DIGITAL SYSTEMS DESIGN (BECE102L)

Dr. Mangaiyarkarasi. R
Assistant Professor Sr. Grade. 2
SENSE- CNR
Cabin @ SMV 231

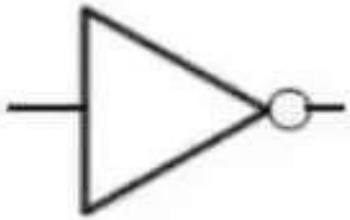
Definition

- Digital Logic is a term used to denote the design and analysis of digital systems.
- Digital Logic is the basis of electronic systems, such as computers and cell phones.
- Digital Logic is rooted in binary code, a series of zeroes and ones.
- This system includes logic gates and facilitates the design of electronic circuits that convey information in the form of digital bits, Digital Logic gate functions include and, or and not.
- Design a digital systems by verilog HDL (Hardware description Language) codes.
- Design of such system is digital logic design

Example:

Temperature Detector
(Above 20°C)

ON/OFF



Central Heating

ON/OFF

If the temperature is above 20°C then the Central Heating is switched off.

If the temperature is below 20°C then the Central Heating is switched on.

Module:1	Digital Logic	8 hours
Boolean Algebra: Basic definitions, Axiomatic definition of Boolean Algebra, Basic Theorems and Properties of Boolean Algebra, Boolean Functions, Canonical and Standard Forms, Simplification of Boolean functions. Gate-Level Minimization: The Map Method (K-map up to 4 variable), Product of Sums and Sum of Products Simplification, NAND and NOR Implementation. Logic Families: Digital Logic Gates, TTL and CMOS logic families.		
Module:2	**Verilog HDL**	**5 hours**
Lexical Conventions, Ports and Modules, Operators, Dataflow Modelling, Gate Level Modelling, Behavioural Modeling, Test Bench.		
Module:3	**Design of Combinational Logic Circuits**	**8 hours**
Design Procedure, Half Adder, Full Adder, Half Subtractor, Full Subtractor, Decoders, Encoders, Multiplexers, De-multiplexers, Parity generator and checker, Applications of Decoder, Multiplexer and De-multiplexer. Modeling of Combinational logic circuits using Verilog HDL.		
Module:4	**Design of data path circuits**	**6 hours**
N-bit Parallel Adder/Subtractor, Carry Look Ahead Adder, Unsigned Array Multiplier, Booth Multiplier, 4-Bit Magnitude comparator. Modeling of data path circuits using Verilog HDL.		

Module:5	Design of Sequential Logic Circuits	8 hours
Latches, Flip-Flops - SR, D, JK & T, Buffer Registers, Shift Registers - SISO, SIPO, PISO, PIPO, Design of synchronous sequential circuits: state table and state diagrams, Design of counters: Modulo-n, Johnson, Ring, Up/Down, Asynchronous counter. Modeling of sequential logic circuits using Verilog HDL.		
Module:6	**Design of FSM**	**4 hours**
Finite state Machine(FSM):Mealy FSM and Moore FSM , Design Example : Sequence detection, Modeling of FSM using Verilog HDL.		
Module:7	**Programmable Logic Devices**	**4 hours**
Types of Programmable Logic Devices: PLA, PAL, CPLD, FPGA Generic Architecture.		
Module:8	**Contemporary issues**	**2 hours**
	Total Lecture hours:	**45 hours**

Textbook(s)

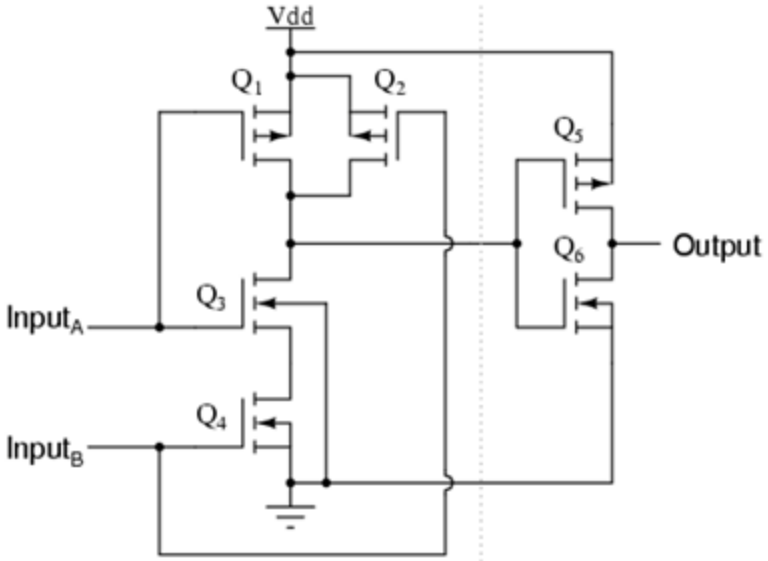
1.	M. Morris Mano and Michael D. Ciletti, Digital Design: With an Introduction to the Verilog HDL and System Verilog, 2018, 6th Edition, Pearson Pvt. Ltd.

Reference Books

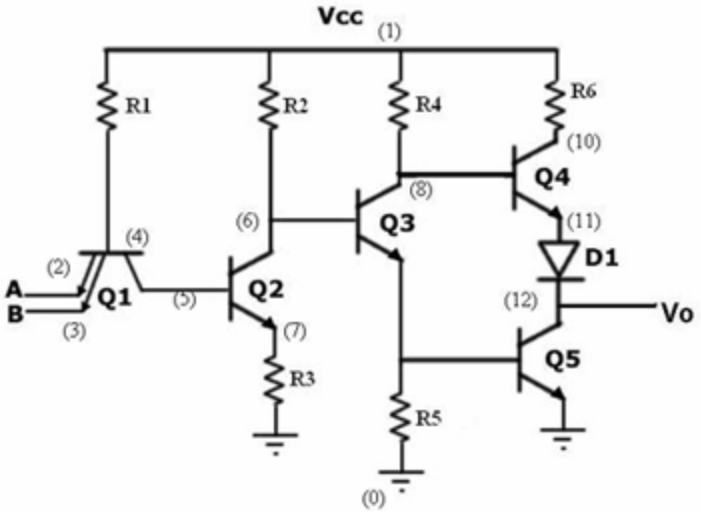
1.	Ming-Bo Lin, Digital Systems Design and Practice: Using Verilog HDL and FPGAs, 2015, 2nd Edition, Create Space Independent Publishing Platform.
2.	Samir Palnitkar, Verilog HDL: A Guide to Digital Design and Synthesis, 2009, 2nd edition, Prentice Hall of India Pvt. Ltd.
3.	Stephen Brown and ZvonkoVranesic, Fundamentals of Digital Logic with Verilog Design, 2013, 3rd Edition, McGraw-Hill Higher Education.

Difference between TTL and CMOS

- CMOS
  - Use FETs
  - Symmetric Device Source and Drain Interchangeable
  - Voltage Controlled Current Device
  - Very high Input Resistance => Very low power consumption when not being used => Imperative for Small Integrated Device
- TTL
  - Use BJTs
  - Non Symmetric Device Non Interchangeable Collector and Emmitor
  - Current Controlled Current Device
  - Lower input Resistance



CMOS Logic



TTL Logic

Course Objectives:

1. Provide an understanding of Boolean algebra and logic functions.
2. Develop the knowledge of combinational and sequential logic circuit design.
3. Design and model the data path circuits for digital systems.
4. Establish a strong understanding of programmable logic.
5. Enable the student to design and model the logic circuits using Verilog HDL.

Course Outcome:

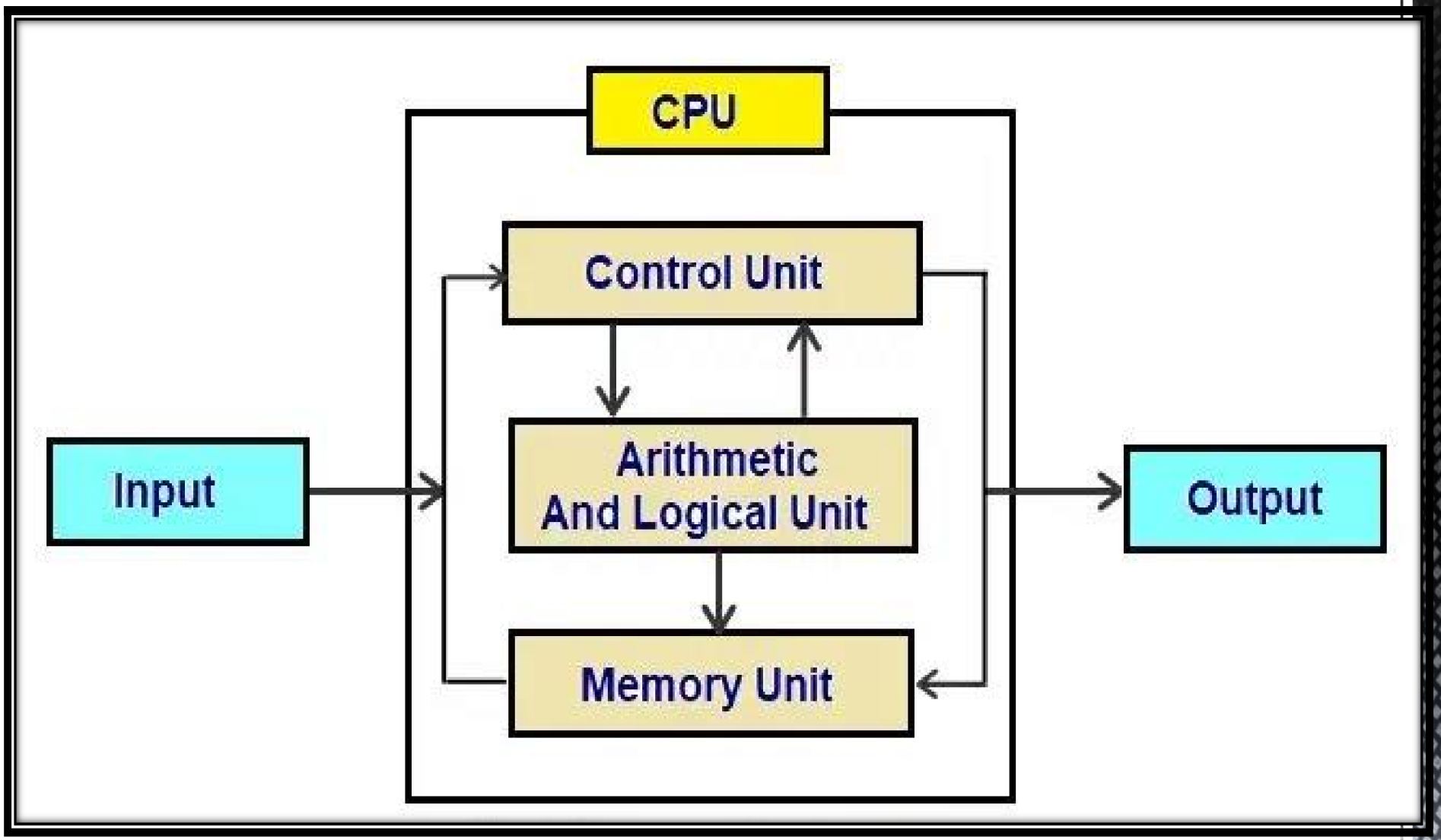
At the end of the course the student will be able to

1. Optimize the logic functions using Boolean principles and K-map
2. Model the Combinational and Sequential logic circuits using Verilog HDL
3. Design the various combinational logic circuits and data path circuits
4. Analyze and apply the design aspects of sequential logic circuits
5. Analyze and apply the design aspects of Finite state machines
6. Examine the basic architectures of programmable logic devices

Analog vs Digital

	Analog System	Digital System
Signal	Analog signal represents physical measurements.	Digital signals are discrete and generated by digital modulation.
Waves	Sine Waves	Square Waves
Representation	Continuous range of values to represent information	Uses discrete values to represent information
Technology	Records waveforms as they are.	Samples analog waveforms into a limited set of numbers and then records them.
Data transmissions	Affected by noise during transmission and write/read cycle.	Noise-immune during transmission and write/read cycle.
Response to Noise	More likely to get affected	Less likely to get affected
Flexibility	Hardware is not flexible.	Hardware is flexible.
Bandwidth	Less bandwidth.	More bandwidth to carry out the same information
Memory	Stored data in the form of wave signal	Stored data in the form of binary bit
Power	Consumes large power	Consumes negligible power
Uses	Best suited for audio and video transmission.	Best suited for Computing and digital electronics.
Cost	Cost is low	Cost is high
Example	Human voice in air, analog electronic	Computers, CDs, DVDs

Block Diagram of a Computer



Module 1 – Boolean Algebra

What is digital System Design?

System: “A set of related components work as a whole to achieve a specific task or goal”
System: Input, behaviour, output
Behaviour: Function that translates input to output
Digital System: A system that takes discrete values in the form of binary bit (0 and 1) to process both numerical and non-numerical information.

Binary system

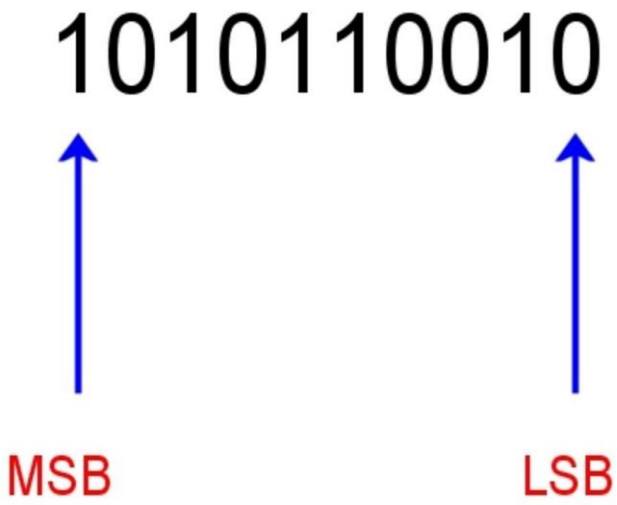
A bit (binary digit) is the smallest unit of data that a computer can process and store.

- 1 bit represents 2 states **‘0’ or ‘1’**
- 2 bits represent 4 states **‘00’ or ‘01’ or ‘10’ or ‘11’**
- 3 bits represent 8 states **‘000’, ‘001’, ‘010’, ‘100’, ‘011’, ‘101’, ‘110’, ‘111’**

- N bits represent 2^N states

LSB and MSB

LSB : Least significant bit MSB: Most significant bit



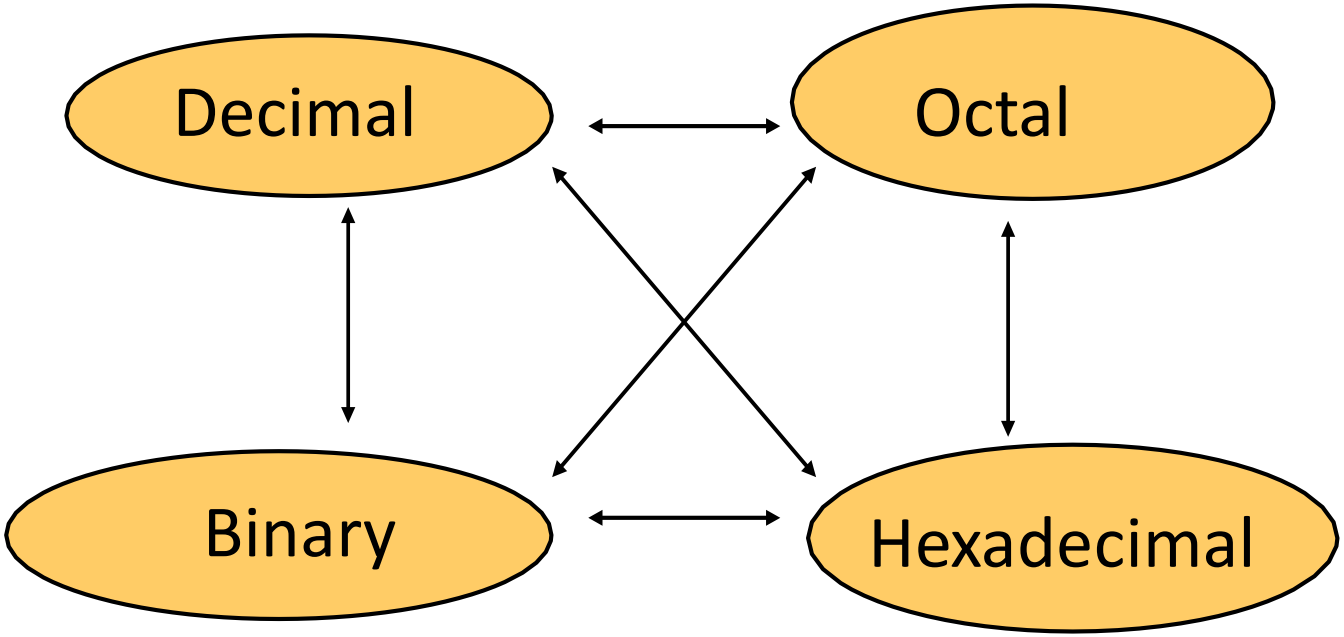
Number systems

Decimal	Binary	Octal	Hexa-decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

Decimal	Binary	Octal	Hexa-decimal
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

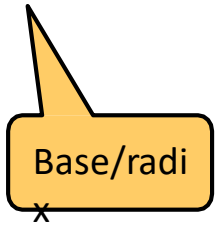
Conversion Among Bases

The possibilities:



Quick Example

$$25_{10} = 11001_2 = 31_8 = 19_{16}$$



Module:1 Digital Logic

Boolean Algebra: Basic definitions, Axiomatic definition of Boolean Algebra, Basic Theorems and Properties of Boolean Algebra, Boolean Functions, Canonical and Standard Forms, Simplification of Boolean functions.

Gate-Level Minimization: The Map Method (K-map up to 4 variable), Product of Sums and Sum of Products Simplification, NAND and NOR Implementation. Logic Families: Digital Logic Gates, TTL and CMOS logic families.

Introduction

- Developed by English Mathematician *George Boole* in between 1815 - 1864.
- It is described as *an algebra of logic or an algebra of two values i.e True or False*
- The term *logic* means a statement having binary decisions i.e *True/Yes*
  False/No

Boolean Algebra

- Logic gates use Boolean algebra to execute logical processes.
- Boolean Algebra is used to analyze and simplify the digital (logic) circuits without changing its functionalities.
- 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.

https://www.geeksforgeeks.org/logic-gates-definition-types-uses/

Boolean Algebra

- Boolean algebra is a mathematical system for the manipulation of variables that can have one of two values.
  - ***In formal logic, these values are "true" and "false."***
  - ***In digital systems, these values are "on" and "off," 1 and 0, or "high" and "low."***
- Boolean expressions are created by performing operations on Boolean variables.

  Common Boolean operators include AND, OR, and NOT.

Boolean Algebra

A Boolean operator can be completely described using a truth table.

The truth table for the Boolean operators AND and OR are shown at the right.

The AND operator is also known as a Boolean product. The OR operator is the Boolean sum.

X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Algebra

The truth table for the Boolean NOT operator is shown at the right.

The NOT operation is most often designated by an overbar. It is sometimes indicated by a prime mark (‘) or an “elbow” (¬).

NOT x

x	$\overline{x}$
0	1
1	0

Boolean Algebra

The following Huntington postulates:

1. *(a)* Closer with respect to the operator (+)

   (b) Closer with respect to the operator (.)

2. *(a)* An identity element with respect to (+) is by 0

   $$A + 0 = 0 + A = A$$

   (b) An identity element with respect to (.) is by 1

   $$A \cdot 1 = 1 \cdot A = A$$

3. *(a)* Commutative with respect to (+)

   $$A + B = B + A$$

   (b) Commutative with respect to (.)

   $$A \cdot B = B \cdot A$$

4. *(a)* (.) is distributive over (+)

   $$A \cdot (B+C) = (A.B) + (A.C)$$

   (b) (+) is distributive over (.)

   $$A + (B.C) = (A+B) \cdot (A+C)$$

Boolean Algebra

A Boolean function has:

- At least one Boolean variable,
- At least one Boolean operator, and
- At least one input from the set {0,1}.

It produces an output that is also a member of the set {0,1}.

Now you know why the binary numbering system is so handy in digital systems.

Application of Boolean algebra

- It is used to perform the logical operations in digital computer.
- In digital computer True represent by 1’ (high volt) and False represent by ‘0’ (low volt)
- Logical operations are performed by logical operators. The fundamental logical operators are:

1. **AND (conjunction)**
2. **OR (disjunction)**
3. **NOT (negation/complement)**

AND operator

- It performs logical multiplication and denoted by (.) dot.

X	Y	X.Y
0	0	0
0	1	0
1	0	0
1	1	1

OR operator

- It performs logical addition and denoted by (+) plus.

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT operator

- It performs logical negation and denoted by (-) bar. It operates on single variable.

X	$\overline{X}$	(means complement of x)
0	1	
1	0	

Truth Table

- Truth table is a table that contains *all possible values of logical variables/statements* in a Boolean expression.

No. of possible combination $=2^n$, where n=number of variables used in a Boolean expression.

Truth Table

- The truth table for XY + Z is as follows:

Dec	X	Y	Z	XY	XY+Z
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	0
3	0	1	1	0	1
4	1	0	0	0	0
5	1	0	1	0	1
6	1	1	0	1	1
7	1	1	1	1	1

XYZ+XZ

Exercise

1. Evaluate the following Boolean expression using Truth Table.

(a) X′Y′+X′Y (b) X′YZ′+XY′

(c) XY′(Z+YZ′)+Z′

2. Verify that P+(PQ)′ is a Tautology.

3. Verify that (X+Y)′=X′Y′

Implementation

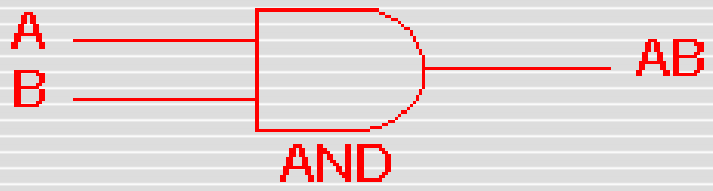
- Boolean Algebra applied in computers electronic circuits. These circuits perform Boolean operations and these are called logic circuits or logic gates.

Logic Gate

- A gate is an digital circuit which operates on one or more signals and produce single output.
- Gates are digital circuits because the input and output signals are denoted by either 1(high voltage) or 0(low voltage).
- Three type of gates are as under:
    1. AND gate
    2. OR gate
    3. NOT gate

AND gate

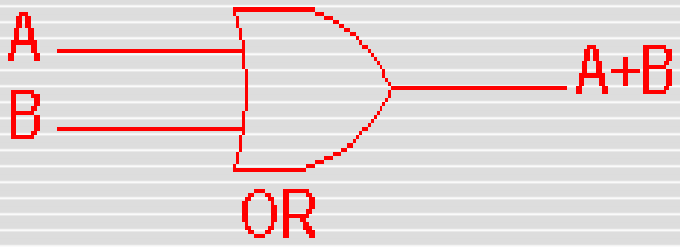
- The AND gate is an electronic circuit that gives a high output (1) only if all its inputs are high.
- AND gate takes two or more input signals and produce only one output signal.



Input A	*Input B*	*Output AB*
0	0	0
0	1	0
1	0	0
1	1	1

OR gate

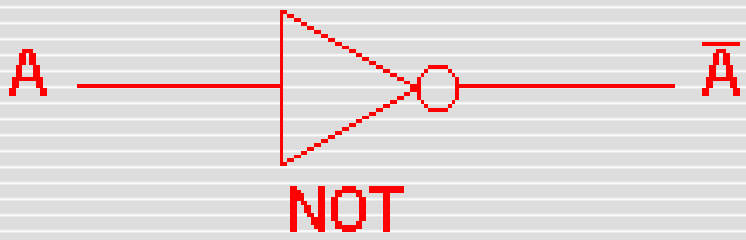
- The OR gate is an electronic circuit that gives a high output (1) if one or more of its inputs are high.
- OR gate also takes two or more input signals and produce only one output signal.



Input A	*Input B*	*Output A+B*
0	0	0
0	1	1
1	0	1
1	1	1

NOT gate

- The NOT gate is an electronic circuit that gives a high output (1) if its input is low .
- NOT gate takes only one input signal and produce only one output signal.
- The output of NOT gate is complement of its input.
- It is also called inverter.



Input A	*Output* $\overline{A}$
0	1
1	0

1. Principal of Duality

- In Boolean algebras the duality Principle is obtained by interchanging AND and OR operators and replacing 0's by 1's and 1's by 0's. Compare the identities on the left side with the identities on the right.

Example

A+1 = 1 then A.0 = 0

Basic Properties and Theorms of Boolean Algebra

1 . Principle of Duality

-important property of Boolean algebra

-means one expression can be obtained from the other in each pair by interchanging every element *i.e.,* every 0 with 1, every 1 with 0, as well as interchanging the operators *i.e.,* every (+) with (.) and every (.) with (+).

Basic Theorem of Boolean Algebra

T1 : Properties of 0

(a) 0 + A = A

(b) 0 A = 0

T2 : Properties of 1

(a) *1 + A = 1*

(b) *1 A = A*

Basic Theorem of Boolean Algebra

T3 : Commutative Law

(a) $A + B = B + A$

(b) $A\,B = B\,A$

T4 : Associate Law

(a) $(A + B) + C = A + (B + C)$

(b) $(A\,B)\,C = A\,(B\,C)$

T5 : Distributive Law

(a) $A\,(B + C) = A\,B + A\,C$

(b) $A + (B\,C) = (A + B)\,(A + C)$

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

Basic Theorem of Boolean Algebra

T6 : Indempotence (Identity) Law

(a) A + A = A

(b) A A = A

T7 : Absorption (Redundance) Law

(a) A + A B = A

(b) A (A + B) = A

Basic Theorem of Boolean Algebra

T8 : Complementary Law

(a) $X+X'=1$

(b) $X.X'=0$

T9 : Involution

(a) $x''=x$

T10 : De Morgan's Theorem

(a) $(X+Y)'=X'.Y'$

(b) $(X.Y)'=X'+Y'$

2. De Morgan's Theorem

Two Theorem

a) The complement of a product is equal to the sum of the complements.

$$(A \cdot B)' = A' + B'$$

b) The complement of a sum is equal to the product of the complements.

$$(A + B)' = A' \cdot B'$$

Exercise

Q 1. State & Verify De Morgan's Law by using truth table and algebraically.

Q 2. State and verify distributive law.

Q 3. Draw a logic diagram for the following expression:

(a) ab+b’c+c’a’

(b) (a+b).(a+b’).c

Boolean Algebra

Our second group of Boolean identities should be familiar to you from your study of algebra:

Identity Name	AND Form	OR Form
Commutative Law	$xy = yx$	$x+y = y+x$
Associative Law	$(xy)z = x(yz)$	$(x+y)+z = x+(y+z)$
Distributive Law	$x+yz = (x+y)(x+z)$	$x(y+z) = xy+xz$

Boolean Algebra

Our last group of Boolean identities are perhaps the most useful.
If you have studied set theory or formal logic, these laws are also familiar to you.

Identity Name	AND Form	OR Form
Absorption Law	$x(x+y) = x$	$x + xy = x$
DeMorgan's Law	$\overline{(xy)} = \bar{x} + \bar{y}$	$\overline{(x+y)} = \bar{x}\bar{y}$
Double Complement Law	$\overline{(\bar{x})} = x$	

Boolean Algebra

Sometimes it is more economical to build a circuit using the complement of a function (and complementing its result) than it is to implement the function directly.

DeMorgan's law provides an easy way of finding the complement of a Boolean function.

Recall DeMorgan's law states:

$$\overline{(xy)} = \bar{x} + \bar{y} \quad \text{and} \quad \overline{(x+y)} = \bar{x}\bar{y}$$

Two-valued Boolean Algebra

- is defined on a set of only two elements, S = {0,1}, with rules for two binary operators (+) and (.) and inversion or complement as shown in the following operator tables:

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

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

A	A’
0	1
1	0

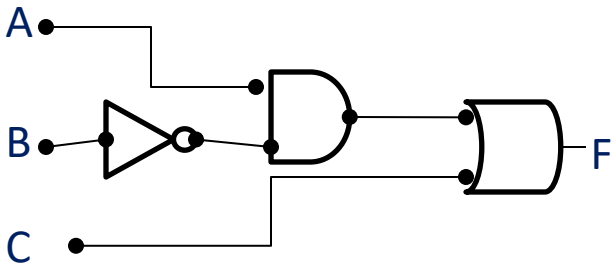
Boolean Function

-is an expression formed with binary variables, the two binary operators AND and OR, one unary operator NOT, parentheses and equal sign

Example: F = AB'C

F will be 1, A = 1, B = 0, and C = 1

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Simplification of Boolean Expressions

There are several methods to minimize the Boolean Function. Simplification or minimization of complex algebraic expressions will be shown with the help of postulate and theorems of Boolean algebra.

Example 1. Simplify the Boolean function F = ***AB+BC+B'C***

Solution: **F = AB + BC +B'C**

=AB + C(B + B')

= AB + C

Complete list of postulates and theorems		
Postulate 2	*(a)* A + 0 = A	*(b)* A . 1 = A
Postulate 5	*(a)* A + A’ = A	*(b)* A . A’ = 0
Theorem 1	*(a)* A + A = A	*(b)* A . A = A
Theorem 2	*(a)* A + 1 = 1	*(b)* A . 0 = 0
Theorem 3, Involution	(A’)’ = A	
Theorem 3, Commutative	*(a)* A + B = B + A	*(b)* A . B = B . A
Theorem 4, Associative	*(a)* A + (B + C)=(A + B) + C	*(b)* A . (B . C)=(A . B) . C
Theorem 6, Distributive	*(a)* A (B + C) = A . B + A . C	*(b)* A + B . C=(A + B) . (A + C)
Theorem 5, DeMorgan	*(a)* (A + B)’ = A’ . B’	*(b)* (A . B)’= A’ + B’
Theorem 6, Absorption	*(a)* A + A . B = A	*(b)* A . (A + B) = A

Product Term

The logical product of several variables on which a function depends is considered to be a product term. In other words, the AND function is referred to as a product term or standard product. The variables in a product term can be either in true form or in complemented form.

ABC’ ---- product term.

Sum Term

An OR function is referred to as a sum term. The logical sum of several variables on which a function depends is considered to be a sum term. Variables in a sum term can also be either in true form or in complemented form.

A+B+C’ ----- sum term

Sum of Products (SOP)

The logical sum of two or more logical product terms is referred to as a sum of products expression. It is basically an OR operation on AND operated variables.

Y=AB+BC+AC ---- sum of products

Y=A’B + BC + AC’ ---- sum of products

Examples

Show the truth table for

$$F = X' + YZ$$

- **Show the followings by constructing truth tables**

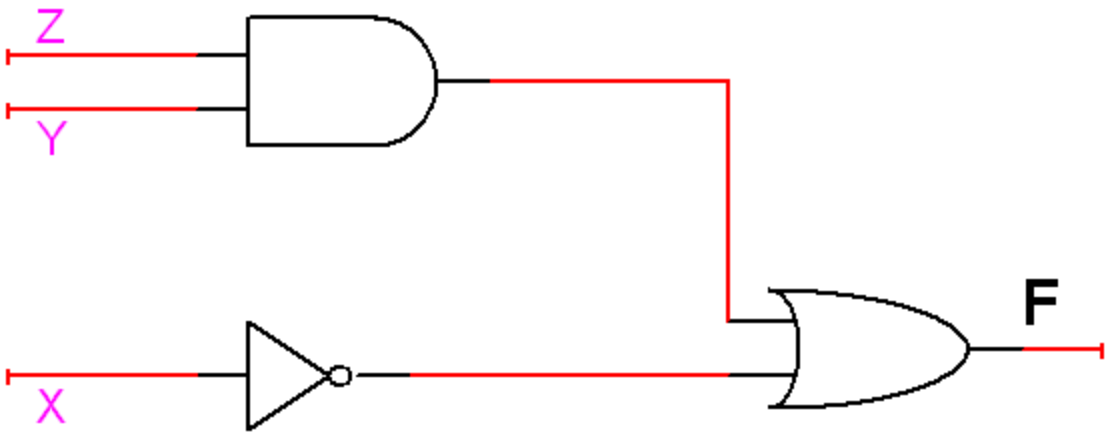
$$X(Y+Z) = XY + XZ$$

$$X + YZ = (X+Y)(X+Z)$$

Example

Draw the network diagram for

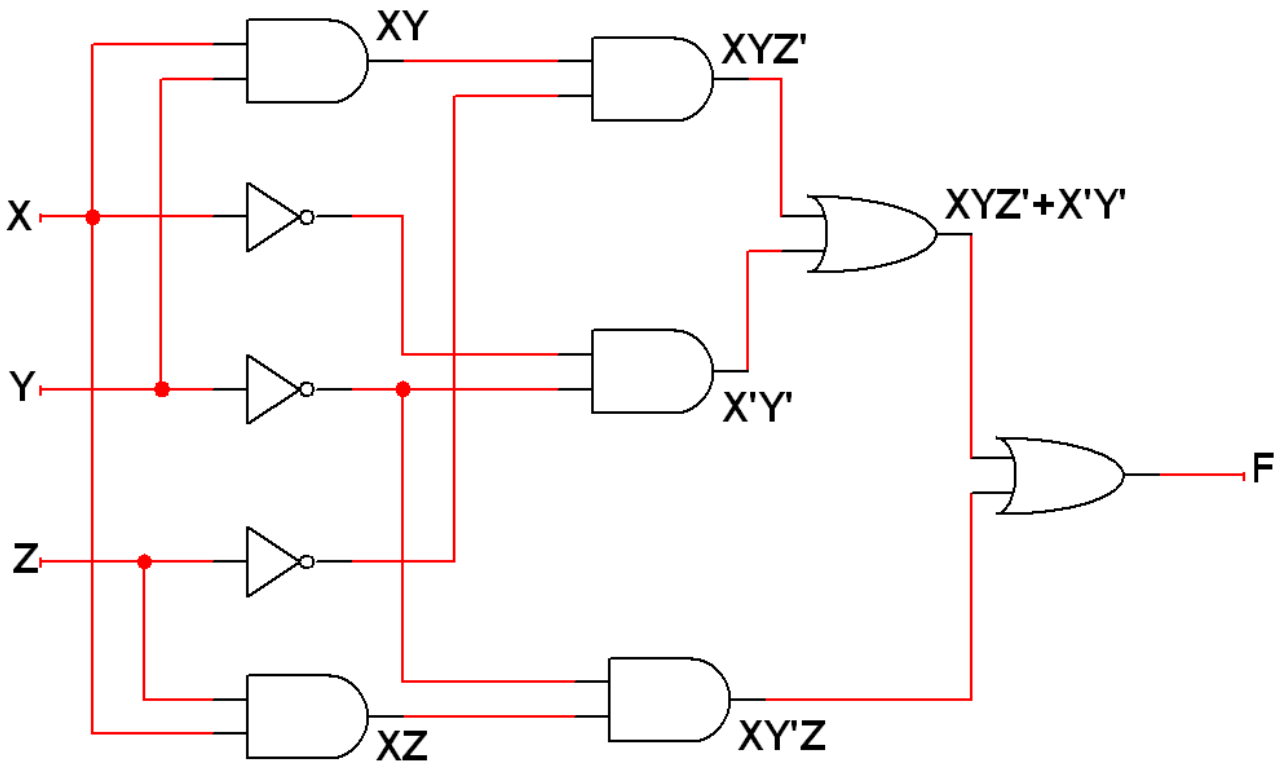
$$F = X' + YZ$$



Example

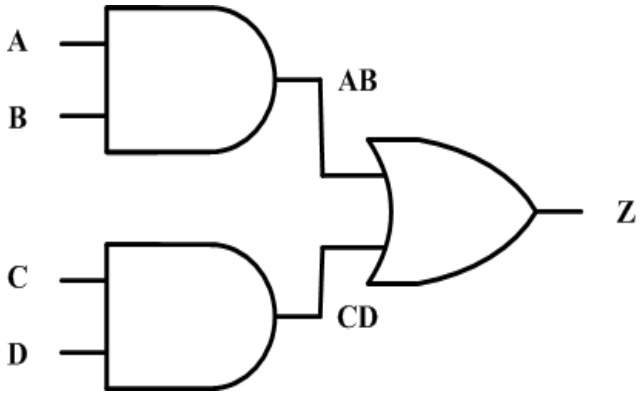
Draw the network diagram for

$$F = XYZ' + XY'Z + X'Y'$$



Truth table expression

Just like we had the truth tables for the basic functions, we can also construct truth tables for any function.



A	B	C	D	Z	AB	CD
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	1	0	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	1

Simplify, simplify

These properties (Laws and Theorems) can be used to simplify equations to their simplest form.

Simplify F=X'YZ+X'YZ'+XZ

$$F = \bar{X}YZ + \bar{X}Y\bar{Z} + XZ$$

$$= \bar{X}Y(Z + \bar{Z}) + XZ \qquad \text{by identity 14}$$

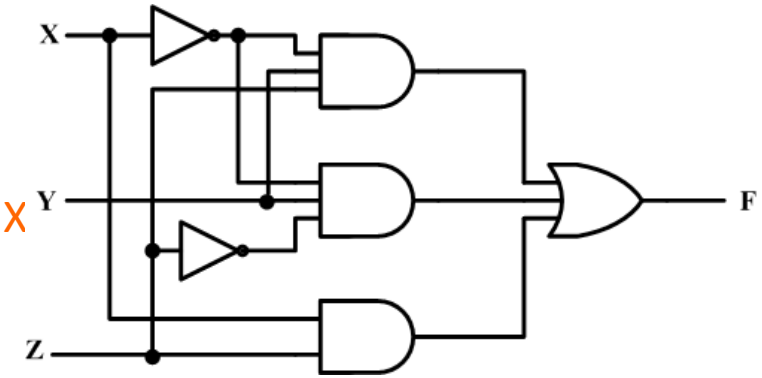
$$= \bar{X}Y \bullet 1 + XZ \qquad \text{by identity 7}$$

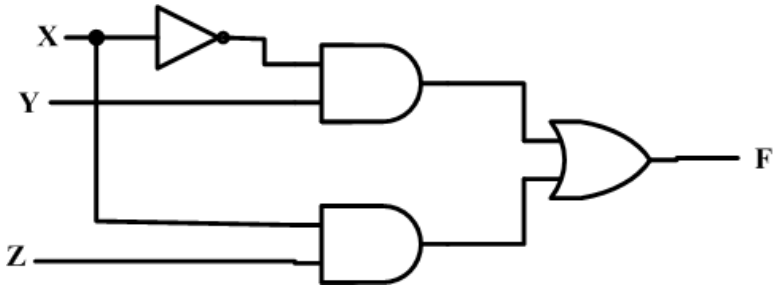
$$= \bar{X}Y + XZ \qquad \text{by identity 2}$$

Effect on implementation

F = X'YZ + X'YZ' + XZ

Reduces to F = X'Y + X





Examples

Find the complements of

$$[(A' + B)C']' = ?$$

$$[(AB' + C)D' + E]' = ?$$

$$\left[A + ((BC')' + D)'\right]' = ?$$

Boolean Algebra Summary

- Recall that the two binary values have different names:
  - True/False
  - On/Off
  - Yes/No
  - 1/0
- We use 1 and 0 to denote the two values.
- The three basic logical operations are:
  - AND
  - OR
  - NOT
- AND is denoted by a dot (·).
- OR is denoted by a plus (+).
- NOT is denoted by an overbar ($\bar{}$), a single quote mark (') after, or (~) before the variable

Boolean Algebra Summary

- We can interpret high or low voltage as representing true or false.
- A variable whose value can be either 1 or 0 is called a Boolean variable.
- AND, OR, and NOT are the basic Boolean operations.
- We can express Boolean functions with either an expression or a truth table.
- Every Boolean expression can be converted to a circuit.
- Now, we' ll look at how Boolean algebra can help simplify expressions, which in turn will lead to simpler circuits.

Boolean Algebra Summary

- Examples:
  - is read "Y is equal to A AND B."
  - is read "z is equal to x OR y."
  - is read "X is equal to NOT A."

Tabular listing of the values of a function for all possible combinations of values on its arguments
Example: Truth tables for the basic logic operations:

AND		
X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

OR		
X	Y	$Z = X+Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT	
X	$Z=\overline{X}$
0	1
1	0

Boolean Operator Precedence

- **The order of evaluation is:**
  1. Parentheses
  2. NOT
  3. AND
  4. OR
- **Consequence: Parentheses appear around OR expressions**
- **Example:** F = A(B + C)(C + D)

Boolean Algebra Postulates

- Commutative Law

$$x \cdot y = y \cdot x \qquad x + y = y + x$$

- Identity Element

$$x \cdot 1 = x \qquad x + 0 = x$$

$$x' \cdot 1 = x' \qquad x' + 0 = x'$$

- Complement

$$x \cdot x' = 0 \qquad x + x' = 1$$

Boolean Algebra Theorems

Theorem 1

- $x \cdot x = x$ $\quad\quad$ $x + x = x$

- Theorem 2
  - $x \cdot 0 = 0$ $\quad\quad$ $x + 1 = 1$

- Theorem 3: *Involution*
  - $(x')' = x$ $\quad\quad$ $\overline{(\overline{x})} = x$

Boolean Algebra Theorems

- Theorem 4:
  - *Associative:* $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
    $(x + y) + z = x + (y + z)$
  - *Distributive* :
    $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
    $x + (y \cdot z) = (x + y) \cdot (x + z)$

- Theorem 5: *DeMorgan*
  - $(x \cdot y)' = x' + y'$ $\quad (x + y)' = x' \cdot y'$
  - $\overline{x \cdot y} = \overline{x} + \overline{y}$ $\quad \overline{(x + y)} = \overline{x} \cdot \overline{y}$

- Theorem 6: *Absorption*
  - $x \cdot (x + y) = x$ $\quad x + (x \cdot y) = x$

Truth Table to Verify DeMorgan's

$$\overline{X + Y} = \overline{X} \cdot \overline{Y} \qquad \overline{X \cdot Y} = \overline{X} + \overline{Y}$$

X	Y	X·Y	X+Y	$\overline{X}$	$\overline{Y}$	$\overline{X+Y}$	$\overline{X} \cdot \overline{Y}$	$\overline{X \cdot Y}$	$\overline{X}+\overline{Y}$
0	0	0	0	1	1	1	1	1	1
0	1	0	1	1	0	0	0	1	1
1	0	0	1	0	1	0	0	1	1
1	1	1	1	0	0	0	0	0	0

- Generalized DeMorgan's Theorem:

$$\overline{X_1 + X_2 + \ldots + X_n} = \overline{X_1} \cdot \overline{X_2} \cdot \ldots \cdot \overline{X_n}$$

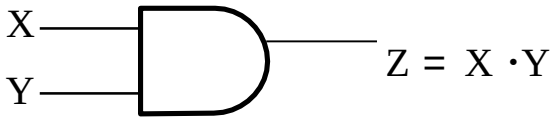
$$\overline{X_1 \cdot X_2 \cdot \ldots \cdot X_n} = \overline{X_1} + \overline{X_2} + \ldots + \overline{X_n}$$

Logic Gates

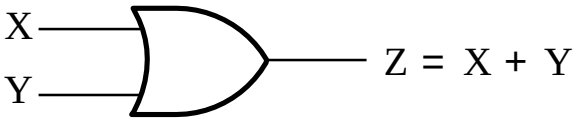
- In the earliest computers, switches were opened and closed by magnetic fields produced by energizing coils in *relays*. The switches in turn opened and closed the current paths.
- Later, *vacuum tubes* that open and close current paths electronically replaced relays.
- Today, *transistors* are used as electronic switches that open and close current paths.

Logic Gate Symbols

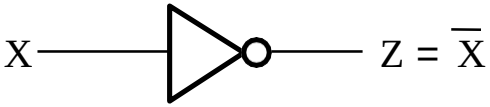
- **Logic gates have special symbols:**



AND gate



OR gate

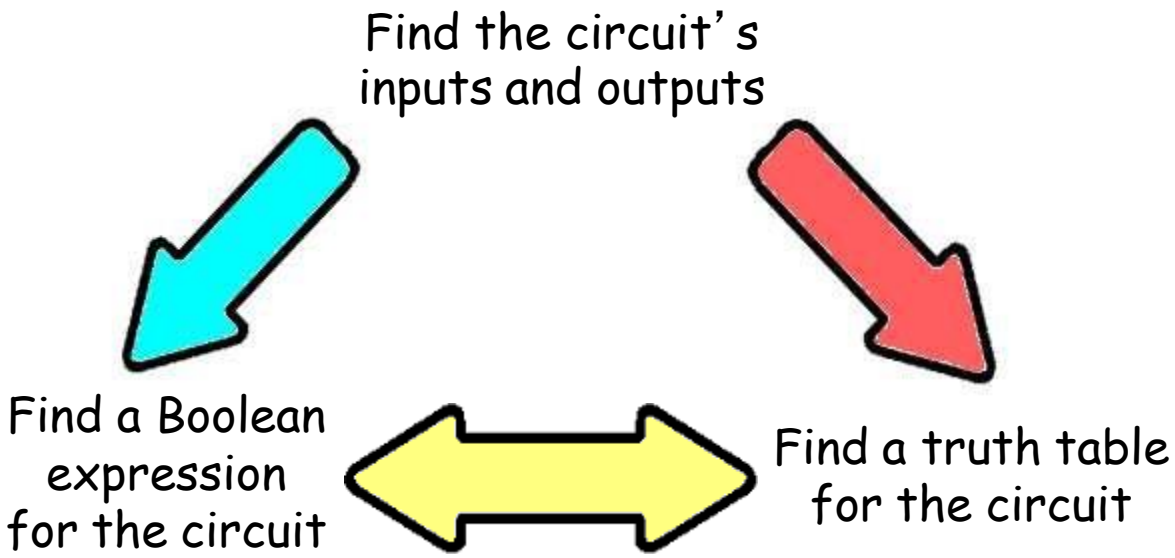


NOT gate or inverter

Boolean Functions

- A ***Boolean function*** is a function whose arguments, as well as the function itself, assume values from a two-element set ({0, 1}).
- ***Example***: $F(x, y) = x'y' + xyz + x'y$

- After finding the circuit inputs and outputs, you can come up with either an expression or a truth table to describe what the circuit does.
- You can easily convert between expressions and truth tables.



Boolean Functions

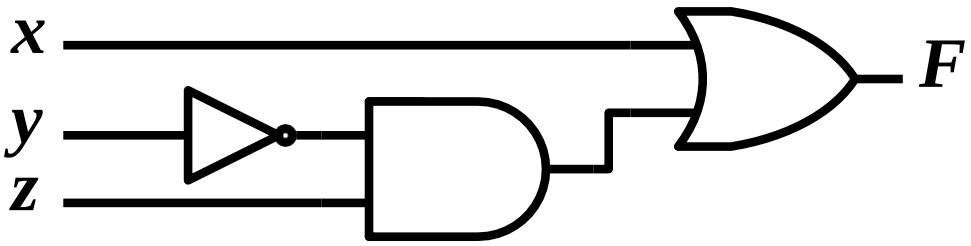
- **Boolean Expression/Function**

  Example: $F(x, y) = x + y' z$

- **Truth Table**

  All possible combinations of input variables

- **Logic Circuit**



x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Logic Diagrams and Expressions

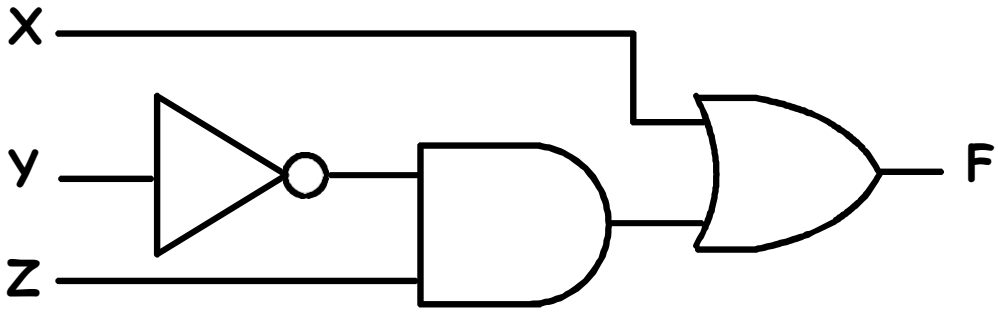
Truth Table

X Y Z	F = X + Y ×Z
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	1

Logic Equation/Boolean Function

$$F = X + \overline{Y}\,Z$$

Logic Circuit



- Boolean equations, truth tables and logic diagrams
- describe the same function!
  Truth tables are unique, but expressions and logic diagrams are not. This gives flexibility in implementing functions.

Boolean Functions Exercise

- The truth table for the function:

$$F(X, Y, Z) = XY + \overline{Y}Z \quad \textit{is:}$$

X	Y	Z	XY	$\overline{Y}$	$\overline{Y}Z$	$F = XY + \overline{Y}Z$
0	0	0	0	1	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	1	0	0
1	0	1	0	1	1	1
1	1	0	1	0	0	1
1	1	1	1	0	0	1

Draw the logic circuit for the boolean function above.

Verify the following Boolean algebraic manipulation. Justify each step with a reference to a postulate or theorem

(X+Y'+XY) (X+Y')X'Y=0

(AB+C+D)(C'+D)(C'+D+E)=ABC'+D

ABC+ABC'+AB'C+A'BC=AB+AC+BC

(a+b)(a'+c)(b+c)=(a+b)(a'+c)

Converting from Truth Table to Boolean Function

- In designing digital circuits, the designer often begins with a truth table describing what the circuit should do.
- The design task is largely to determine what type of circuit will perform the function described in the truth table.
- While some people seem to have a natural ability to look at a truth table and immediately envision the necessary logic gate or relay logic circuitry for the task, there are procedural techniques available for the rest of us.
- Here, Boolean algebra proves its utility in a most dramatic way!

Converting from Truth Table to Boolean Function

- This problem will be solved by showing that any Boolean function can be represented by a Boolean sum of Boolean products of the variables and their complements or the product of sums.

- There are two ways to convert from truth tables to Boolean functions:
  1. Using Sum of Products /Minterms
  2. Using Product of Sums /Maxterms

Canonical and Standard Forms

Logical functions – are generally expressed in terms of different combinations of logical variables with their true forms as well as the complement forms.

Arbitrary logic function

1. **Sum of the Products (SOP)**
2. **Product of the Sums (POS)**

Representation of Boolean expression

Boolean expression can be represented by either

(i)Sum of Product(SOP) form or

(ii)Product of Sum (POS form)

e.g.

AB+AC · SOP

(A+B)(A+C) · POS

In above examples both are in SOP and POS respectively but they are not in Standard SOP and POS.

Canonical form of Boolean Expression (Standard form)

- In standard SOP and POS each term of Boolean expression must contain all the literals (with and without bar) that has been used in Boolean expression.
- If the above condition is satisfied by the Boolean expression, that expression is called Canonical form of Boolean expression.

Canonical form of Boolean Expression (Standard form) Contd..

- In Boolean expression **AB+AC** the literal C is missing in the 1st term **AB** and B is missing in 2nd term **AC**. That is why AB+AC is not a Canonical SOP.

Canonical form of Boolean Expression (Standard form) Contd..

Convert AB+AC in Canonical SOP

Sol. AB + AC

AB(C+C′) + AC(B+B′)

ABC+ABC′+ABC+AB′C

ABC+ABC′+AB′C

Canonical form of Boolean Expression (Standard form) Contd..

Convert (A+B)(A+C) in Canonical SOP

Sol. (A+B).(A+C)

(A+B)+(C.C′) . (A+C)+(B.B′)

(A+B+C).(A+B+C′).(A+B+C)(A+B′+C) **Distributive law**

(A+B+C).(A+B+C′)(A+B′+C) **Remove duplicates**

Canonical form of Boolean Expression (Standard form) Contd..

Minterm and Maxterm

Individual term of Canonical Sum of Products (SOP) is called Minterm. In otherwords minterm is a product of all the literals (with or without bar) within the Boolean expression.

Individual term of Canonical Products of Sum (POS) is called Maxterm. In otherwords maxterm is a sum of all the literals (with or without bar) within the Boolean expression.

Minterms & Maxterms for 2 variables (Derivation of Boolean function from Truth Table)

x	y	Index	Minterm	Maxterm
0	0	0	$m_0 = x' y'$	$M_0 = x + y$
0	1	1	$m_1 = x' y$	$M_1 = x + y'$
1	0	2	$m_2 = x y'$	$M_2 = x' + y$
1	1	3	$m_3 = x y$	$M_3 = x' + y'$

The minterm m_i should evaluate to 1 for each combination of x and y.

The maxterm is the complement of the minterm

Minterms & Maxterms for 3 variables

x	y	z	Index	Minterm	Maxterm
0	0	0	0	$m_0 = \bar{x}\,\bar{y}\,\bar{z}$	$M_0 = x + y + z$
0	0	1	1	$m_1 = \bar{x}\,\bar{y}\,z$	$M_1 = x + y + \bar{z}$
0	1	0	2	$m_2 = \bar{x}\,y\,\bar{z}$	$M_2 = x + \bar{y} + z$
0	1	1	3	$m_3 = \bar{x}\,y\,z$	$M_3 = x + \bar{y} + \bar{z}$
1	0	0	4	$m_4 = x\,\bar{y}\,\bar{z}$	$M_4 = \bar{x} + y + z$
1	0	1	5	$m_5 = x\,\bar{y}\,z$	$M_5 = \bar{x} + y + \bar{z}$
1	1	0	6	$m_6 = x\,y\,\bar{z}$	$M_6 = \bar{x} + \bar{y} + z$
1	1	1	7	$m_7 = x\,y\,z$	$M_7 = \bar{x} + \bar{y} + \bar{z}$

Maxterm M_i is the complement of minterm m_i

$$M_i = \overline{m_i} \text{ and } m_i = \overline{M_i}$$

Minterms and Maxterms

Each individual term in canonical SOP form is called *minterm* and each individual term in canonical POS form is called *maxterm*. The concept of minterms and maxterms allows us to introduce a very convenient shorthand notations to express logical functions. Table 3.1 gives the minterms and maxterms for a three literal/variable logical function where the number of minterms as well as maxterms is $2^3 = 8$. In general, for an n-variable logical function there are 2^n minterms and an equal number of maxterms.

Variables			Minterms	Maxterms
A	B	C	m_i	M_i
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 FORM

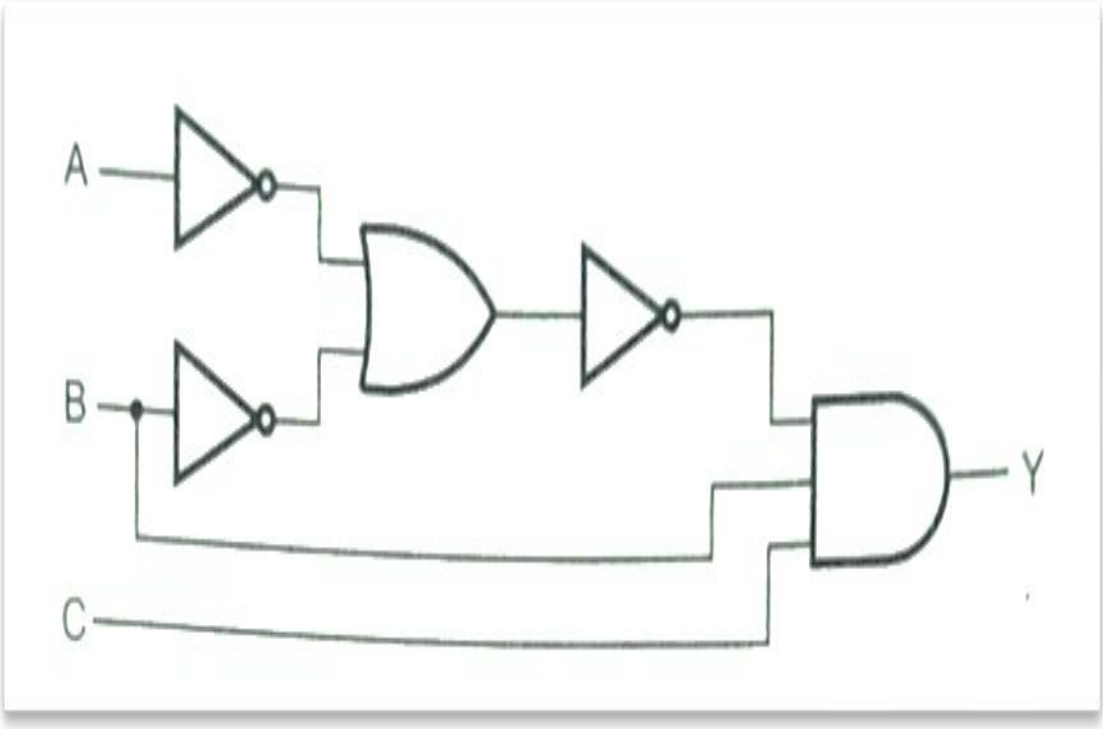
VERSUS

STANDARD FORM

CANONICAL FORM	STANDARD FORM
A representation that helps to describe Boolean outputs of digital circuits using Boolean functions	A simplified version of Canonical form
Divides into Canonical SoP form and Canonical PoS form	Divides into Standard SoP form and Standard PoS form
More complex	Simple

Examples

1. Write a Boolean Expression for the output Y in figure below. And determine the truth table



Example 2 Determine the truth table for the circuit shown below

Determine the truth table for the circuit shown in

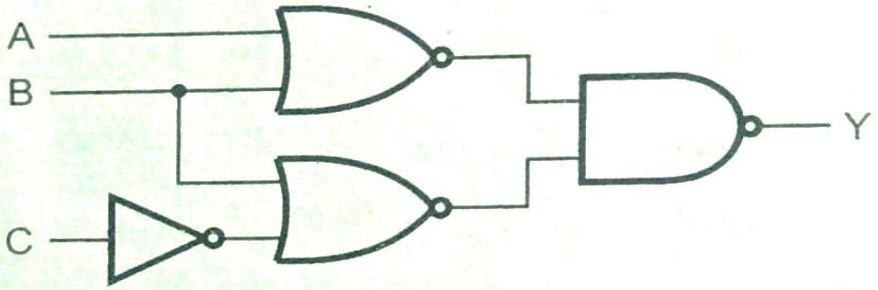


Fig. 3.67

$$Y = \overline{\overline{A+B}\cdot\overline{B+\overline{C}}}$$

$$= \overline{\overline{A+B}} + \overline{\overline{B+\overline{C}}} \qquad \text{De Morgan's Theorem 1}$$

$$= A + B + B + \overline{C} \qquad \text{Rule 9 : } \left[\overline{\overline{A}} = A\right]$$

$$= A + B + \overline{C} \qquad \text{Rule 5 : } [A + A = A]$$

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

Truth table 3.24

Example 3

Simplify the following expression

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

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

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

Rul

$$= A\overline{B}C + AC\overline{C} + \overline{A}B\overline{B} + \overline{A}B\overline{C} + B\overline{B}C + BC$$

Example 2: Simply Y = (A+B) (A'+C) (B'+C')

Y = AB'C + A'BC'

For the following expressions, construct the corresponding logic circuits, using AND/ OR/INVERTER logic.

Sol. : a) $Y = \overline{AB\,(C+D)}$

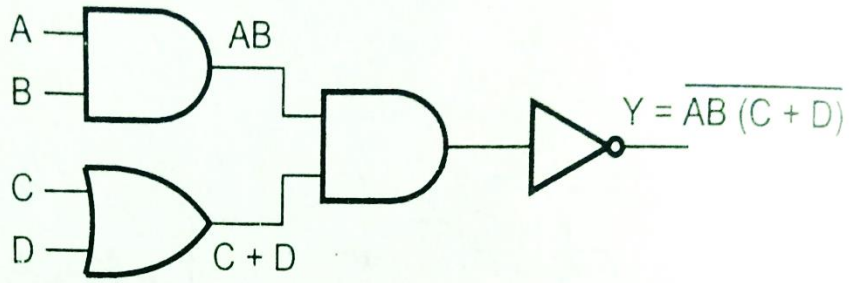
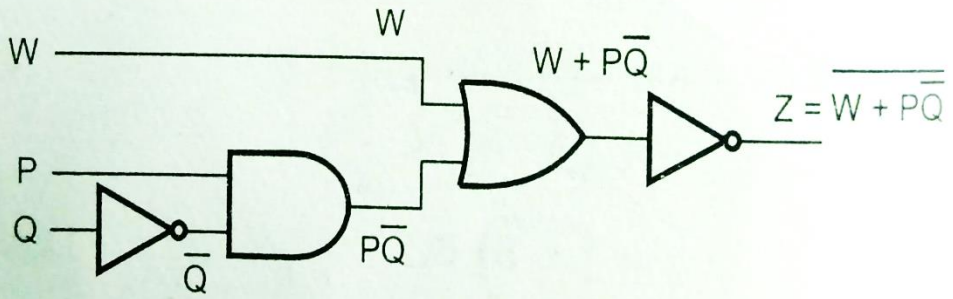


Fig. 3.65

b) $Z = \overline{W + P\overline{Q}}$



Canonical SOP and POS form

Convert the given expression in canonical SOP Form

Y=A+AB+ABC

Convert the expression into canonical POS form

$Y=(A+B)(B+C)(A+C)$

$Y= A(A+B)(A+B+C)$

1. $$Y = \bar{A}\,\bar{B}\bar{C} + \bar{A}\,\bar{B}C + \bar{A}\,BC + A\,B\bar{C}$$
$$= m_0 + m_1 + m_3 + m_6$$
$$= \sum m(0, 1, 3, 6)$$

2. $$Y = (A + B + \bar{C})\,(A + \bar{B} + \bar{C})\,(\bar{A} + \bar{B} + C)$$
$$= M_1 + M_3 + M_6$$
$$= \pi M\,(1, 3, 6)$$

where Σ denotes **sum of product** while π denotes **product of sum**.

We know that logical expression can be represented in the truth table form. It is possible to write logic expression in canonical SOP or POS form corresponding to a given truth table. The logic expression corresponding to a given truth table can be written in a canonical sum of products form by writing one product term for each input combination that produces an output of 1. These product terms are ORed together to create the canonical sum of products. Consider, for example, the following truth table :

A	B	C	Y	
0	0	0	0	
0	0	1	0	
0	1	0	1	← $\bar{A}B\bar{C}$
0	1	1	1	← $\bar{A}BC$
1	0	0	0	
1	0	1	0	
1	1	0	1	← $AB\bar{C}$
1	1	1	0	

A product of sums form derived from a truth table is logically equivalent to a sum of products form derived from the same truth table. Let us write the standard SOP and POS from the previous truth table :

SOP form : $Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C} + A\,B\,C$

POS form : $Y = (A + \overline{B} + C)\,(\overline{A} + B + \overline{C})$

Now by simplifying equation in the POS form we have,

$$\begin{aligned} Y &= A\,\overline{A} + A\,B + A\,\overline{C} + \overline{A}\,\overline{B} + B\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,C + B\,C + C\,\overline{C} \\ &= A\,B + A\,\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,C + B\,C \end{aligned}$$

Coverting to standard sum of products we have,

$$\begin{aligned} Y &= A\,B\,(C + \overline{C}) + A\,\overline{C}\,(B + \overline{B}) + \overline{A}\,\overline{B}\,(C + \overline{C}) + \overline{B}\,\overline{C}\,(A + \overline{A}) \\ &\quad + \overline{A}\,C\,(B + \overline{B}) + B\,C\,(A + \overline{A}) \\ &= A\,B\,C + A\,B\,\overline{C} + A\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,C \\ &\quad + \overline{A}\,\overline{B}\,C + A\,B\,C + \overline{A}\,B\,C \\ &= A\,B\,C + A\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,C \\ &= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C} + A\,B\,C \end{aligned}$$

Therefore, we can say that POS and SOP derived from the same truth table are logically equivalent. In terms of minterms and maxterms we can then write

$$\begin{aligned} Y &= m_0 + m_1 + m_3 + m_4 + m_6 + m_7 = M_2 + M_5 \\ \therefore \quad Y &= \sum m(0, 1, 3, 4, 6, 7) \\ &= \pi\, M(2, 5) \end{aligned}$$

From the above expressions, we can easily notice that there is a complementary type of relationship between a function expressed in terms of maxterms. Using this complementary Relationship, we can find logical function in terms of maxterms if function in minterms is known or Vice-versa.

For example, for a four variables if

$Y = \Sigma\, m(0,2,4,6,8,10,12,14)$

$Y = \pi\, M\,(1,3,5,7,9,11,13,15)$

Other Logic Operators

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary
$F_1 = A\ B$	A . B	AND	A and B
$F_2 = A\ B'$	A / B	Inhibition	A but not B
$F_3 = A$		Transfer	A
$F_4 = A'\ B$	B / A	Inhibition	B but not A
$F_5 = B$		Transfer	B
$F_6 = A\ B' + A'\ B$	A ⊕ B	Exclusive-OR	A or B but not both
$F_7 = A + B$	A + B	OR	A or B
$F_8 = (A + B)'$	A ↓ B	NOR	Not OR
$F_9 = A\ B + A'\ B'$	A B	Equivalence *	A equals B
$F_{10} = B'$	B'	Complement	Not B
$F_{11} = A + B'$	A ⊂ B	Implication	If B then A
$F_{12} = A'$	A'	Complement	Not A
$F_{13} = A' + B$	A ⊃ B	Implication	If A then B
$F_{14} = (A\ B)'$	A ↑ B	NAND	Not AND
$F_{15} = 1$		Identity	Binary constant 1

Digital Logic Gates

1. Tables of Logic gates
2. Extension to Multiple Inputs
3. Universal gates
4. Realization of Logic Functions by NAND Gates
5. Realization of Logic Functions by NOR Gates
6. Two-level Implementation of Logic Networks
7. Multilevel Gating Networks

Positive and Negative Logic

For a *positive logic* system, the most positive voltage level represents logic 1 state or HIGH level (H) and the lowest voltage level represents logic 0 state or LOW level (L).

For a *negative logic* system, the most positive voltage level represents logic 0 state and lowest voltage level represents logic 1 state.

3. **Other Important Theorems**

Theorem 1*(a)*: A + A = A

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

Theorem 2*(a)*: A + 1 = 1

Theorem 1*(b)*: A . A = A

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

Theorem 2*(b)*: A . 0 = 0

Theorem 3*(a)*: A + A . B = A

A + A . B = A . 1 + A . B
= A (1 + B)
= A . 1
= A

Theorem 3*(b)*: A (A + B) = A

Example:4: Show how Y=ABC’ can be implemented with one two-input NOR and one two input NAND gate

Implement Y=ABCD using two inputs NAND gates

i) Implement AB+C'D'=F with three NAND gates.
Draw the logic circuits

ii) Show that the NAND connections is not associative