MCA Part 1

Paper 2: Computer Organisation

Prepared by : Dr. Kiran Pandey

 School of Computer Science

Email-id: kiranpandey.nou@gmail.com

Lecture 03

Boolean Algebra and Logic Gates

Boolean Algebra is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1. It is also called as **Binary Algebra** or **logical Algebra**. Boolean algebra was invented by **George Boole** in 1854.

Rule in Boolean Algebra

Following are the important rules used in Boolean algebra.

* Variable used can have only two values. Binary 1 for HIGH and Binary 0 for LOW.
* Complement of a variable is represented by an overbar (-). Thus, complement of variable B is represented as . Thus if B = 0 then  = 1 and B = 1 then  = 0.
* ORing of the variables is represented by a plus (+) sign between them. For example ORing of A, B, C is represented as A + B + C.
* Logical ANDing of the two or more variable is represented by writing a dot between them such as A.B.C. Sometime the dot may be omitted like ABC.

Boolean Laws

There are six types of Boolean Laws.

Commutative law

Any binary operation which satisfies the following expression is referred to as commutative operation.



Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.

Associative law

This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.



Distributive law

Distributive law states the following condition.



AND law

These laws use the AND operation. Therefore they are called as **AND** laws.



OR law

These laws use the OR operation. Therefore they are called as **OR** laws.



INVERSION law

This law uses the NOT operation. The inversion law states that double inversion of a variable results in the original variable itself.



LOGIC GATES

Logic gates are the basic building blocks of any digital system. It is an electronic circuit having one or more than one input and only one output. The relationship between the input and the output is based on a **certain logic**. Based on this, logic gates are named as AND gate, OR gate, NOT gate etc.

AND Gate

A circuit which performs an AND operation is shown in figure. It has n input (n >= 2) and one output.



Logic diagram



Truth Table



OR Gate

A circuit which performs an OR operation is shown in figure. It has n input (n >= 2) and one output.



Logic diagram



Truth Table



NOT Gate

NOT gate is also known as **Inverter**. It has one input A and one output Y.



Logic diagram



Truth Table



NAND Gate

A NOT-AND operation is known as NAND operation. It has n input (n >= 2) and one output.



Logic diagram



Truth Table



NOR Gate

A NOT-OR operation is known as NOR operation. It has n input (n >= 2) and one output.



Logic diagram



Truth Table



XOR Gate

XOR or Ex-OR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-OR gate is abbreviated as EX-OR gate or sometime as X-OR gate. It has n input (n >= 2) and one output.



Logic diagram



Truth Table



XNOR Gate

XNOR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-NOR gate is abbreviated as EX-NOR gate or sometime as X-NOR gate. It has n input (n >= 2) and one output.



Logic diagram



Truth Table



De Morgan’s Theorem

De Morgan has suggested two theorems which are extremely useful in Boolean Algebra. The two theorems are discussed below.

Theorem 1



* The left hand side (LHS) of this theorem represents a NAND gate with inputs A and B, whereas the right hand side (RHS) of the theorem represents an OR gate with inverted inputs.
* This OR gate is called as **Bubbled OR**.



Table showing verification of the De Morgan's first theorem −



Theorem 2



* The LHS of this theorem represents a NOR gate with inputs A and B, whereas the RHS represents an AND gate with inverted inputs.
* This AND gate is called as **Bubbled AND**.



Table showing verification of the De Morgan's second theorem −

