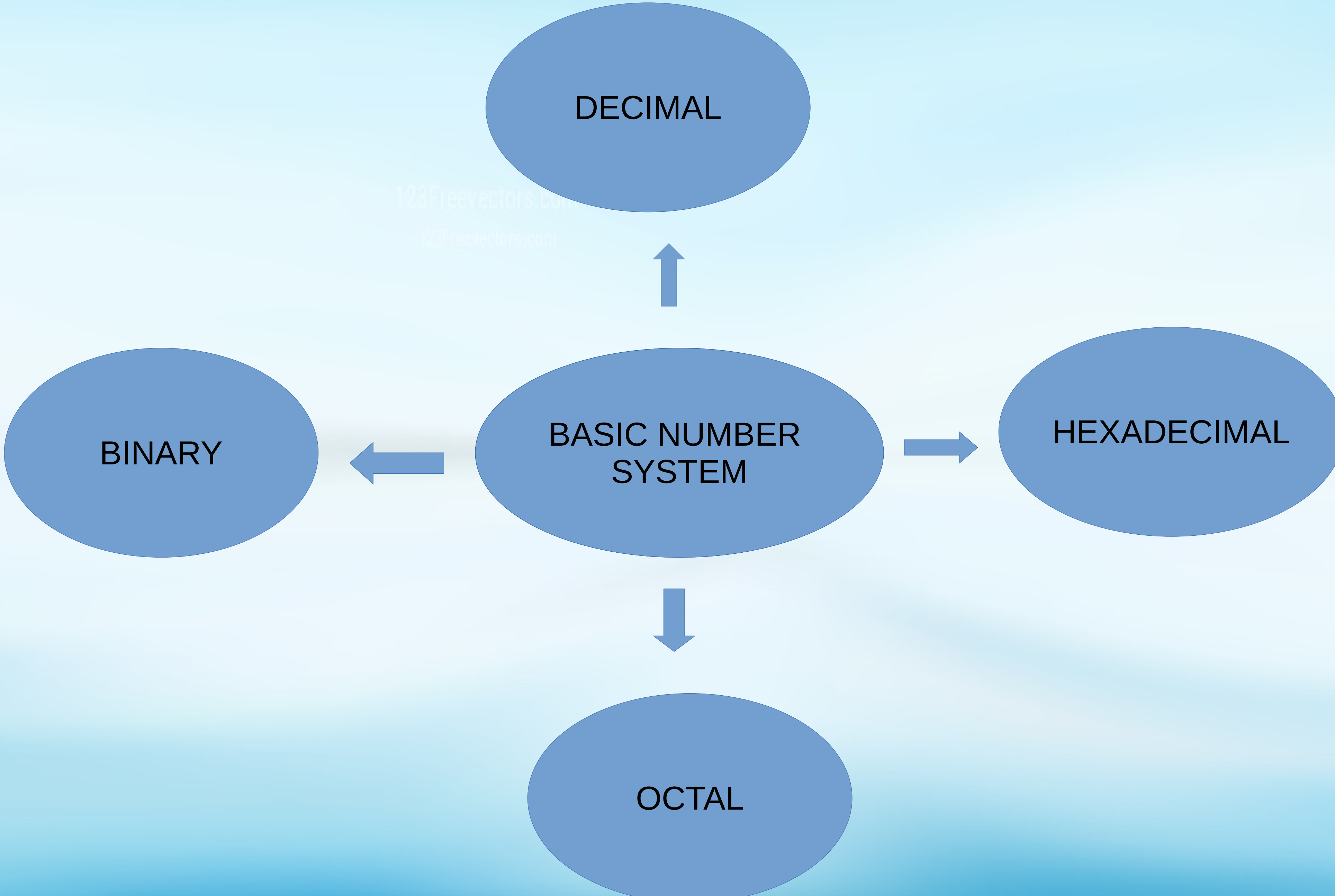
Digital Electronics

Dr Anjan Kumar Chandra
Assistant Professor
Department of Physics
Ramakrishna Mission Vivekananda Centenary College

- A computer understands 0 and 1
- Binary digits


DECIMAL
BINARY
BASIC NUMBER SYSTEM
HEXADECIMAL
OCTAL

DECIMAL NUMBER SYSTEM

DIGITS	0,1,2,3,4,5,6,7,8,9
BASE	10

Weight	10^4	10^3	10^2	10^1	10^0
Value	2	5	3	6	7

→ 25367

BINARY NUMBER SYSTEM

DIGITS 0,1

BASE 2

Weight	2^4	2^3	2^2	2^1	2^0
Value	1	0	1	1	0

→ $0*2^0 + 1*2^1 + 1*2^2 + 0*2^3 + 1*2^4 = 21$

OCTAL NUMBER SYSTEM

DIGITS: 0,1,2,3,4,5,6,7

BASE: 8

Weight	8^4	8^3	8^2	8^1	8^0
Value	2	5	3	6	7

25367 → $2*8^4 + 5*8^3 + 3*8^2 + 6*8^1 + 7*8^0$

HEXADECIMAL NUMBER SYSTEM

DIGITS

0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F

BASE

16

Weight	16^4	16^3	16^2	16^1	16^0
Value	A	5	C	6	7

A5C67 → $10*16^4 + 5*16^3 + 12*16^2 + 6*16^1 + 7*16^0$

BINARY TO DECIMAL

10101

BINARY NUMBER

Multiply the binary values (0/1) with their positional weightage and then sum

Weight	2^4	2^3	2^2	2^1	2^0
Value	1	0	1	0	1

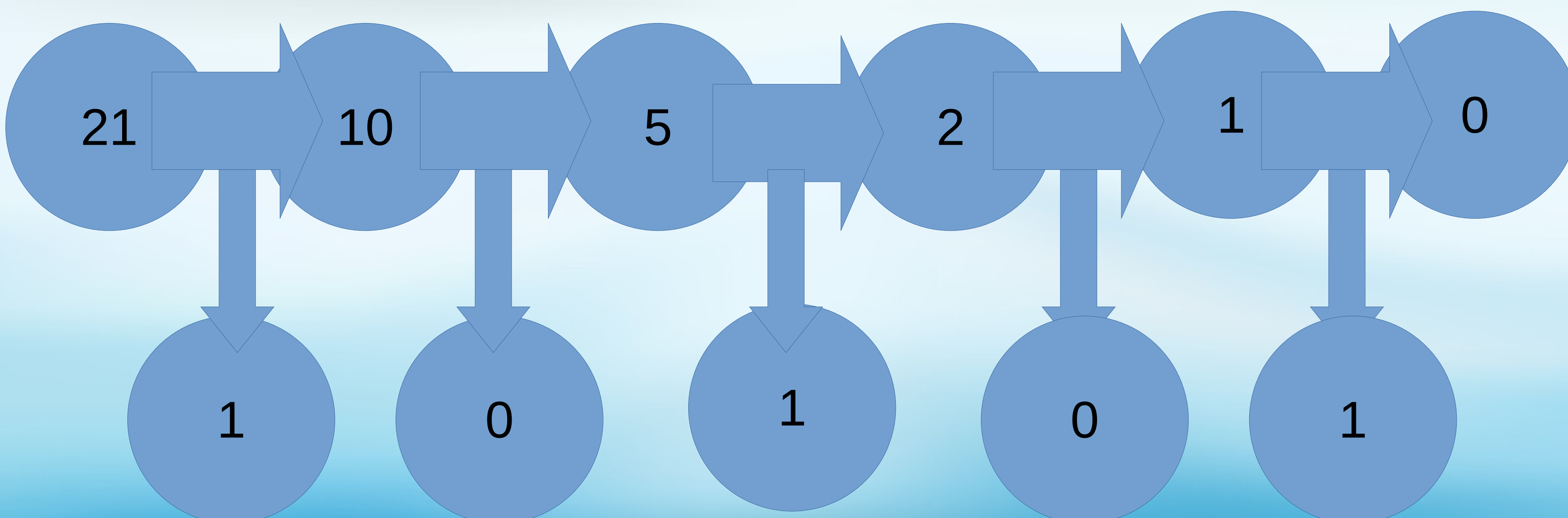
10101 → $1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 16 + 0 + 4 + 0 + 1 = 21$

DECIMAL TO BINARY

21

DECIMAL NUMBER

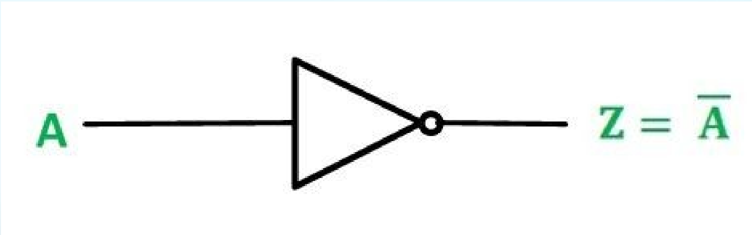
Divide the decimal number by 2 until the final result equals 0



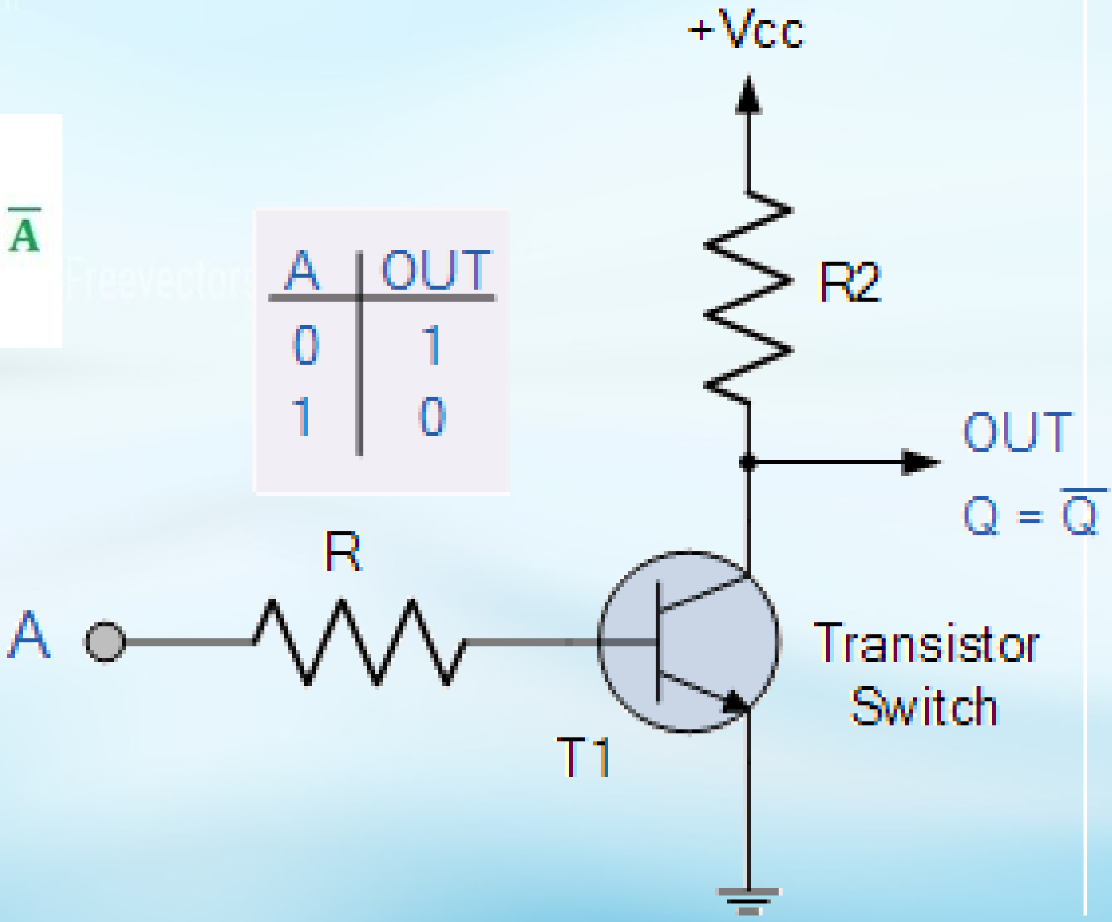
LOGIC GATES

- Logic gates are the fundamental building blocks of integrated circuits.
- They perform basic logical functions.
- They take input as 1 or 0 and gives output as 1 or 0 following certain logical rules.
-

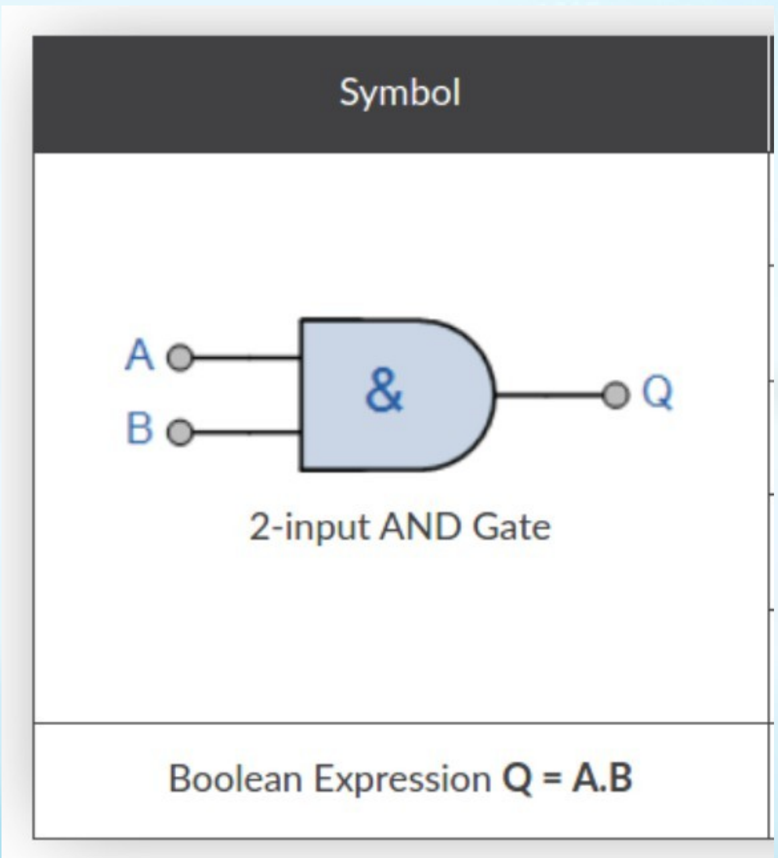
NOT GATE

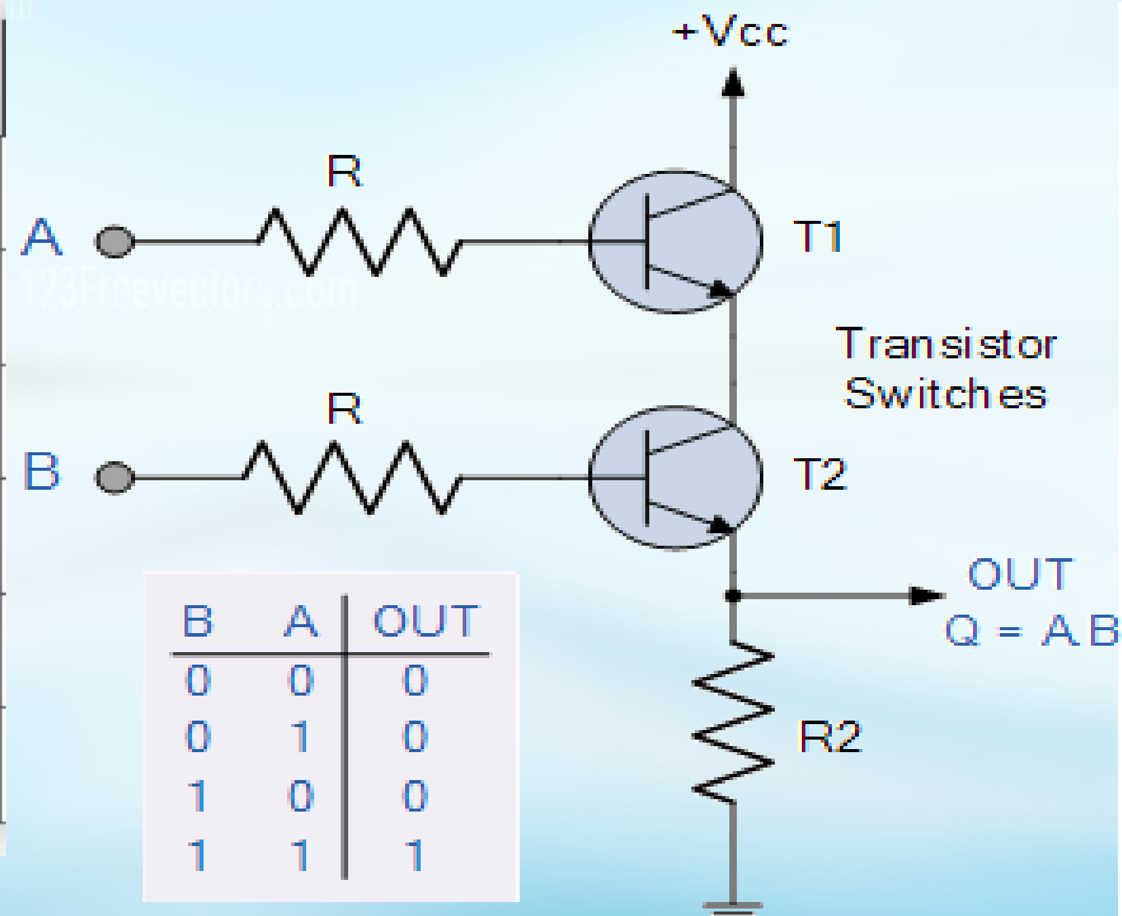


A	OUT
0	1
1	0



AND GATE

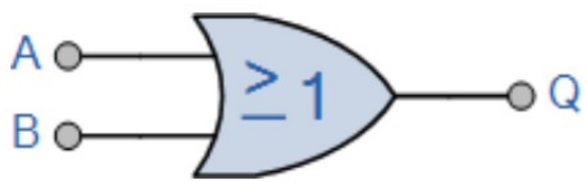




B	A	OUT
0	0	0
0	1	0
1	0	0
1	1	1

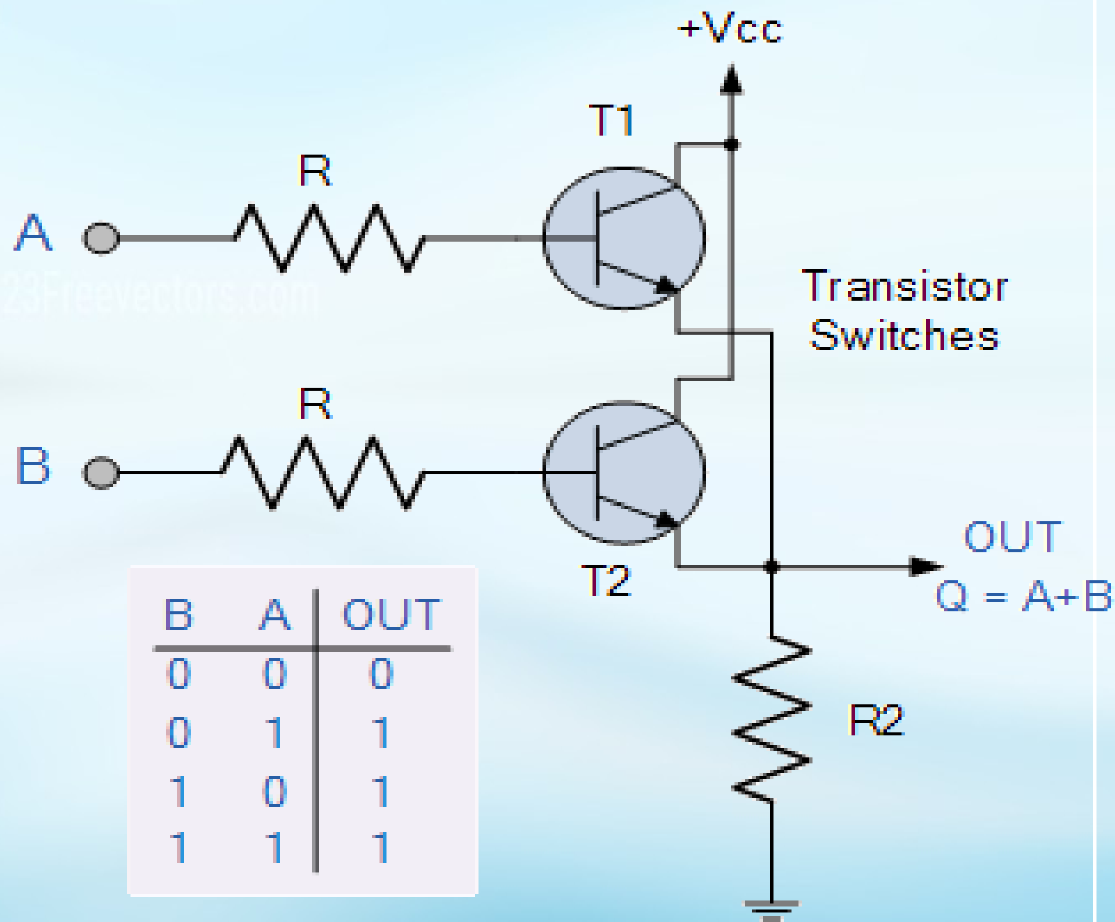
OR GATE

Symbol



2-input OR Gate

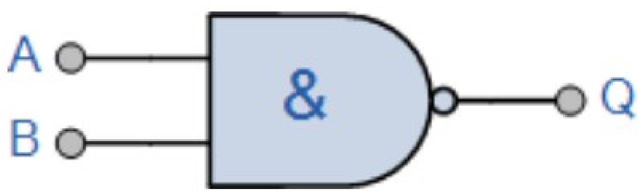
Boolean Expression **Q = A+B**



B	A	OUT
0	0	0
0	1	1
1	0	1
1	1	1

NAND GATE

Symbol



2-input NAND Gate

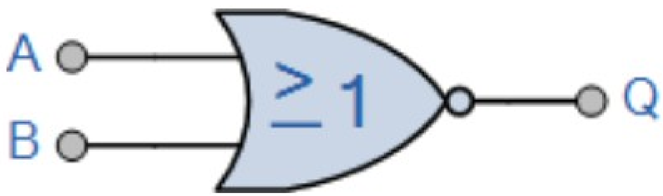
Truth Table

B	A	Q
0	0	1
0	1	1
1	0	1
1	1	0

Boolean Expression **Q** = $\overline{\mathbf{A.B}}$

Read as A **AND** B gives **NOT** Q

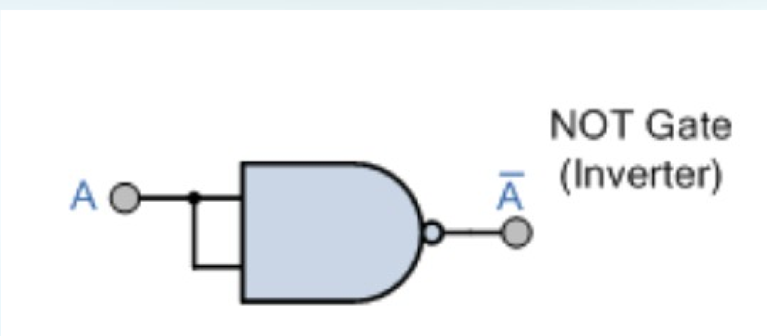
NOR GATE

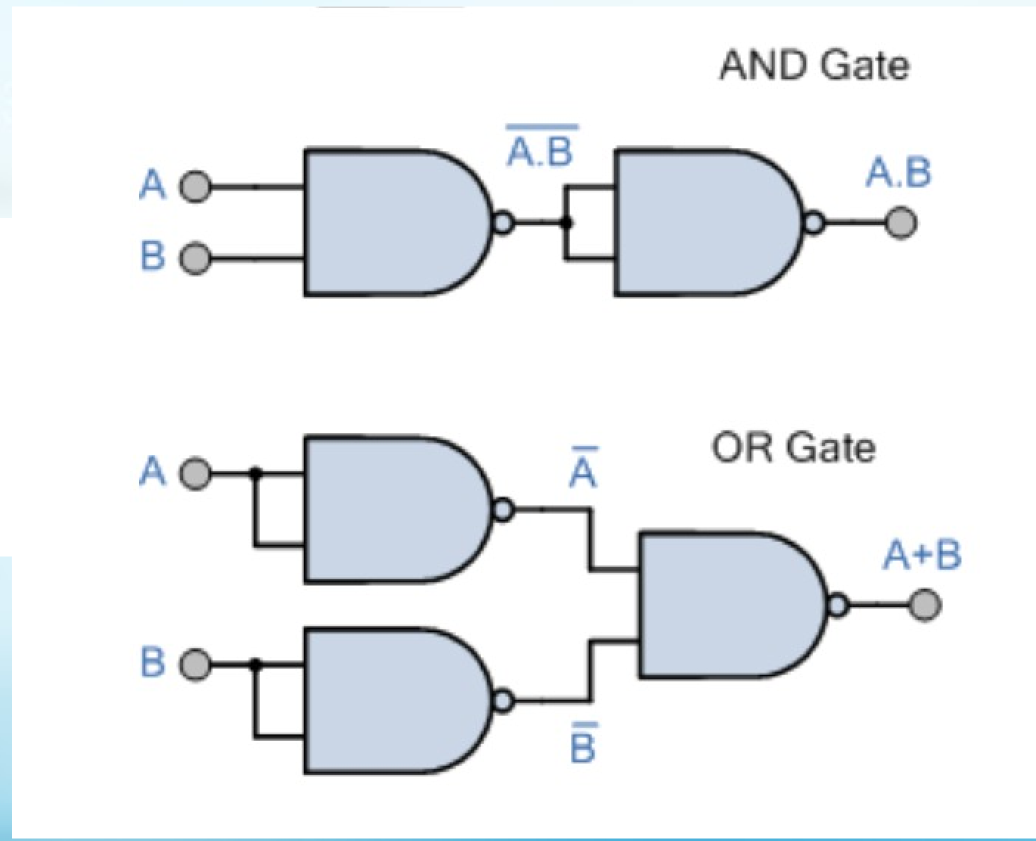
<table>
<tr><th>Symbol</th><th colspan="3">Truth Table</th></tr>
<tr><td rowspan="5">

2-input NOR Gate</td><td>B</td><td>A</td><td>Q</td></tr>
<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>
<tr><td>Boolean Expression Q = $\overline{A+B}$</td><td colspan="3">Read as A OR B gives NOT Q</td></tr>
</table>

NAND AS UNIVERSAL GATE

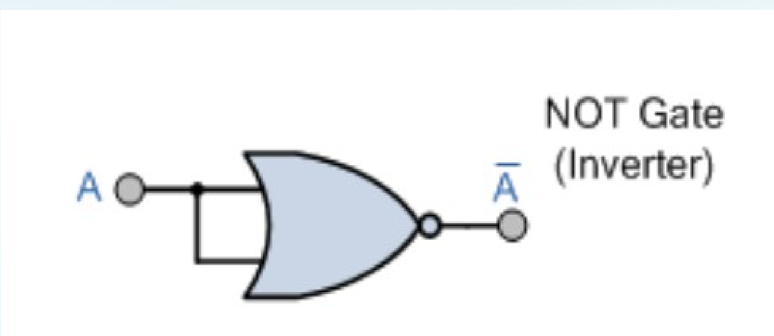
We can realise all of the other Boolean functions and gates by using just one single type of universal logic gate, the NAND (NOT AND) gate.

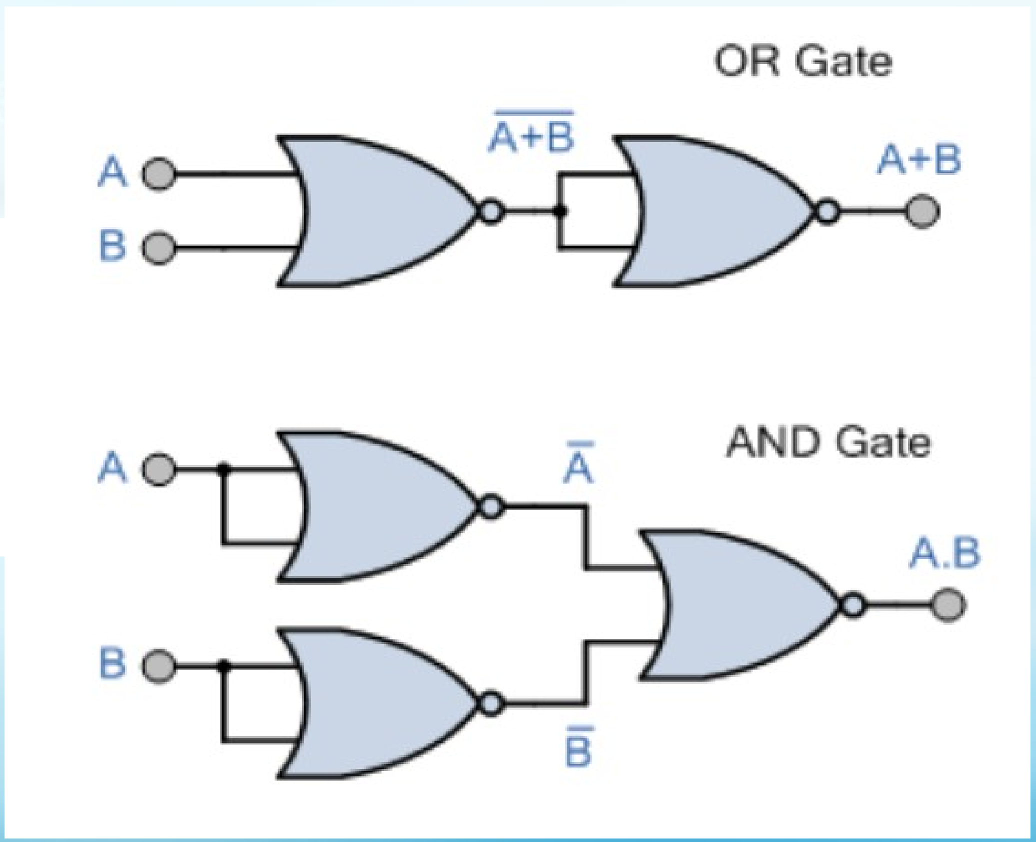




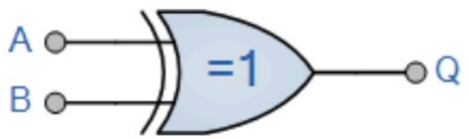
NOR AS UNIVERSAL GATE

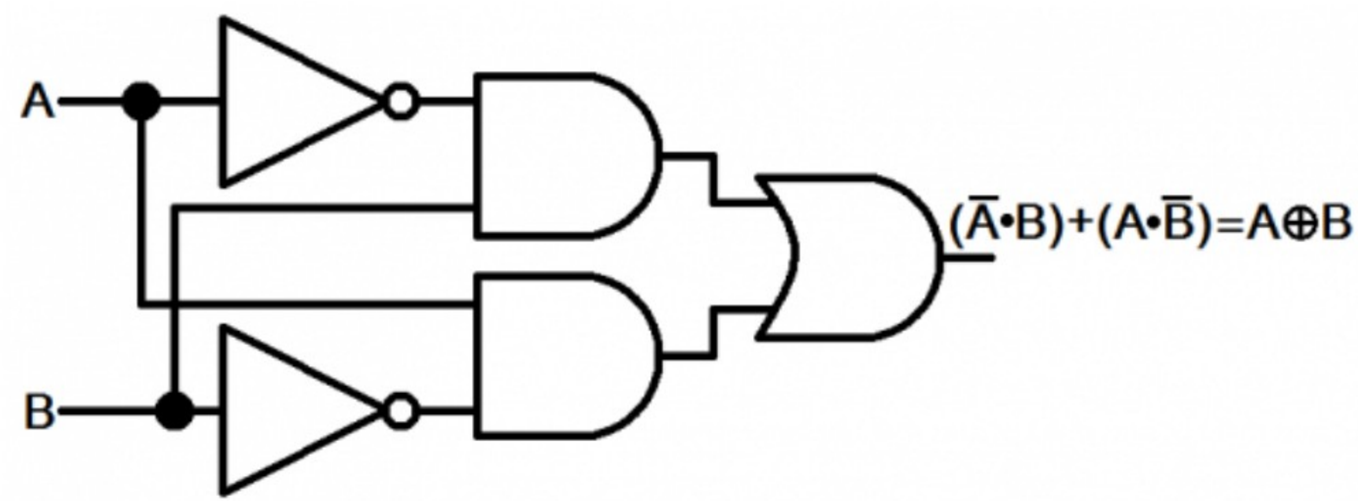
We can realise all of the other Boolean functions and gates by using just one single type of universal logic gate, the NOR (NOT OR) gate.





EXCLUSIVE-OR (XOR) GATE

Symbol	Truth Table		
	B	A	Q
 2-input Ex-OR Gate	0	0	0
	0	1	1
	1	0	1
	1	1	0
Boolean Expression Q = A ⊕ B	A **OR** B but NOT **BOTH** gives Q		

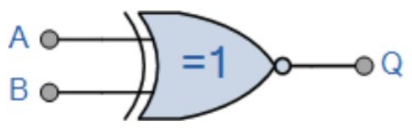


PARITY CHECKER

- If the sum of the binary bits in a word is odd (even), the word is said to have odd (even) parity.
- The circuit that checks this parity is called parity checker.
-

EXCLUSIVE-NOR (XNOR) GATE

Symbol	Truth Table



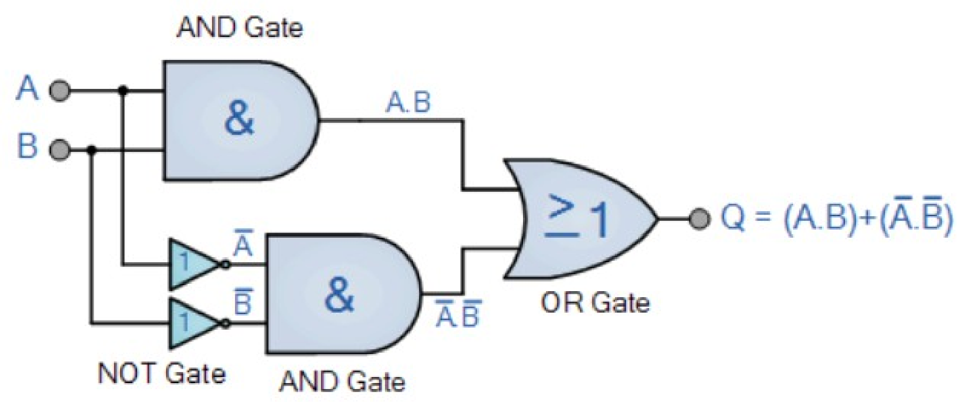
2-input Ex-NOR Gate

B	A	Q
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Expression Q = $\overline{A \oplus B}$

Read if A **AND** B the **SAME** gives Q

Ex-NOR Gate Equivalent Circuit



BOOLEAN ALGEBRA

- *In 1854, Logical algebra was published by George Boole, today known as Boolean algebra*
- *Only two states or values of a variable are allowed -*

  1 and 0 (corresponding to "ON" and "OFF" states in a electronic circuit)
- *In 1938, Claude Shannon was the first to apply Boole's work to the analysis and design of logic circuits.*

BASIC OPERATIONS

- *OR addition (indicated by a "+" sign) :*

$Y = A + B$ → Y equals A or B (Y is 'ON' when any of the inputs is 'ON')

If A = 0 and B = 0, then Y = 0 (0 + 0 = 0)

If A = 0 and B = 1, then Y = 1 (0 + 1 = 1)

If A = 1 and B = 0, then Y = 1 (1 + 0 = 1)

If A = 1 and B = 1, then Y = 1 (1 + 1 = 1)

In general, if there are N inputs and one output, then the output is 1, if any of the N inputs is in 1 state.

This operation is accomplished by a OR gate.

BASIC OPERATIONS

- *AND multiplication (indicated by a "." sign) :*

  Y = A.B → Y equals A or B (Y is 'ON' when any of the inputs is 'ON')

  If A = 0 and B = 0, then Y = 0 (0.0 = 0)

  If A = 0 and B = 1, then Y = 1 (0.1 = 1)

  If A = 1 and B = 0, then Y = 1 (1.0 = 1)

  If A = 1 and B = 1, then Y = 1 (1.1 = 1)

  In general, if there are N inputs and one output, then the output is 1, if all of the N inputs is in 1 state.

  This operation is accomplished by a AND gate.

BASIC OPERATIONS

- *AND multiplication (indicated by a "." sign) :*

$Y = A.B$ → Y equals A or B (Y is 'ON' when any of the inputs is 'ON')

If $A = 0$ and $B = 0$, then $Y = 0$ ($0.0 = 0$)

If $A = 0$ and $B = 1$, then $Y = 1$ ($0.1 = 1$)

If $A = 1$ and $B = 0$, then $Y = 1$ ($1.0 = 1$)

If $A = 1$ and $B = 1$, then $Y = 1$ ($1.1 = 1$)

In general, if there are N inputs and one output, then the output is 1,

if all of the N inputs is in 1 state.

This operation is accomplished by a AND gate.

DE MORGAN’S THEOREM

- De Morgan’s first theorem :

If A and B are two variables, then

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

Proof :

A	B	$\overline{A + B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

DE MORGAN’S THEOREM

- De Morgan’s second theorem :

If A and B are two variables, then

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

Proof :

A	B	$\overline{A.B}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

- *NOT operation (indicated by a "‾" sign) :*

  $Y = \overline{A}$ → Y equals complement of A (Y is 'OFF' (ON) when A is 'ON' OFF)

  If A = 0 and Y = 1

  If A = 1 and Y = 0

  This operation is accomplished by a NOT gate.

BINARY ADDITION

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

$$\begin{array}{r} 1\;0^{1}\;0^{1}\;1 \\ +\;\;1\;0\;1\;1 \\ \hline 1\;0\;1\;0\;0 \end{array}$$

$$\begin{array}{r} {}^{1}\;1^{1}\;1^{1}\;1 \\ +\;\;1\;1\;0\;1 \\ \hline 1\;0\;1\;0\;0 \end{array}$$

BINARY SUBTRACTION

- 1’s complement of a binary number is obtained by changing each 0 of the number to 1 and each 1 to 0.

  1’s complement of 11001 is 00110.

- 2’s complement of a binary number is obtained by adding 1 to its 1’s complement.

  2’s complement of 11001 is 00110+1 = 00111.

BINARY SUBTRACTION by 1's complement

A) The 1's complement of the number to be subtracted is determined.

B) The 1's complement is added to the number from which the subtraction is desired.

C) When there is 1 carry in the last position of the result of addition in step B, the carry is added to the result without the carry to obtain the final result.

Let us subtract 1011 from 1101.

- Step A → The number to be subtracted is 1011

  1's complement of 1011 is 0100

- Step B →

$$\begin{array}{r} 1^{1}\,1\,0\,1 \\ +\;0\,1\,0\,0 \\ \hline 1\;0\,0\,0\,1 \end{array}$$

$$\begin{array}{r} 0\,0\,0\,1 \\ +\qquad 1 \\ \hline 0\,0\,1\,0 \end{array}$$

← Step C

BINARY SUBTRACTION by 2's complement

A) The 2's complement of the number to be subtracted is determined.

B) The 2's complement is added to the number from which the subtraction is desired.

C) (i) If there is 1 carry in the last position of the result of addition in step B, the carry is just rejected to get the final result and the answer is positive.

(ii) If there is no carry in the last position of the result of addition in step B, the answer is negative. The 2's complement of the added result (got in C(i)) is found and a minus sign is assigned to get the final result.

BINARY SUBTRACTION by 2's complement

Let us subtract 1011 from 1101.

- Step A — The number to be subtracted is 1011

  2's complement of 1011 is 0101

- Step B →

$$\begin{array}{r} 1^{1}1\ 0^{1}\ 1 \\ \underline{+\ \ 0\ \ 1\ 0\ 1} \\ \color{red}{1}\ \ 0\ 0\ \ 1\ 0 \end{array}$$

0 0 1 0 ← Step C(i)

- Let us subtract 1101 from 1011.

  Step A — The number to be subtracted is 1101

  2's complement of 1101 is 0011

- Step B

$$\begin{array}{r} 1\ 0\ 1^{1}\ 1 \\ \underline{+\ \ 0\ 0\ 1\ \ 1} \\ 1\ 1\ 1\ \ 0 \end{array}$$

- 0 0 1 0 ← Step C(ii)

Basic laws of Boolean algebra

A) The Commutative laws :

Commutative law of addition : $A + B = B + A$

Commutative law of multiplication : $A.B = B.A$

B) The Associative laws :

Associative law of addition : $A + (B + C) = (A + B) + C$

Associative law of multiplication : $A(BC) = (AB)C$

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

Some basic relations of Boolean algebra

OR	AND
$A + 0 = A$	$A.0 = 0$
$A + 1 = 1$	$A.1 = A$
$A + A = A$	$A.A = A$
$A + \overline{A} = 1$	$A.\overline{A} = 0$

Prove : $A + AB = A$ and $A.(A + B) = A$

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

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

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

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

[Product of the sums (POS) = Sum of their products (SOP)]

Canonical and Standard Forms of Boolean functions

Two binary variables A and B can be combined with an AND operation as

$\overline{A}\,\overline{B}$, $\overline{A}\,B$, $A\,\overline{B}$, $A\,B$ $\longrightarrow$ Each of these terms is called a *minterm* or *standard product.*

- Each variable is overlined if it is a 0 and not overlined if it is 1.

In general *n* binary variables can be combined to give 2^n minterms.

Two binary variables A and B can be combined with an OR operation as

$\overline{A}+\overline{B}$, $\overline{A}+B$, $A+\overline{B}$, $A+B$ Each of these terms is called a *maxterm* or *standard sum.*

In general *n* binary variables can be combined to give 2^n maxterms.

- Each variable is not overlined if it is a 0 and overlined if it is 1.

Canonical and Standard Forms of Boolean functions

The symbol for a minterm is m_j and for a maxterm is M_j

where *j* represents the decimal equivalent of the concerned term.

- The three variable minterm $\overline{A}\ \overline{B}\ \overline{C}$ has the decimal equivalent 0.

  So its symbol is m_0

- The corresponding maxterm A + B + C is designated by M_0

  The minterm $\overline{A}$ B C has the decimal equivalent 3 (A=0, B=1, C=1)

  So it is denoted by m_3

- The corresponding maxterm $A + \overline{B} + \overline{C}$ is represented by M_3

Canonical and Standard Forms of Boolean functions

Binary variables A B C	Minterm	Minterm designation	Maxterm	Maxterm designation
0 0 0	$\bar{A}\,\bar{B}\,\bar{C}$	m_0	$A + B + C$	M_0
0 0 1	$\bar{A}\,\bar{B}\,C$	m_1	$A + B + \bar{C}$	M_1
0 1 0	$\bar{A}\,B\,\bar{C}$	m_2	$A + \bar{B} + C$	M_2
0 1 1	$\bar{A}\,B\,C$	m_3	$A + \bar{B} + \bar{C}$	M_3
1 0 0	$A\,\bar{B}\,\bar{C}$	m_4	$\bar{A} + B + C$	M_4
1 0 1	$A\,\bar{B}\,C$	m_5	$\bar{A} + B + \bar{C}$	M_5
1 1 0	$A\,B\,\bar{C}$	m_6	$\bar{A} + \bar{B} + C$	M_6
1 1 1	$A\,B\,C$	m_7	$\bar{A} + \bar{B} + \bar{C}$	M_7

Canonical and Standard Forms of Boolean functions

A Boolean function can be expressed from a truth table by

- forming a minterm of each combination of the variables giving a 1 in the function and
- Then taking the OR of all these terms.

Thus any Boolean function can be expressed as a sum of minterms.