

Chapter 3
**Boolean
Logic**

Computer Science
Class XI (As per CBSE Board)

Visit : python.mykvs.in for regular updates

Boolean Logic



What does a Computer Understands

Computers do not understand natural languages nor programming languages. They only understand the language of bits. A bit is the most basic unit in computer machine language. All instructions that the computer executes and the data that it processes is made up of a group of bits. Bits are represented in many forms either through electrical voltage, current pulses, or by the state of an electronic flip-flop circuit in form of 0 or 1.

1 Bit = Binary Digit(0 or 1)

8 Bits = 1 Byte

1024 Bytes = 1 KB (Kilo Byte)

1024 KB = 1 MB (Mega Byte)

1024 MB = 1 GB(Giga Byte)

1024 GB = 1 TB(Terra Byte)

1024 TB = 1 PB(Peta Byte)

1024 PB = 1 EB(Exa Byte)

1024 EB = 1 ZB(Zetta Byte)

1024 ZB = 1 YB (Yotta Byte)

1024 YB = 1 (Bronto Byte)

1024 Brontobyte = 1 (Geop Byte)



Boolean Logic

Because of computer understands machine language(0/1) which is binary value so every operation is done with the help of these binary value by the computer.

George Boole, Boolean logic is a form of algebra in which all values are reduced to either 1 or 0.

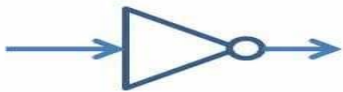
To understand boolean logic properly we have to understand Boolean logic rule, Truth table and logic gates

Boolean Logic

Boolean Logic :- NOT ,AND,OR,XOR,NAND,NOR operation can be understood through below table.Assume 0 as False and 1 as True.Not reverse the value 0 to 1 and 1 to 0,AND multiply value,OR add values,XOR Compare values return 0 in case same value else return 1

NOT

X	F
0	1
1	0



AND

X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

XOR

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

Boolean Logic



Boolean Logic :- NAND return the inverted value of AND operation and NOR return inverted value of OR operation

NAND

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

NOR

Input		Output
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Boolean Logic

Boolean Logic rules

Boolean Algebra is the mathematics we use to analyse digital gates and circuits. We can use these “Laws of Boolean” to both reduce and simplify a complex Boolean expression in an attempt to reduce the number of logic gates required.

Boolean Expression	Boolean Algebra Law or Rule
$A + 1 = 1$	Annulment
$A + 0 = A$	Identity
$A \cdot 1 = A$	Identity
$A \cdot 0 = 0$	Annulment
$A + A = A$	Idempotent
$A \cdot A = A$	Idempotent
$\text{NOT } \text{NOT } A = A$	Double Negation
$A + \bar{A} = 1$	Complement
$A \cdot \bar{A} = 0$	Complement
$A+B = B+A$	Commutative
$A \cdot B = B \cdot A$	Commutative
$\overline{A+B} = \bar{A} \cdot \bar{B}$	de Morgan's Theorem
$\overline{A \cdot B} = \bar{A} + \bar{B}$	de Morgan's Theorem



Boolean Expression

A Boolean expression is a logical statement that is either TRUE or FALSE .

A Boolean expression can consist of Boolean data, such as the following:

- * BOOLEAN values (YES and NO, and their synonyms, ON and OFF, and TRUE and FALSE)
 - * BOOLEAN variables or formulas
 - * Functions that yield BOOLEAN results
-
- BOOLEAN values calculated by comparison operators. E.g.
 1. $F(x, y, z) = x' y' z' + x y' z + x y z' + x y z$
 2. $F'(x, y, z) = x' y z + x' y' z + x' y z' + x y' z'$
 3. $F(x, y, z) = (x + y + z) \cdot (x+y+z')$. $(x+y'+z)$. $(x'+y+z)$



De Morgan's Law

The complement of the union of two sets is equal to the intersection of their complements and the complement of the intersection of two sets is equal to the union of their complements. These are called De Morgan's laws.

For any two finite sets A and B

(i) $(A \cup B)' = A' \cap B'$ (which is a De Morgan's law of union).

OR

$$(A+B)'=A'.B'$$

(ii) $(A \cap B)' = A' \cup B'$ (which is a De Morgan's law of intersection).

OR

$$(A . B)'=A'+B'$$

Boolean Logic



Proof of De Morgan's law: $(A \cup B)' = A' \cap B'$

Let $P = (A \cup B)'$ and $Q = A' \cap B'$

Let x be an arbitrary element of P then $x \in P \Rightarrow x \in (A \cup B)'$

$\Rightarrow x \notin (A \cup B)$

$\Rightarrow x \notin A$ and $x \notin B$

$\Rightarrow x \in A'$ and $x \in B'$

$\Rightarrow x \in A' \cap B'$

$\Rightarrow x \in Q$

Therefore, $P \subset Q$ (i)

Again, let y be an arbitrary element of Q then $y \in Q \Rightarrow y \in A' \cap B'$

$\Rightarrow y \in A'$ and $y \in B'$

$\Rightarrow y \notin A$ and $y \notin B$

$\Rightarrow y \notin (A \cup B)$

$\Rightarrow y \in (A \cup B)'$

$\Rightarrow y \in P$

Therefore, $Q \subset P$ (ii)

Now combine (i) and (ii) we get; $P \equiv Q$ i.e. $(A \cup B)' \equiv A' \cap B'$

Visit : python.mykvs.in for regular updates

Boolean Logic



Proof of De Morgan's law: $(A \cap B)' = A' \cup B'$

Let $M = (A \cap B)'$ and $N = A' \cup B'$

Let x be an arbitrary element of M then $x \in M \Rightarrow x \in (A \cap B)'$

$\Rightarrow x \notin (A \cap B)$

$\Rightarrow x \notin A$ or $x \notin B$

$\Rightarrow x \in A'$ or $x \in B'$

$\Rightarrow x \in A' \cup B'$

$\Rightarrow x \in N$

Therefore, $M \subset N$ (i)

Again, let y be an arbitrary element of N then $y \in N \Rightarrow y \in A' \cup B'$

$\Rightarrow y \in A'$ or $y \in B'$

$\Rightarrow y \notin A$ or $y \notin B$

$\Rightarrow y \notin (A \cap B)$

$\Rightarrow y \in (A \cap B)'$

$\Rightarrow y \in M$

Therefore, $N \subset M$ (ii)

Now combine (i) and (ii) we get; $M = N$ i.e. $(A \cap B)' = A' \cup B'$



Truth table

A truth table is a mathematical table used in logic.

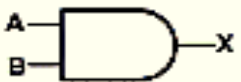

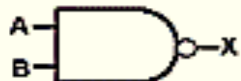



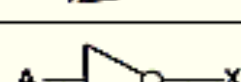
e.g.

A	B	(A and B)	(A or B)	not(A and B)	not(A or B)
True	True	True	True	False	False
True	False	False	True	True	False
False	True	False	True	True	False
False	False	False	False	True	True

Boolean Logic

Logic Gates

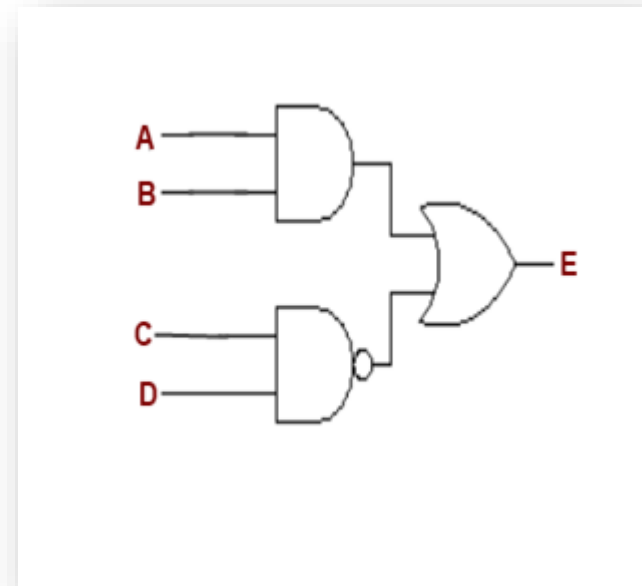
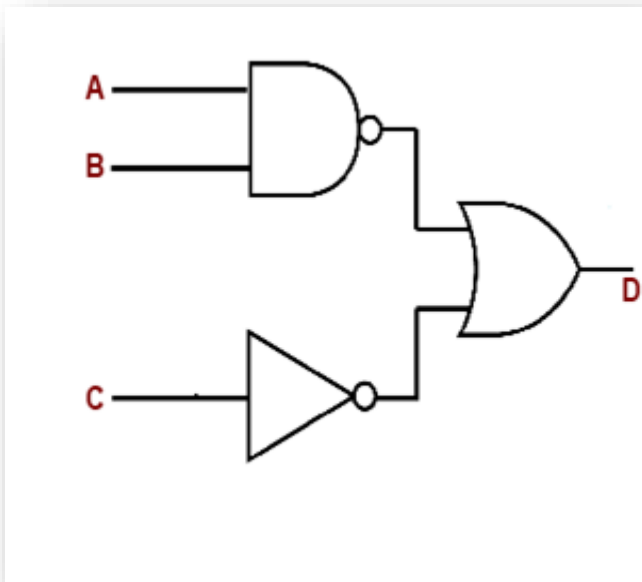
Logic gate is an idealized or physical device implementing a Boolean function. These are used to construct logic circuit.

LOGIC GATES		
Logic gate symbol	Description	Boolean
	The AND gate output is at logic 1 when, and only when all its inputs are at logic 1, otherwise the output is at logic 0.	$X = A \cdot B$
	The OR gate output is at logic 1 when one or more of its inputs are at logic 1. If all the inputs are at logic 0, the output is at logic 0.	$X = A + B$
	The NAND Gate output is at logic 0 when, and only when all its inputs are at logic 1, otherwise the output is at logic 1.	$X = \overline{A \cdot B}$
	The NOR gate output is at logic 0 when one or more of its inputs are at logic 1. If all the inputs are at logic 0, the output is at logic 1.	$X = \overline{A + B}$
	The XOR gate output is at logic 1 when one and ONLY ONE of its inputs is at logic 1. Otherwise the output is logic 0.	$X = A \oplus B$
	The XNOR gate output is at logic 0 when one and ONLY ONE of its inputs is at logic 1. Otherwise the output is logic 1. (It is similar to the XOR gate, but its output is inverted).	$X = \overline{A \oplus B}$
	The NOT gate output is at logic 0 when its only input is at logic 1, and at logic 1 when its only input is at logic 0. For this reason it is often called an INVERTER.	$X = \overline{A}$

Boolean Logic

Logic circuit

Construct a truth tables for following circuits of logic gates



Construct the logic circuit of following

1. $C + BC$:
2. $AB + BC(B + C)$



Universal gates are the logic gates which are capable of implementing any Boolean function without requiring any other type of gate.

Types of Universal Gates-

In digital electronics, there are only two universal gates which are-

1. NAND Gate
2. NOR Gate