SYLLABUS (PART-A)

Combinational Circuits:

Review of switching algebra: Definitions, Theorems, Functions of n variable, Logic Detailed Diagram and Symbols minimization, Minimization Techniques: optimal combinations with K-map and tabular methods, simplification & minimization, complimentary approach with map method, map method for multi-output functions, Tabular and Iterative consensus method for obtaining prime implicants for single and multi-output functions.

Error Correction and Detection:

Error detection and correction techniques, Single error detection, Single error correction with double error

Fault detection and Location in combinational circuits:

Different methods of detecting and locating Faults in combinational circuits.

SYLLABUS (PART-B)

SECTION-B

Sequential Circuits: Synchronous circuits: Concept of state diagram and state table, state assignment, Analysis and synthesis of sequential circuits, designs of Next state decoder and output decoder, state reduction, Machine minimization of completely and incompletely specified machines.

Asynchronous Circuits: Analysis and Synthesis of Asynchronous circuits, Races and Cycles, hazards in asynchronous circuits. Sequential Machine Flow Charts, synthesis using sequential machine flow charts.

Fault detection and Location in sequential circuits.

TEXT BOOKS

1. Switching and Finite Automata Theory By: Kohavi, TMH publisher

2. Digital Circuits and Logic Design By: Lee, PHI publisher

BASICS OF DIGITAL:

Digital Circuit: An electronic circuit in which a state changes (switches) between two distinct states as there is a change in the input states.

There are two states/levels in the circuit as:

Low state = Level '0' = False = No

High state = Level '1' = True = Yes

Digital voltage: 0 to 5 volts

Level '0' is represented between voltage range: **0 - 1 volt**

Level '1' is represented between voltage range: **3.5 to 5 volts**

Value varies depending upon the type of Logic families (RTL, DTL, TTL, etc)

BASICS OF DIGITAL:

Two types of Logic:

1. **Positive Logic:** **Low = '0' and High = '1'**

 It is the default logic system.

2. **Negative Logic:** **Low = '1' and High = '0'**

BOOLEAN ALGEBRA AND LOGICAL OPERATORS

Algebra: variables, values, operations

In Boolean algebra, the values are the symbols 0 and 1

If a logic statement is false, it has value 0

If a logic statement is true, it has value 1

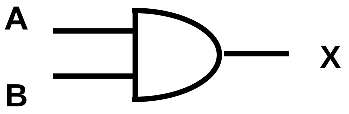
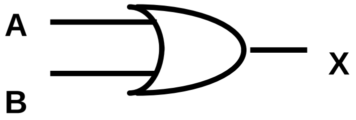
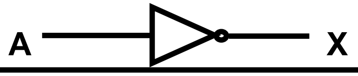
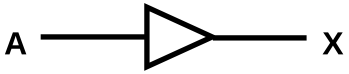
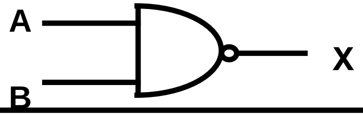
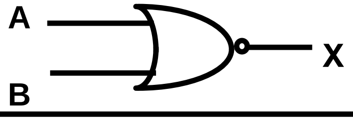
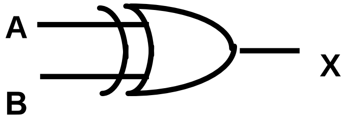
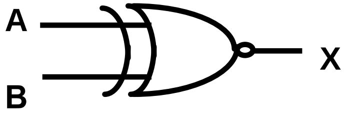
Operations: AND, OR, NOT

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

X	NOT X
0	1
1	0

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 ⊕ 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 ⊕ 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

- 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 INVERT

Truth Table

* The most elementary specification of the function.

- Table that describes the Output Values for all the combinations of the Input Values, called MINTERMS
- n input variables → $(2)^N$ minterms

REPRESENTATIONS OF DIGITAL DESIGN: BOOLEAN ALGEBRA

Values: 0, 1
Variables: A, B, C, . . ., X, Y, Z
Operations: NOT, AND, OR, . . .

NOT X is written as X'
X AND Y is written as X & Y, or sometimes X Y
X OR Y is written as X + Y

Deriving Boolean equations from truth tables:

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Sum = $\bar{A}\,B + A\,\bar{B}$

Carry = A B

OR'd together product terms
for each truth table
row where the function is 1

if input variable is 0, it appears in
complemented form;
if 1, it appears uncomplemented

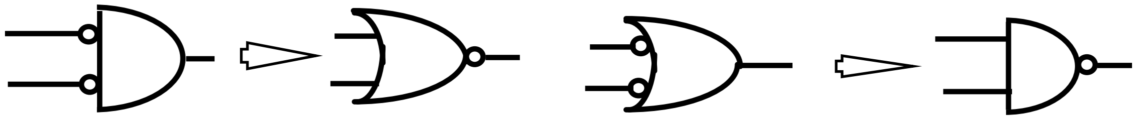
BASIC IDENTITIES OF BOOLEAN ALGEBRA

[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] $(x')' = x$	

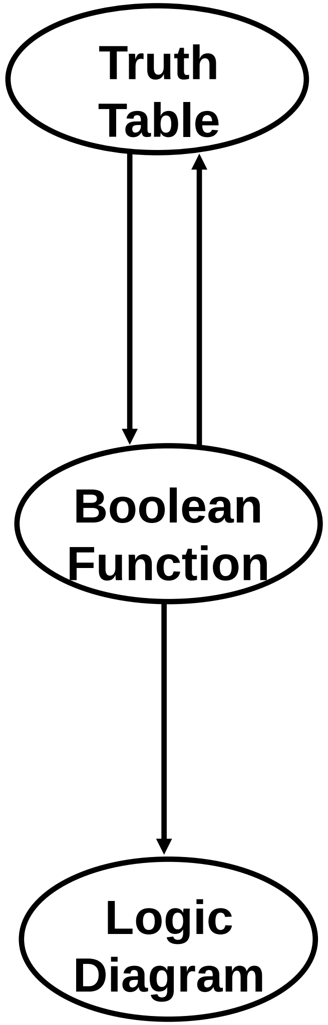
[15] and [16] : De Morgan's Theorem

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

I, AND → NOR I, OR → NAND

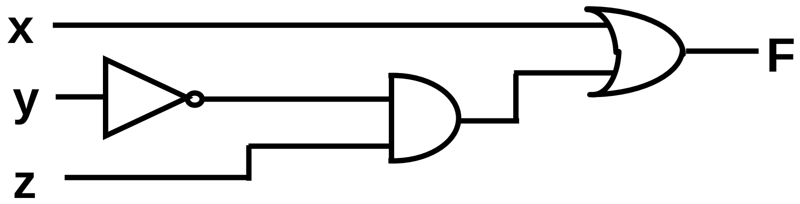


LOGIC CIRCUIT DESIGN



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

$F = x + y'z$

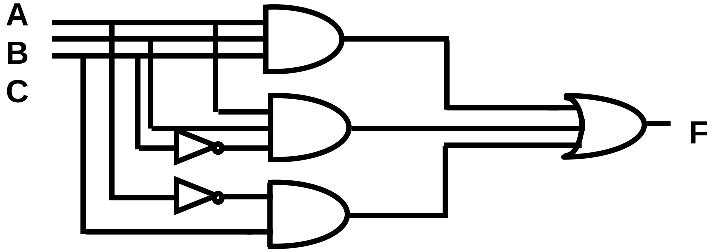


EQUIVALENT CIRCUITS

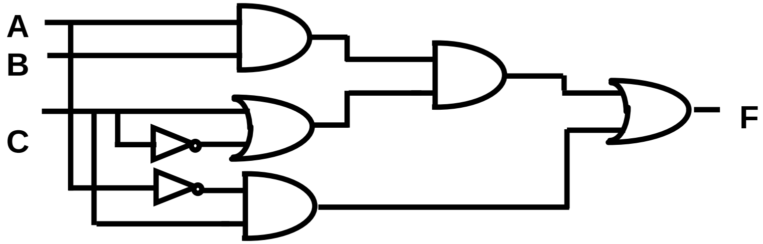
Many different logic diagrams are possible for a given Function

$$F = ABC + ABC' + A'C \qquad \ldots\ldots\ldots\ldots\ (1)$$
$$= AB(C + C') + A'C \qquad [13]\ \ldots\ldots\ (2)$$
$$= AB \bullet 1 + A'C \qquad [7]$$
$$= AB + A'C \qquad [4]\ \ldots\ldots\ (3)$$

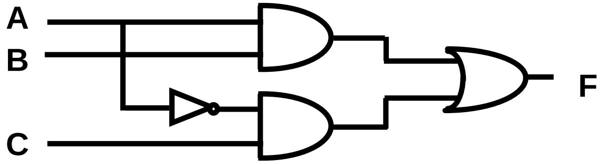
(1)



(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.

$\rightarrow$ Complement of a Boolean function

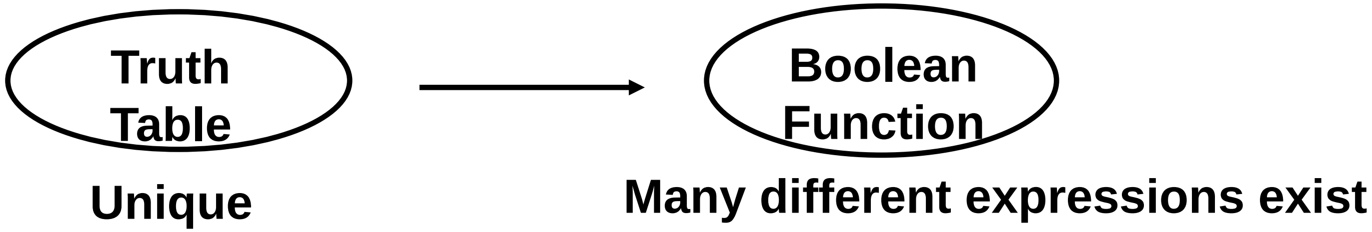
- Replace all the variables and subexpressions in the parentheses appearing in the function expression with their respective complements

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

- Replace all the operators with their respective complementary operators

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

SIMPLIFICATION



Simplification from Boolean function

- Finding an equivalent expression that is least expensive to implement
- For a simple function, it is easy
- But, with complex functions, it is a very difficult task

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