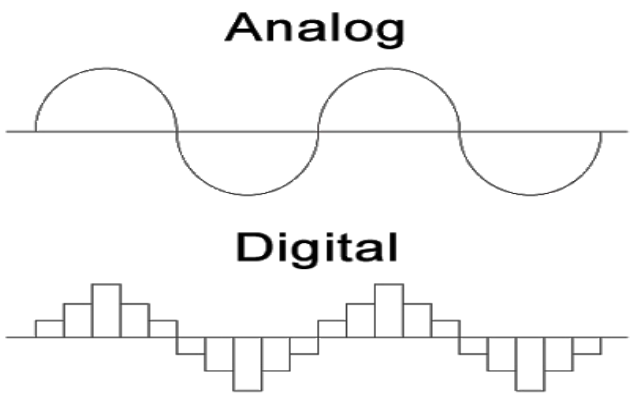
Digital Circuit
&
Logic Design
(CSDC-0101)

Number System, Boolean Algebra and Logic Gates

Numerical values can be represented by analog or digital.

A digital signal is a discrete signal (step by step), and an analog signal is a continuous signal.



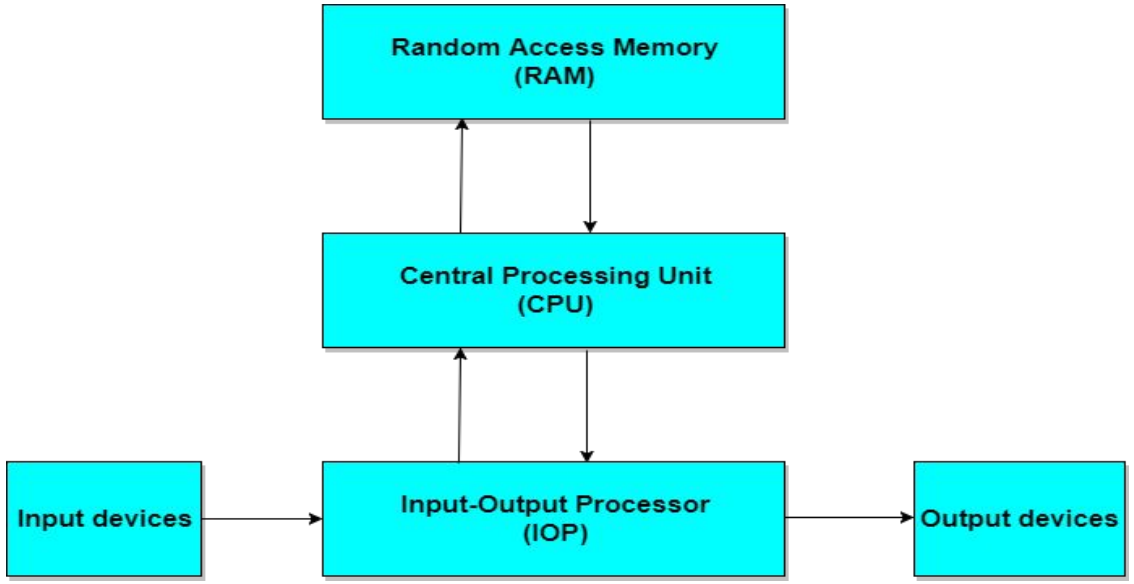
Digital means electronic technology that generates, stores, and processes data in terms of two states: positive (represented by the number 1) and non-positive (represented by the number 0).

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Computer Components

A computer consists of two components: **hardware and software**.

- **Hardware** refers to the physical components of a computer such as the keyboard, CPU, and memory.
- **Software** refers to programs run by CPU including operating systems and application programs.



Source:
Internet

The **input device** is used for entering information into memory. The input device converts information into bits, and the bits are stored in memory.

A computer's memory transfers information to the **output device** in the form of bits. The output device converts bits to characters, images, and voices which can be interpreted by humans.

Memory is used to store information and programs. Memory comes in the form of solid-state electronics such as RAM, ROM, flash drive, or hard disk.

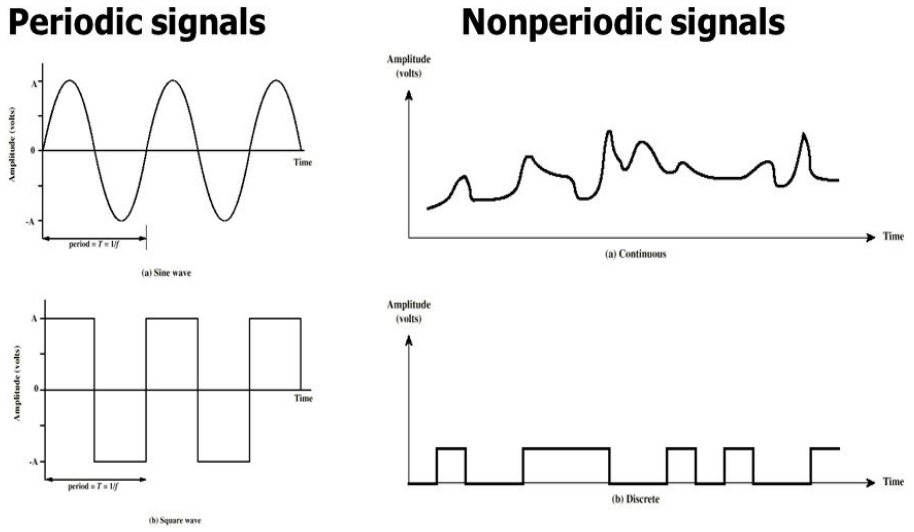
The **Central Processing Unit (CPU)** is used to perform computations on information.

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Analog Signals

An analog signal is a signal whose amplitude is a function of time and changes gradually as time changes.

- **Nonperiodic Signal:** In a nonperiodic signal, there is no repeated pattern in the signal.
- **Periodic Signal:** A signal that repeats a pattern within a measurable time period is called a periodic signal, and completion of a full pattern is called a cycle.



Characteristics of an Analog Signal

Frequency

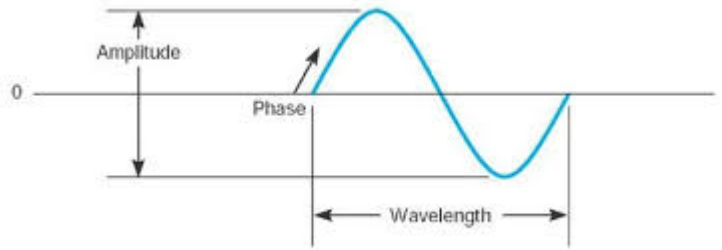
Frequency (F) is the number of cycles in 1 s, F =1/T, represented in Hz (Hertz). If each cycle of an analog signal is repeated every 1 s, the frequency of the signal is 1 Hz.

Amplitude

The amplitude of an analog signal is a function of time and may be represented in volts (unit of voltage). In other words, the amplitude is its voltage value at any given time. At the time t1, the amplitude of the signal is V1.

Phase

Two signals with the same frequency can differ in phase. This means that one of the signals starts at a different time from the other one. This difference can be represented in degrees (0° to 360°) or by radians. A phase angle of 0° indicates that the sine wave starts at time 0, and a phase angle of 90º indicates that the signal starts at 90°.



Digital Signals

Digital signals carry the data although it is a bit different. These signals are discrete or not continuous.

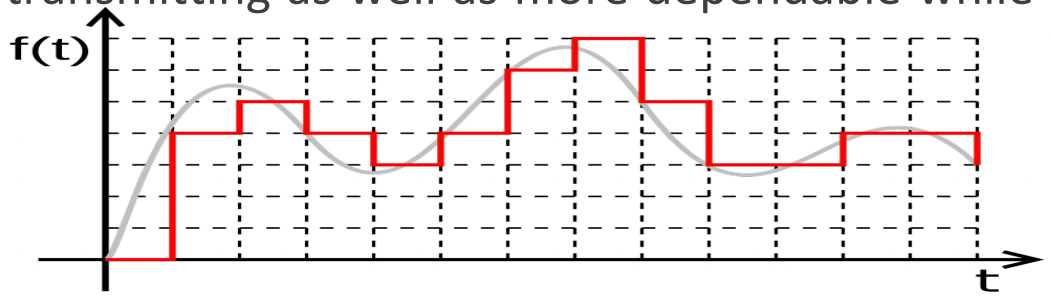
A digital signal carries the data in the form of binary because it signifies in the bits. These signals can be decomposed into sine waves which are termed as harmonics.

Every digital signal has amplitude, frequency, & phase like the analog signal.

This signal can be defined by bit interval as well as bit rate. Here, bit interval in nothing but the required time for transmitting an only bit, whereas the bit rate is bit interval frequency.

Digital signals are more resistant toward the noise; therefore, it barely faces some distortion.

These waves are simple in transmitting as well as more dependable while contrasted to analog waves.



Analog vs Digital

Analog information is made up of a continuum of values within a given range.

At its most basic, digital information can assume only one of two possible values: one/zero, on/off, high/low, true/false, etc.

Digital Information is less susceptible to noise than analog information

Exact voltage values are not important, only their class (1 or 0)

The complexity of operations is reduced, thus it is easier to implement them with high accuracy in digital form.

Analog Signal	Digital Signal
An analog signal signifies a continuous signal that keeps changes with a time period.	A digital signal signifies a discrete signal that carries binary data and has discrete values.
Analog signals are continuous sine waves.	Digital signal is square waves.
Analog signals describe the behavior of the wave with respect to amplitude, time period, & phase of the signal.	Digital signals describe the behavior of the signal with respect to the rate of a bit as well as bit interval.
Analog signal range will not be set.	Digital signal is limited as well as ranges from 0 to 1.
Analog signal is further horizontal toward distortion during the response to noise.	A digital signal has resistance in response toward the noise, therefore, it does not often face distortion.
An analog signal broadcasts the information in the signal form.	A digital signal broadcasts the information in the form of binary that is bits.
The example of an analog signal is the human voice.	The example of a digital signal is the data transmission in a computer.

Digital benefits

Digital representation is more accurate.

Digital information are easier to store

Digital systems are easier to design.

Noise has less effect.

Digital systems can easily be fabricated in an integrated circuit.

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Digital Systems

A **digital system** is a data technology that uses discrete (discontinuous) values represented by high and low states known as bits. By contrast, non-digital (or analog) systems use a continuous range of values to represent information.

Although digital representations are discrete, the information represented can be either discrete, such as numbers, letters or icons, or continuous, such as sounds, images, and other measurements of continuous systems.

Digital systems are widely used and its applications can be seen in computers, calculators, and cell phones.

In a digital system, information is transferred between components of the digital system in the form of digital signals.

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Number System

The number system is a way to represent or express numbers.

Based on the different symbol used to represent numbers, there are various types of number systems:

- The decimal number system
- The binary number system
- The octal number system and
- The hexadecimal number system
- Binary Coded Decimal or BCD Numbering System

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Converting from Binary to Decimal

- decimal = $d_0 \times 2^0 + d_1 \times 2^1 + d_2 \times 2^2 + \ldots$
- Example

Find the decimal value of 111001_2:

binary number:	1	1	1	0	0	1
power of 2:	2^5	2^4	2^3	2^2	2^1	2^0

$$111001_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 57_{10}$$

SOURCE: INTERNET

Converting from Decimal Integer to Binary

Conversion steps:

- Divide the number by 2.
- Get the integer quotient for the next iteration.
- Get the remainder for the binary digit.
- Repeat the steps until the quotient is equal to 0.

Example

Convert 13_{10} to binary :

Division by 2	Quotient	Remainder	Bit #
13/2	6	1	0
6/2	3	0	1
3/2	1	1	2
1/2	0	1	3

$13_{10} = 1101_2$

Converting Decimal Fraction to Binary

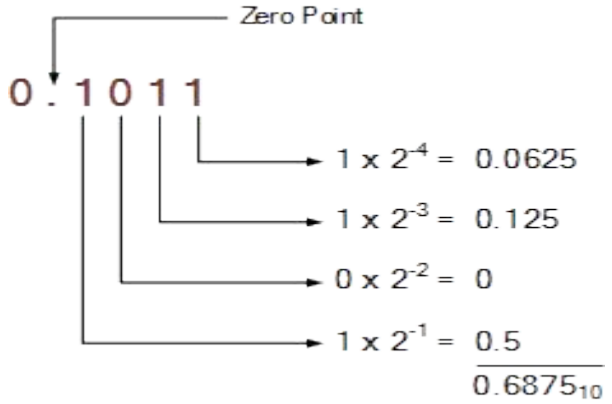
- Binary Fractions use the same weighting principle as decimal numbers except that each binary digit uses the base-2 numbering system.
- A decimal number representation of $(0.XY)_{10}$ can be converted into base of 2 and represented by $(0.\ a_1\ ,\ a_2\ ,\ a_3\ ,\ \text{etc.})_2$.
- The fraction number is multiplied by 2, the result of integer part is a_1 and fraction part multiply by 2, and then separate integer part from fraction, the integer part represents a_1 ; this process continues until the fraction becomes 0.

Example: $(0.625)_{10}$

		Integer		Fraction	Coefficient
0.625	* 2 =	1	.	25	$a_{-1} = 1$
0.25	* 2 =	0	.	5	$a_{-2} = 0$
0.5	* 2 =	1	.	0	$a_{-3} = 1$

Answer: $(0.625)_{10} = (0.a_{-1}\ a_{-2}\ a_{-3})_2 = (0.101)_2$

MSB LSB

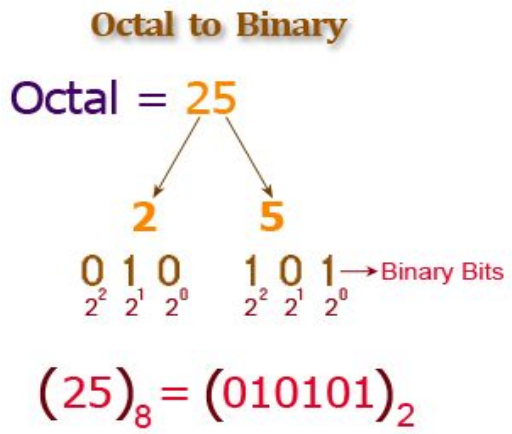


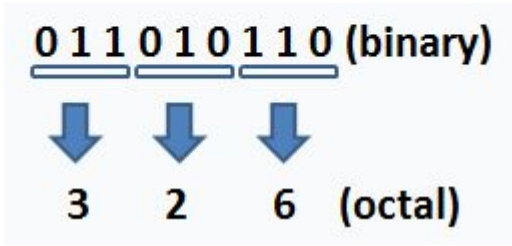
Converting from Octal to Binary

The octal numeral system, or oct for short, is the base-8 number system, and uses the digits 0 to 7.

Octal numerals can be made from binary numerals by grouping consecutive binary digits into groups of three (starting from the right).

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



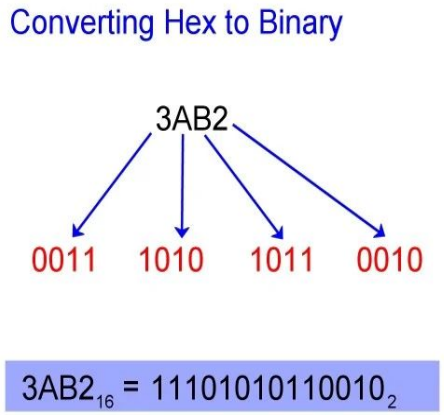


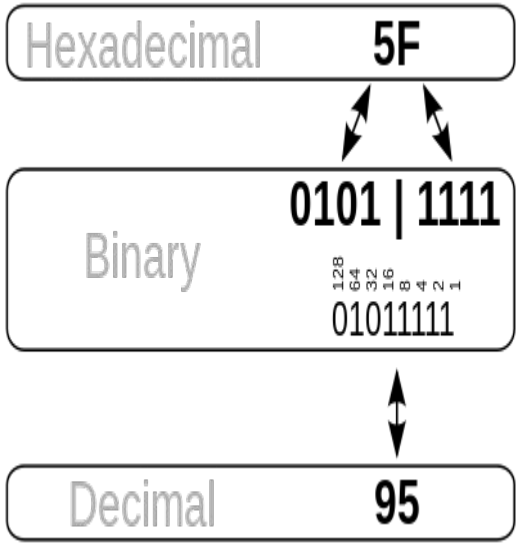
SOURCE: INTERNET

Converting from Hex to Binary

Hexadecimal (also base 16, or hex) is a positional numeral system with a radix, or base, of 16. It uses sixteen distinct symbols, most often the symbols 0–9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen.

Decimal	Hexadecimal	Binary
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111



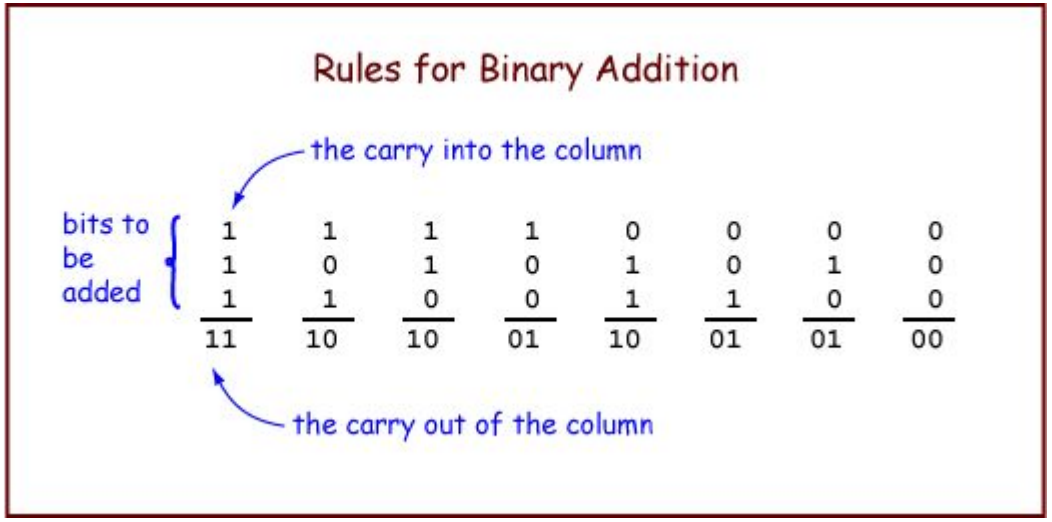


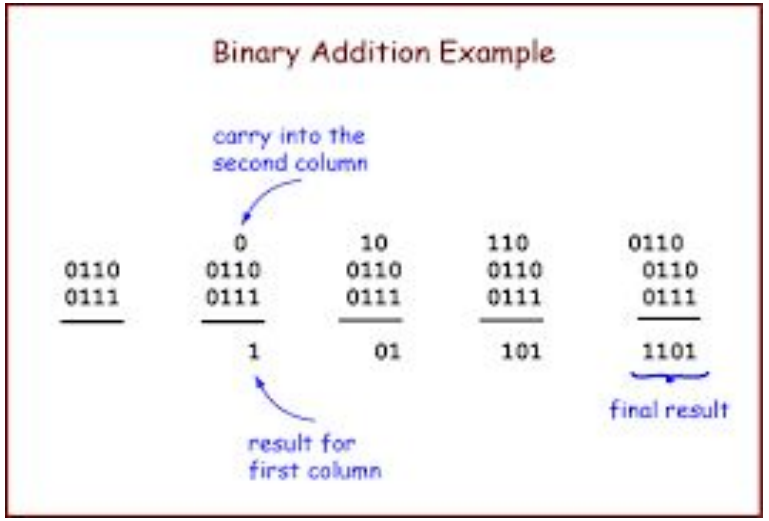
SOURCE: INTERNET

Binary Addition

Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

Example:





Binary Coded Decimal (BCD)

BCD or **Binary Coded Decimal** is that number system or code which has the binary numbers or digits to represent a decimal number where each digit is represented by a fixed number of binary bits, usually between four and eight.

A decimal number contains 10 digits (0-9). Now the equivalent binary numbers can be found out of these 10 decimal numbers. In case of BCD the binary number formed by four binary digits, will be the equivalent code for the given decimal digits. In BCD we can use the binary number from 0000-1001 only, which are the decimal equivalent from 0-9 respectively.

The BCD_{8421} code is so called because each of the four bits is given a 'weighting' according to its column value in the binary system. The least significant bit (lsb) has the weight or value 1, the next bit, going left, the value 2. The next bit has the value 4, and the most significant bit (msb) the value 8. E.g.,

24_{10} in 8 bit binary would be 00011000 but in BCD_{8421} is 0010 0100.

992_{10} in 16 bit binary would be 0000001111100000 but in BCD_{8421} is 1001 1001 0010.

Decimal	BCD	
0	0 0 0 0	
1	0 0 0 1	
2	0 0 1 0	
3	0 0 1 1	
4	0 1 0 0	
5	0 1 0 1	
6	0 1 1 0	
7	0 1 1 1	
8	1 0 0 0	
9	1 0 0 1	
-	1 0 1 0	Unused
-	1 0 1 1	
-	1 1 0 0	
-	1 1 0 1	
-	1 1 1 0	
-	1 1 1 1	

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Binary-Coded Decimal vs. Binary to Decimal Conversion

Decimal Number	BCD	Binary
0	0000 0000 0000 0000	0000 0000 0000 0000
1	0000 0000 0000 0001	0000 0000 0000 0001
2	0000 0000 0000 0010	0000 0000 0000 0010
3	0000 0000 0000 0011	0000 0000 0000 0011
4	0000 0000 0000 0100	0000 0000 0000 0100
5	0000 0000 0000 0101	0000 0000 0000 0101
6	0000 0000 0000 0110	0000 0000 0000 0110
7	0000 0000 0000 0111	0000 0000 0000 0111
8	0000 0000 0000 1000	0000 0000 0000 1000
9	0000 0000 0000 1001	0000 0000 0000 1001
⋮		
9620	1001 0110 0010 0000	0010 0101 1001 0100
120	0001 0010 0000	0000 0000 0111 1000
→ 4568	0100 0101 0110 1000	0001 0001 1101 1000

REALPARS

SOURCE: HTTPS://REALPARS.COM/BCD/

Binary Base-2	Decimal Base-10	Hexa-Decimal Base-16	Octal Base-8	BCD Code
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	8	8	10	8
1001	9	9	11	9
1010	10	A	12	---
1011	11	B	13	---
1100	12	C	14	---
1101	13	D	15	---
1110	14	E	16	---
1111	15	F	17	---

SOURCE: INTERNET

Logic Gates

Binary logic deals with binary variables and with operations that assume a logical meaning.

It is used to describe, in algebraic or tabular form, the manipulation and processing of binary information.

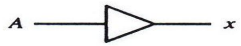
The manipulation of binary information is done by logic circuits called **gates** .

Gates are blocks of hardware that produce signals of binary 1 or 0 when input logic requirements are satisfied.

Each gate has a distinct graphic symbol and its operation can be described by means of an algebraic expression.

The input-output relationship of the binary variables for each gate can be represented in tabular form by a **truth table**.

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Name	Graphic symbol	Algebraic function	Truth table
AND		$x = A \cdot B$ or $x = AB$	A B | x 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$x = A + B$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$x = A'$	A | x 0 | 1 1 | 0
Buffer		$x = A$	A | x 0 | 0 1 | 1
NAND		$x = (AB)'$	A B | x 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$x = (A + B)'$	A B | x 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$x = A \oplus B$ or $x = A'B + AB'$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR or equivalence		$x = (A \oplus B)'$ or $x = A'B' + AB$	A B | x 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

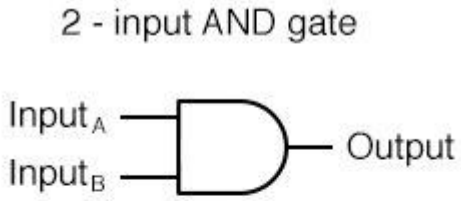
AND Gate

The AND gate produces the AND logic function: that is, the output is 1 if input A and input B are both equal to 1; otherwise, the output is 0.

These conditions are also specified in the truth table for the AND gate. The table shows that output x is 1 only when both input A and input B are 1.

The algebraic operation symbol of the AND function is the same as the multiplication symbol (.) of ordinary arithmetic.

AND gates may have more than two inputs, and by definition, the output is 1 if and only if all inputs are 1.



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

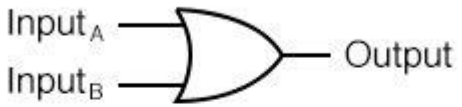
OR Gate

The OR gate produces the inclusive-OR function; that is, the output is 1 if input A or input B or both inputs are 1; otherwise, the output is 0.

The algebraic symbol of the OR function is +, similar to arithmetic addition.

OR gates may have more than two inputs, and by definition, the output is 1 if any input is 1.

2 - input OR gate



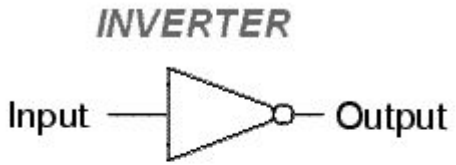
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

Inverter (NOT)

The inverter circuit inverts the logic sense of a binary signal.

It produces the NOT, or complement function.

The algebraic symbol used for the logic complement is either a prime or a bar over the variable symbol.



Input	Output
1	0
0	1

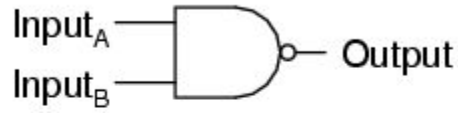
NAND Gate

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate.

The outputs of all NAND gates are high if any of the inputs are low.

The NAND function is the complement of the AND function, as indicated by the graphic symbol, which consists of an AND graphic symbol followed by a small circle.

NAND gate



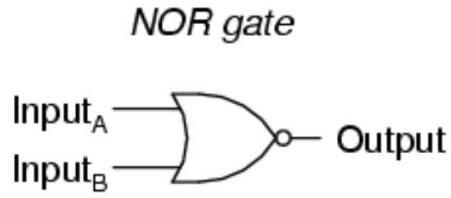
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate

This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate.

The outputs of all NOR gates are low if any of the inputs are high.

The NOR gate is the complement of the OR gate and uses an OR graphic symbol followed by a small circle.



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

Exclusive-OR (XOR)

It produces a high output if either, but not both, of its two inputs are high.

The exclusive-OR gate has a graphic symbol similar to the OR gate except for the additional curved line on the input side.

The output of this gate is 1 if any input is 1 but excludes the combination when both inputs are 1.

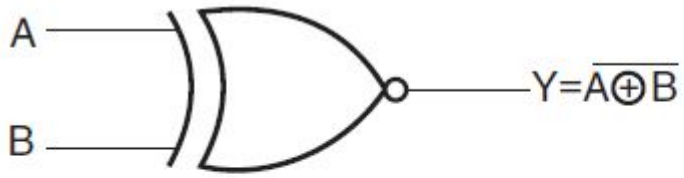
Exclusive-OR gate

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

Exclusive-NOR (XNOR)

The 'Exclusive-NOR' gate circuit does the opposite to the XOR gate.

It will give a low output if either, but not both, of its two inputs are high.



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

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

Boolean Algebra

Boolean algebra is an algebra that deals with binary variables and logic operations.

This forms the algebraic expression showing the operation of the logic circuit for each input variable either True or False that results in a logic “1” output.

The three basic logic operations are AND, OR, and complement (NOT).

The purpose of Boolean algebra is to facilitate the analysis and design of digital circuits. It provides a convenient tool to:

- Express in algebraic form a truth table relationship between binary variables.
- Express in algebraic form the input-output relationship of logic diagrams.
- Find simpler circuits for the same function.

Boolean algebra is the branch of algebra in which the values of the variables are the truth values true and false, usually denoted 1 and 0 respectively.

By manipulating a Boolean expression according to Boolean algebra rules, one may obtain a simpler expression that will require fewer gates.

It is used to analyze and simplify the digital (logic) circuits.

	AND Form	OR Form
Commutative Law	A . B = B . A	A + B = B + A
Associate Law	(A . B) . C = A . (B . C)	(A + B) + C = A + (B + C)
Distributive Law	(A+B)+ C = (A+C).(B+C)	(A + B) . C = (A . C) + (B . C)
Identity Law	A . 1 = A	A + 0 = A
Zero and One Law	A . 0 = 0	A + 1 = 1
Inverse Law	A . A' = 0	A + A' = 1
Idempotent Law	A . A = A	A + A = A
Absorption Law	A(A+B) = A	A + A.B = A A + A'B = A+B
DeMorgan's Law	(A . B)' = (A)' + (B)'	(A+ B)' = (A)' . (B)'
Double Complement Law	$\overline{\overline{x}} = x$	

SOURCE: MORRIS MANO, THIRD EDITION

Truth Table

- The relationship between a function and its binary variables can be represented in a **truth table**.
- A truth table shows how the truth or falsity of a compound statement depends on the truth or falsity of the simple statements from which it's constructed.
- It defines the function of a logic gate by providing a concise list that shows all the output states in tabular form for each possible combination of input variable that the gate could encounter.
- To represent a function in a truth table we need a list of the 2^n combinations of the **n** binary variables.

Boolean Function

Boolean function can be expressed algebraically with binary variables, the logic operation symbols, parentheses, and equal sign.

For a given value of the variables, the Boolean function can be either 1 or 0.

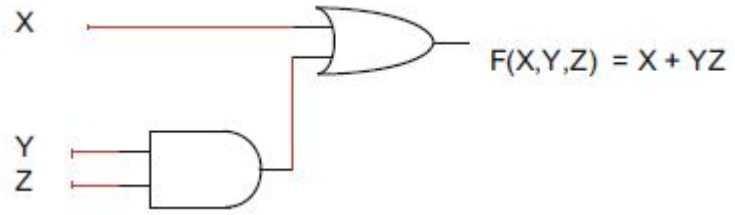
E.g.,

$$F = x + y' z$$

The function F is equal to 1 if x is 1 or if both y' and z are equal to 1; F is equal to 0 otherwise. But saying that y' = 1 is equivalent to saying that y = 0 since y' is the complement of y. Therefore, we may say that F is equal to 1 if x = 1 or if yz = 01.

.

F (X,Y,Z)= X+ YZ is a boolean function



X	Y	Z	YZ	X + YZ
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The truth table shows the characteristics of function F; the function F = 1 when the inputs to the circuit are 011 or 100 or 101 or 110 or 111

Logic Diagram

Boolean function can be transformed from an algebraic expression into a logic diagram composed of AND, OR, and inverter/NOT gates.

The logic diagram consists of gates and symbols that can directly replace an expression in Boolean arithmetic.

This is a graphical representation of a logic circuit that shows the wiring and connections of each individual logic gate, represented by a specific graphical symbol that implements the logic circuit.

Example:

$F(X, Y, Z) = XY'Z + XY'Z' + XY$

$F(X, Y, Z) = XY'(Z + Z') + XY$; where $Z + Z' = 1$ then

$F(X, Y, Z) = XY' + XY = X(Y + Y') = X$