flop Diagram]

![JK Flip Flop Diagram]
Important:

(A) SR to:
   ∘ JK : \[ S = \bar{J}Q \]
   \[ R = Q \bar{A} \]
   ∘ D : \[ S = D \]
   \[ R = D \]
   ∘ T : \[ S = \bar{T}Q \]
   \[ R = \bar{T}Q \]

(B) JK to:
   ∘ SR : \[ J = S \]
   \[ K = R \]
   ∘ D : \[ J = D \]
   \[ K = D \]
   ∘ T : \[ J = T \]
   \[ K = T \]

(C) D to:
   ∘ SR : \[ D = S + RQ \]
   \[ \bar{R}Q \]
   ∘ JK : \[ D = \bar{J}Q + \bar{K}Q \]
   \[ \bar{K}Q \]
   \[ K \]
   \[ T : \]
   \[ D = T \bar{Q} \]

(D) T to:
   ∘ SR : \[ T = S\bar{Q} + RQ \]
   \[ \bar{R}Q \]
   ∘ JK : \[ T = \bar{J}Q + \bar{K}Q \]
   \[ \bar{K}Q \]
   \[ K \]
   \[ D : \]
   \[ T = D + \bar{Q} \]
JK FF to SR FF :-

\[ S \quad R \quad Q_n \quad Q_{n+1} \quad J \quad K \]

\[
0 \quad 0 \quad 0 \quad 0 \quad x \quad 0 \quad 0 \quad 0 \quad x \quad 0 \quad 0 \quad 1 \quad 0 \quad x \quad 1 \quad 0 \quad 0 \quad 1 \quad x \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad x \quad x \quad 0 \quad 1 \quad 1 \quad 0 \quad x \quad x \quad x \quad x \quad x \quad x \quad x \quad x
\]

\[
J \rightarrow S \quad x \quad x \quad x \quad x \quad k \rightarrow x \quad x \quad x \quad x \quad x \quad x \quad x \quad x \quad x
\]

\[ J = S \quad k = R \]

Implementation:-

\[
S \quad J \quad k
\]

\[
\alpha \rightarrow S \quad J \quad k
\]

\[ J = T \quad K = T \]

Implementation:-

\[
T \quad Q_n \quad Q_{n+1} \quad J \quad K
\]

\[
0 \quad 0 \quad 0 \quad 0 \quad x \quad 0 \quad 0 \quad 0 \quad x \quad 0 \quad 0 \quad 1 \quad 0 \quad x \quad 1 \quad 0 \quad 0 \quad 1 \quad x \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad x \quad x \quad 0 \quad 1 \quad 1 \quad 0 \quad x \quad x \quad x \quad x \quad x \quad x
\]

\[
J \rightarrow T \quad x \quad x \quad x \quad x \quad K \rightarrow T \quad x \quad x \quad x \quad x \quad x \quad x \quad x \quad x \quad x
\]

\[
J = T \quad K = T
\]

Implementation:-
SR to JK FF:

\[
\begin{array}{cccccc}
J & S & R & \bar{Q}_n & \bar{Q}_{n+1} & S' & R' \\
0 & 0 & 0 & 0 & 0 & x & \\
0 & 0 & 1 & 1 & x & 0 & \\
0 & 1 & 0 & 0 & x & 0 & \\
0 & 1 & 1 & 0 & 0 & 1 & \\
1 & 0 & 0 & 1 & 1 & 0 & \\
1 & 0 & 1 & 1 & x & 0 & \\
1 & 1 & 0 & 1 & 1 & 0 & \\
1 & 1 & 0 & 1 & 0 & 0 & \\
\end{array}
\]

\[S = J\bar{Q}_n\]

\[R = J\bar{Q}_n\]

Implementation:

SR FF to D FF:

\[
\begin{array}{cccccc}
D & Q_n & \bar{Q}_{n+1} & S & R \\
0 & 0 & 0 & 0 & x & \\
0 & 1 & 0 & 0 & 1 & \\
1 & 0 & 1 & 1 & 0 & \\
1 & 1 & 1 & x & 0 & \\
\end{array}
\]

\[D = Q_n\]

\[S = D\bar{Q}_n\]

\[R = \bar{D}\]
(vi) SR to TFF :-

\[
\begin{array}{cccc}
T & Q_n & \bar{Q}_n & SR \\
0 & 0 & 0 & 0 & X \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
S = T \bar{Q}_n \\
R = T Q_n
\]

Implementation:

(vii) D FF to SR FF :-

\[
\begin{array}{cccc}
S & R & Q_n & \bar{Q}_n & D \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & X & X \\
1 & 1 & 0 & X & X \\
\end{array}
\]

\[D = S + R Q_n\]
(viii) \[ \text{D FF to JK FF:} \]

\[
\begin{array}{cccc}
J & K & \bar{Q}_n & Q_{n+1} \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 \\
\end{array}
\]

\[ D = J\bar{Q} + \bar{K}Q \]

**Implementation:**

![JK FF Circuit Diagram]

(ix) \[ \text{D-FF to T-FF:} \]

\[
\begin{array}{cccc}
T & \bar{Q}_n & \bar{Q}_{n+1} & D \\
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
\end{array}
\]

\[ D = \overline{TQ} + T\bar{Q} \]

**Implementation:**

![T FF Circuit Diagram]
(x) T FF to SR FF:

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Qn</th>
<th>Qn+1</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ T = RQ + S\bar{Q} \]

Implementation:

(k) T FF to JK FF:

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Qn</th>
<th>Qn+1</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ T = J\bar{Q} + KQ \]

Implementation:
$T-$FF to D-$FF$ :-

\[
\begin{align*}
D & \quad \bar{D} \\
\bar{D} & \quad \bar{D} \\
T & \quad \bar{T} \\
\bar{T} & \quad \bar{T}
\end{align*}
\]

\[
T = DQ + \bar{D}Q.
\]
<table>
<thead>
<tr>
<th>Latch</th>
<th>Flip Flop</th>
</tr>
</thead>
<tbody>
<tr>
<td>level triggered</td>
<td>Edge triggered</td>
</tr>
<tr>
<td>Asynchronous clk</td>
<td>Synchronous clk</td>
</tr>
</tbody>
</table>

Setup time:

The min. time required to keep I/p at proper level before applying clock.

Hold time:

The min. time required to keep I/p is same level after applying clock.

Note: If we give Data first then we apply clk.
 hari

Register:

- Registers are used to store a group of bits.
- To store n bits, n FF are cascaded in a register.
- Register has four types (depending on I/P and O/P):
  1. SISO (Serial in Serial out)
  2. SIPO (most imp)
  3. PISO
  4. PIPO

Depending on application, the register are two types:

1. Shift register
2. Storage register

(A) SISO (Serial in Serial out):

```
\[
\begin{array}{c|cccc}
   & D_0 & D_1 & D_2 & D_3 \\
\hline
\text{I/P} & \bar{a}_0 & \bar{a}_1 & \bar{a}_2 & \bar{a}_3 \\
\text{O/P} & a_0 & a_1 & a_2 & a_3 \\
\end{array}
\]
```

Data: \( a_3, a_2, a_1, a_0 \)

CLK:

\[
\begin{array}{c|cccc}
10101 & 0000 & 0000 & 0000 & 0000 \\
10000 & 0000 & 0000 & 0000 & 0000 \\
11000 & 0000 & 0000 & 0000 & 0000 \\
01100 & 0000 & 0000 & 0000 & 0000 \\
10111 & 0000 & 0000 & 0000 & 0000 \\
\end{array}
\]

- For serial in register, the n-bit data storage requires n clock pulse.
- In SISO register, to store n-bit data, it requires n clock pulse.
- SISO register is used to provide n clock pulse delay to I/P data.

\[
\text{delay} = n \cdot \text{Clock}
\]
To provide n bit data serially out it requires (n-1) clock pulse.

(A) SIPO (Serial in parallel out):

⇒ In SIPO register to provide n bit data serially in it requires n clock pulse and provide parallel output it requires 0 clk pulse required.
⇒ It is used to serial to parallel converter
⇒ SIPO is used to convert Temporal code to Spacial code.
⇒ Slow to fast converter.

serial :=
Parallel = spacial code

a. The ckt shown in the fig. is a 4 bit SIPO register which is initially loaded with 1010. If three clk pulses applied then the data if the system is:

(a) 1010
(b) 1101
(c) 1111
(d) 0130
Initially loaded 1011. If clk pulse applied continuously after how many clk pulse again the data become 1010.

SOL: o/p of 3 variable x-or is 1 if no. of 1's is odd.

<table>
<thead>
<tr>
<th>clk</th>
<th>Q3</th>
<th>Q2</th>
<th>Q1</th>
<th>Q0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(c) PISO (Parallel in serial out):

\[ \text{control} = 0 = \text{Parallel In} \]
\[ \text{control} = 1 = \text{Serial out} \]

In PISO register to provide parallel in it require 1 clock pulse and to provide for serial out \((n-1)\) clock pulse.
PISO is also used to convert spatial code to temporal code.

**PIPO (Parallel in parallel out):**

- PIPO is used as storage register.
- Parallel in it requires 1 clock pulse.
- Parallel out it requires a clock pulse.

![Diagram of PIPO circuit]

### Important:

<table>
<thead>
<tr>
<th></th>
<th>I/P</th>
<th>O/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>SISO</td>
<td>n</td>
<td>n-1</td>
</tr>
<tr>
<td>SIPO</td>
<td>n</td>
<td>0</td>
</tr>
<tr>
<td>PISO</td>
<td>1</td>
<td>n-1</td>
</tr>
<tr>
<td>PIPO</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Each shift left register operation provides multiplication by 2.
- If n shift left operation performed then data is multiplied by $2^n$.
- Each shift right operation performed then data is divided by 2.
- Each shift right operation performed then data is divided by $2^n$. 
COUNTERS:

- Counters are basically used to count the number of clock pulse applied. It can also be used for frequency divider, time measurement, frequency measurement, range measurement, pulse width count.

  \[ 16 \times \text{Pulse width} = \text{Total width} \]

- Also used for waveform generator.

- With \( N \) FF, max. possible stage in the counter is \( 2^n \)

  \[ N \leq 2^n \]

  \[ n \geq \log_2 N \]

  where \( N = \) no. of stage

  \( n = \) no. of FF.

Depending on clock pulse applied counters of two types:

1. Asynchronous
2. Synchronous

<table>
<thead>
<tr>
<th>Asynchronous</th>
<th>Synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Different FF are applied with different clock.</td>
<td>1. All FF are applied same clock.</td>
</tr>
<tr>
<td>2. It is slower.</td>
<td>2. It is faster</td>
</tr>
<tr>
<td>3. Fixed count sequence, i.e. up or down.</td>
<td>3. Any count sequence is possible</td>
</tr>
<tr>
<td>4. Decoding errors will present.</td>
<td>4. No decoding error will present</td>
</tr>
<tr>
<td>5. Ripple counter</td>
<td>5. Ring counter</td>
</tr>
</tbody>
</table>
No. of stage use in counter mean modulus of counter
i.e. if Mod 5 counter = 5 stage.
Mod n counter = n stage

A decade counter is applied with frequency of 10 MHz then
O/P frequency is...

\[ \text{f}_{\text{out}} = \frac{\text{f}_{\text{in}}}{10} = \frac{10 \text{ MHz}}{10} = 1 \text{ MHz} \]

Let Mod M and Mod N are cascaded then it will act as
Mod MN counter.

Content:
- Basic
- Ripple counter
- Non binary ripple counter
- Ring counter
- Johnson counter
- Synchronous series carry
- Synchronous parallel carry
- Synchronous counter design and analysis

(A) Ripple counter:
- It is a Asynchronous counter.
- Different FF used different clock pulse.
- Toggle mode.
- Only one FF is applied with external clk. and other
  FF's are clk is from previous FF O/P. (whether 0 or 1).
- The FF applied with external clk. will acts as LSB.
3-bit ripple counter: (up counter)

The circuit shown in fig. $Q_0$ toggle for every clk pulse.

An change when $Q_{n-1}$ change from 1-0. i.e. $Q_1$ changes when $a$, changes from 1-0.

Truth table:

<table>
<thead>
<tr>
<th>Clk</th>
<th>$Q_3$</th>
<th>$Q_1$</th>
<th>$Q_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

This is up counter.

It is also called Mod 8 ripple counter.

Timing Diagram:

- $clk$:
- $Q_0$:
- $Q_1$:
- $Q_2$: 4T
- $Q_3$: 8T
In n bit ripple counter propagation delay of each ff is \( t_{pdff} \), hence time period of \( .clk \) is:

\[
\begin{align*}
\tau_{clk} &= n \cdot t_{pdff} \\
\frac{1}{f_{clk}} &= n \cdot t_{pdff} \\
\frac{1}{f_{max}} &= \frac{1}{n \cdot t_{pdff}}
\end{align*}
\]

Note:

1. **I** - ve edge trigger \( \rightarrow \) \( a \) as clock \( \rightarrow \) up counter
2. **I** - ve **" "** \( \rightarrow \) \( \bar{a} \) as clock \( \rightarrow \) up counter
3. **I** - ve **" "** \( \rightarrow \) \( a \) as clock \( \rightarrow \) down counter
4. **I** - ve **" "** \( \rightarrow \) \( \bar{a} \) as clock \( \rightarrow \) down counter

3-Bit Ripple counter (Down counter):

\[
\begin{array}{c}
\begin{array}{c}
T_0 \quad Q_0 \\
T_1 \quad Q_1 \\
T_2 \quad Q_2
\end{array}
\end{array}
\]

Explanation:

1. The ckt shown in fig. \( Q_0 \) toggles for every clock pulse.
2. \( Q_1 \) toggles when \( Q_0 \) changes from 0 to 1.
3. \( Q_2 \) toggles when \( Q_1 \) changes from 0 to 1.

Truth table:

<table>
<thead>
<tr>
<th>Clock</th>
<th>( Q_2 )</th>
<th>( Q_1 )</th>
<th>( Q_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
This is called ripple counter because the input clock is the op of previous FF output. This is just like ripple then called ripple counter.

If the op is (000) and clock is applied then tpd is delay

\[
\begin{align*}
\text{strobe} & \rightarrow \begin{cases} 001 \end{cases} \text{ unwanted or decoding error.} \\
\text{clk} & \rightarrow \begin{cases} 011 \end{cases} \text{ also called transient state.}
\end{cases}
\]

Decoding errors or transient state present in ripple counter due to propagation delay.

To avoid decoding error strobe signal is used.

\[
T_{\text{clk}} \geq n t_{\text{pd}} + T_s
\]

i.e. strobe signal is zero for \(n t_{\text{pd}}\) and after that it is one for next clock. Then all the op is zero for the transient time. Therefore due to strobe signal we can remove decoding error.

In ripple counter with \(n\) FF, \(2^n\) possible state

Frequency after \(n\) FF in the ripple counter is \(f_{\text{op}} = \frac{f_{\text{clock}}}{2^n}\). (i.e. for 3FF op is \(f/8\)).
= Clear and preset are known as Asynchronous I/p.
S.R.J.K.D.T are Synchronous I/p.
Clear: clear is use to reset our FF or counter.
Preset: preset is use to set our FF or counter.

\[ \begin{align*}
\text{RC} & \quad \text{CLR} \\
\text{CLR} & \quad \text{CLR}
\end{align*} \]

\[ \text{CLR} = 0 \quad \text{no effect} \quad \text{CLR} = 0 \quad \text{FF is zero} \\
\text{CLR} = 1 \quad \text{no effect} \quad \text{CLR} = 1 \quad \text{FF is zero} \]

(8) Non Binary Ripple counter:

\[ \begin{align*}
\text{B}(i) \quad \text{BCD counter} & \quad \text{(Decade counter)} \\
\Rightarrow 4 \quad \text{flip flop used}
\end{align*} \]

\[ \begin{align*}
\text{CLK} & \quad \bar{Q}_3 \quad \bar{Q}_2 \quad Q_1 \quad Q_0 \\
0 & \quad 0 \quad 0 \quad 0 \quad 0 \\
1 & \quad 0 \quad 0 \quad 0 \quad 1 \\
2 & \quad 0 \quad 0 \quad 1 \quad 0 \\
3 & \quad 0 \quad 0 \quad 1 \quad 1 \\
4 & \quad 0 \quad 1 \quad 0 \quad 0 \\
5 & \quad 0 \quad 1 \quad 0 \quad 1 \\
6 & \quad 0 \quad 1 \quad 1 \quad 0 \\
7 & \quad 0 \quad 1 \quad 1 \quad 1 \\
8 & \quad 1 \quad 0 \quad 0 \quad 0 \\
9 & \quad 1 \quad 0 \quad 0 \quad 1 \\
\text{classmate} & \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \\
\bar{Q}_3 \quad \bar{Q}_2 \quad \bar{Q}_1 \quad \bar{Q}_0 & \quad \text{PAGE} \end{align*} \]
if we use only $a_3$ and $a_1$, we also use this as clear ext.

⇒ All BCD counter is Decade counter but reverse is not true.
⇒ BCD counter is Asymmetric o/p time diagram.
⇒ o/p frequency of BCD counter is $f/10$.
⇒ low for 8 clock and high for 2 clock in $a_3$.
⇒ duty cycle is 20%.

⇒ In Asynchronous counter follow steps:
1. Trigger $\rightarrow$ +ive
2. clock $\rightarrow$ $\bar{a}$
3. counter $\rightarrow$ up/Dawn
4. Presel/clear $\rightarrow$ clear $\rightarrow$ 000
   $\rightarrow$ Preset $\rightarrow$ 111
5. Decoding logic (Terminating logic)
(iii) Ring Counter: (Synchronous counter)

⇒ The last FF o/p is connected to first FF i/p.

![Diagram of ring counter]

(iv) Explanation:

⇒ Only one FF o/p is high and remaining FF are low.

⇒ In 4 bit ring counter 4 states are there (i.e. for n FF there is n states).

(v) Truth table:

<table>
<thead>
<tr>
<th>clk</th>
<th>a_3</th>
<th>a_2</th>
<th>a_1</th>
<th>a_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

⇒ In synchronous counter the +ive edge or -ive edge, the o/p remains same.

(vi) Time diagram:

![Time diagram for ring counter]
\[
\text{\( n \) bit} \Rightarrow \text{\( n \) state} \Rightarrow \text{\( f_0 = \frac{f}{n} \)}
\]

- Phase shift \( \phi \) in generated waveform is \( \frac{360}{n} \).

\[
\phi = \frac{360}{n}
\]

Application:
- Used in stepper motor control.
- In Analog to Digital converter.

No. of unused state in ring counter is \( 2^n - n \).

Ring counter using J-K:

* Self starting ring counter:

Advantage of ring counter is decoding is simple. To decode no logic gate are required.

Last o/p cannot be connected in the I/P of self start ring counter.
(c) Johnson Counter:

- Symmetric o/p waveform.
- 8 stages are there for 4 bit counter.
- Phase shift = \( \frac{360}{4} = 90^\circ \)
- It is just like SISO register.

\[\text{Truth Table:}\]

<table>
<thead>
<tr>
<th>Clock (clk)</th>
<th>A_3</th>
<th>A_2</th>
<th>Q_1</th>
<th>Q_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Total no. of used state = 8
- Total no. of unused state = \( 2^4 - 8 = 16 - 8 = 8 \) state
- Also called Twisted Ring counter, Mobies counter or, creeping counter or, walking counter or, switch tail counter.
<table>
<thead>
<tr>
<th>CLOCK</th>
<th>( \bar{a}_3 \bar{a}_2 \bar{a}_1 \bar{a}_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( \begin{array}{c} 0 \ 0 \ 0 \ 0 \end{array} ) \rightarrow ( \bar{a}_3 \bar{a}_0 )</td>
</tr>
<tr>
<td>1</td>
<td>( \begin{array}{c} 0 \ 0 \ 0 \ 0 \end{array} ) \rightarrow ( a_3 \bar{a}_2 )</td>
</tr>
<tr>
<td>2</td>
<td>( \begin{array}{c} 1 \ 0 \ 0 \ 0 \end{array} ) \rightarrow ( a_2 \bar{a}_1 )</td>
</tr>
<tr>
<td>3</td>
<td>( \begin{array}{c} 1 \ 1 \ 1 \ 0 \end{array} ) \rightarrow ( a_1 \bar{a}_0 )</td>
</tr>
<tr>
<td>4</td>
<td>( \begin{array}{c} 0 \ 1 \ 1 \ 0 \end{array} ) \rightarrow ( \bar{a}_3 a_0 )</td>
</tr>
<tr>
<td>5</td>
<td>( \begin{array}{c} 0 \ 0 \ 1 \ 1 \end{array} ) \rightarrow ( \bar{a}_2 a_2 )</td>
</tr>
<tr>
<td>6</td>
<td>( \begin{array}{c} 0 \ 0 \ 0 \ 1 \end{array} ) \rightarrow ( \bar{a}_1 a_1 )</td>
</tr>
<tr>
<td>7</td>
<td>( \begin{array}{c} 0 \ 0 \ 0 \ 0 \end{array} ) \rightarrow ( \bar{a}_1 a_0 )</td>
</tr>
</tbody>
</table>

- In Johnson counter to decode each state one two

I/p. AND/NOR gate used.

Disadvantage:

- Lock out may occur. (when counter enter into unused state)

Note: In synchronous counter propagation delay of each counter is \( t_{pf} \), then

\[
T_{clk} \geq t_{pf} \]

\[
t_{clk} \leq \frac{1}{t_{pf}} \]

\[
t_{max} = \frac{1}{t_{pf}} \]

In synchronous counter.
(A) Synchronous Series carry counter:

Explanation:

1. The circuit shown in the figure is a synchronous series carry up counter.
2. In this counter, $a_0$ toggles for every clock pulse.
3. $a_1$ toggles when $a_0 = 1$ and clock is applied.
4. $a_2$ toggles when $a_1 = a_0 = 1$ and clock applied.
5. $a_3$ will toggle when $a_2 = a_1 = a_0 = 1$ and clock applied.
6. This circuit may be a down counter when $\bar{a}$ is connected to $T$.

Truth Table:

<table>
<thead>
<tr>
<th>Clock</th>
<th>$a_3$</th>
<th>$a_2$</th>
<th>$a_1$</th>
<th>$a_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

To provide down counter, use $\bar{a}$ as input to provide 'next stage'.

Note: This is a handwritten note. The content is transcribed as accurately as possible.
(8) Synchronous parallel carry counter:

\[ T_{CLK} \geq t_{PDFF} + (n-2) t_{PAND} \]

- Faster than series carry counter.
- Disadvantage: increased I/O pin of AND Gate.

\[ T_{CLK} \geq t_{PDFF} + t_{PAND} \]

- Same

⇒ Ripple counter < Synchronous serial carry < Synchronous parallel carry counter for faster logic.
(A) Synchronous Series carry counter:

⇒ Explanation:

⇒ Ckt shown in fig. is synchronous series carry up counter.

⇒ In this counter $a_0$ toggles for every clock pulse.

⇒ $a_1$ toggles when $a_0=1$ and clock is applied.

⇒ $a_2$ toggles when $a_1=a_0=1$ and clock applied.

⇒ $a_3$ will toggle when $a_2=a_1=a_0=1$ and clock applied.

⇒ This ckt may be down counter when $\bar{a}$ is connected to $\bar{T}$.

⇒ Truth table:

<table>
<thead>
<tr>
<th>clock</th>
<th>$a_3$</th>
<th>$a_2$</th>
<th>$a_1$</th>
<th>$a_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

⇒ To provide down counter use $\bar{a}$ o/p to provide next stage.
(a) Synchronous parallel carry counter:

\[ T_{CLK} \geq t_{pdFF} + (n-2) t_{pdAND} \]

- Faster than series carry counter.
- Disadvantage is increased I/p pin of AND gate.

\[ T_{CLK} \geq t_{pdFF} + t_{pdAND} \]

- Same
- Ripple counter < Synchronous serial carry < Synchronous parallel carry counter for faster logic.
Synchronous counter design for the given sequence:

Problem: Design a synchronous counter for the count sequence 0 → 3 → 1 → 2 → 0.

Solution: Design a synchronous counter using positive edge trigger D-FF.

Procedure:
1. Identify no. of FF and I/p and o/p.
2. Construct state table.
3. Logical expression for I/p.
4. Minimize.
5. Implement the circuit.

Now,
1. Implement the circuit.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>1 1</td>
</tr>
<tr>
<td>1 1</td>
<td>0 1</td>
</tr>
<tr>
<td>0 1</td>
<td>1 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0</td>
</tr>
</tbody>
</table>

Logical expression:

\[ D_1 = \overline{q}_1 \overline{q}_0 + \overline{q}_1 q_0 = \overline{q}_1 (q_0 + \overline{q}_0) = \overline{q}_1 \]

\[ D_0 = \overline{q}_1 \overline{q}_0 + q_1 q_0 = q_1 \oplus q_0 = q_1 q_0 \]

Implementation:
(ii) State Table:

<table>
<thead>
<tr>
<th>$A_1, A_0$</th>
<th>$A_{1+}, A_{0+}$</th>
<th>$T_1$</th>
<th>$T_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

(ii) Logical Expression:

\[
T_1 = 1
\]

\[
T_0 = \overline{A_1}A_0 + \overline{A_{1+}}A_{0+} = \overline{A_1}
\]

(iv) Implementation:

![Implementation Diagram]
Content:

- Various no. system.
- Arithmetic operation: complement
  - Add, sub.
- Various codes.
- Data representation: unsigned
  - signed
  - signed magnitude
  - \(^{12}S\)
  - \(^{2}S\)

Number system and codes:

<table>
<thead>
<tr>
<th>Base</th>
<th>Different Digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0, 1</td>
</tr>
<tr>
<td>8</td>
<td>0, ..., 7</td>
</tr>
<tr>
<td>10</td>
<td>0, ..., 9</td>
</tr>
<tr>
<td>12</td>
<td>0, ..., 9, A, B</td>
</tr>
<tr>
<td>16</td>
<td>0, ..., 9, A, B, C, D, E, F</td>
</tr>
<tr>
<td>4</td>
<td>0, ..., 3</td>
</tr>
<tr>
<td>6</td>
<td>0, 1, 2, 3, 4, 5</td>
</tr>
</tbody>
</table>
Conversion (Various number systems):

1. Decimal to others:
   - To convert decimal no. into any other base r divide integer part multiply fractional part with r.
   - E.g.
     - Convert \((25.625)_{10} \rightarrow (\_\_\_)_2\).

   \[\begin{array}{c|c}
   \text{Divide by 2} & \text{Remainder} \\
   \hline
   25 & 1 \\
   12 & 0 \\
   6 & 0 \\
   3 & 0 \\
   1 & 1 \\
   0 & 1 \\
   \end{array}\]

   Ans: \((11001.101)_2\)

2. Convert \((25.625)_{10} \rightarrow (\_\_\_)_8\).

   \[\begin{array}{c|c}
   \text{Divide by 8} & \text{Remainder} \\
   \hline
   25 & 3 \\
   3 & -3 \uparrow \\
   0 & -3 \\
   \end{array}\]

   Ans: \((31.5)_8\).

3. When we go from higher to lower base the no. is increased.

- Convert \((25.625)_{10} \rightarrow (\_\_\_)_{16}\).

   \[\begin{array}{c|c}
   \text{Divide by 16} & \text{Remainder} \\
   \hline
   25 & 9 \\
   1 & -1 \uparrow \\
   0 & -1 \\
   \end{array}\]

   Ans: \((19.A)_{16}\).

- Convert \((254)_{16} \rightarrow (\_\_\_)_{16}\).

   \[\begin{array}{c|c}
   \text{Divide by 16} & \text{Remainder} \\
   \hline
   15 & 14 = E \uparrow \\
   0 & 15 = F \\
   \end{array}\]

   = \((FE)_{16}\)
a: convert \((27.4)_{10} = (-)_{4}\)

\[
\begin{align*}
    27 & = 1 \cdot 4^2 + 3 \cdot 4^1 + 1 \cdot 4^0 \\
    6 & = 1 \cdot 4^1 + 2 \cdot 4^0 \\
    0 & = 0 \cdot 4^1 + 0 \cdot 4^0
\end{align*}
\]

\[
\begin{align*}
    0.4 & = 0 \cdot 4^1 + 4 \cdot 4^0
\end{align*}
\]

\[
\begin{align*}
    0.6 & = 0 \cdot 4^1 + 6 \cdot 4^0
\end{align*}
\]

\[
\begin{align*}
    0.4 & = 0 \cdot 4^1 + 4 \cdot 4^0
\end{align*}
\]

\[
\text{Ans.:} \quad (123.121)_4
\]

2. Others to Decimal :-

\[
(\text{X}_1\text{X}_2\text{X}_3, \text{Y}_1, \text{Y}_2)_{\text{r}} = (\text{ })_{10}
\]

\[\Rightarrow\] To convert any other base \(r\) to decimal multiply each digit with positional weighted then add.

\[
(\text{--})_{10} = \text{X}_1 \cdot r^2 + \text{X}_2 \cdot r^1 + \text{X}_3 \cdot r^0 + \text{Y}_1 \cdot r^{-1} + \text{Y}_2 \cdot r^{-2}
\]

a1. Convert \((101.1011)_{2} = (-)_{10}\).

\[
\begin{align*}
    1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 & + 1 \times 2^{-1} + 1 \times 2^{-2} \\
    16 + 0 + 4 + 0 + 1 + 0.5 + 0.25 & = 21.75
\end{align*}
\]

\[
\text{Ans.:} \quad (21.75)_{10}
\]

a1. Convert \((57.4)_{8} = (\text{ })_{10}\).

\[
\begin{align*}
    5 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} & \\
    40 + 7 + 0.5 & = 47.5
\end{align*}
\]

\[
\text{Ans.:} \quad (47.5)_{10}
\]

a1. Convert \((57.4)_{16} = (\text{ })_{10}\).

\[
\begin{align*}
    5 \times 16^1 + 7 + \frac{4}{16} & = 87 + 0.25 = (87.25)_{10}
\end{align*}
\]

a1. Convert \((BAD)_{16} = (\text{ })_{10}\).

\[
\begin{align*}
    11 \times 16^2 + A \times 16^1 + 10 \times 16 + 13 & \\
    256 + 160 + 13 & = 2819
\end{align*}
\]

\[
\text{Ans.:} \quad (2819)_{10}
\]
8. \[ \text{convert } (35)_6 = (-)_{10} \]

\[ 3 \times 6 + 5 \times 1 = 18 + 5 = (23)_{10} \]

9. Octal to Binary & Binary to Octal:

\[ (xyz)_8 = (-)_2 \]

Each no. is represented by its 3 bit binary format.

Guess:\convert (37.45)_8 = ( )_2

So:\[ (011111110010101)_2 \]

Binary to Octal:

Guess:\convert (10110.11)_2

So:\[ (0100110.110) \]

= \((26.6)_8\)

4. Hexadecimal to Binary and Binary to Hexadecimal:

Each digit is represented by 4 bit binary.

Guess:\convert (259A)_{16} = (-)_2

So:\[ (001001010110011010) \]

5. Hexadecimal to Octal or Octal to Hexa:

for Hexa \[ \rightarrow \] octodecimal

\[ \rightarrow \] Binary \[ \rightarrow \]

\[ 0: \text{convert } (CAD)_{16} = (08) \]

So:\[ (CAD)_{16} \]

= \((110010101101)_2 \]

= \((6255)\)
ARITHMETIC OPERATIONS:

(a) Binary Addition, Subtraction, Multiplication:

Add:

\[
\begin{array}{c}
11011.0 \\
+ 101101 \\
\hline
1100011\end{array}
\]

Sub:

\[
\begin{array}{c}
1101 \\
- 10110 \\
\hline
00110\end{array}
\]

Multiply:

\[
\begin{array}{c}
1010 \\
\times 101 \\
\hline
1010 \\
0000 \\
1010 \\
\hline
110010\end{array}
\]

(b) Octal Addition, Subtraction:

\[
\begin{array}{c}
0+0 = 0 \\
0+1 = 1 \\
1+1 = 2 \\
1+7 = 10 \\
7+2 = 11 \\
\hline
\text{Sum of 2 Octal no.} \\
\hline
\end{array}
\]

\[
\begin{array}{c}
7+1 = (8)_{10} = 8 \\
7+6 = 13_{10} \\
\hline
\end{array}
\]

\[
\begin{array}{c}
243 \\
+ 212 \\
\hline
455 \\
\hline
\end{array}
\]

\[
\begin{array}{c}
567 \\
+ 243 \\
\hline
1032 \\
\hline
\end{array}
\]

Subtract:

\[
\begin{array}{c}
743 \\
\underline{564} \\
157 \bigg{)}
\end{array}
\]
Hexadecimal (Add, Subtraction):

Addition:

\[ 1 + 1 = 2 \]
\[ 1 + 9 = A \]
\[ 1 + B = C \]

\[ A + A = (20)_{10} = (14)_{16} \]

Q: 5 6 8 9

1. \[ ADD \]
2. \[ DAD \]
3. \[ 18 \cdot 8 \cdot A \]

Subtract:

\[ \begin{array}{c}
\text{Soll} \\
9 7 4 8 \\
5 8 7 C \\
\hline
3 E C F \\
\end{array} \]

\[ \begin{array}{c}
\text{Soll} \\
9 6 5 4 \\
- 5 3 2 1 \\
\hline
4 3 3 3 \\
\end{array} \]
Complements:

\[ 1 = \rightarrow (r-1)'s \text{ complement} \]
\[ \rightarrow r's \text{ complement} \]

Binary \[ \rightarrow 1's \quad \rightarrow 2's \]

Octa \[ \rightarrow 7's \quad \rightarrow 8's \]

Decimal \[ \rightarrow 9's \quad \rightarrow 10's \]

Hexa \[ \rightarrow F's \quad \rightarrow 16's \]

(r-1)'s complement:

\[ \Rightarrow \text{Subtract from max. no. to the given no.} \]

\[ \text{e.g. comp of } (1010) = \]
\[ \begin{array}{c}
1111 \\
1010 \\
0101
\end{array} \]

To determine \((r-1)'s\) complement subtract given no from max. no. possible in the given base. (max. no. \(\emptyset\))

\((r^n - 1)\)

e.g. 1's complement of 101101 is,

\[ \text{Sol.:- } \]
\[ \begin{array}{c}
111111 \\
101101 \\
010010
\end{array} \]

\[ \text{q1: determine 7's complement of octal no. 5674.} \]
\[ \text{Sol:- } \]
\[ \begin{array}{c}
7777 \\
-5674 \\
2103
\end{array} \]

\[ \text{q2: Determine 9's complement of decimal 2679.} \]
\[ \text{Sol:- } \]
\[ \begin{array}{c}
9999 \\
-2679 \\
7320
\end{array} \]

\[ \text{q3: Det. F's comp. of Hexa. 2689.} \]
\[ \text{Sol:- } \]
\[ \begin{array}{c}
FFFF \\
-2689 \\
0976
\end{array} \]
1: r's complement:
To determine r's complement first write (r-1)'s complement then add 1 at LSB (at right most)

6: Determine 2's complement of 10100.
Sol: 1's complement = 01011

\[ \begin{array}{c}
-10100 \\
01011 \\
\hline
01100 \text{ Ans.}
\end{array} \]

6: Determine 2's complement of 10110.11
Sol: 1's complement = 01001.00
\[ \begin{array}{c}
+1 \\
01001.01 \text{ Ans.}
\end{array} \]

6: Determine 8's complement of octal 2670.
Sol: \( \overline{7777} \)
\[ \begin{array}{c}
-2670 \\
5101 \\
\hline
5110 \text{ Ans.}
\end{array} \]

6: Determine 10's Complement of Decimal 5690.
Sol: 9's = 9999
\[ \begin{array}{c}
-5690 \\
4309 \\
\hline
4310 \text{ Ans.}
\end{array} \]

6: Determine 16's complement of Hexadecimal 5289
Sol: \( \overline{FF} = F_8' = \)
\[ \begin{array}{c}
-5289 \\
AD76 \\
\hline
AD77 \text{ Ans.}
\end{array} \]
**CODES:**

1. **BCD code:**

   - Binary coded decimal
   - Weighted code
   - 4 bit code
   - 8421 code

   Each decimal digit is represented with 4 bit

<table>
<thead>
<tr>
<th>Decimal</th>
<th>BCD</th>
<th>Excess -3 code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0011</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>0100</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>0101</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>0110</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>0111</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>1000</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>1001</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>1010</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1011</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>1100</td>
</tr>
</tbody>
</table>

   - 1010
   - 1011
   - 1100
   - 1101
   - 1110
   - 1111

   Invalid BCD code or don't care.

   During Arithmetic operation if invalid BCD present then add 0110 to get correct result.

   A combinational ckt is applied with 4 bit BCD code which is represented as $D_3D_2D_1D_0$, o/p is $Y$. $Y=1$ then $I/p$ BCD is divisible by 3. then logical expression for $Y$ is.
\[ Y = D_3D_4D_2D_1 + D_5D_6D_2D_1 = D_8D_3D_1 + D_2D_1D_4 + D_2D_3D_4 \]

\[ Y = \bar{D}_8\bar{D}_4\bar{D}_2\bar{D}_1 + D_1D_8 + D_1D_2\bar{D}_4 + D_1D_2D_4 \]

For write BCD code each digit (decimal) is write separately in BCD.

\[ \text{e.g. } (534)^{10} = (010100110100)^{BCD} \]

2. Excess -3 code:

\[ \text{Excess -3 code} = \text{BCD} + 3 \]

- Unweighted code
- 4bit code

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Excess -3 code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0011</td>
</tr>
<tr>
<td>1</td>
<td>0100</td>
</tr>
<tr>
<td>2</td>
<td>0101</td>
</tr>
<tr>
<td>3</td>
<td>0110</td>
</tr>
<tr>
<td>4</td>
<td>0111</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
</tr>
<tr>
<td>6</td>
<td>1001</td>
</tr>
<tr>
<td>7</td>
<td>1010</td>
</tr>
<tr>
<td>8</td>
<td>1011</td>
</tr>
<tr>
<td>9</td>
<td>1100</td>
</tr>
</tbody>
</table>

It is self complement code.

- Only unweighted code which is self complement is Excess -3 code.

The code which addition is 9 is self complement code.

\[ \text{e.g. } 2481 \quad \left\{ \begin{array}{l} \text{weighted} \\
3321 \quad \text{self complemented} \\
4311 \\
5211 \end{array} \right. \]
Write 2421 weighted code.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>2421</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>0111</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
</tr>
<tr>
<td>5</td>
<td>1011</td>
</tr>
<tr>
<td>6</td>
<td>1100</td>
</tr>
<tr>
<td>7</td>
<td>1111</td>
</tr>
<tr>
<td>8</td>
<td>1110</td>
</tr>
<tr>
<td>9</td>
<td>1111</td>
</tr>
</tbody>
</table>

Self Completer

3. Binary to Gray code:
   (A) Binary to Gray:
   - Unweighted code
   - Successive no. is differ by 1 bit
   - Also called unit distance code
   - Also cyclic code, reflective code, and minimum error code.

   Binary: 1011
   Gray: 1110

   \[ B_3 \quad B_2 \quad B_1 \quad B_0 \]
   \[ G_3 \quad G_2 \quad G_1 \quad G_0 \]

(B) Gray to Binary:

   Binary: 1110
   Gray: 1011
Data Representation:

Data representation

Magnitude

\[ \overset{\text{unsigned}}{\rightarrow} \overset{\text{signed}}{\rightarrow} \overset{\text{1's complement}}{\rightarrow} \overset{\text{2's complement}}{\rightarrow} \]

+ive, -ive

no sign bit

one extra bit → sign

sign → MSB

logic 0 → +ive

-1 → -ive.

+6  \[ \rightarrow \]

\[ 0110 \]

0110  \[ \rightarrow \]

0110

-6  \[ \times \]

\[ 1110 \]

1001  \[ \rightarrow \]

1010

(can't write)

In all representation +ive no. are represented in similar way. To represent -ive no. in sign magnitude, only sign bit change. In 1's complement represent -ive no. first write positive no. and then 1's complement to it.

And in 2's complement first write +ive no. and then 2's complement to it.

A no. is represent in its 2's complement for 1011

the equivalent decimal value.

\[ 1011 = -(0101) = -(0101) \]

\[ = -5. \]
To find $5 - 4$?

**Solution:**

1. $5 + (-4 = 0101 + 0011 = 0101$
2. $-4 = 1100$
3. $1100 * 0001$

In 2's complement addition, if any carry present it is discarded.

In 2's complement to extend no. of bit, copy MSB bit.

---

**Table:**

<table>
<thead>
<tr>
<th>Binary</th>
<th>Sign</th>
<th>Mag</th>
<th>1's</th>
<th>2's</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>+0</td>
<td>+0</td>
<td>+0</td>
<td>+0</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>0010</td>
<td>+2</td>
<td>+2</td>
<td>+2</td>
<td>-2</td>
</tr>
<tr>
<td>0011</td>
<td>+3</td>
<td>+3</td>
<td>+3</td>
<td>-3</td>
</tr>
<tr>
<td>0100</td>
<td>+4</td>
<td>+4</td>
<td>+4</td>
<td>-4</td>
</tr>
<tr>
<td>0101</td>
<td>+5</td>
<td>+5</td>
<td>+5</td>
<td>-5</td>
</tr>
<tr>
<td>0110</td>
<td>+6</td>
<td>+6</td>
<td>+6</td>
<td>-6</td>
</tr>
<tr>
<td>0111</td>
<td>+7</td>
<td>+7</td>
<td>+7</td>
<td>-7</td>
</tr>
<tr>
<td>1000</td>
<td>-0</td>
<td>-7</td>
<td>-8</td>
<td>-8</td>
</tr>
<tr>
<td>1001</td>
<td>-1</td>
<td>-6</td>
<td>-7</td>
<td>-7</td>
</tr>
<tr>
<td>1010</td>
<td>-2</td>
<td>-5</td>
<td>-6</td>
<td>-6</td>
</tr>
<tr>
<td>1011</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
<td>-5</td>
</tr>
<tr>
<td>1100</td>
<td>-4</td>
<td>-3</td>
<td>-4</td>
<td>-4</td>
</tr>
<tr>
<td>1101</td>
<td>-5</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
</tr>
<tr>
<td>1110</td>
<td>-6</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>1111</td>
<td>-7</td>
<td>-0</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
4 bit:

- Range of signed mag: -7 → +7
- 1's complement: -7 → +7
- 2's complement: -8 → +7

3. For signed mag. and 1's complement:

\[ n \text{ bit} \rightarrow -(2^{n-1}) \text{ to } + (2^{n-1}) \]

4. For 2's complement:

\[ n \text{ bit} \rightarrow -(2^{n-1}) \text{ to } + (2^{n-1}) \]

5. Perform 5+4 using 2's complement:

\[ 5 = 0101 \]
\[ 4 = 0100 \]

\[ 
\begin{array}{r}
\hline
& 0 & 0 & 0 & 1 \\
\hline
& 0 & 1 & 0 & 0 \\
\hline
\end{array}
\]

Overflow may occur when same (two because sign no. are added in signed representation, because for 4-bit we can only represent -\(2^{n-1}\) to +\(2^{n-1}\) = (-8 to +7)

Let x and y are sign bit of two no. and z is resultant sign no. then condition for overflow is

\[ z = x'y'z + xy'\overline{z} \] condition of overflow.

\[ \begin{array}{c}
\text{Cin} \\
\text{Cout}
\end{array} \quad \begin{array}{l}
0 - \text{No overflow} \\
1 - \text{Overflow}
\end{array} \]
let $C_{in}$ = carry into MSB
$C_{out}$ = carry from MSB
Q: Design a synchronous counter using D-FF for the sequence.

\[0 \rightarrow 2 \rightarrow 5 \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 0\]

So \(\n\)

since 1, 6 is unused

used lock out condition

i.e. \(1 \rightarrow 0\)

6 \(\rightarrow 0\)

| PS | NS
|----|----|
| \(Q_2, Q_1, Q_0\) | \(Q_{2\,t}, Q_{1\,t}, Q_{0\,t}\)
| 0 0 0 | 0 1 0 |
| 0 1 0 | 1 0 1 |
| 1 0 1 | 0 1 1 |
| 0 1 1 | 1 0 0 |
| 1 0 0 | 1 1 1 |
| 1 1 1 | 0 0 0 |
| 0 0 1 | 0 0 0 |
| 1 1 0 | 0 0 0 |

⇒ To avoid lock out change unused states into one of used states in state table.

Q: 10 or 31.

\[
\begin{array}{c|c|c|c|c}
0 & 0 & 1 & 1 & 0
\end{array}
\]

⇒ 4bit ⇒ 4 state requires ⇒ 2FF. required.
Moore & Mealy:

State Machines:

- Moore
- Mealy

- Output depends on present state
- Design easy
- More no. of states

- Output depends on present state
- Design complex
- Less no. of states
1. **Resolution/Step size**:

It change in analog voltage corresponding one LSB increment in the I/P.

\[
\text{Resolution} = \frac{V_r}{2^{n-1}}
\]

where \( V_r \) = reference voltage corresponding to logic 1

\( n \) = no. of bits

2. **Analog o/p voltage**:

\[ V_{\text{analog}} = \text{Resolution} \times \text{Decimal equivalent of binary data} \]

3. In a 4 bit DAC reference voltage 5V. if binary data 1001 is applied then analog voltage is.

\[
\text{Resolution} = \frac{V_r}{2^{n-1}} = \frac{5}{16} = \frac{1}{3}
\]

\[
V_{\text{analog}} = \frac{1}{3} \times 9 = 3V
\]
(3) $V_{FS} =$

Full scale voltage is the max. op voltage of DAC.

$$V_{FS} = V_T \times 2^{n-1} \quad \frac{1}{2^{n-1}}$$

$$V_{FS} > V_T$$

(4) Resolution:

$$\begin{align*}
\text{y. Resolution} &= \frac{\text{Resolution}}{V_{FS}} \times 100 \\
\text{y. Resolution} &= \frac{1}{2^{n-1}} \times 100
\end{align*}$$

(5) Bits/\% Accuracy:

Error acceptable in ADC's or DAC's is equal to resolution or step size.

(8) Analog to Digital converter:

Characteristics of ADC's:

$$V_{ref} \rightarrow \text{ADC} \rightarrow n$$

(9) Resolution = \frac{\text{range}}{2^{n-1}}

where

$$V_{range} = V_{max} - V_{min}$$

(2) y. Resolution = \frac{1}{2^{n-1}} \times 100

(3) Dynamic range = (6n+1.76) dB

\approx 6n \text{ dB}$$
Resolution of R-2R ladder type DAC's is

\[ \text{Resolution} = \frac{V_r}{2^n} \]

\[ V_{FS} = 10.24 \]
\[ n = 10 \]
\[ \text{Resolution} = \frac{10.24}{2^{10}} = 10 \text{mV} \]

\[ \text{error} = \frac{1.28}{2} = \frac{10 \text{mV}}{2} = \pm 5 \text{mV} \]

Calibrate at 25°C i.e. error at 25°C is zero.

\[ +25°C \Rightarrow 5 \text{mV} \]
\[ 1°C = \frac{5 \text{mV}}{25°C} \]
\[ = 0.2 \text{mV/°C} = 200 \text{mV/°C} \]
(A) Digital to Analog Circuit:–

Digital to Analog circuits:–

A) Weighted Resistor DAC:– (4 Bit):–

\\begin{align*}
  b_3 & = \text{MSB} = \text{more current} \\
  b_0 & = \text{LSB} = \text{less current} \\
  I_3 & = \frac{V_r \times b_3}{R} \\
  I_2 & = \frac{V_r \times b_2}{2R} \\
  I_1 & = \frac{V_r \times b_1}{4R} \\
  I_0 & = \frac{V_r \times b_0}{8R} \\
  I_f & = I_3 + I_2 + I_1 + I_0 \\
  v_o & = -I_f R_f
\end{align*}

\textbf{LSB resistance} = \left(2^{n-1}\right) \text{-MSB resistance}.

\Rightarrow \text{In weighted resistor DAC the accuracy is less due to use of different resistance.}

\Rightarrow \text{To overcome this we use R-2R ladder used.}
(B) R-2R ladder -

* Normal ladder
* Inverted ladder

* Non inverting
* Inverting

(8) 3 Bit R-2R ladder -

\[ V_0 = \left( 1 + \frac{R_f}{R_1} \right) V_\alpha \]

\[ V_\alpha = \text{Resolution} \times \text{Decimal equivalent of binary data} \]

\[ V_\alpha = \frac{V_r}{2^n} \geq 2^{i}\hat{b}_i \]

(decimal equivalent = \( b_2 b_1 b_0 \) (binary data))

\[ V_0 = \frac{V_r}{2^n} \geq \sum_{i=0}^{n-1} 2^i\hat{b}_i \left( 1 + \frac{R_f}{R_1} \right) \]

\[ V_0 = \text{Resolution} \times \text{Decimal} \times \text{gain} \]

(B) 3 Bit R-2R ladder (Inverting) -
\[ V_o = \text{Resolution} \times \text{decimal} \times \text{gain}. \]

\[ V_o = V_r \times \sum_{i=0}^{n-1} 2^i b_i \times \left[ \frac{-R_f}{R_1 + R_f} \right] \]

\[ I'_f = \frac{V_r}{2^n} \times \sum_{i=0}^{n-1} 2^i b_i \times \left[ \frac{-1}{R_1 + R_f} \right] \]

**3) Inverted ladder type DAC circuit:**

Since A and A' both are ground then (logical or virtual ground and ground) the switch is at some potential then charging and discharging of switch problem removed in previous ext.

\[ I = \frac{V_r}{R} \]

\[ I_3 = \frac{I_2}{b_3} \times \frac{1}{2} x b_3 \]

\[ I_2 = \frac{I_4}{b_2} x b_2 \]

\[ I_1 = \frac{I_8}{b_1} x b_1 \]

\[ I_0 = \frac{I_16}{b_0} x b_0 \]
\[ I_j = I_0 + I_1 + I_2 + I_3 \]

\[ = \frac{I}{16} \left[ 8b_3 + 4b_2 + 2b_1 + b_0 \right] \]

\[ = \frac{V_r}{2^n} \left( \sum_{i=0}^{n-1} 2^i b_i \right) \times \frac{1}{R} \]

\[ I_1 = \frac{V_r}{2^n} \left( \sum_{i=0}^{n-1} 2^i b_i \right) \frac{1}{R} \]

\[ V_0 = \frac{V_r}{2^n} \left( \sum_{i=0}^{n-1} 2^i b_i \right) \left( -\frac{R_+}{R} \right) \]
Analog to Digital Circuit:

(a) Counter type ADC:

\[ V_1 > V_2 \implies V_o = V_{ref} \]
\[ V_1 < V_2 \implies V_o = -V_{ref} \]

- It is a one-bit quantizer.

- In counter type ADC, a comparator is used in the input stage to compare the input analog voltage with the reference voltage provided by the DAC feedback.

- A counter is used to count the number of clock pulses applied to the comparator when the analog voltage \( V_a \) is greater than the DAC voltage. If \( V_a \) is 1 then the counter counts and if \( V_a \) is less than the reference voltage (DAC voltage) then \( V_a \) is 0 and the counter stops counting and gives the comparative digital output.

- Max. no. of clock pulses required for N-bit conversion is \( 2^n - 1 \).

- Max. conversion time = \((2^n - 1) T_{clk}\).

- Conversion time depends on input analog voltage.

- Also called ramp type ADC.
Parallel comparator type:

- For n bit \(2^{n-1}\) comparator required.
- \(2^n\) resistor required.
- \(2^n\times n\) priority encoder.

Also called Flash ADC (fastest ADC).

3 bit parallel comparator:

No clock pulse is required.

Therefore, it is fastest ADC among all.

Max no. of clock pulse required for n bit conversion is which is inside PIPO.
Range of analog | O/P
---|---
$V_a > 7V/8$ | 111
$7V/8 > V_a > 6V/8$ | 110
$6V/8 > V_a > 5V/8$ | 101
$5V/8 > V_a > 4V/8$ | 100
$4V/8 > V_a > 3V/8$ | 011
$3V/8 > V_a > 2V/8$ | 010
$2V/8 > V_a > V_r/8$ | 001
$V_r/8 > V_o$ | 000

**SAR Type (Successive Approximation Register):**

![SAR Type Diagram]

- **soc** - Start of conversion
- **eoc** - End of conversion

1. Ring counter is used to set the base.
2. Control ckt is used to reset $(V_a < V_r)$.
3. In SAR Type ADC, ring counter will present to successively set the base.
4. Control ckt is used to reset; previously set bit when $V_a < V_r$.
5. In SAR Type ADC, $n$ clock pulse required for $n$ bit conversion.
6. Conversion time $= nT_{CLK}$

SAR Type, conversion time uniform for any analog voltage.
SAR is mostly used in digital ckt to provide interface with microprocessor.

**Dual slope Integrating type ADC:**

\[ T_1 = 2^n T_{CLK} \]

\[ T_2 = 2^m \cdot N T_{CLK} \]

- \( v_r \) slope is always greater than \( v_a \) slope.

In Dual slope, a counter is used to count clock pulse conversion started initially counter is reset to zero and switch \( s \) is connected to \( v_a \) (analog voltage) when integrator is integrating analog voltage \( y_p \) of integrator will become -ive voltage. due to this comparator \( y_p \) will become 1 and counter continues to add clock pulses, after 8 \( T_{CLK} \) pulses again counter value become zero. at this time \( T_1 \) control ckt connect switch \( s \) to \( -v_e \). During \( v_e \) integration up to \( T_2 \) time o/p of integrator is -ive. due to this counter again contain clock pulses. at time \( T_2 \) o/p of integrator become +1 and comparator o/p become 0 due to this counter...
will stop. Let N is count when counter stops.

Then,

\[ V_o = \frac{-V_a \cdot T_1 + V_r (T_2 - T_1)}{RC} \]

at time \( t = T_2 \), \( V_o = 0 \).

\[ \Rightarrow \quad 0 = \frac{-V_a \cdot T_1 + V_r (T_2 - T_1)}{RC} \]

\[ \Rightarrow \quad V_a \cdot T_1 = V_r (T_2 - T_1) \]

\[ \Rightarrow \quad V_a \cdot 2^n \cdot T_{CLK} = V_r (N \cdot T_{CLK}) \]

\[ \Rightarrow \quad N = \frac{V_a \cdot 2^n}{V_r} = \frac{V_a 2^n}{V_r} \]

\[ \Rightarrow \quad V_a = \frac{V_r}{2^n} \cdot N. \]

\[ \begin{array}{c}
V_a = \frac{V_r \cdot N}{2^n}
\end{array} \]

If \( V_r = 2^n \) then.

\[ \begin{array}{c}
V_a = N
\end{array} \]

\[ \Rightarrow \quad \text{This is the most accurate AOC among all.} \]

\[ \Rightarrow \quad \text{All ripple and noise is separated or compressed by capacitor. (Therefore this have more accuracy due to integrator).} \]

\[ \Rightarrow \quad \text{max no. of clock pulse} = \frac{2^n + 2^n - 1}{2^n + 2^n} = 2^n + 1. \]

\[ \Rightarrow \quad \text{It is slowest among all.} \]

Application:

\[ \Rightarrow \quad \text{Mostly used in digital voltmeter.} \]

Clock pulse:

\[ \begin{array}{l}
\text{counter type} = 2^n - 1 \quad \text{Flash} = 1 \\
\text{SAR} = n \quad \text{Dual type} = 2^{m+1}
\end{array} \]
1. **Diode AND Gate**

\[
\begin{align*}
A & \quad B & \quad D_A & \quad D_B & \quad Y \\
0 & \quad 0 & \quad \text{on} & \quad \text{on} & \quad 0 \\
0 & \quad 1 & \quad \text{on} & \quad \text{off} & \quad 0 \\
1 & \quad 0 & \quad \text{off} & \quad \text{on} & \quad 0 \\
1 & \quad 1 & \quad \text{off} & \quad \text{off} & \quad 1 \\
\end{align*}
\]

\[V_o = A \cdot B\]

**Diode OR Gate**

2. For **Not gate** we use transistor.

3. **NOT**:

\[V_o = A \overline{B} + C \overline{D}\]
4. NAND Gate:

When logic gate output is 1 (Tr. OFF) it will act as current source.

When logic gate output is 0 (Tr. ON) it will act as current sink.

5. NOR Gate:

In cutoff and saturation region transistor will act as switch.

<table>
<thead>
<tr>
<th>$J_E$</th>
<th>$J_C$</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_B$</td>
<td>$R_B$</td>
<td>cutoff</td>
</tr>
<tr>
<td>$R_B$</td>
<td>$F_B$</td>
<td>reverse active</td>
</tr>
<tr>
<td>$F_B$</td>
<td>$R_B$</td>
<td>active</td>
</tr>
<tr>
<td>$F_B$</td>
<td>$F_B$</td>
<td>saturation</td>
</tr>
</tbody>
</table>

Page 6
Characteristics of logic family:

1) Propagation delay: \( t_{pd} \):
   - It is measured in nsec.
   \[
   t_{pd} = \frac{t_{PHL} + t_{PLH}}{2}
   \]

2) Propagation delay is always measured from 50% value of the signal.

3) In a CMOS, the time to switch from OFF to ON is more compared to OFF to ON time due to saturation or storage time.

4) Power dissipation:
   - Power dissipation by each logic gate.
   \[
   P_{diss} = mW
   \]
   \[
   P_{diss} = V_{cc} \cdot I_{avg}
   \]

5) Figure of Merit:
   \[
   FOM = P_{diss} \cdot t_{pd} = P_{Joule}
   \]

- \( I^2t \) have best FOM.
- The value of FOM is low, the logic family is best.
Fanout:

It is max. no. of logic gate that can be given by a logic gate.

\[
\text{fanout}_H = \frac{I_{OH}}{I_{IL}}
\]

\[
\text{fanout}_L = \frac{I_{OL}}{I_{IL}}
\]

Max. fanout is \( \min \) value of \( (\text{fanout}_H, \text{fanout}_L) \).

Question: if \( I_{OH} = 400 \mu A, I_{IL} = 40 \mu A, I_{OL} = -16 \mu A, I_{OH} = 16 \), find fanout.

Solution:

\[
\text{fanout}_H = \frac{400}{40} = 10
\]

\[
\text{fanout}_L = \frac{16}{1.6} = 10
\]

Max. fanout = \( (10, 10) \), \( \min = 10 \).

⇒ TTL have max. fanout.

(v) Noise Margin:

It is the max. noise voltage that can be added to the logic family which will not affect the o/p.
\[
\begin{align*}
NM_H &= V_{OH} - V_{IH} \\
NM_L &= V_{IL} - V_{OL}
\end{align*}
\]

overall noise margin = \((NM_H, NM_L)_{min}\)

(A) RTL (Register Transistor logic) family:

\[\overline{A \cdot B} \cdot \overline{C \cdot D}\] (wired AND)

\[V_{cc}\]

- Basic gate - NOR gate.
- \(t_{pd} = 50\) ns
- \(P_{diss} = 10\) mW
- \(FOM = 500\) PJ
- \(NM = 0.2\) V
- Fanout = 3
- wired AND used

Disadvantage:
1. Lower speed of operation
2. Low noise margin
3. Lowest fanout
(E) DCTL (Direct Coupled Transistor Logic): -

1. In RTL logic family if \( r_p \) resistance removed then resulting is DCTL.
2. \( t_{pd} = 40 \text{ nsec} \)
3. Disadvantage: -

5. In DCTL logic, if tr. switch different characteristics are used the Tr having lower \( V_{BEsat} \) then first on and it will not allow other Tr to be on. this phenomenon is known as current Hogging.

Integrated Injection Logic (I\(^2\)L): -

1. It's injecting the current into base.

2. When \( A \) is high the current flows through the base of Tr.
3. i.e. Tr. must be on.
4. \( I^2L \) covers less space.
5. i.e \( I^2L \) have high density.
6. It is equivalent to NOT gate.
There is no problem of current hogging.

- FOM = 0.1 P/J - 0.7 P/J.
  - Best FOM among all logic family.
- tpd = 40ns.
- Fan out = 8.

<table>
<thead>
<tr>
<th>SSI</th>
<th>1-12</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSI</td>
<td>18-99</td>
</tr>
<tr>
<td>LSI</td>
<td>100-1000</td>
</tr>
<tr>
<td>VLSI</td>
<td>&gt;1000</td>
</tr>
</tbody>
</table>

In TTL logic, due to integration of PNP and NPN tr. it occupies less area hence density are more in TTL logic. It is mostly used in MSI and LSI logic family.

Also called MTL (Merged logic family) due to integration of Transistor.
DTL (Diode Transistor logic) family:

AND gate followed by NOT gate.

20k resistor used only for discharging the junction capacitance. The capacitance which is discharged is Transition cap. Cc.

The out is called basic DTL gate.

In this any one of the I/p is low or all the I/p are low. D_A and or D_B will become forward bias whereas D_1 and D_2 will become reverse bias due to T. T1 is off and o/p is 1.

When all the I/p's are high then D_A and D_B become reverse bias and D_1 and D_2 will become forward bias and T1 is on and o/p is low.

The basic gate is NAND gate.

- \( t_{pd} = 30 \text{ ns} \)
- \( P_{diss} = 8 \text{ mw} \)
- \( R_{OM} = 240 \text{ Pf} \)
- \( N_{M} = 0.75 \text{ V} \)
- Fanout = 3

It provides wired AND operation. \( V_{cc} \)

To increase fanout we introduce T in place of Diode.

5k\( \Omega \) resistor used to lower the I_1 current.

called STANDARD LOGIC
High Threshold Logic (HTL) family:

\[ V_{CC} = 15 \text{V} \]

- Zener Diode is used in place of D2
- NM = 4 - 5V (Highest noise margin)
- Since in DTL all diode and transistor is -ive temp coefficient \( \frac{dv}{dT} = 0 - 2.5 \text{mv/}^\circ \text{C} \)
- logic 0 = 2V, logic 1 = 12V
- \( t_{pd} = 90 \text{ns} \)
- \( P_{diss} = 55 \text{mW} \)
- \( FOM = 4950 \text{ PJ} \times 5000 \text{ PJ} \)
- Fanout = 8
- Basic gate = NAND gate
- Noise margin = 4V - 5V
TTL (Transistor Transistor logic) family:

\[ V_{cc} \]

\[ 4k \]
\[ 1.6k \]
\[ 130\Omega \]

\[ T_1 \]
\[ T_2 \]
\[ T_3 \]
\[ T_4 \]
\[ Y \]

\[ A \]
\[ B \]
\[ T_1 \]
\[ T_2 \]
\[ T_3 \]
\[ T_4 \]
\[ Y \]

\[ 0 \]
\[ 0 \]
\[ 0 \]
\[ 1 \]
\[ 0 \]
\[ 1 \]
\[ R \]
\[ S \]
\[ S \]
\[ C \]
\[ O \]

\[ T_1 = \text{Multiemitter transistor.} \]

The circuit shown in fig. is standard TTL logic family; it basically have three stages:

i) Multiemitter I/p stage

ii) Phase splitter

iii) Totem pole or active pull up, o/p stage.

active = use of \( T_1, T_4 \)

Pullup = \( T_1, T_4 \) connect to \( V_{cc} \).

Operation:

\( \Rightarrow \) Any one of I/p low or all I/p’s are low then EB junction is FB (\( J_e = FB \)) and collector base (\( J_o = R_1 \) is RB). \( T_1 \) is in active mode due to this \( T_2 \) and \( T_3 \) are off (in cutoff region) whereas \( T_4 \) is SAT hence o/p is 1.

\( \Rightarrow \) When all the I/p’s are high then \( J_e \) (EB junct”) of \( T_1 \) is RB and \( J_o \) (CB junct”) is FB. (The mode of operation is reverse active.)
T2 and T3 are in saturation and T4 is in cutoff. Hence \( V_{CE} \) is zero.

\[
\begin{align*}
V_{IN} &= 8 \text{ V} & \text{I/P voltage at which } T_r \text{ takes logic} \\
V_{OH} &= 2.4 \text{ V} \\
V_{IL} &= 0.8 \\
V_{IH} &= 0.4 \\
\end{align*}
\]

\[t_{pd} = 10 \text{ ns} \]
\[P_{diss} = 10 \mu \text{W} \]
\[FOM = 100 \mu \text{J} \]
\[\text{Fanout} = 10 \]
\[V_{NM} = 0.4 \text{ V} \]

1. Diode D is used to cutoff \( T_r \) \( T_4 \) when \( T_3 \) is ON.

2. Advantage of Totem pole:
   1. Lower power dissipation
   2. Higher speed of operation
   3. Higher fan out

3. Disadvantage of Totem pole:
   It is not used in wired logic

4. To provide wired AND logic open collector configuration is used.
130 kΩ resistor used in collector in O/P stage to reduce ripple or noise generation so in high frequency of operation.

In TTL if any I/P is open it behaves as logical 0.

Clamping diodes are connected in I/P stage to protect transistor during high frequency of operation.

Claumping D. removes ringing of high frequency operation.

There are different types of TTL:

(a) Standard TTL
(b) High speed
(c) Low power
(d) Schottky TTL
High speed TTL:

In standard TTL logic family if resistor value reduces then $t_{pd}$ reduces and known as High speed logic family.

$t_{pd} = 6\text{nsec.}$

$\Rightarrow$ Power dissipation increases.

Low speed power TTL:

In TTL logic family if resistor value increased then power dissipation reduced and resultant is known as low power logic family.

Schottky diode:

If Schottky diode is used b/w collector and base region then it will remove storage time and saturation delay. The family known as Schottky diode TTL.

$t_{pd} = 2\text{nsec.}$
ECL (Emitter coupled logic family):

1. It is never go in saturation region.
2. Work only in cutoff and active region.
3. It is fastest logic family due to work in active and cutoff region. (Because it is non-saturated)

\[ V_{CE} = -1.17V \]

\[ t_{pd} = 1\text{ns} \]
\[ \text{fanout} = 25 \]

4. It basically contains contains two stage.
   1. Differential amp' I/p stage.
   2. CC or Emitter follower O/p stage.

5. Due to use of D.A. complementry o/p are available in ECL logic family. (NOR/OR) gate.

6. Due to use of CC stage in the O/p fanout is high.

7. Negative spikes do not affect the transistor due to -ive power supply.

8. ECL uses -ive power supply. Due to this any spike or negative voltage not affect operation.

\[ t_{pd} = 1\text{ns} \]
\[ P_{diss} = 55\text{mW} \]
\[ FOM = 55\text{pJ} \]
\[ \text{Fanout} = 25 \]
\[ N_{M} = 0.3\text{V} \]
\[ \text{logic}_0 = -1.7 \text{V} \]
\[ \text{logic}_1 = -0.85 \text{V} \]

It is logic 1 mode only when voltage supply is negative.

ECL provides wired AND logic.

\[ (A+B) + (C+D) \]

If any input is open then it is logic 0.
NMOS:

N-channel:
- Logic 0' = OFF
- Logic 1' = ON

P-channel MOS:
- Logic 0' = ON
- Logic 1' = OFF

Since FET is a voltage variable resistor hence in MOS circuit in place of regulators we use MOSFET.

NMOS NOT gate:

A T2 \rightarrow
O \quad OFF \rightarrow 1
I \quad ON \rightarrow 0

NMOS NAND gate:

A B T3 T3 \rightarrow
0 0 \quad OFF \rightarrow 1
0 1 \quad OFF \rightarrow 1
1 0 \quad ON \rightarrow 0
1 1 \quad ON \rightarrow 0
NMOS NOR Gate:

- $t_{pd} = 250\, \text{ns}$
- $P_{diss} = 1\, \text{mW}$
- $FOM = 250\, \text{PJ}$
- $fanout = 5$
- $NM = 1.5\, \text{V}$
PMOS NOR Gate:

\[ y = \overline{a \cdot b} \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ t_{pd} = 300 \text{ nsec} \]

\[ P_{diss} = 0.2 \text{ mW} \]

\[ FOM = 60 \text{ PJ} \]

PMOS

- Strong in 1
- Weak in 0

NMOS

- Strong in 0
- Weak in 1

where, \( V_T = \text{Threshold voltage} \).
CMOS NOT gate:

\[ \text{Transfer characteristics:} \]

\[ P_{\text{diss}} = 0.01 \text{ mw} \]
\[ t_{\text{pd}} = 70 \text{ nsec} \]
\[ FOM = 0.7 \text{ PJ} \]
\[ \text{Fanout} = 50 \]
\[ NM = \frac{V_{DD}}{2} \]

Power dissipation:

(i) Static \( PD = \)

During logic '0' or logic '1'.

(ii) Dynamic \( PD = \)

During transition from 0→1 or 1→0.

\[ PD = C_f V_{DD}^2 \]
GMOS NAND Gate

\[ V_{DD} \]

\[ A \quad B \quad T_1 \quad T_2 \quad T_3 \quad T_4 \quad Y \]

\[ 0 \quad 0 \quad \text{ON} \quad \text{ON} \quad \text{OFF} \quad \text{OFF} \quad 1 \]

\[ 0 \quad 1 \quad \text{ON} \quad \text{OFF} \quad \text{OFF} \quad \text{ON} \quad 1 \]

\[ 1 \quad 0 \quad \text{OFF} \quad \text{ON} \quad \text{ON} \quad \text{OFF} \quad 1 \]

\[ Y \quad 1 \quad 1 \quad \text{OFF} \quad \text{OFF} \quad \text{ON} \quad \text{ON} \quad 0 \]
Transmission Gate:

\[ \text{Control} \quad A \quad Y \]

\[
\begin{align*}
0 & \quad \text{High impedance} \quad 0 \\
1 & \quad 0 \quad 0 \\
1 & \quad 1 \quad 1
\end{align*}
\]

Symbol of Transmission gate:

\[ A \quad \xrightarrow{\text{___}} \quad Y \]

\[ S \]

\[ I_0 \quad I_1 \]

Total no. of transistor = 2 + 2 + 2 = 6.

CMOS monostable multivibrator:

\[ V_{DC} \quad \text{NAND} \quad \text{AND} \]