LOGIC DESIGN

CSC 221

What is Logic?

Logic is a science that aims to identify principles for good and bad reasoning. As such, logic is a prescriptive study in that one of its goals is to tell you how you ought to reason.

What is Symbolic Logic?

Symbolic logic is a branch of logic that represents how we ought to reason by using a formal language consisting of abstract symbols. These abstract symbols, their method of combination, and their meanings (interpretations) provide a precise, widely applicable, and highly efficient language in which to reason.

Logic deals with statements, and statements vary extensively in the precision with which they may be made. If someone says, "That is a good book," that is a statement. It is far less precise, however, than a statement such as "Osogbo is the capital of Osun State."

In symbolic logic, a letter such as **p** stands for an entire statement. It may, for example, represent the statement, "*A triangle has three sides.*"

Every statement is either **True** or **False**

Logic is built from simple statements using **logical connectives**

CONJUNCTION

If p stands for the statement, **"All right angles are equal,"** and q the statement, **"Parallel lines never meet,"** one can make a single statement by joining them with "**and**": "All right angles are equal and parallel lines never meet." This can be symbolized **p ∧ q**, using the inverted V-shaped symbol to stand for the conjunction "and." Both the combined statement and the word "and" itself are called "conjunctions." In ordinary English, there are several words in addition to "and" which can used for joining two statements conjunctively,In logic the only conjunctive term is "and."

DISJUNCTION

Another word used in both ordinary English and in logic is "**or**." Someone who says, "Either he did not hear me, or he is being rude," is saying that at least one of those two possibilities is true. By connecting the two possibilities about which he or she is unsure, the speaker can make a statement of which he or she is sure. **In logic, "or" means "and/or."** If p and q are statements, **p V q** is the statement, called a "disjunction," formed by connecting p and q with "or," symbolized by "**V**."

For example if p is the statement, **"Mary Doe may draw money from this account,"** and q is the statement, **"John Doe may draw money from this account,"** then p V q is the statement, **"Mary Doe may draw money from this account, or John Doe may draw money from this account."**

The disjunction p V q is true when p, q, or both are true. In the example above, for instance, an account set up in the name of Mary or John Doe may be drawn on by both while they are alive and by the survivor if one of them should die. Had their account been set up in the name Mary and John Doe, both of them would have to sign the withdrawal slip, and the death of either one would freeze the account.

NEGATION

Negation is another logical "operation." Unlike conjunction and disjunction, however, it is applied to a single statement. If one were to say, "She is friendly," the negation of that statement would be, "She is not friendly." The symbol for negation is " ~." It is placed in front of the statement to be negated, as in ~ (p∧q) or ~ p. If p were the statement, "She is friendly," ~ p means "She is not friendly,"

"**implies**" symbolized by **"=>."**
A statement p=> q means that whenever p is true, q is true also.
For example, if p is the statement, "x is in Osun State," and q is the statement "x is in Nigeria," then p=> q is the statement, "If x is in Osun state, then x is in Nigeria." It can be read "p and q are equivalent;
" p is true if and only if q is true;"
"p implies and is implied by q;" "p is a necessary and sufficient condition for q;" and "p implies q, and conversely."
In p=> q, p is called the "antecedent" and q the "consequent."

Equivalent propositions or statements can be symbolized with the two-headed arrow $\leftrightarrow$ ("if and only if..."**).**

Truth Tables

P	¬P
T	F
F	T

P	Q	P ∧ Q
T	T	T
T	F	F
F	T	F
F	F	F

cont. Truth Tables

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

cont. Truth Table

P	Q	$P \leftrightarrow Q$
T	T	T
T	F	F
F	T	F
F	F	T

Compound statement with three components P, Q, and R

P	Q	R
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

Construct a truth table for the formula $\neg P \land (P \to Q)$

P	Q	$\neg P$	$P \to Q$	$\neg P \land (P \to Q)$
T	T	F	T	F
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Tautology

Example. Show that **$(P \to Q) \lor (Q \to P)$ is** a tautology.

P	**Q**	**$(P \to Q)$**	**$(Q \to P)$**	**$(P \to Q) \lor (Q \to P)$**
T	T	T	T	T
T	F	F	T	T
F	T	T	F	T
F	F	T	T	T

Compound Statement Example.

Construct a truth table for **$(P \to Q) \land (Q \to R)$**

P	Q	R	$(P \to Q)$	$(Q \to R)$	$(P \to Q) \land (Q \to R)$
T	T	T	T	T	T
T	T	F	T	F	F
T	F	T	F	T	F
T	F	F	F	T	F
F	T	T	T	T	T
F	T	F	T	F	F
F	F	T	T	T	T
F	F	F	T	T	T

Boolean Algebra

Boolean algebra is a division of mathematics that deals with operations on logical values and incorporates binary variables. Boolean algebra traces its origins to an 1854 book by mathematician George Boole.

Boolean Algebra is a set of rules used to simplify logic expression without changing its functionality. It is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1.

$$F = A'B + BC + ABC$$

$$F = A'B + BC$$

Rules in Boolean Algebra

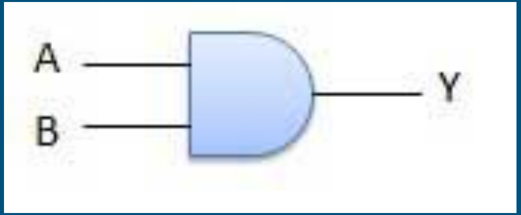
- Variable used can have only two values. Binary 1 for HIGH and Binary 0 for LOW.
- **Complement of a variable** is represented by an overbar (-). Thus, complement of variable B is represented as $\overline{B}$. Thus if B = 0 then $\overline{B}$ = 1 and B = 1 then $\overline{B}$ = 0.
- ORing of the variables 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.

LOGIC GATES

- Logic gates are made of transistors, and they are the basic components of integrated circuit (IC).
- Logic gates are used for designing digital system; there are **three basic logic operations** and they are called **AND, OR,** and **NOT**.
- The characteristic of a digital system can be represented by a function or truth table.
- Boolean theorems are used to simplify Boolean function in order to use fewer logic gates.

AND Gate

Logic Diagram



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

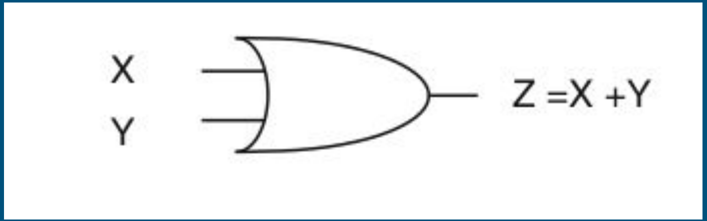
AND Law

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

- A . 0 = 0
- A . 1 = A
- A. A = A
- A . A’= 0

OR GATE

Logic Diagram



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

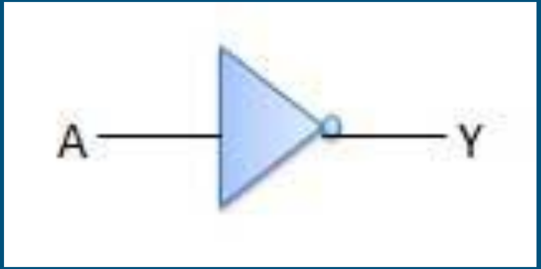
OR Law

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

- $A + 0 = A$
- $A + 1 = 1$
- $A + A = A$
- $A + A' = 1$

NOT GATE

Logic Diagram



A	Y
0	1
1	0

Laws of Boolean Algebra

Distributive Law

Distributive law states the following conditions:

- A. (B + C) = (A. B) + (A. C)
- A + (B. C) = (A + B) . (A + C)

Commutative Law

Any binary operation which satisfies the following expression is referred to as a commutative operation. Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.

- A. B = B. A
- A + B = B + A

Associative Law

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

- $(A.B).C = A.(B.C)$
- $(A+B)+C = A+(B+C)$

Inversion Law

In Boolean algebra, the inversion law states that double inversion of variable results in the original variable itself.

$$(A')' = A$$

Boolean Algebra Theorem

The two important theorems which are extremely used in Boolean algebra are De Morgan’s First law and De Morgan’s second law. These two theorems are used to change the Boolean expression. This theorem basically helps to reduce the given Boolean expression in the simplified form.

De Morgan’s First Law:

De Morgan’s First Law states that **$(A.B)' = A'+B'$.**

The first law states that the complement of the product of the variables is equal to the sum of their individual complements of a variable.

The truth table that shows the verification of De Morgan’s First law is given as follows:

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

De Morgan’s Second Law:

De Morgan’s Second law states that (A+B)’ = A’. B’.

The second law states that the complement of the sum of variables is equal to the product of their individual complements of a variable.

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

$$F = A'B + BC + ABC$$

$$A'B + BC(1 + A)$$

$$A'B + BC(1)$$

$$F = A'B + BC$$

$F=(A+B+C)(A+B'+C)(A+B+C')$

Let $X= A + B$

$F = (X+C)(A+B'+C)(X+C')$

$A.B = B.A,\ A.B.C=B.C.A$

$(X+C)(X+C')(A+B'+C)$ ……….Distributive Law

$(X+C.C').(A+B'+C)$

$(X + 0).(A+B'+C)$

$X.(A+B'+C)$

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

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

$A + B.B'+ B.C$

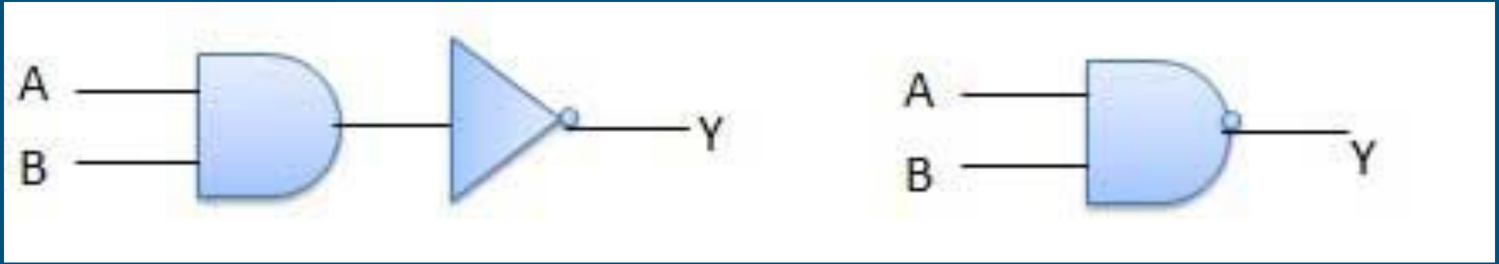
$A + 0 + B.C$

$F= A+B.C$

UNIVERSAL GATES

NAND GATE

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

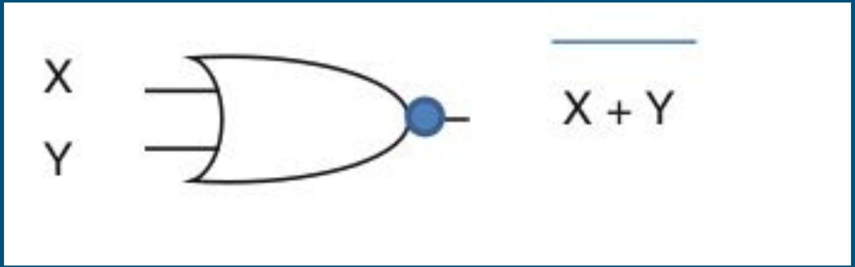


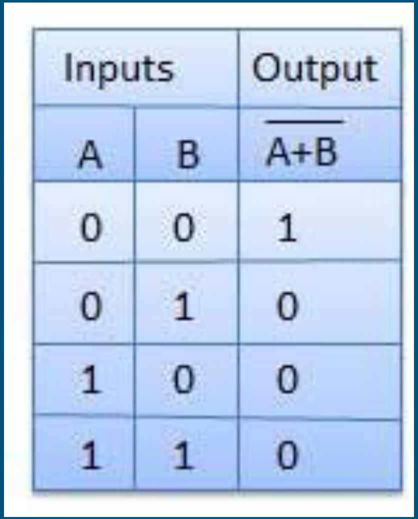
Nand gate Truth Table

A	B	(AB)’
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate

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



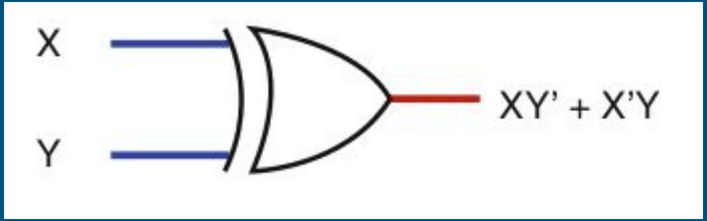


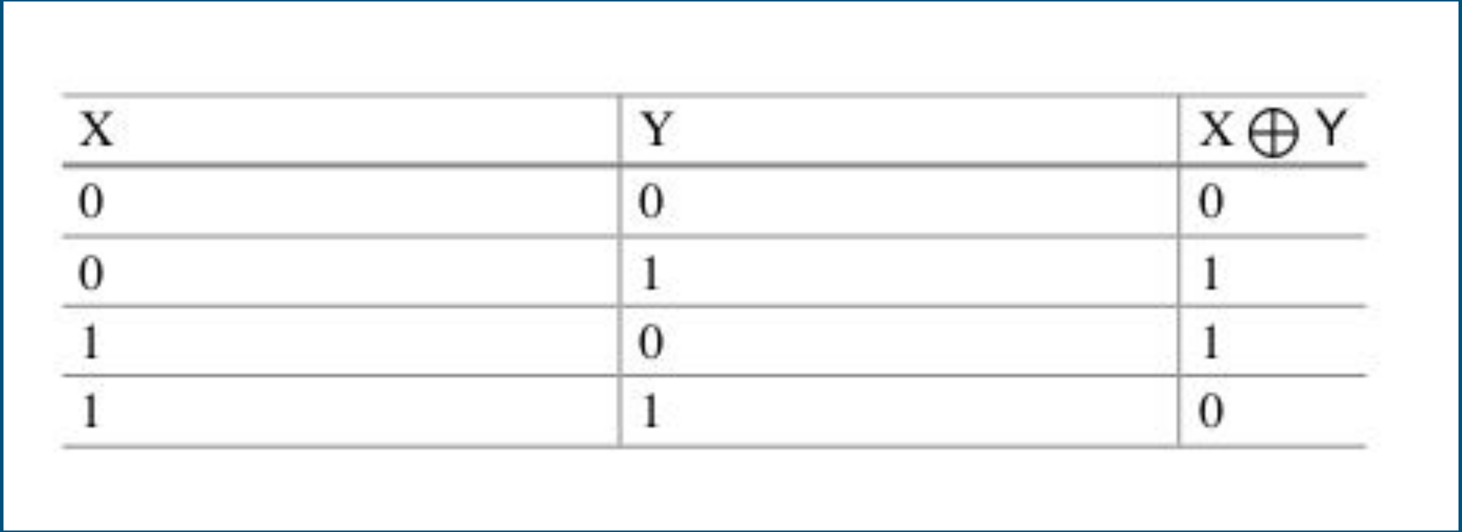
Inputs		Output
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

ADVANCED GATE

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.

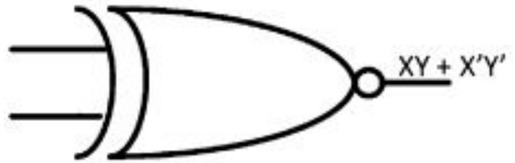




X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

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.



X	Y	$X \odot Y$
0	0	1
0	1	0
1	0	0
1	1	1