DIGITAL LOGIC CIRCUITS

Logic Gates

Boolean Algebra

Map Specification

Combinational Circuits

Flip-Flops

Sequential Circuits

Memory Components

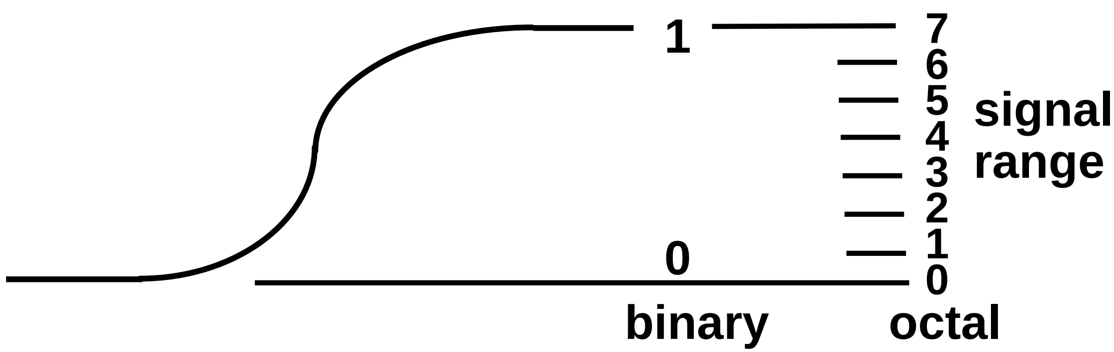
Integrated Circuits

LOGIC GATES

Digital Computers

- Imply that the computer deals with digital information, i.e., it deals with the information that is represented by binary digits
- Why *BINARY* ? instead of Decimal or other number system ?

* Consider electronic signal

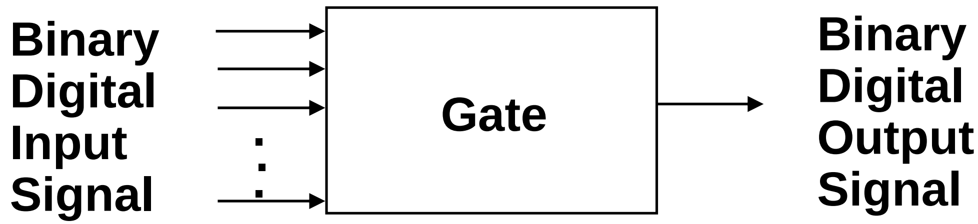


* Consider the calculation cost - Add

	0	1
0	0	1
1	1	10

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

BASIC LOGIC BLOCK - GATE -



Types of Basic Logic Blocks

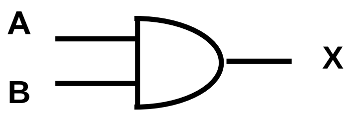
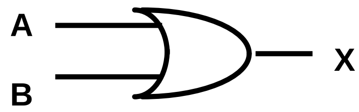
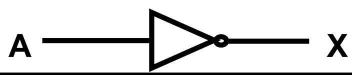
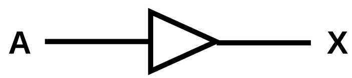
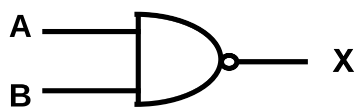
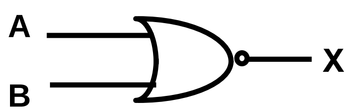
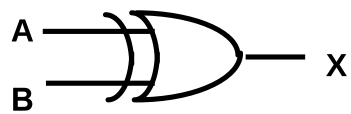
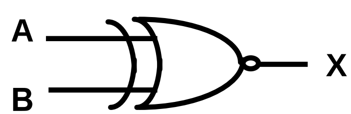
- **Combinational Logic Block**
  Logic Blocks whose output logic value depends only on the input logic values

- **Sequential Logic Block**
  Logic Blocks whose output logic value depends on the input values and the state (stored information) of the blocks

Functions of Gates can be described by

- **Truth Table**
- **Boolean Function**
- **Karnaugh Map**

COMBINATIONAL GATES

Name	Symbol	Function	Truth Table
AND		X = A • 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
I		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
XOR Exclusive OR		$X = A \oplus B$ or X = A’B + AB’	A B X 0 0 0 0 1 1 1 0 1 1 1 0
XNOR 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

BOOLEAN ALGEBRA

Boolean Algebra

* Algebra with Binary(Boolean) Variable and Logic Operations
* Boolean Algebra is useful in Analysis and Synthesis of Digital Logic Circuits

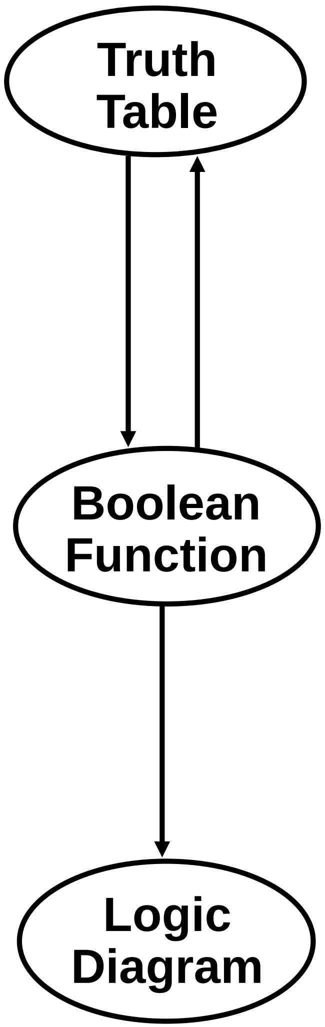
  - Input and Output signals can be represented by Boolean Variables, and
  - Function of the Digital Logic Circuits can be represented by Logic Operations, i.e., Boolean Function(s)
  - From a Boolean function, a logic diagram can be constructed using AND, OR, and I

Truth Table

* The most elementary specification of the function of a Digital Logic Circuit is the Truth Table

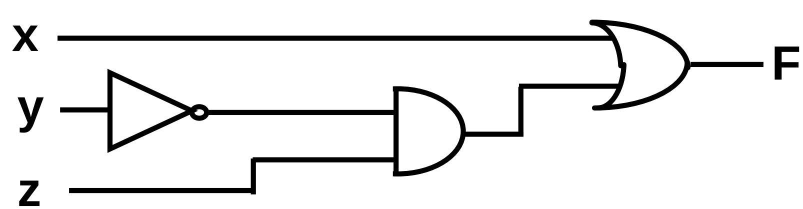
  - Table that describes the Output Values for all the combinations of the Input Values, called *MINTERMS*
  - n input variables $\rightarrow 2^n$ minterms

LOGIC CIRCUIT DESIGN



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

$F = x + y'z$



BASIC IDENTITIES OF BOOLEAN ALGEBRA

[1] $x + 0 = x$	[2] $x \bullet 0 = 0$
[3] $x + 1 = 1$	[4] $x \bullet 1 = x$
[5] $x + x = x$	[6] $x \bullet x = x$
[7] $x + x' = 1$	[8] $x \bullet X' = 0$
[9] $x + y = y + x$	[10] $xy = yx$
[11] $x + (y + z) = (x + y) + z$	[12] $x(yz) = (xy)z$
[13] $x(y + z) = xy + xz$	[14] $x + yz = (x + y)(x + z)$
[15] $(x + y)' = x'y'$	[16] $(xy)' = x' + y'$
[17] $(x')' = x$	

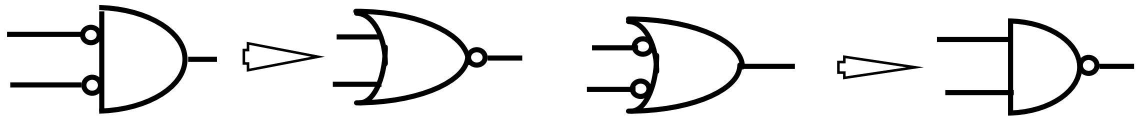
[15] and [16] : De Morgan's Theorem

Usefulness of this Table

- Simplification of the Boolean function
- Derivation of equivalent Boolean functions to obtain logic diagrams utilizing different logic gates
  - -- Ordinarily ANDs, ORs, and Inverters
  - -- But a certain different form of Boolean function may be convenient to obtain circuits with NANDs or NORs
  - → Applications of De Morgans Theorem

$x'y' = (x + y)'$ $x' + y' = (xy)'$

I, AND → NOR   **I, OR → NAND**

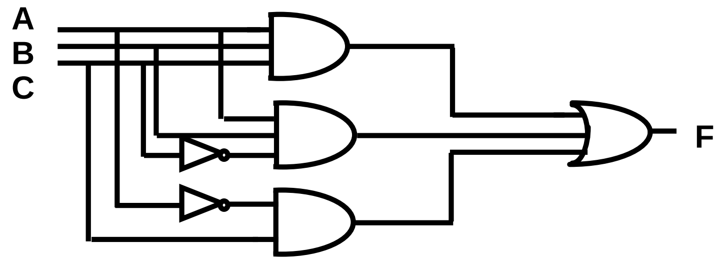


EQUIVALENT CIRCUITS

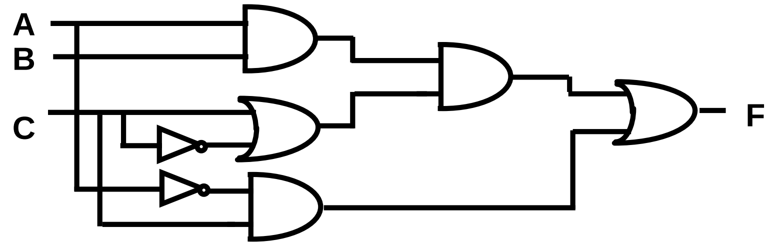
Many different logic diagrams are possible for a given Function

$$
\begin{aligned}
F &= ABC + ABC' + A'C & &\text{.............}\quad (1)\\
&= AB(C + C') + A'C & &[13] \text{......}\quad (2)\\
&= AB \bullet 1 + A'C & &[7]\\
&= AB + A'C & &[4]\ \text{.......}\quad (3)
\end{aligned}
$$

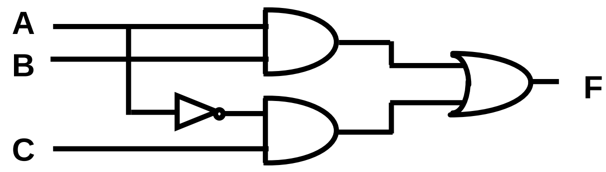
(1)



(2)



(3)



COMPLEMENT OF FUNCTIONS

A Boolean function of a digital logic circuit is represented by only using logical variables and AND, OR, and Invert operators.

→ Complement of a Boolean function

- Replace all the variables and subexpressions in the parentheses appearing in the function expression with their respective complements

$$A,B,\ldots,Z,a,b,\ldots,z \Rightarrow A',B',\ldots,Z',a',b',\ldots,z'$$
$$(p + q) \Rightarrow (p + q)'$$

- Replace all the operators with their respective complementary operators

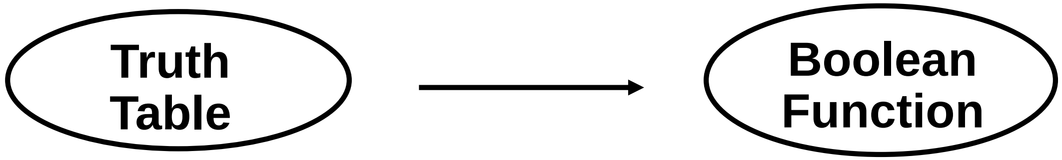
$$\text{AND} \Rightarrow \text{OR}$$
$$\text{OR} \Rightarrow \text{AND}$$

- Basically, extensive applications of the De Morgan's theorem

$$(x_1 + x_2 + \ldots + x_n)' \Rightarrow x_1'x_2'\ldots x_n'$$

$$(x_1x_2 \ldots x_n)' \Rightarrow x_1' + x_2' + \ldots + x_n'$$

SIMPLIFICATION



Unique **Many different expressions exist**

Simplification from Boolean function

- Finding an equivalent expression that is least expensive to implement
- For a simple function, it is possible to obtain a simple expression for low cost implementation
- But, with complex functions, it is a very difficult task

Karnaugh Map (K-map) is a simple procedure for simplifying Boolean expressions.