Date:	White Board Plan Digital Logic Design (UNIT-1) Lecture no:	Teaching Methodology:
Today's Topics: - Introduction of Boolean Algebra. - Basic Definitions	**Boolean Algebra and Logic Gates** - Mathematical methods that simplify circuits rely primarily on Boolean Algebra - To reduce the overall cost of the design, we need to make simple and cheap circuits for the implementation of binary logic. * Boolean Algebra will enable you to optimize simple circuits and to understand the purpose of algorithms. • Software tools utilize these algorithms in order to optimize complex circuits involving millions of logic gates.	**Industrial Applications:** • Digital computers • Digital Devices.
Glimpse of Next Lecture:		**Practical Applications:**
Text Books: 1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc. 2. Donald D. Givone (2002). Digital Principles and Design, TataMcGraw Hill	**Basic Definitions** - Boolean Algebra may be defined with a set of elements, a set of operators, and a number of unproved axioms or postulates. $S = \{1, 2, 3, 4\}$ | $x \in S \Rightarrow x$ is a member of S $y \notin S \Rightarrow y$ is not an element of S.	**Class Activities:** S. is a set x and y are certain objects
Reference Books: 1. C. V. S. Rao (2009). Switching and Logic Design.		**Summary:**

⇒ $a * b = c$

Here, * is a binary operator

↳ * is binary operator if & only if $(a, b, c) \in S$

↳ * is not binary operator if any one element $\notin S$.

like, $(a, b) \in S$ but $c \notin S$

① Closure :- A set S is closed w.r.t. a binary operator if, for every pair of elements of S, the binary operator specifies a rule ~~obtain~~ for obtaining a unique element of S. For eg -

$N = \{1, 2, 3, 4, \dots\}$ → set of Natural nos.

- N is closed w.r.t. binary operator + (addition)

  i.e $a + b = c$ (for every pair)

- N is not closed w.r.t. binary operator − (subtraction)

  i.e $a - b \neq c$ (for every pair).

  $3 - 2 = 1$, here $1, 2, 3 \in N$

  and, $2 - 3 = -1$, here $2, 3 \in N$ but $-1 \notin N$

② Associative Law :- A binary operator * on a set S is said to be associative whenever,

$$(x * y) * z = x * (y * z) \quad \forall\ x, y, z \in S$$

Date:	White Board Plan Digital Logic Design (UNIT-) Lecture no:	Teaching Methodology:
Today's Topics:		
	③ Commutative Law :- A binary operator * on a set S, is said to be ~~associative~~ commutative whenever, $x * y = y * x \quad \forall \; x, y \in S$	
		Industrial Applications:
	④ Identity element :- A set S is said to have an identity element w.r.t. a binary operation * on S if there exists an element $e \in S$ with the property that - $e * x = x * e = x \quad \forall \; x \in S$	
Glimpse of Next Lecture:		**Practical Applications:**
	Eg. ① set of Integers, $I = \{\ldots -3, -2, -1, 0, 1, 2, 3, \ldots\}$ Here, the element 0 is and identity element w.r.t. binary operator +, as - $x + 0 = 0 + x = x \quad \forall \; x \in I$	
Text Books:		**Class Activities:**
1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc. 2. Donald D. Givone (2002), Digital Principles and Design, TataMcGraw Hill	② set of ~~Natural~~ Natural numbers, N has no identity element.	
Reference Books:	⑤ Inverse :- A set S having the identity element e w.r.t. a binary operator * is said to have an inverse whenever, for every $x \in S$, there exists an element $y \in S$ such that $x * y = e$	**Summary:**
1. C. V. S. Rao (2009), Switching and Logic Design,		

Eg. In the set of integers, I, and the operator $+$, with $e = 0$, the inverse of an element a is $(-a)$, since

$$a + (-a) = 0$$

⑥ Distributive Law :- If $*$ and $\cdot$ are two binary operators on a set S, $*$ is said to be distributive over $\cdot$ whenever

$$x * (y \cdot z) = (x * y) \cdot (y * z)$$

Eg. $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

Summary

1) The binary operator $+$ defines addition
2) The additive identity is 0.
3) The additive inverse defines subtraction.
4) The binary operator $\cdot$ defines multiplication.
5) The multiplicative identity is 1.
6) For $a \neq 0$, the multiplicative inverse of $a = 1/a$ defines division.
7) The only distributive law applicable is that of $\cdot$ over $+$:-

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

Date:	White Board Plan Digital Logic Design (UNIT-) Lecture no:	Teaching Methodology:
Today's Topics:	**Axiomatic Definition of Boolean Algebra** Boolean algebra is an algebraic structure defined by a set of elements, B, together with two binary operators, + and •, provided that the following postulates are satisfied. 1. (a) The structure is closed w.r.t. the operator +. (b) The structure is closed w.r.t. the operator •. 2. (a) The element 0 is an identity element w.r.t. + i.e $x+0 = 0+x = x$ (b) The element 1 is an identity element w.r.t. • i.e $x \cdot 1 = 1 \cdot x = x$ 3. (a) The structure is commutative w.r.t. + i.e. $x+y = y+x$ (b) The structure is commutative w.r.t. • i.e $x \cdot y = y \cdot x$ 4. (a) The operator • is distributive over + i.e. $x \cdot (y+z) = (x \cdot y) + (x \cdot z)$ (b) The operator + is distributive over • i.e $x + (y \cdot z) = (x+y) \cdot (x+z)$	**Industrial Applications:** **Practical Applications:** **Class Activities:** **Summary:**
Glimpse of Next Lecture:		
Text Books: 1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc. 2. Donald D. Givone (2002), Digital Principles and Design, TataMcGraw Hill		
Reference Books: 1. C. V. S. Rao (2009), Switching and Logic Design,		

5. For every element $x \in B$, there exists an element $x' \in B$ such that
   (a) $x + x' = 1$
   (b) $x \cdot x' = 0$
6. There exist atleast two elements $x, y \in B$ such that $x \neq y$.

Basic Theorems and Properties of Boolean Algebra

(1) Duality : It states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and the elements ~~of the set B~~ are interchanged.

Example
1.) (a) $x + 0 = x$
   (b) $x \cdot 1 = x$
2.) (a) $x + x' = 1$
   (b) $x \cdot x' = 0$

Basic Theorems

6 Theorems of Boolean algebra and 4 of its postulates are list in the following table.

Table : Postulates and Theorems of Boolean Algebra

• Postulate 2	(a) $x + 0 = x$	(b) $x \cdot 1 = x$
• Postulate 5	(a) $x + x' = 1$	(b) $x \cdot x' = 0$
Theorem 1	(a) $x + x = x$	(b) $x \cdot x = x$
Theorem 2	(a) $x + 1 = 1$	(b) $x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
• Postulate 3, commutative	(a) $x + y = y + x$	(b) $xy = yx$
Theorem 4, associative	(a) $x + (y + z) = (x + y) + z$	(b) $x(yz) = (xy)z$
• Postulate 4, distributive	(a) $x(y + z) = xy + xz$	(b) $x + yz = (x + y)(x + z)$
Theorem 5, De Morgan	(a) $(x + y)' = x'y'$	(b) $(xy)' = x' + y'$
Theorem 6, absorption	(a) $x + xy = x$	(b) $x(x + y) = x$

Date: 07/10/22 (AI&ML)

White Board Plan
Digital Logic Design (UNIT-)
Lecture no:

Today's Topics:

- Boolean Functions

Glimpse of Next Lecture:

- Canonical and Standard Forms

Text Books:

1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc.

2. Donald D. Givone (2002), Digital Principles and Design, TataMcGraw Hill

Reference Books:

1. C. V. S. Rao (2009), Switching and Logic Design,

Boolean Functions

- Boolean algebra is an algebra that deals with variables and logic operations.
- A Boolean function described by an algebraic expression consists of binary variables, the constants 0 and 1, and the logic operation symbols.
- For a given value of the binary variables, the function can be equal to either 1 or 0.

For eg.-

$$F_1 = x + y'z$$

$$F_2 = x'y'z + x'yz + xy'$$

Truth Table for F_1 and F_2

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

Teaching Methodology:

Industrial Applications:

- Digital Computers
- Electronic appliances

Practical Applications:

Class Activities:

Summary:

- A Boolean function can be transformed from an algebraic expression into a circuit diagram composed of logic gates connected in a particular structure.
- The logic diagram is also called a schematic.

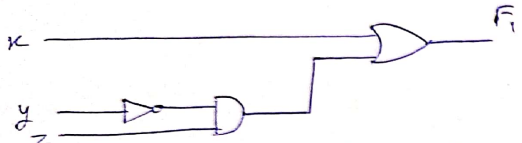


Fig 1 Gate implementation of F_1

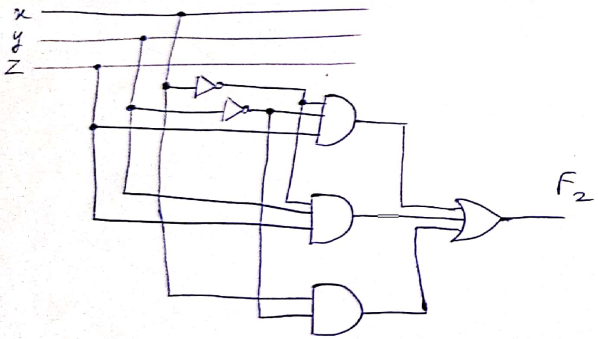
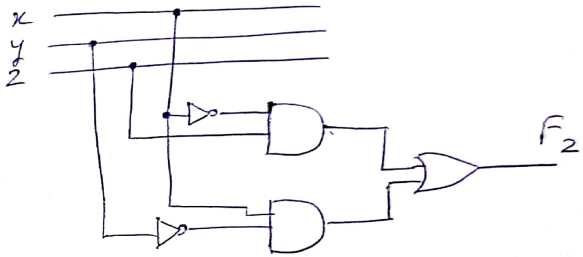


Fig 2. Gate implementation of F_2

Now, according to the rules of Boolean algebra the Boolean function F_2 can be simplified as,

$$F_2 = x'y'z + x'yz + xy'$$
$$= x'z(y'+y) + xy'$$
$$= x'z + xy'$$



Advantage
- Fewer gates
- Fewer wires
- Reduces the costing
- simpler circuit

Note : ① Function $F_2 = x'y'z + x'yz + xy'$ has 3 terms and 8 literals.

② Function $F_2 = x'z + xy'$ has 2 terms and 4 literals.

Date:	White Board Plan Digital Logic Design (UNIT-) Lecture no:	Teaching Methodology:
Today's Topics:	• The algebraic manipulation of Boolean algebra helps in finding the simplified expression. Examples Simplify the Boolean functions to a minimum no. of literals. 1) $x(x'+y)$ 2) $x+x'y$ 3) $(x+y)(x+y')$ 4) $xy+x'z+yz$ 5) $(x+y)(x'+z)(y+z)$ Sol 1) $x(x'+y) = xx'+xy = 0+xy = xy$ 2) $x+x'y = (x+x')(x+y)$ [using Postulate 4(b)] $= 1.(x+y)$ $= x+y$ 3) $(x+y)(x+y') = x+yy' = x$ 4) $xy+x'z+yz = xy+x'z+yz(x+x')$ $= xy+x'z+xyz+x'yz$ $= xy(1+z)+x'z(1+y)$ $= xy+x'z$ 5) $(x+y)(x'+z)(y+z) = (x+y)(x'+z)$	**Industrial Applications:** **Practical Applications:** **Class Activities:** **Summary:**
Glimpse of Next Lecture:		
Text Books: 1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc. 2. Donald D. Givone (2002), Digital Principles and Design, TataMcGraw Hill		
Reference Books: 1. C. V. S. Rao (2009), Switching and Logic Design,		

- Complement of a function F : F' and is obtained from an interchange of 0's for 1's and 1's for 0's in the value of F.
- The complement of a function can be derived algebraically through DeMorgan's Theorems. For eg -

Let $F = A + B + C$. Then

$$\left\{ \begin{aligned} F' &= (A+B+C)' \\ &= A'.B'.C' \end{aligned} \right\} \text{ OR } \left\{ \begin{aligned} \bar{F} &= \overline{A+B+C} \\ &= \bar{A}.\bar{B}.\bar{C} \end{aligned} \right\}$$

- The generalized form of DeMorgan's Theorems states that the complement of a function is obtained by interchanging AND and OR operators, and complementing each literal.

Examples

1) Find the complement of the functions using DeMorgan's Theorem

(a) $F_1 = x'yz' + x'y'z$ (b) $F_2 = x(y'z' + yz)$

Sol. (a) $F_1' = (x'yz' + x'y'z)'$

$= (x'yz')'(x'y'z)'$

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

(b) $F_2' = [x(y'z' + yz)]'$

$= x' + (y'z' + yz)'$

$= x' + (y'z')'(yz)'$

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

$= x' + yz' + y'z$

2) Find the complement of the functions using Duality Principle

(a) $F_1 = x'yz' + x'y'z$

(b) $F_2 = x(y'z' + yz)$

Sol. (a) $F_1 = x'yz' + x'y'z$

The dual of F_1 is $(x'+y+z')(x'+y'+z)$

Complementing each literal, we get

$(x+y'+z)(x+y+z') = F_1'$

(b) $F_2 = x(y'z' + yz)$

The dual of F_2 is $x + (y'+z')(y+z)$

Complementing each literal, we get

$x' + (y+z)(y'+z') = F_2'$

Date:	White Board Plan Digital Logic Design (UNIT-) Lecture no:	Teaching Methodology:
Today's Topics:	✱ Canonical and Standard Forms	
	• Minterms :- 'n' variables forming an AND term, with each variable being primed or unprimed, provide 2^n possible combinations are called as minterms (or standard products).	**Industrial Applications:**
Glimpse of Next Lecture:	• Maxterms :- 'n' variables forming an OR term, with each variable being primed or unprimed, provide 2^n possible combinations are called as maxterms (or standard sums).	**Practical Applications:**
Text Books:	Table 1 : Minterms and Maxterms for 3 Binary Variables	**Class Activities:**
1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc. 2. Donald D. Givone (2002), Digital Principles and Design, TataMcGraw Hill		**Summary:**
Reference Books:		
1. C. V. S. Rao (2009), Switching and Logic Design,		

x	y	z	Minterms		Maxterms	
			Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x+y+z$	M_0
0	0	1	$x'y'z$	m_1	$x+y+z'$	M_1
0	1	0	$x'yz'$	m_2	$x+y'+z$	M_2
0	1	1	$x'yz$	m_3	$x+y'+z'$	M_3
1	0	0	$xy'z'$	m_4	$x'+y+z$	M_4
1	0	1	$xy'z$	m_5	$x'+y+z'$	M_5
1	1	0	xyz'	m_6	$x'+y'+z$	M_6
1	1	1	xyz	m_7	$x'+y'+z'$	M_7

Note :- Each maxterm is the complement of its corresponding minterm and vice-versa.

⇒ A Boolean function can be expressed algebraically from a given Truth Table.

Table 2: Given Truth Table for 3 variables

x	y	z	Function f_1	Function, F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$f_1 = m_1 + m_4 + m_7$$
$$= x'y'z + xy'z' + xyz$$

$$F_2 = M_0 \cdot M_1 \cdot M_2 \cdot M_4$$
$$= (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)$$

Note: Total no. of functions formed with 'n' variables $= 2^{2^n}$

Meaning :- Let's say $n = 2$, then

$2^{2^n} = 2^{2\times 2} = 16$

Total 16 functions can be formed with 2 variables

Sum of Products (SOP's)

- Each of the 2^{2^n} functions of 'n' binary variables can be expressed as sum-of-products (SOPs)
- To express a Boolean function as a SOPs, it must be brought into a form of AND terms.

Example :- Express the Boolean function $f = A + B'C$ as a sum of minterms or SOPs.

Sol. Given, $f = A + B'C$

This function has 3 variables.

$$\therefore f = A(B+B')(C+C') + B'C(A+A')$$
$$= (AB+AB')(C+C') + AB'C + A'B'C$$
$$= ABC + ABC' + \underline{AB'C} + AB'C' + \underline{AB'C} + A'B$$
$$= ABC + ABC' + AB'C + AB'C' + A'B'C$$

($\because x + x = 1$)

$$= A'B'C + AB'C + AB'C' + ABC' + ABC$$
$$f = m_1 + m_4 + m_5 + m_6 + m_7$$
$$\therefore f = \Sigma(m_1, m_4, m_5, m_6, m_7)$$

OR

$$f = \Sigma(1, 4, 5, 6, 7)$$

OR

$$\therefore f(A, B, C) = \Sigma(m_1, m_4, m_5, m_6, m_7)$$
$$= \Sigma(1, 4, 5, 6, 7)$$

Date:	White Board Plan Digital Logic Design (UNIT-) Lecture no:	Teaching Methodology:
Today's Topics:	✱ Product of Sums (POS's) - Each of the 2^{2^n} functions of 'n' binary variables can be expressed as product-of-sums (POS's) - To express a Boolean function as a POS's, it must be brought into a form of OR terms. Example:- Express the Boolean function $F = AB + A'C$ as a POS's or product of max terms. Sol. Given, $F = AB + A'C$ • Use Distributive Law i.e $x+yz = (x+y)(x+z)$ $\Rightarrow F = (AB + A')(AB + C)$ $= (A'+AB)(C+AB)$ $= (A'+A)(A'+B)(C+A)(C+B)$ $= (A'+B)(A+C)(B+C)$ $= (A'+B+CC')(A+C+BB')(B+C+AA')$ [$\because xx' = 0$] $= (A'+B+C)(A'+B+C')(A+B+C)(A+B'+C)$ $(A+B+C)(A'+B+C)$ $= (A'+B+C)(A'+B+C')(A+B+C)(A+B'+C)$ $= (A+B+C)(A+B'+C)(A'+B+C)(A'+B+C')$	
Glimpse of Next Lecture:		**Industrial Applications:**
		Practical Applications:
Text Books: 1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc. 2. Donald D. Givone (2002), Digital Principles and Design, TataMcGraw Hill		**Class Activities:**
Reference Books: 1. C. V. S. Rao (2009), Switching and Logic Design,		**Summary:**

$\Rightarrow F = M_0 M_2 M_4 M_5$

~~$= \pi(0,2,4,5)$~~

Also $F(A,B,C) = \pi(M_0, M_2, M_4, M_5)$

$= \pi(0,2,4,5)$

Note: ① Σ denotes the ORing of minterms, and the numbers are the minterms of the function.

② π denotes the ANDing of maxterms, and the numbers are the maxterms of the function.

* Conversion between Canonical Forms

The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function. For eg -

① Given, $F(A,B,C) = \Sigma(1,4,5,6,7)$

Then the complement of this function is expressed as -

$F'(A,B,C) = \Sigma(0,2,3) = m_0 + m_2 + m_3$

Now, if we take the complement of F' by DeMorgan's Theorem, we get

$F = (M_0 + m_2 + m_3)'$ ~~$= M_0 M_2 M_3$~~

$= m_0' \cdot m_2' \cdot m_3'$

$= M_0 M_2 M_3$

$= \pi(0,2,3)$

Note: To convert from one canonical form to another, interchange the symbols Σ and π, and list those numbers missing from the original form.

Example

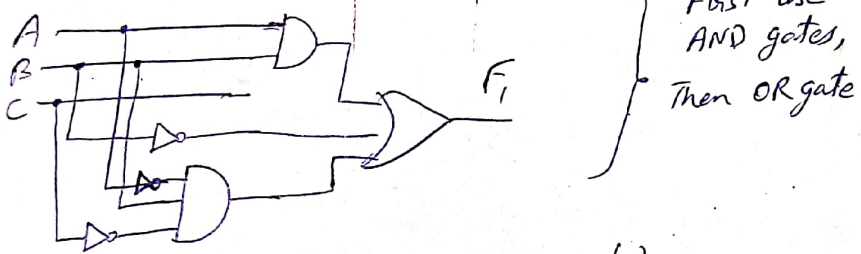
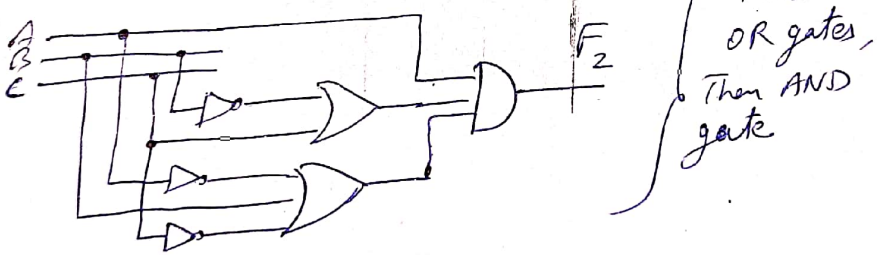
Make Truth Table for the Boolean expression $F = xy + x'z$. Find canonical forms.

Sol.

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

$F(x,y,z) = \Sigma(1,3,6,7)$

& $F(x,y,z) = \pi(0,2,4,5)$

Date:	White Board Plan Digital Logic Design (UNIT-) Lecture no:	Teaching Methodology:
Today's Topics:	• Standard Forms ① Sum of Products (SOP's) ② Product of Sums (POS's) ~~$F = A' + BC$~~ $F_1 = B' + AB + A'BC'$  First use AND gates, Then OR gate $F_2 = A(B'+C)(A'+B+C')$  First use OR gates, Then AND gate	**Industrial Applications:** **Practical Applications:** **Class Activities:** **Summary:**
Glimpse of Next Lecture:		
Text Books: 1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc. 2. Donald D. Givone (2002), Digital Principles and Design,TataMcGraw Hill		
Reference Books: 1. C. V. S. Rao (2009), Switching and Logic Design.		

Other Logic Operations

- We know that for 'n' binary variables, there are 2^{2^n} functions. Let $n = 2$, the no. of possible Boolean functions is 16.
- The truth tables for the 16 functions formed with 2 binary variables are listed in table below :-

A	B	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- The 16 functions can be expressed algebraically by means of Boolean Functions, as shown in the first column of the table below.

Note : The Boolean expressions are simplified to their minimum number of literals.

Date:	White Board Plan Digital Logic Design (UNIT-) Lecture no:	Teaching Methodology:
Today's Topics:	Table :- Boolean Expressions for the 16 Functions of Two Variables	

Boolean Functions	Operator symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x.y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x+y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

Glimpse of Next Lecture:

Text Books:

1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc.
2. Donald D. Givone (2002), Digital Principles and Design, TataMcGraw Hill

Reference Books:

1. C. V. S. Rao (2009), Switching and Logic Design,

Industrial Applications:

Practical Applications:

Class Activities:

Summary:

The 16 functions listed can be subdivided into 3 categories -

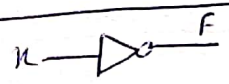
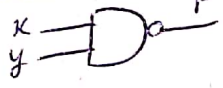
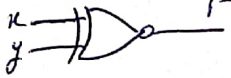
1) Two functions that produce a constant 0 or 1.
2) Four functions with unary operations: complement and transfer.
3) Ten functions with binary operators that define eight different operations: AND, OR, NAND, NOR, XOR, equivalence, inhibition, and implication.

Digital Logic Gates

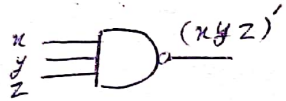
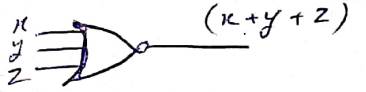
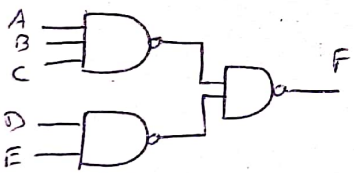
- It is easier to implement a Boolean function with AND, OR, and NOT logic gates.
- Factors to be weighted in considering the construction of other types of logic gates are -

(1) The feasibility and economy of producing the gate with physical components.
(2) The possibility of extending the gate to more than two inputs.
(3) The basic properties of the binary operator, such as commutativity and associativity,
(4) The ability of the gate to implement Boolean functions alone or in conjunction with other gates.

Table : Digital Logic Gates

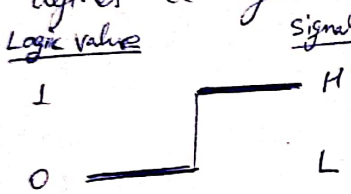
Name	Graphic Symbol	Algebraic Function	Truth Table
AND		$F = xy$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	x | F 0 | 1 1 | 0
Buffer		$F = x$	x | F 0 | 0 1 | 1
NAND		$F = (xy)'$	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = (x + y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = x \oplus y$ $= xy' + x'y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR or equivalence		$F = (x \oplus y)'$ $= xy + x'y'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Date:

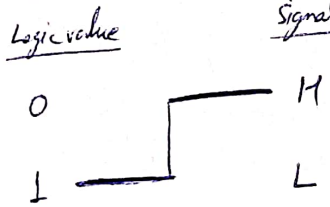
Date:	White Board Plan Digital Logic Design (UNIT-) Lecture no:	Teaching Methodology:
Today's Topics:	• The gates shown in above table, except for the inverter and buffer, can be extended to have more than 2 inputs. Condition for extension to multiple inputs A gate can be extended to have multiple inputs if binary operation it represents is commutative and associative.  3-input NAND gate  3-input NOR gate Q. Find the Boolean Function from the cascaded NAND gates:  $F = [(ABC)'.(DE)']'$ $= ABC + DE$ Note:- XOR is an odd function (ie. it is equal to 1 if the input variables have an odd number of 1's).	**Industrial Applications:** **Practical Applications:** **Class Activities:** **Summary:**
Glimpse of Next Lecture:		
Text Books: 1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc. 2. Donald D. Givone (2002). Digital Principles and Design, TataMcGraw Hill		
Reference Books: 1. C. V. S. Rao (2009). Switching and Logic Design,		

Positive and Negative Logic

- Hardware digital gates are defined in signal values such as H and L.
- Choosing the high-level H to represent logic 1 defines a positive logic system.
- Choosing the low-level L to represent logic 1 defines a negative logic system.



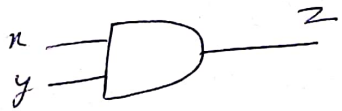
(a) Positive Logic



(b) Negative Logic

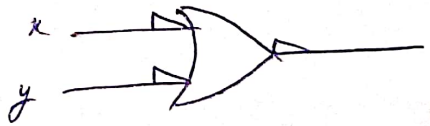
x	y	z
L	L	L
L	H	L
H	L	L
H	H	H

- ✓ H can be 3 volts
- ✓ L can be 0 volt



Positive Logic AND Gate

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



Negative Logic OR Gate

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

Date: Today

Date:

White Board Plan
Digital Logic Design (UNIT-)
Lecture no:

Today's Topics:

Glimpse of Next Lecture:

Text Books:

1. Digital Design, 4th edition, by M. Monis Mano Pearson Education Inc.

2. Donald D. Givone (2002), Digital Principles and Design, TataMcGraw Hill

Reference Books:

1. C. V. S. Rao (2009), Switching and Logic Design,

Teaching Methodology:

Industrial Applications:

Practical Applications:

Class Activities:

Summary:

Error-Correction Code – Hamming Code

Q.1 Construct Hamming code for the BCD word 0110 using even parity.

Sol Here, $n = 4$

We know that

$$2^k \geq n + k + 1$$

$$2^k > 5 + k \Rightarrow \boxed{k = 3}$$

These three parity bits P_3, P_2 and P_1 are placed at positions 4, 2, 1.

∴ Hamming code becomes:

Bit position	7	6	5	4	3	2	1
	0	1	1	P_3	0	P_2	P_1

Method 1: By three parity checks, the values of P_1, P_2, P_3 are fixed

For even parity at positions 1, 3, 5, 7, $P_1 = 1$

For even parity at positions 2, 3, 6, 7, $P_2 = 1$

For even parity at positions 4, 5, 6, 7, $P_3 = 0$

∴ Hamming code for BCD word 0110 is 0110011

Method 2 : For the determination of three parity bits (P_1, P_2 & P_3), bit position numbers of the data where 1's are placed are noted and their binary equivalents are taken.

i.e

6	110
5	101
	011 (Ex-OR operation)

∴ Resultant is the three parity bits

011

P_3 P_2 P_1

Now the seven-bit Hamming Code is–

Bit position	7	6	5	4	3	2	1
	D_4	D_3	D_2	P_3	D_1	P_2	P_1
	0	1	1	0	0	1	1

Error position	Position Number C_1	C_2	C_3
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
	1	1	

Q.2 A Hamming Code constructed in even parity is received as 0110111. Find the error position & give the correct word.

Sol The received Hamming Code is

7	6	5	4	3	2	1
0	1	1	0	1	1	1

Method 1 : The error position is expressed as $C_1 C_2 C_3$. Now parity checks are conducted.

1st check at positions 4, 5, 6, 7 (0110) gives $C_1 = 0$

2nd " " 2, 3, 6, 7 (1110) gives $C_2 = 1$

3rd " " 1, 3, 5, 7 (1110) gives $C_3 = 1$

∴, The error position

$C_1 C_2 C_3$ = 011 (3rd place)

So, the error has occurred at 3rd position bit. This bit is complemented for correction.

The correct word is 0110011

Method 2: Take binary equivalents of 1's placed at bit positions and take their Ex-OR

→ 011 (3rd place)