The Electronic System

1- Binary System \((0,1)_2\)
2- Ternary System \((0,1,2)_3\)
3- Octal System \((0,1,2,3,4,5,6,7)_8\)
4- Decimal System \((0,1,\ldots, 8,9)_{10}\)
5- Hexadecimal System \((0,\ldots, 8,9,A,B,C,D,E,F)_{16}\)

Conversion from binary to decimal

e.g. (1) Convert the binary \((10111)_2\) to decimal.

\[
\begin{align*}
\text{Sol.:} & \quad (10111)_2 \\
& = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
& = 16 + 0 + 4 + 2 + 1 = 23_{10}
\end{align*}
\]

Notice \(2^0 = 1\) for all numbers, e.g., \((10)^0 = 1\) and \((986)^0 = 1\).
\[ \text{e.g. (2)} \quad (0.011)_2 = 0.75 \]

\[ \Rightarrow 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 0.75_{10} \]

\[ 0 \times 0.25 + 0.125 = (0.375)_{10} \]

\[ \therefore (0.011) = (0.375)_{10} \]

\[ \text{e.g. (3)} \quad (101110.011)_2 = 23.375_{10} \]

\[ \Rightarrow (101110.011)_2 = 23.375_{10} \]

---

**Conversion from decimal to binary**

\[ \text{e.g. (1)} \quad \text{Convert the decimal } (53)_{10} \text{ to binary.} \]

\[ \begin{array}{c|c|c}
2 & 53 & 1 \text{ least} \\
2 & 26 & 0 \\
2 & 13 & 1 \\
2 & 6 & 0 \\
2 & 3 & 1 \\
1 & 1 & 1 \text{ most} \\
\end{array} \]

\[ = (110101)_{2} \]

\[ \Rightarrow 0\% (53)_{10} = (110101)_{2} \]

\[ \text{e.g. (2)} \quad (0.375)_{10} \]

\[ \begin{array}{c|c|c}
0.375 & \times 2 = 0.75 & 0 \text{ most} \\
0.75 & \times 2 = 1.5 & 1 \\
0.5 & \times 2 = 1 & 1 \text{ least} \\
\end{array} \]

\[ = (0.11)_2 = (0.011)_2 \]

\[ \Rightarrow 0\% (0.375)_{10} = (0.011)_2 \]

\[ \text{e.g. (3)} \quad (53.375)_{10} \]

\[ \Rightarrow (53.375)_{10} = (110101.011)_2 \]
Conversion from octal to decimal

\[ \text{e.g. (1)} \quad (256)_8 \]

\[ = 2 \times 8^2 + 5 \times 8^1 + 6 \times 8^0 \]
\[ = 2 \times 64 + 5 \times 8 + 6 \]
\[ = 128 + 40 + 6 \]
\[ = 174 \]

\[ \therefore (256)_8 = (174)_{10} \]

\[ \text{e.g. (2)} \quad (72.4)_8 \]

\[ = 7 \times 8^1 + 2 \times 8^0 + 4 \times 8^{-1} \]
\[ = 56 + 2 \times \frac{4}{8} \]
\[ = 56 + 1 \]
\[ = 58 \]

\[ \therefore (72.4)_8 = (58.5)_{10} \]

Conversion from decimal to octal

\[ \text{e.g. (1)} \quad (518)_{10} \]

\[ \begin{array}{c|ccc}
8 & 518 & 6 & \text{Lead} \\
8 & 64 & 0 & \\
8 & 8 & 0 & \\
\hline
1 & & \text{Most} & \\
\end{array} \]

\[ = (1006)_8 \]

\[ \therefore (518)_{10} = (1006)_8 \]
Conversion from hexadecimal to decimal

\[ \text{e.g. (1)} \quad (2A)_{16} \]

\[
\text{Sol.} \quad (2A)_{16} = 16 \times 2 + 16 \times A \\
= 16 \times 2 + 1 \times 10 \\
= 32 + 10 \quad \text{= (42)}_{10} \\
\therefore (2A)_{16} = (42)_{10}
\]

\[ \text{e.g. (2)} \quad (5A - 8)_{16} \]

\[
\text{Sol.} \quad (5A - 8)_{16} = 16 \times 5 + 16 \times 10 - 16 \times 8 \\
\quad = 80 + 10 - \frac{8}{16} \quad \text{= (90 - 0.5)}_{10} \\
\therefore (5A - 8)_{16} = (90.5)_{10}
\]

Conversion from decimal to hexadecimal

\[ \text{e.g. (1)} \quad (295)_{10} \]

\[
\text{Sol.} \quad 16 \div 295 \\
16 \div 18 \quad 7 \quad \text{= (127)}_{16} \\
2 \quad \therefore (295)_{10} = (127)_{16}
\]
Conversion from octal to binary

\[ \begin{align*}
\text{e.g. (1)} & \quad (23)_8 \quad \ldots \quad 28, 64, 32, 16, 8, 4, 2, 1, 1 \\
& \downarrow \downarrow \\
& (101, 011)_2 \\
& 00 (23)_8 = (10, 011)_2
\end{align*} \]

\[ \begin{align*}
\text{e.g. (2)} & \quad (462, 52)_8 \\
& \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\
& (100, 110, 010, 101, 010)_2
\end{align*} \]

Conversion from binary to octal

\[ \begin{align*}
\text{e.g. (1)} & \quad (1011)_2 \\
& \downarrow \downarrow \downarrow \\
& 001 (11)_8, 00 (1011)_2 = (13)_8
\end{align*} \]

\[ \begin{align*}
\text{e.g. (2)} & \quad (110, 101 - 011, 011)_2 \\
& \downarrow \downarrow \downarrow \downarrow \downarrow \\
& (011, 101, 011, 011)_2 \\
& (153, 3, 3)_8
\end{align*} \]
Conversion from hexadecimal to binary

\[\text{e.g. (1)} \quad (D\ F\ C)_{16}\]

\[\begin{align*}
S_0 \times 16 & = 12D \quad 15F \quad 1C \\
(1101) & \quad (1111) \quad (1100) \quad 00 \quad (D\ F\ C)_{16} = (1101)_{16} (1111)_{16} (1100)_{16}
\end{align*}\]

Conversion from binary to hexadecimal

\[\text{e.g. (1)} \quad (11010000)_{2}\]

\[\begin{align*}
S_0 \times 16 & = 0010 \quad 1011 \quad 0001 \\
(0000) & \quad (1011) \quad (0001) \quad 00 \quad (35B7)_{16}
\end{align*}\]

\[\text{e.g. (2)} \quad (10110000)_{2}\]

\[\begin{align*}
S_0 \times 16 & = 0011 \quad 1010 \quad 0100 \\
(1011) & \quad (0000) \quad (110) \quad 00 \quad (30D6)_{16}
\end{align*}\]

\[\text{e.g. (3)} \quad (11010100)_{2} = (BO\ DC)_{16}\]

\[\text{e.g. (4)} \quad (11010000)_{2} = (BO\ DC)_{16}\]

Binary Addition

| \[\text{0} + \text{0} = \text{0}\] | \[\text{e.g.} \quad \text{0} \quad \text{0} \quad \text{0}\] |
| \[\text{0} + \text{1} = \text{1}\] | \[\text{e.g.} \quad \text{0} \quad \text{1} \quad \text{1}\] |
| \[\text{1} + \text{1} = \text{10}\] | \[\text{e.g.} \quad \text{1} \quad \text{0} \quad \text{1} \quad \text{0}\] |

\[\begin{array}{c}
\text{0} + \text{0} = \text{0} \\
\text{0} + \text{1} = \text{1} \\
\text{1} + \text{1} = \text{10}
\end{array}\]

\[\begin{array}{c}
\text{1} + \text{1} = \text{10}
\end{array}\]
Binary Subtraction

1. 1's Complement Subtraction

The 1's complement of a binary number is found by simply changing all 1s to 0s and all 0s to 1s, as illustrated by a few examples.

<table>
<thead>
<tr>
<th>Binary number</th>
<th>1's Complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>01010</td>
</tr>
<tr>
<td>10111</td>
<td>01000</td>
</tr>
<tr>
<td>111100</td>
<td>000011</td>
</tr>
</tbody>
</table>

Example (1): Subtract 10101 from 11011 using the 1's complement method.

Solution:

\[
\begin{align*}
11011 & \quad - \quad 10101 \\
\hline
01000 & \quad + \quad 10100 & \quad \text{First complement of } 10101
\end{align*}
\]

\[
11011 = 01010
\]

Example (2): Subtract 1102 from 10102 using the 1's complement method.

Solution:

\[
\begin{align*}
1010 & \quad - \quad 0110 \\
\hline
1000 & \quad + \quad 1001 & \quad \text{First complement of } 0110
\end{align*}
\]

\[
\begin{align*}
\hline
100011 & \quad + \quad 1 \\
\hline
100100 & \quad \text{Final result}
\end{align*}
\]
2's Complement Subtraction

The 2's complement of a binary number is found by adding 1 to the 1's complement. An example will show this is done.

<table>
<thead>
<tr>
<th>Binary number</th>
<th>2's Complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>001 1's complement of 110</td>
</tr>
<tr>
<td></td>
<td>+1 2's complement of 110</td>
</tr>
<tr>
<td>10101</td>
<td>0110 1's complement of 10101</td>
</tr>
<tr>
<td></td>
<td>+1 2's complement of 10101</td>
</tr>
<tr>
<td></td>
<td>01011 Final result</td>
</tr>
</tbody>
</table>

E.g. (i) Subtract 10101 from 1011 using the 2's complement method.

\[ 0111 - 10101 \rightarrow 01010 +1 \]

\[ 01011 \]

\[ 110 \]

E.g. (ii) Subtract 1102 from 10102 using the 2's complement method.

\[ -1010 \rightarrow 1010 +1 \]

\[ 110 \]

\[ \overline{1010} \] 2's complement of 1100
**Logic Functions Gates**

1. **AND-gate**
   - Symbol: \( A \rightarrow y \, \text{AND B} \)
   - Truth Table:
     | A | B | y |
     |---|---|---|
     | 0 | 0 | 0 |
     | 0 | 1 | 0 |
     | 1 | 0 | 0 |
     | 1 | 1 | 1 |
   - Output Equation: \( y = A \cdot B \)

2. **OR-gate**
   - Symbol: \( A \rightarrow y \, \text{OR B} \)
   - Truth Table:
     | A | B | y |
     |---|---|---|
     | 0 | 0 | 0 |
     | 0 | 1 | 1 |
     | 1 | 0 | 1 |
     | 1 | 1 | 1 |
   - Output Equation: \( y = A + B \)

3. **NOT-gate (Inverter)**
   - Symbol: \( A \rightarrow y \, \text{NOT B} \)
   - Truth Table:
     | A | y |
     |---|---|
     | 0 | 1 |
     | 1 | 0 |
   - Output Equation: \( y = \overline{A} \)

4. **NAND-gate**
   - Symbol: \( A \rightarrow y \, \text{NAND B} \)
   - Truth Table:
     | A | B | y |
     |---|---|---|
     | 0 | 0 | 1 |
     | 0 | 1 | 0 |
     | 1 | 0 | 0 |
     | 1 | 1 | 0 |
   - Output Equation: \( y = \overline{A} \cdot \overline{B} \)

5. **NOR-gate**
   - Symbol: \( A \rightarrow y \, \text{NOR B} \)
   - Truth Table:
     | A | B | y |
     |---|---|---|
     | 0 | 0 | 1 |
     | 0 | 1 | 0 |
     | 1 | 0 | 0 |
     | 1 | 1 | 0 |
   - Output Equation: \( y = \overline{A} + \overline{B} \)

6. **XOR-gate**
   - Symbol: \( A \rightarrow y \, \text{XOR B} \)
   - Truth Table:
     | A | B | y |
     |---|---|---|
     | 0 | 0 | 0 |
     | 0 | 1 | 1 |
     | 1 | 0 | 1 |
     | 1 | 1 | 0 |
   - Output Equation: \( y = A \oplus B \)

7. **X-NOR-gate**
   - Symbol: \( A \rightarrow y \, \text{X-NOR B} \)
   - Truth Table:
     | A | B | y |
     |---|---|---|
     | 0 | 0 | 1 |
     | 0 | 1 | 0 |
     | 1 | 0 | 0 |
     | 1 | 1 | 1 |
   - Output Equation: \( y = A \oplus B \)
   - Output Equation: \( y = A \otimes B \)
Number of Variables

Number of inputs = 2

1 - if number of variables one (A)

\[ \text{Number of inputs} < 2 \Rightarrow A \]

2 - if number of variables two (A, B)

\[ \text{Number of variables} = 2 < 4 \Rightarrow \begin{array}{c} \text{A} \\ \text{B} \end{array} \]

3 - if number of variables three (A, B, and C)

\[ \text{Number of variables} = 2 < 8 \Rightarrow \begin{array}{c} \text{A} \\ \text{B} \\ \text{C} \end{array} \]
1- OR Rules
\[ A + 0 = A, \quad A + 1 = 1, \quad A + A = A, \quad A + \bar{A} = 1 \]

2- AND Rules
\[ A \cdot 0 = 0, \quad A \cdot 1 = A, \quad A \cdot A = A, \quad A \cdot \bar{A} = 0 \]

3- Complementarity Rules
\[ \bar{0} = 1, \quad \bar{1} = 0, \quad \text{if } A = 0 \text{ then } \bar{A} = 1, \quad \bar{A} = A \]

4- Commutative Rules
\[ A + B = B + A, \quad A \cdot B = B \cdot A \]

5- Distributive Rules
\[ A \cdot (B + C) = AB + AC, \quad A + BC = (A + B)(A + C) \]
\[ A + \bar{A} \cdot B = A + B \]

6- Absorptive Rules
\[ A + AB = A, \quad A \cdot \bar{A} = A + B \]
\[ A \cdot (A + B) = A, \quad A \cdot (A + B) = A \cdot \bar{A} = AB \]

\[ \text{e.g. (1)} - \text{prove that } A + \bar{A}B = A + B \]
\[ \text{Sol.} \quad (A + \bar{A})(A + B) = A + B \]

\[ \text{e.g. (2)} - \text{prove that } A \cdot (A + B) = A \]
\[ \text{Sol.} \quad A \cdot A + AB = A + AB = A \cdot (A + B) = A \]

\[ \text{e.g. (3)} - \text{prove that } A \cdot (\bar{A} + B) \]
\[ \text{Sol.} \quad A \cdot \bar{A} + AB = A \cdot B \]
e.g. (4). Prove that $AC + ABC = AC$

$\text{Sol. } AC + ABC = AC$

e.g. (5). Prove that $ABC + A\overline{B}C + A\overline{B}\overline{C} = A(B + C)$

$\text{Sol. } AC(B + C) + A\overline{B}C = AC + A\overline{B}\overline{C} = A(C + B\overline{C})$

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

e.g. (6). Prove that $(A + B)(A + C) = A + BC$

$\text{Sol. } A(A + AC + AB + BC) = A(1 + B) + AC + BC$

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

e.g. (7). Draw and implement the logic circuit. Then describe its truth table.

1. $F = AB + A\overline{B}$

$\text{Sol. }$ 

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

2. $F = \overline{A}C + B\overline{C} + \overline{A}B$

$\text{Sol. }$ 

<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>1</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>
Simplify the following Boolean expression:

\( a - F = \overline{A}BC + ABC + ABC + BC \)

\[ S01. \quad F = \overline{A}BC + ABC + ABC + BC \]

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

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

\[ F = B + ABC \]

\( b - F = \overline{B} \cdot (A + C) + C \cdot (\overline{A} + B) + AC \)

\[ S02. \quad F = \overline{B} \cdot (A + C) + C \cdot (\overline{A} + B) + AC \]

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

**De Morgan's Laws**

1. \[ \overline{A \cdot B} = \overline{A} + \overline{B} \]
2. \[ \overline{A} + \overline{B} = \overline{A} \cdot \overline{B} \]

**Example (1)**

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

So,
\[ (\overline{AB} + \overline{A}) \cdot \overline{B} = \overline{AB} + \overline{A} \cdot \overline{B} \]
\[ = (\overline{AB}) \cdot (\overline{A} + \overline{B}) + \overline{B} \]
\[ = (\overline{AB}) \cdot (\overline{A} + \overline{B}) + \overline{B} \]
\[ = (A + \overline{B}) \cdot (\overline{A} + \overline{B}) + \overline{B} \]
\[ = A \overline{A} + AB + A \overline{B} + B \overline{B} + \overline{B} \]
\[ = AB + \overline{B} (\overline{A} + 1) = AB + \overline{B} = (\overline{AB} \cdot (B + \overline{B}) \]
\[ = A + \overline{B} \]
Simplification using Karnaugh Map.

1- K.M of two Variable (x, y)

2- K.M of three Variable (x, y, z)

3- K.M of Four Variable (x, y, z, and w)

4- K.M of Five Variable (x, y, z, w, and l)

Example: Simplify Circuit given this result.
Simplify circuit given this result: \( A \cdot B \cdot Y \)

\[ y = A + B \]

Using K-M to simplify the following function:
\[ F = \overline{AB} + AB + \overline{AB} + AB \]

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

Using K-M simplify the following results:
\[ A \cdot B \cdot Y \]

\[ y = A \cdot C + AB + BC \]

Using K-M simplify the following function:
\[ Z = \overline{A} \cdot B \cdot C + A \cdot B \cdot C + \overline{A} \cdot B \cdot C + \overline{A} \cdot B \cdot C + A \cdot B \cdot C + \overline{A} \cdot B \cdot C \]

\[ z = \overline{B} \cdot C \]
\[ z = \overline{x}yz + \overline{x}yz + \overline{x}yz + \overline{y}z + \overline{x}yz + xyz \]

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

\[ F = \overline{x}yz + \overline{x}yz + \overline{y}z + \overline{y}z \]

\[ y = \overline{x}y + xy = x \oplus y \]

\[ F = \overline{x}yz + \overline{x}yz + \overline{y}z + \overline{y}z \]

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

\[ F = \overline{AC} + \overline{AB} + ABC + BC \]

\[ y = C + AB \]

\[ F(x, y, z) = \overline{z}(0, 2, 4, 5, 6) \]

\[ y = \overline{z} + xy \]
\[ Y = A \bar{D} + \bar{A}B + A\bar{A} \]

\[ Y = D + \bar{B} \]

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

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

\[ F = \bar{A} \bar{B} \bar{C} + \bar{B} \bar{C} \bar{D} + \bar{A} \bar{B} \bar{C} \bar{D} + A \bar{B} \]

\[ F = \bar{B} \bar{C} + \bar{B} D + \bar{A} C \bar{D} \]
Applications of Ex-Or gate

1. Parity checker

\[ y(\text{odd}) = \begin{cases} 
0 & \text{if number of ones even} \\
1 & \text{if number of ones odd} 
\end{cases} \]

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

E.g. (i). Design 3-bits parity checker.
Sol.

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

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

\[ y = A(BC + BC) + A(BC + BC) \]

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

\[ y = \bar{A}D + A\bar{B} \]

\[ y = A \oplus D \Rightarrow y = A \oplus B \oplus C \]

2. Control Inverter

Let \( z \Rightarrow \text{control inverter} \)

\[ z = \begin{cases} 
0 & \text{odd} = \text{odd} \\
1 & \text{odd} = \text{even} 
\end{cases} \]

\[
\begin{array}{c|c|c|c}
\text{z} & \text{odd} & \text{even} \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]
Example (1): Design three bits Control Inverter.

Solution:

1. If \( z = 0 \), \( x = A \), \( y = B \), \( z = C \)

2. If \( z = 1 \), \( x = \bar{A} \), \( y = \bar{B} \), \( z = \bar{C} \)

3. Binary to Gray and Gray to Binary

- Binary to Gray

Example (1): Design three bits Binary to Gray

Solution:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Gray</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>100</td>
<td>111</td>
</tr>
<tr>
<td>101</td>
<td>100</td>
</tr>
<tr>
<td>110</td>
<td>101</td>
</tr>
<tr>
<td>111</td>
<td>110</td>
</tr>
</tbody>
</table>

Diagram for Binary to Gray conversion:
b- Gray to Binary

\[ \text{Gray} \quad \rightarrow \quad \text{Binary} \]

e.g. (1) - Design three bits Gray to binary

\[
\begin{array}{c|ccc|cccc}
\text{Gray} & x_1 & x_2 & x_3 & A & B & C \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]

4- Comparator

e.g. (1) - Design two bits Comparator.

\[
\begin{array}{c|cc|ccc}
A & B & A > B & A < B & A = B \\
\hline
0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 \\
\end{array}
\]
\[ A \geq B = \overline{A} \overline{B} \quad A \leq B = \overline{A} \overline{B} \quad A = B = \overline{A} \overline{B} + \overline{A} \overline{B} = A \oplus B \]

\[ A > B \]
\[ A < B \]
\[ A = B \]

E.g. (2). Design between \( A (A_1 A_0) \), \( B (B_1 B_0) \) Comparator.

\[ A > B = A_1 B_1 \text{ or } A_0 B_0 \text{ AND } A_1 = B_1 \]
\[ = A_1 \overline{B}_1 + (A_0 \overline{B}_0) \cdot (A_1 \oplus B_1) \]

\[ A < B = A_1 \overline{B}_1 \text{ or } A_0 B_0 \text{ AND } A_1 = B_1 \]
\[ = \overline{A}_1 B_1 + (\overline{A}_0 B_0) \cdot (A_1 \oplus B_1) \]

\[ A = B = (A_1 = B_1) \text{ AND } (A_0 = B_0) \]
\[ = (A_1 \oplus B_1) \cdot (A_0 \oplus B_0) \]
1- Half-Adder

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>S</th>
<th>C</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>

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

2- Full-Adder

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

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

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

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

\[ S = A \cdot C + A \cdot \overline{D} \]

\[ S = A \oplus D \]

\[ S = A \oplus B \oplus C \]

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

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

\[ C = c (A \oplus B) + AB \]
3. Half Subtraction

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>D</th>
<th>B</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>

\[ D = \bar{A}B + AB \]
\[ D = A \oplus B \]
\[ B = \bar{A}B \]

4. Full Subtraction

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

\[ D = \bar{A}BC + \bar{A}BC + \bar{A}BC + ABC \]
\[ D = A \oplus B \oplus C \]
\[ B = \bar{A}BC + \bar{A}BC + \bar{A}BC + ABC \]
\[ B = \bar{A}(BC + BC) + BC(\bar{A} + A) \]
\[ B = \bar{A}(B \oplus C) + BC \]

Block diagram:

Full Subtractor

| Block diagram | Half-subtractor | Half-subtractor | Full-subtractor | Block diagram |
Parallel Binary Adders

\[ A_1 \cdot A_0 \iff A \quad + B_1 \cdot B_0 \iff B \quad \text{e.g.} \quad 01 \]
\[ S_2 \quad S_1 \quad S_0 \quad \text{e.g.} \quad 00 \]

**Example 1:** Design block diagram of a three-bit parallel adder.

**Example 2:** Design block diagram of a three-bit parallel subtractor.
e.g. (3): Design block diagram of a three bits parallel adder/subtractor.

![Block diagram of a three bits parallel adder/subtractor.]

SOL.

e.g. (4): Considered 4-bit, 2-binary $A = 1011$, $B = 0011$, what is the sum and carry of each circuit?

![Block diagram of four circuits with inputs and outputs.]

SOL.

e.g. (5): Add two number, each consist two bits using half adder and full adder.

![Block diagram showing the addition of two numbers using half and full adders.]

SOL.
A decoder is a circuit that converts information from (n) inputs to a maximum of \(2^n\) unique output lines.

Binary \(\rightarrow\) Decimal

inputs \(n\) \(\rightarrow\) Decoder \(\rightarrow\) outputs

*Example (1): Design \((2\times4)\) decoder.

<table>
<thead>
<tr>
<th>A B</th>
<th>D_0 D_1 D_2 D_3</th>
<th>OR</th>
<th>A B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>1 0 0 0</td>
<td></td>
<td>0 0</td>
<td>D_0 = \bar{A}\bar{B}</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1 0 0</td>
<td></td>
<td>0 1</td>
<td>D_1 = \bar{A}B</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0 1 0</td>
<td></td>
<td>1 0</td>
<td>D_2 = AB</td>
</tr>
<tr>
<td>1 1</td>
<td>0 0 0 1</td>
<td></td>
<td>1 1</td>
<td>D_3 = A\bar{B}</td>
</tr>
</tbody>
</table>

![Diagram of (2x4) decoder](image-url)
e.g. (2). Design half adder using decoder.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>S</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>D₀</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>D₁</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>D₂</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>D₃</td>
</tr>
</tbody>
</table>

\( A \) \( B \) \( \overline{S} \) \( C \) \( (2 \times 4) \) \( \text{Dec.} \) 2 \( \bar{D}_0 \) \( D_1 \) \( D_2 \) \( D_3 \)

\( S \) \( C \)

\( e.g. (3). \) Using the decoder to find
\[ F = \begin{cases} 0, 1, 4, 6, 7 \end{cases} \]

\( S o l.: \) Maximum number (7) \( \Rightarrow \) using three bits \((A, B, \text{and} C)\):

<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>1</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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( (8 \times 8) \) \( \text{Dec.} \)

\( A \) \( B \) \( C \) \( F \)

\( F = \frac{1}{7} (5, 9, 2) \)

\( e.g. (4). \) Using the decoder to find
\[ F = \frac{1}{7} (1, 5, 9, 2) \]
Design a (3×8) decoder using (2×4) decoder.

**Solution:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>D₀</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D₁</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>D₂</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>D₃</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>D₄</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>D₅</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>D₆</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>D₇</td>
</tr>
</tbody>
</table>

1. If $A = 0$, Dec. (1) on; Dec. (2) off.
2. If $A = 1$, Dec. (1) off; Dec. (2) on.

**Encoders**

An encoder is a circuit that has $(2^n)$ or less input lines and $(n)$ output lines.

Decimal $\rightarrow$ Binary

Inputs $(2^n)$ $\rightarrow$ Encoder $\rightarrow$ Outputs $(n)$
e.g. (1). Design (4x2) Encoders.

<table>
<thead>
<tr>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</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>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

OR

<table>
<thead>
<tr>
<th>D</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>D1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>D2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>D3</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ A = D_2 + D_3 \quad \land \quad B = D_1 + D_3 \]

\[ \text{H.W} \]

e.g. (2). Design (8x3) Encoder.
**Multiplexer**

A multiplexer (mux) is a device that allows information from one of many input lines and directs it to output lines. It also has select lines.

\[
\text{number of inputs} = 2 \quad \Rightarrow \quad (N = 2)
\]

\[
\begin{align*}
\text{e.g. } & N = 2, \quad 2 = 2 \quad \Rightarrow \quad n = 1 \\
\text{e.g. } & N = 4, \quad 2 = 2^2 \quad \Rightarrow \quad n = 2 \\
\text{e.g. } & N = 8, \quad 2 = 2^3 \quad \Rightarrow \quad n = 3
\end{align*}
\]

\[
\text{e.g. (1). Design (2x1) multiplexer.} \quad \Rightarrow \quad N = 2, \quad 2^1 = 2, \quad \Rightarrow \quad n = 1
\]

<table>
<thead>
<tr>
<th>A</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D₀</td>
</tr>
<tr>
<td>1</td>
<td>D₁</td>
</tr>
</tbody>
</table>

\[
\text{if } A = 0, \quad F = D₀ \quad \text{if } A = 1, \quad F = D₁
\]
e.g (2): Design a (4x1) multiplexer.

Solution:
\[ N = 4, \quad 2^2 = 4 \implies n = 2 \]

\[
\begin{array}{c|c|c}
A & B & F \\
0 & 0 & D_0 \\
0 & 1 & D_1 \\
1 & 0 & D_2 \\
1 & 1 & D_3 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
A & B & C & F \\
\hline
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 \\
\end{array}
\]

e.g (3): Implement the following function using multiplexer. \( F = \overline{A}C + B\overline{C} + ABC \)

Solution:

\[
\begin{array}{c|c|c}
D_0 & D_1 & F \\
\hline
0 & 0 & D_0 \\
0 & 1 & D_1 \\
1 & 0 & D_2 \\
1 & 1 & D_3 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
D_4 & D_5 & D_6 & D_7 & F \\
\hline
0 & 0 & 0 & 0 & \text{mux} \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 \\
\end{array}
\]
Implement the following function using multiplexer, using (4x1) multiplexer.
\[ F = \bar{A}C + B\bar{C} + AB\bar{C} \]

**Solution:**
1. Let \((A, B)\) select line

\[
\begin{array}{ccc|c}
A & B & C & F \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 \\
\end{array}
\]

2. Let \((B, C)\) select line

\[
\begin{array}{ccc|c}
& D_0 & 4x1 & F \\
& D_1 & \text{MUX} & \\
& D_2 & \\
& D_3 & \\
A & 0 & & \\
\bar{A} & 1 & & \\
B & 0 & & \\
C & 1 & & \\
\end{array}
\]
A demultiplexer basically reverses multiplexing function. It takes data from one line and distributes them to a given number of output lines.

\[ F \rightarrow \text{Demux} \rightarrow D_0, D_1, \ldots, D_n \]

\[ \text{Select line} \]

\[ \begin{array}{c|c|c|c}
A & D_0 & D_1 \\
\hline
0 & F & 0 \\
1 & 0 & F \\
\end{array} \]

\[ F = \sum (1, 3, 7) \]

\[ \begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c}
A & B & C & F \\
\hline
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
\end{array} \]
Digital electronics involves circuits and systems in which there are only two possible states. These states are represented by two different voltage levels: a HIGH and a LOW.

The two states also be represented by current levels. The two states number system is called binary, and its two digits are 0 and 1. The two digits in the binary system, 1 and 0, are called bits.

**Positive Logic levels**

The system in which a 1 is represented by a HIGH and a 0 is represented by a LOW:

\[ \text{HIGH} = 1 \text{ and Low} = 0 \]

**Negative Logic levels**

The system in which a 1 is represented by a Low, and a 0 is represented by a HIGH:

\[ \text{HIGH} = 0 \text{ and Low} = 1 \]

**Logic levels**

The voltage used to represent a 1 and a 0 are called logic levels. Ideally, one voltage level represents a HIGH and another voltage level represents a LOW.
A HIGH can be any voltage between a specified minimum value and a specified maximum value. Likewise, a LOW can be any voltage as specified minimum and a specified maximum value. Figure 1 illustrates the general range of LOWs and HIGHs for a digital circuit.

The variable $V_H(\text{max})$ represents the maximum HIGH voltage value, and $V_L(\text{min})$ represents the minimum HIGH voltage value. The minimum LOW voltage value is represented by $V_L(\text{min})$, and the minimum LOW voltage value is represented by $V_H(\text{max})$. The voltage value between $V_L(\text{max})$ and $V_H(\text{min})$ are unacceptable for proper operation. A voltage in the unacceptable range can appear as either a HIGH or a LOW to a digital circuit. Therefore, these unacceptable values are never used.

For example, the HIGH values for a certain type of digital circuit called TTL may range from 2V to 5V and the low values may range from 0V to 0.8V. So, for example, if a voltage of 2.5V is applied, the circuit will accept it as a HIGH or binary 1.

If a voltage of 0.5V is applied, the circuit will accept it as a LOW or binary 0. For this type of circuit, voltages between 0.8V and 2V are unacceptable and are never used.
Propagation Delay

This parameter is a result of the limitation on switching speed or frequency at which a logic circuit can operate. The terms low speed and high speed applied to logic circuit refer to the propagation delay time. The shorter the propagation delay, the higher the speed of the circuit and the higher the frequency at which it can operate.

i.e. Propagation delay It is the time required to execute an operation on the gate.

\[ t_P = \frac{t_{PHL} + t_{PLH}}{2} \]

Propagation time delay.

Fan-out The maximum number of a similar gate that a gate can drive and still functionally properly.

\[ \text{N-Similar Gates,} \]
Power Dissipation

A logic gate draws current from the dc supply voltage source, as indicated in Figure (2). When the gate is in HIGH output state, an amount of current designated by \( I_{CH} \) is drawn, and in the low output state, a different amount of current, \( I_{CL} \) is drawn.

\[ \text{Fig. (2) currents from the dc supply} \]

As an example, if \( I_{CH} \) is specified as 1.5mA when \( V_{CC} \) is 5V and if the gate is in a static (unchanging) HIGH output state, the power dissipation \( (P_d) \) of the gate is

\[ P_d = V_{CC} I_{CH} = (5V)(1.5mA) = 7.5mW \]

When a gate is pulsed, its output switches back and forth between HIGH and LOW, and the amount of supply current variable between \( I_{CH} \) and \( I_{CL} \). The average power dissipation depends on the duty cycle and is usually specified for a duty cycle of 50%. 

\[ \text{Figure a) and b) show the current flow in the gate.} \]
when the duty cycle is 50%, the output is HIGH half the time and LOW the other half.

The average supply current is therefore

\[ I_{cc} = \frac{I_{ccH} + I_{ccL}}{2} \]

The average power dissipation is

\[ P_d = V_{cc} I_{cc} \]

**Example:** A certain gate draws 2.4 mA when its output is HIGH and 3.6 mA when its output is LOW. What is its average power dissipation if \( V_{cc} \) is 5 V and the gate is operated on a 50% duty cycle?

**Solution:**

The average \( I_{cc} \) is

\[ I_{cc} = \frac{I_{ccH} + I_{ccL}}{2} = \frac{2.4\text{ mA} + 3.6\text{ mA}}{2} = 2.8\text{ mA} \]

The average power dissipation is

\[ P_d = V_{cc} I_{cc} = (5\text{ V})(2.8\text{ mA}) = 14\mu\text{W} \]

**Noise Margin**

Noise is unwanted voltage that is induced in electrical circuits and can present a threat to the proper operation of the circuit. In order not to be adversely affected by noise, a logic circuit must have a certain amount of noise.
This is the ability to tolerate a certain amount of uninvited voltage fluctuation on its inputs without changing its output state, as shown in Fig. (3).

(a)

(b)

Fig. (3) Illustration of the effects of input noise on gate operation.

There are two values of noise margin specified for a given logic circuit. The HIGH level noise margin ($V_{IH}$) and low level noise margin ($V_{IL}$).
Determine the HIGH-level and LOW-level noise margins for CMOS and for TTL by using the information below.

**For CMOS**

\[
\begin{align*}
V_{IH}(\text{min}) &= 3.5 \text{V} \\
V_{IL}(\text{max}) &= 1.5 \text{V} \\
V_{OH}(\text{min}) &= 4.4 \text{V} \\
V_{OL}(\text{max}) &= 0.33 \text{V}
\end{align*}
\]

**For TTL**

\[
\begin{align*}
V_{IH}(\text{min}) &= 2 \text{V} \\
V_{IL}(\text{max}) &= 0.8 \text{V} \\
V_{OH}(\text{min}) &= 2.4 \text{V} \\
V_{OL}(\text{max}) &= 0.4 \text{V}
\end{align*}
\]

**Solution**

**1. For CMOS**

\[
\begin{align*}
V_{NH} &= V_{OH}(\text{min}) - V_{IH}(\text{min}) \\
&= 4.4 \text{V} - 3.5 \text{V} = 0.9 \text{V}
\end{align*}
\]

\[
V_{NL} = V_{IL}(\text{max}) - V_{OL}(\text{max})
\]

\[
V_{NL} = 1.5 \text{V} - 0.33 \text{V} = 1.17 \text{V}
\]

**2. For TTL**

\[
\begin{align*}
V_{NH} &= V_{OH}(\text{min}) - V_{IH}(\text{min}) \\
&= 2.4 \text{V} - 2 \text{V} = 0.4 \text{V}
\end{align*}
\]

\[
V_{NL} = V_{IL}(\text{max}) - V_{OL}(\text{max})
\]

\[
V_{NL} = 0.8 - 0.4 = 0.4 \text{V}
\]
These parameters are defined by the following equations:

\[ V_{\text{RH}} = V_{\text{OH}}(\text{min}) - V_{\text{IH}}(\text{min}) \]

\[ V_{\text{NL}} = V_{\text{IL}}(\text{max}) - V_{\text{OL}}(\text{max}) \]

From equations, \( V_{\text{RH}} \) is difference between the lowest possible HIGH output and the lowest possible HIGH input, \( V_{\text{NL}} \) is difference between the maximum possible low input and the maximum possible low output. Noise margins are illustrated in Figure (4).

The voltage on this line will never be less than \( V_{\text{RH}} \) unless noise or improper operation is introduced.

(a) High-level noise margin

(b) Low-level noise margin

For example, published noise margins values are for 5V CMOS, but
Transistor-transistor logic (TTL or T²L) is one of the most widely used integrated circuit technologies. The basic element in TTL circuit is the bipolar transistor. TTL has replaced earlier circuit technologies such as resistor-transistor logic (RTL) and diode-transistor logic (DTL).

ECL (Emitter-coupled logic), ECL is much faster than TTL because the transistor does not operate in saturation and is used in more high-speed applications.

**TTL Inverter Circuit**

![TTL Inverter Circuit Diagram](image)
TTL NAND Gate Circuit

TTL NOR Gate Circuit