Unit 2: Boolean algebra and Logic Gates

Logic Function and Boolean Algebra

Logic Function

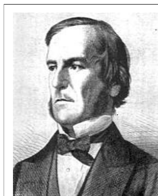
A digital circuit represents and manipulates information encoded as electric signals that can assume one of two voltages: logic high (V_{dd}) or logic low (GND). A digital circuit requires a power supply that can produce these two voltages, and these same supply voltages are also used to encode information in the form of two-state, or binary signals. Thus, if a given circuit node is at V_{dd}, then that signal is said to carry a logic '1'; if the node is at GND (Ground or earthing), then the node carries a logic '0'. The components in digital circuits are simple on/off switches that can pass logic '1' and logic '0' signals from one circuit node to another. Most typically, these switches are arranged to combine input signals to produce an output signal according to basic logic relationships.

The three primary or basic logic relationships, AND, OR, and NOT (or inversion) can be used to express any logical relationship between any number of variables. These simple logic functions form the basis for all digital electronic devices—from a simple microwave oven controller to a desktop PC. We can write a collection of logic equations of the form $F=A \cdot B$ (F equals A and B) that use these three relationships to specify the behaviour of any given digital system.

Logic equations provide an abstract model of actual logic circuits. They are used to show how an output logic signal should be driven in response to changes on one or more input signals. The equal sign ('=') is typically used as an assignment operator to indicate how information should flow through a logic circuit. For example, the simple logic equation 'F = A' specifies that the output signal F should be assigned whatever voltage is currently on signal A. Thus, the logic equation 'F = A' dictates that a change on the signal A will result in a change on the signal F.

Most logic equations specify an output signal that is some function of input signals. For example, the logic equation $F=A \cdot B$ specifies a logic circuit whose output will be driven to '1' only when both inputs are driven to '1'.

Introduction to Boolean Algebra


George Boole

In 1854, **George Boole** performed an investigation into the "laws of thought" which were based around a simplified version of the "group" or "set" theory, and from this Boolean Algebra was developed. Boolean algebra is algebra of logics. Boolean algebra is fundamental to computer operations. It was developed by George Boole. Initially, Boolean algebra was developed just to determine whether logical propositions are true or false. The variables in Boolean algebra can have only one of two values true or false. Abbreviated by T or F. We can also write 1 for T and 0 for F. No division, subtraction, exponent, coefficient, negative numbers are allowed in Boolean algebra.

Introduction to Boolean values

The two values of Boolean Algebra could represented in different forms in digital logical as below:

True	False
1	0
On	Off
High	Low
Yes	No

Unit 2: Boolean algebra and Logic Gates

Truth Table

A truth table is a table that shows the relationship between the Boolean values when used together with the Boolean operator and gives a particular output. A truth table is widely used for proving Boolean expression and show the relationship between the Boolean values. A truth table with N inputs requires 2^N rows to list all possible input combinations.

Boolean expression and Boolean function

A Boolean function is a Boolean expression that is a combination of Boolean variables and operators combined together to perform a particular task. The variables in a Boolean function as expression could be single variables or a complement of a variable. The output achieved from one function could be used as an input for another and so on to achieve any particular output.

F= (a + b).(b + c) ___________ (i)

F= (a.b).(b.c) ___________ (ii)

Laws (Identities) of Boolean algebra

If B is a non empty set then for a, b, c ∈ B we have following laws which satisfies in Boolean Algebra.

1. Commutative Law
2. Associative law
3. Distributive law
4. Idempotence law
5. Involution law
6. Boundedness law
7. Absorption law
8. Identity law
9. Complement law
10. De Morgan's law

Each of these laws are discussed as follows

1. Commutative Laws

This law says the order in which Boolean variables are added is immaterial

i. A + B = B + A

ii. A. B = B. A

where a and b are the Boolean variables

Proof of (i)

A	B	A+B	B+A
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

All the values of third column of above table is equal to values in fourth column. Hence (i) part of the law is proved.

Proof of (ii)

A	B	A.B	B.A
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

All the values of third column of the above table is equal to value in fourth column. Hence (ii) part of law is proved.

2. Associative law

This law says that in the process of logically multiplying three variables, A, B, C it is immaterial whether we do this by multiplying C to the product of A and B or by multiplying A to the product of B and C.

i.e. **A.(A.C) = (A.B).C**

A	B	C	A.B	B.C	A.(B.C)	(A.B)C
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	0

The values of column A.(B. C) is equal to (A. B).C. Hence the first statement of associative law is proved.

Also this law has another statement which says in the process of logically adding two variables, it is immaterial whether we do this by adding C to the sum of A and B or by adding A to sum of B and C i.e **A+(B+C) = (A+B)+C**

A	B	C	A+B	B+C	A+(B+C)	(A+B)+C
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	0	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

The values of column A + (B + C) is equal to values in column (A + B) + C. Hence second statement of associative law is proved.

3. Distributive law

Distributive law says that if Boolean expression is a sum of two or more than two terms having a common variable then that common variable can be taken common like ordinary variable i.e. A.(B+C) = A.B + A.C

A	B	C	A.B	A.C	B+C	A.(B+C)	A.B+A.C
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

The values of column A(B+C) is equal to values of column A. B + A. C. hence 1st statement of distributive law is proved.

Similarly, this law also states that Boolean expression can be expanded term by term.

i.e. **A +(B.C) = (A + B) . (A + C)**

A	B	C	A+B	A+C	B.C	A+(B.C)	(A+B).(A+C)
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1
1	0	1	1	1	0	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1

The value in column A+(B+C) are equal to values in column (A + B).(A + C). Hence statement is proved.

4. Idempotent law

This law says if a Boolean variables is multiplied with itself, we get that variable only.

i.e. A. A=A

A	A. A
0	0
1	1

Here values of column A and A.A are equal. Hence statement proved.

similarly, this law also states that if we add a Boolean variables with itself we get that variable only.

i.e. A+A=A

A	A + A
0	0
1	1

Here values of column A and A + A are equal. Hence statement proved.

5. Involution law

This law states that double complement of a Boolean variable is equal to the variable itself.

i.e. (A')' = A.

Proof

A	A'	(A')'
0	1	0
1	0	1

Values in column A and (A')' are equal. Hence equal.

6. Boundedness law

This law says if 1 is added to a Boolean variable we get output 1.

i.e. A + 1= 1

A		A+1
0	1	1
1	1	1

All values of column A+1 are 1. Hence statement proved.

Also this law says that if 0 is multiplied to a Boolean variable we get output 0.

i.e. A.0 = 0

A		A.0
0	0	0
1	0	0

All values of column A.0 are 0. Hence 2nd statement proved.

7. Absorption law

If two Boolean variables A and B are given then if we add A to the product of A and B we get Boolean variable A as the result.

i.e. A+(A. B) =A

A	B	A. B	A+(A .B)
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

All values of column A+(A. B) are equal to values in column A. Hence statement proved.

Similarly, the other statement of absorption law says if two Boolean variables, A and B are given, then if we multiply A to the sum of A and B then we get A.

i.e. A.(A + B) = A

A	B	A + B	A.(A + B)

0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

In above table the values in column A.(A + B) are equal to column A. Hence statement proved.

8. Identity law

This law says of 0 is added to any Boolean variable 'a' then we obtain that variable only.

i.e. A+0=A

A	A+0
0	0
1	1

Here values in column A+0 is equal to column A. Hence statement proved

Similarly, another statement of identify law says if 1 in multiplied to any Boolean variable 'A' then we get that variable only.

i.e. A.1=A

A	A.1
0	0
1	1

Here values in column A and A.1 are equal. Hence statement proved.

9.Complement law

This law says that if a Boolean variable is added with its complement we obtain 1.

i.e. A + A' = 1

A	A'	A + A'
0	1	1
1	0	1

In the table all the values of column A + A' are 1. Hence statement proved.

Similarly if a Boolean variable is multiplied with its complement we obtain 0

i.e. A. A'=0

A	A'	A. A'
0	1	0
1	0	0

Here, in the table all the values of column A. A' are 0. Hence statement proved.

10. De Morgan's law

De Morgan law is the most renowned logical theorem for digital electronics using Boolean algebra. It has two following theorem.

a. Theorem 1

This theorem states that "The complement of a sum of Boolean variable equal to the product of complement of those variables. If a and b are two Boolean variables then (a + b)' = a'. b' .

truth table proof:

A	B	A'	B'	A + B	(A + B)'	A'. B'
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

The values in column (A + B)' are equal to values in column A'. B'. Hence proved.

b. Theorem 2

This theorem states that complement of product of Boolean variables equals to sum of the complement of those variables. If A, B and C are the Boolean variables then

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

truth table proof:

A	B	A'	B'	A. B	(A. B)'	A' + B'
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

The values in column (A.B)' are equal to the values in column A'+B'. Hence Theorem 2 is proved.

Principles of Duality

Principles of Duality states that a duality of a Booleans expression can be formed by:

i. Replacing OR (+) with AND (–)

ii. Replacing AND (–) with OR (+)

iii. Replacing 0 with 1 and 1 with 0.

But in the process, variables and the complements needs to remain same.

Examples of duals formed from Boolean expression:

Boolean expression	Duals
I) (A + B).C	(A. B) + C
II) A. B + C	(A + B). C
III) (X+0) + (1+A')	(X. 1). (1. A')

Logical Gates

Logical gates are the electronic circuit which operates on one or more input signal to produce single output . AND, OR, NOT are basic gates.

a) AND gates:

It is a digital circuit whose output is a value of 1 only when all input are 1. On all other cases of input output 0 is returned. If we consider 3 inputs a, b and c with s being the output then the resulting AND gate is

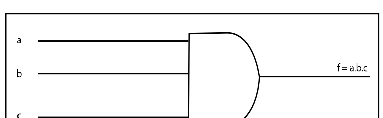


Fig: AND Gate

shown below:

The resulting truth table is shown below:

Inputs			Outputs
A	B	C	F = A.B.C
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	1

The resulting Boolean expression is F = A.B.C

b) OR gate:

It is a digital circuit whose output is a value of 1; if any one of the input is 1. The output in OR gate is 0 only when all inputs are 0. If we consider 3 inputs a, b, c and if d is the output then the resulting or gate is shown below.

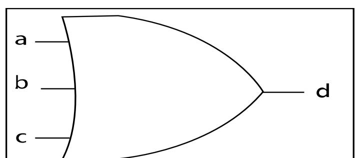


Figure: OR Gate

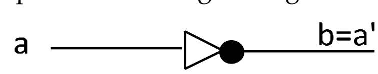
The resulting truth table is shown below:

Inputs			Outputs
a	b	c	d = a + b + c
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

The resulting Boolean expression is
d= a + b + c

c. NOT gate:

It is a digital circuit whose output is a value of 1; if input is 0 and output is 0 if input is 1. If we consider 'a' as input and 'b' as output the resulting NOT gate is shown below:



The resulting truth table is shown below:

Input A	*Output B=A'*
0	*1*
1	*0*

The resulting Boolean expression is:

b = a'

In addition to the above basic logical gates; we can have some more logic gates by combining basic gates. These gates are NOR, NAND, XOR, XNOR which are described as followed:

i. NOR gate(logical complement of OR)

It is a digital circuit whose output is 1 only if all inputs are 0. On all other cases output is 0. A NOR gate is basically an OR circuit followed by a NOT circuit. If we consider x and y as input and F as the output then the resulting NOR gate is shown below:

Figure: of NOR gate

The resulting truth table is shown below:

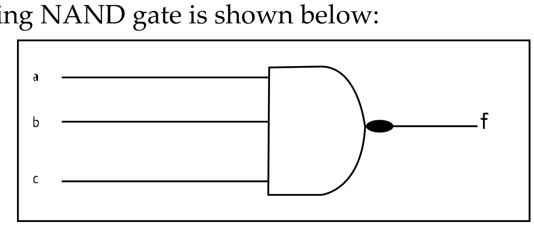
Inputs		Outputs
X	Y	F
0	*0*	*1*
0	*1*	*0*
1	*0*	*0*
1	*1*	*0*

The resulting Boolean expression is:

F= (X + Y)'

ii. NAND gate(Logical complement of AND)

It is a digital circuit whose output is 0 when all input signal are 1 and on all other cases output is 1. A NAND gate is basically AND circuit followed by a NOT circuit. If we consider A, B, C as input and s as output then the resulting NAND gate is shown below:



The resulting truth table is shown below:

Inputs			Outputs
A	B	C	F
0	0	0	1
0	0	1	1

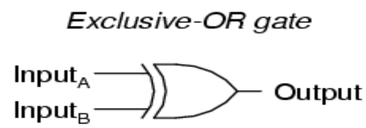
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

The resulting Boolean expression is

F = (A.B.C)'

iii. XOR gate: (Exclusive OR gate)

It is a digital circuit which yields true (1) if and only if the number of 1's on input is in odd. Otherwise output 0 is yielded.



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

The above figure shows the symbol of XOR gate and its truth table

Resulting Boolean expression is

F= A.'B +A.B'

Where F is assumed to be output

iv. XNOR gate (Exclusive NOR gate) or Equivalence gate:

It is the digital circuit which yields output 1 if all the inputs are 1 or all the inputs are 0.And on all other cases output 0 is obtained.

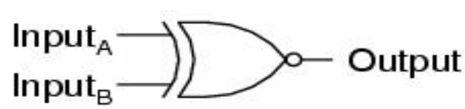


Figure XNOR gate

The resulting truth table is

Inputs		Output F
A	B	
0	*0*	*1*
0	*1*	*0*
1	*0*	*0*
1	*1*	*1*

Resulting Boolean expression is

F=A.B + A'B' where F is assumed as output.

BUFFER GATE

Unit 2: Boolean algebra and Logic Gates

Sometimes in digital electronic circuits we need to isolate logic gates from each other or have them drive or switch higher than normal loads, such as relays, solenoids and lamps without the need for inversion. A weak signal source (one that is not capable of sourcing or sinking very much current to a load) may be boosted by means of two inverters like the pair shown in the following illustration. One type of single input logic gate that allows us to do just that is called the **Digital Buffer**.

The "Buffer" performs no inversion or decision making capabilities (like logic gates with two or more inputs) but instead produces an output which exactly matches that of its input. In other words, a digital buffer does nothing as its output state equals its input state.

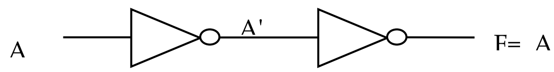


Figure a. Double inversion NOT gate

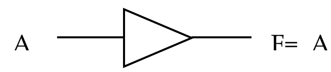


Figure b. Buffer gate

Input	Output
0	0
1	1

Figure c. Truth Table of Buffer

Integrated Circuit (ICs)

Texas Instruments is celebrating the North Texas man who made the integrated circuit – the microchip — possible. On Sept. 12, 1958, Jack Kilby, a IT engineer, invented the integrated circuit. It would revolutionize the electronics industry, helping make cell phones and computers widespread today.

An integrated circuit (ICs) is an electronic device comprising numerous functional elements such as transistors, resistors, condensers, etc. on a piece of silicon semiconductor substrate, and is sealed inside a package with multiple terminals. At present, IC critical dimensions (or smallest dimensions of IC elements) are in the order of 10 nanometers (nm: 10-9m), which is extremely small. Transistor radios that fascinated boys in the old days consisted of a piece of printed board with discrete transistors, resistors, condensers and diodes inserted, which were wired to each other. The current IC is highly integrated and miniaturized, about 1/55000 of the size and 3 billionths of the area of the transistor radio. Owing to their high integration, ICs with various functions embedded have dramatically enhanced the performance of electronics.

In the manufacturing of ICs, many ICs are made on a silicon wafer and then cut (diced) into numerous IC chips (dies). The IC chips are sealed inside packages because they are too small to be electrically bonded to a printed circuit board, and also because IC chips would get broken if left unprotected. If you open the cover of a personal computer, you will see objects with multiple legs sticking out. These are the ICs hidden inside the packages. Depending on the number of components, ICs are classified as LSI (large-scale integration), VLSI（very large scale integration), and ULSI（ultra large scale integration）. Normally the "VLSI" term covers "ULSI".