DIGITAL ELECTRONICS CIRCUIT(BCA 103)

DEPARTMENT OF COMPUTER SCIENCE

PROGRAMME: BCA

CENTRAL UNIVERSITY OF ODISHA

KORAPUT

BINARY LOGIC

- Binary logic deals with variables that take on two discrete values and with operations that assume logical meaning.
- The two values the variables assume may be *true* and *false*, *yes* and *no* or 1 and 0.
- Binary logic consists of binary variables and a set of logical operations.
- The variables are designated by letters of the alphabet, such as *A*, *B*, *C*, *x*, *y*, *z*, etc., with each variable having two and only two distinct possible values: 1 and 0.
- There are three basic logical operations: AND, OR, and NOT. Each operation produces a binary result, denoted by *z*.

- Binary logic resembles binary arithmetic, and the operations AND and OR have similarities to multiplication and addition, respectively.
- The symbols used for AND and OR are the same as those used for multiplication and addition.
- However, **binary logic should not be confused with binary arithmetic.**
- An arithmetic variable designates a number that may consist of many digits. But a logic variable is always either 1 or 0.
- Ex: **Binary Arithmetic :** 1 + 1 = 10 **Binary Logic :** 1 + 1 = 1

- A truth table is a table of all possible combinations of the variables, showing the relation between the values that the variables may take and the result of the operation.

Truth Tables of Logical Operations

AND			OR			NOT	
x	y	$x \cdot y$	x	y	$x + y$	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

Boolean Algebra

- Boolean algebra may be defined with a set of elements, a set of operators, and a number of unproved axioms or postulates.
- A *set* of elements is any collection of objects, usually having a common property.
- If S is a set, and x and y are certain objects, then the notation $x \varepsilon S$ means that x is a member of the set S and y x S means that y is not an element of S.
- The postulates of a mathematical system form the basic assumptions from which it is possible to deduce the rules, theorems, and properties of the system.

Common Postulates

- ***Closure:*** A set S is closed with respect to a binary operator if, for every pair of elements of S, the binary operator specifies a rule for obtaining a unique element of S.

Ex: the set of natural numbers $N = \{1, 2, 3, 4, c\}$ is closed with respect to the binary operator + by the rules of arithmetic addition, since, for any $a, b \, \varepsilon \, N$, there is a unique $c \, \varepsilon \, N$ such that $a + b = c$.

- ***Associative law:*** A binary operator * on a set S is said to be associative whenever $(x * y) * z = x * (y * z)$ for all $x, y, z, \varepsilon \, S$.
- ***Commutative law:*** A binary operator * on a set S is said to be commutative whenever $x * y = y * x$ for all $x, y \, \varepsilon \, S$.
- ***Identity element:*** A set S is said to have an identity element with respect to a binary operation * on S if there exists an element $e \, \varepsilon \, S$ with the property that $e * x = x * e = x$ for every $x \, \varepsilon \, S$.

- ***Inverse:*** A set *S* having the identity element *e* with respect to a binary operator * is said to have an inverse whenever, for every $x \varepsilon S$, there exists an element $y \varepsilon S$
  such that $x * y = e$.
- ***Distributive law:*** If * and . are two binary operators on a set *S*, * is said to be distributive over . whenever $x * (y . z) = (x * y) . (x * z)$

Two-Valued Boolean Algebra

- A two-valued Boolean algebra is defined on a set of two elements, $B = \{0, 1\}$, with rules for the two binary operators + and .

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

- Distributive law: $x \,.\, (y + z) = (x \,.\, y) + (x \,.\, z)$

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

$y + z$	$x \cdot (y + z)$
0	0
1	0
1	0
1	0
0	0
1	1
1	1
1	1

$x \cdot y$	$x \cdot z$	$(x \cdot y) + (x \cdot z)$
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	1	1
1	0	1
1	1	1

Theorems of Boolean Algebra

- *Duality principle: It* states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged.
- If the *dual* of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

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, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

- The postulates are basic axioms of the algebraic structure and need no proof.
- The theorems must be proven from the postulates.
- **Theorem: x + x = x.**

Statement	Justification
L.H.S: $x + x = (x + x) .1$	$(x . 1 = 1)$
$= (x + x)(x + x')$	$(x + x' = 1)$
$= x + xx'$	$(x + yz = (x + y) . (x + z))$
$= x + 0$	$(x . x' = 0)$
$= x$	$(x + 0 = x)$

- **Theorem: $x . x = x$.**

Statement	Justification
L.H.S: $x . x = (x . x) + 0$	$(x + 0 = x)$
$= (x . x) + (x . x')$	$(x . x' = 0)$
$= x . (x + x')$	$(x . (y + z)) = (x . y) + (x . z))$
$= x . 1$	$(x + x' = 1)$
$= x$	$(x . 1 = x)$

- **Theorem: x + 1 = 1.**

Statement	Justification
L.H.S: $x + 1 = (x + 1) \cdot 1$	$(x \cdot 1 = x)$
$= (x + 1) \cdot (x + x')$	$(x + x' = 1)$
$= x + (1 \cdot x')$	$(x + yz = (x + y) \cdot (x + z))$
$= x + x'$	$(1 \cdot x = x)$
$= 1$	$(x + x' = 1)$

- **Theorem: x + xy = x.**

Statement	Justification
L.H.S: $x + xy = (x \cdot 1) + (x \cdot y)$	$(x \cdot 1 = x)$
$= x \cdot (1 + y)$	$((x \cdot y) + (x \cdot z) = x \cdot (y + z))$
$= x \cdot (y + 1)$	$(x + y = y + x)$
$= x \cdot 1$	$(x + 1 = 1)$
$= x$	$(x \cdot 1 = x)$

-

- The theorems of Boolean algebra can be proven by means of truth tables.
- Theorem: x + xy =x DeMorgan's Theorem: (x+ y)' = x'y'

x	y	xy	x+ xy
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

x	y	x + y	(x+ y)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

x'	y'	x'y'
1	1	1
1	0	0
0	1	0
0	0	0

- The operator precedence for evaluating Boolean expressions is :

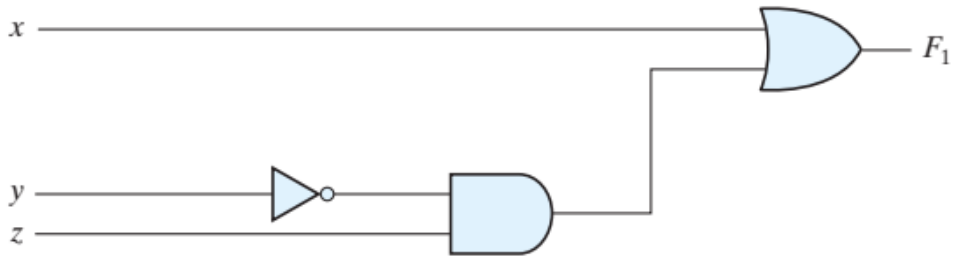
  (1) parentheses, (2) NOT, (3) AND, and (4) OR.
- In other words, expressions inside parentheses must be evaluated before all other operations.

Boolean Function

- Boolean algebra is an algebra that deals with binary 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.
- *Ex:* $\boldsymbol{F_1 = x + y'z}$
- The function $\boldsymbol{F_1}$ is equal to 1 if x is equal to 1 or if both y and z are equal to 1.
- $\boldsymbol{F_1}$ is equal to 0 otherwise. The complement operation dictates that when $y' = 1$, $y = 0$.
- Therefore, $\boldsymbol{F_1} = 1$ if $x = 1$ or if $y = 0$ and $z = 1$.
- A Boolean function expresses the logical relationship between binary variables and is evaluated by determining the binary value of the expression for all possible values of the variables.

- A Boolean function can be represented in a truth table. The number of rows in the truth table is 2^n, where n is the number of variables in the function.
- The binary combinations for the truth table are obtained from the binary numbers by counting from 0 through 2^n - 1.
-

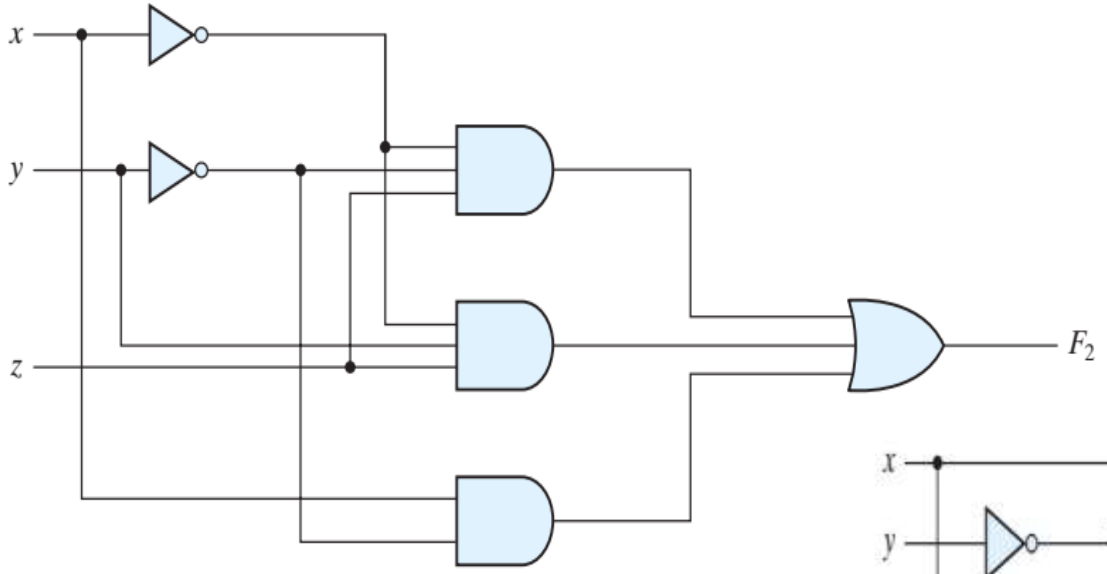
X	Y	Z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0



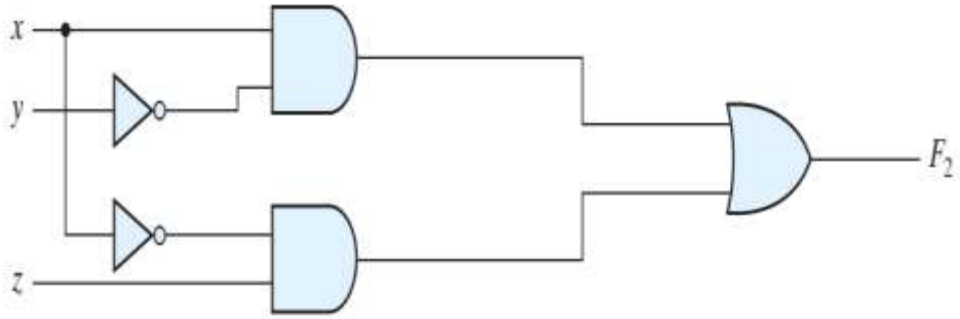
$F_1 = X + Y'Z$

- 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-circuit diagram is also called a schematic.
- In logic-circuit diagrams, the variables of the function are taken as the inputs of the circuit and the binary variable F_1 is taken as the output of the circuit.
- The schematic expresses the relationship between the output of the circuit and its inputs.
- There is only one way that a Boolean function can be represented in a truth table.
- However, when the function is in algebraic form, it can be expressed in a variety of ways, all of which have equivalent logic.

- **Ex:** $F_2 = x'y'z + x'yz + xy'$
- The possible simplification of the function by applying some of the identities of Boolean algebra:
  $F_2 = x'y'z + x'yz + xy' = x'z(y' + y) + xy' = x'z + xy'$
- By means of a truth table, it is possible to verify that the two expressions are equivalent.



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



(b) $F_2 = xy' + x'z$

Algebraic Manipulation

- When a Boolean expression is implemented with logic gates, each term requires a gate and each variable within the term designates an input to the gate.
- We define a *literal* to be a single variable within a term, in complemented or un-complemented form.

- $F_2 = x'y'z + x'yz + xy'$ (Function has 3 terms and 8 literals)
- $F_2 = x'z + xy'$ (Function has 2 terms and 4 literals)

- By reducing the number of terms, the number of literals, or both in a Boolean expression, it is often possible to obtain a simpler circuit.
- The manipulation of Boolean algebra consists mostly of reducing an expression for the purpose of obtaining a simpler circuit.

- Simplify the following Boolean functions to a minimum number of literals.

1. $x(x' + y) = xx' + xy = 0 + xy = xy.$

2. $x + x'y = (x + x')(x + y) = 1(x + y) = x + y.$

3. $(x + y)(x + y') = xx + xy + xy' + yy' = x(1 + y + y') + 0 = x.$

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

Functions 4 and 5 are together known as the *consensus theorem*.

Complement of a Function

- The complement of a function F is 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 may be derived algebraically through DeMorgan's theorems.
- $(A + B + C)' = (A + x)'$ let $B + C = x$
  $= A'x'$ by DeMorgan theorem (A+B)'=A'B'
  $= A'(B + C)'$ substitute $B + C = x$
  $= A'(B'C')$ by DeMorgan theorem (A+B)'=A'B'
  $= A'B'C'$ by associative

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.

$$(A + B + C + D + \ldots\ldots + F)' = A'B'C'D'\ldots\ldots F'$$
$$(ABCD \ldots\ldots F)' = A' + B' + C' + D' + \ldots\ldots + F'$$

- Find the complement of the functions:

  $F_1 = x'yz' + x'y'z$ **and** $F_2 = x(y'z' + yz)$
- **Ans:** $F'_1 = (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x + y' + z)(x + y + z')$

- $F'_2 = [x(y'z' + yz)]' = x' + (y'z' + yz)' = x' + (y'z')' (yz)'$
  $= x' + (y + z)(y' + z')$
  $= x' + yz' + y'z$

- A simpler procedure for deriving the complement of a function is to take the dual of the function and complement each literal.
- The dual of a function is obtained from the interchange of AND and OR operators and 1's and 0's.

- A simpler procedure for deriving the complement of a function is to take the dual of the function and complement each literal.
- The dual of a function is obtained from the interchange of AND and OR operators and 1's and 0's.
- Find the complement of the functions:

  $F_1 = x'yz' + x'y'z$ **and** $F_2 = x(y'z' + yz)$
- **Ans:**
- The dual of F_1 is $(x' + y + z')(x' + y' + z)$.
  Complement each literal: $(x + y' + z)(x + y + z') = F'_1$

- $F_2 = x(y'z' + yz)$
  The dual of F_2 is $x + (y' + z')(y + z)$
  Complement each literal: $x' + (y + z)(y' + z') = F'_2$

Minterms and Maxterms

- A binary variable may appear either in its normal form (x) or in its complement form (x').
- Two binary variables x and y combined with an AND operation.
- Since each variable may appear in either form, there are four possible combinations:

  x'y', x'y, xy',* and *xy.

- Each of these four AND terms is called a ***minterm*, or a *standard product*.**
- In a similar manner, n variables can be combined to form 2^n minterms.
- In a similar fashion, n variables forming an OR term, with each variable being primed or unprimed, provide 2^n possible combinations, called ***maxterms*, or *standard sums***.

  x+y, x'+y, x+y', and x'+y'

- The binary numbers from 0 to 2^n - 1 are listed under the *n* variables.
- Each minterm is obtained from an AND term of the *n* variables, with each variable being primed if the corresponding bit of the binary number is a 0 and unprimed if a 1.
- A symbol for each minterm is also shown in the table and is of the form m_j, where the subscript *j* denotes the decimal equivalent of the binary number of the minterm designated.

- **Note:** (1) each maxterm is obtained from an OR term of the *n* variables, with each variable being unprimed if the corresponding bit is a 0 and primed if a 1, and

(2) each maxterm is the complement of its corresponding minterm and vice versa.

			Minterms			Maxterms	
x	y	z	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:** A Boolean function can be expressed algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those terms.
- Ex: $f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$

  $f_2 = xy'z' + x'yz' + x'y'z + x'y'z' = m_3 + m_5 + m_6 + m_7$

x	y	z		f_1	f_2
0	0	0	m_0	0	0
0	0	1	m_1	1	0
0	1	0	m_2	0	0
0	1	1	m_3	0	1
1	0	0	m_4	1	0
1	0	1	m_5	0	1
1	1	0	m_6	0	1
1	1	1	m_7	1	1

- From the truth table by forming a minterm for each combination that produces a 0 in the function and then ORing those terms.
- The complement of f_1 is read as : $f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$
- If we take the complement of f_1', we obtain the function f_1:

  $f_1 = (x + y + z)(x + y' + z)(x + y' + z')(x' + y' + z')\ (x' + y' + z)$

  $= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$
- Similarly, it is possible to read the expression for f_2 from the table:

  $f_2 = (x + y + z)(x + y + z')(x + y' + z)(x' + y + z) = M_0M_1M_2M_4$

x	y	z		f_1	f_2
0	0	0	M_0	0	0
0	0	1	M_1	1	0
0	1	0	M_2	0	0
0	1	1	M_3	0	1
1	0	0	M_4	1	0
1	0	1	M_5	0	1
1	1	0	M_6	0	1
1	1	1	M_7	1	1

- These examples demonstrate first property of Boolean algebra: Any Boolean function can be expressed as a sum of minterms (with "sum" meaning the ORing of terms).
- Second property of Boolean algebra: Any Boolean function can be expressed as a product of maxterms (with "product" meaning the ANDing of terms).
- The procedure for obtaining the product of maxterms directly from the truth table is as follows: Form a maxterm for each combination of the variables that produces a 0 in the function, and then form the AND of all those maxterms.
- **Boolean functions expressed as a sum of minterms or product of maxterms are said to be in *canonical form*.**

Sum of Minterms

- For *n* binary variables, one can obtain 2^n distinct minterms and that any Boolean function can be expressed as a sum of minterms.
- The minterms whose sum defines the Boolean function are those which give the 1's of the function in a truth table.
- Since the function can be either 1 or 0 for each minterm, and since there are 2^n minterms, one can calculate all the functions that can be formed with *n* variables to be 2^{2^n}.
- It is sometimes convenient to express a Boolean function in its sum-of-minterms form.
- If the function is not in this form, it can be made so by first expanding the expression into a sum of AND terms.
- Each term is then inspected to see if it contains all the variables.
- If it misses one or more variables, it is ANDed with an expression such as ***x + x'***, where *x* is one of the missing variables.

- **Express the Boolean function $F = A + B'C$ as a sum of minterms.**
- The function has three variables: A, B, and C.
- First Term A: is missing two variables; therefore,
  $A = A(B + B') = AB + AB'$
- This function is still missing one variable i.e. C, so
  $A = AB(C + C') + AB'(C + C') = ABC + ABC' + AB'C + AB'C'$
- Second term BC : is missing one variable; hence,
  $B'C = B'C(A + A') = AB'C + A'B'C$
  Combining all terms, we have :
  $F = A + B'C = ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C$
- $F = ABC + ABC' + AB'C + AB'C' + A'B'C$
  But $AB'C$ appears twice, and according to theorem 1: ($x + x = x$), it is possible to remove one of those occurrences
- Rearranging the minterms in ascending order, we finally obtain
  $F = A'B'C + AB'C' + AB'C + ABC' + ABC = m_1 + m_4 + m_5 + m_6 + m_7$

- When a Boolean function is in its sum-of-minterms form, it is sometimes convenient to express the function in the following brief notation:

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

- The summation symbol g stands for the ORing of terms; the numbers following it are the indices of the minterms of the function.
- The letters in parentheses following *F* form a list of the variables in the order taken when the minterm is converted to an AND term.

Alternative Procedure For Deriving the minterms

- **Note:** An alternative procedure for deriving the minterms of a Boolean function is to obtain the truth table of the function directly from the algebraic expression and then read the minterms from the truth table.
- Ex: **Express the Boolean function *F = A + B'C* as a sum of minterms.**
- Inserting 1's under *F* for those combinations for which *A* = 1 and *BC* = 01. From the truth table, we can then read the five minterms of the function to be 1, 4, 5, 6, and 7
- $F(A, B, C) = \sum(1, 4, 5, 6, 7)$

A	B	C	B'	B'C		F
0	0	0	1	0	m_0	0
0	0	1	1	1	m_1	1
0	1	0	0	0	m_2	0
0	1	1	0	0	m_3	0
1	0	0	1	0	m_4	1
1	0	1	1	1	m_5	1
1	1	0	0	0	m_6	1
1	1	1	0	0	m_7	1

- **Each of the 2^{2n} functions of *n* binary variables can be also expressed as a product of maxterms.**
- To express a Boolean function as a product of maxterms, it must first be brought into a form of OR terms.
- This may be done by using the distributive law,
  $x + yz = (x + y)(x + z)$.
- Then any missing variable *x* in each OR term is ORed with ***xx'***.

- **Express the Boolean function $F = xy + x'z$ as a product of maxterms.**
- First, convert the function into OR terms by using the distributive law:

  $F = xy + x'z = (xy + x')(xy + z)$

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

  $= (x' + y)(x + z)(y + z)$
- The function has three variables: x, y, and z. Each OR term is missing one variable; therefore,

  $x' + y = x' + y + zz' = \mathbf{(x' + y + z)}(x' + y + z')$

  $x + z = x + z + yy' = \mathbf{(x + y + z)}(x + y' + z)$

  $y + z = y + z + xx' = \mathbf{(x + y + z)(x' + y + z)}$

  Combining all the terms and removing those which appear more than once, we finally obtain:

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

  $= M_0M_2M_4M_5$

- $F(x, y, z) = \prod(0, 2, 4, 5)$
- The product symbol, $\prod$, denotes the ANDing of maxterms; the numbers are the indices of the maxterms of the function.

Conversion between Canonical Forms

- To convert from one canonical form to another, interchange the symbols $\sum$ and $\prod$ and list those numbers missing from the original form.
- In order to find the missing terms, one must realize that the total number of minterms or maxterms is 2^n, where n is the number of binary variables in 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.
- This is because the original function is expressed by those minterms which make the function equal to 1, whereas its complement is a 1 for those minterms for which the function is a 0.
- **Ex: $F(A, B, C) = \sum(1, 4, 5, 6, 7)$**
- This function has a complement that can be expressed as
  $F'(A, B, C) = \sum(0, 2, 3) = m_0 + m_2 + m_3$
- Now, if we take the complement of F' by DeMorgan's theorem, we obtain F in a different form:
  $F = (m_0 + m_2 + m_3)' = m_0'.m_2'.m_3' = M_0M_2M_3 = \prod(0, 2, 3)$
- $m_j' = M_j$
  That is, the **maxterm with subscript j is a complement of the minterm with the same subscript j and vice versa.**

- A Boolean function can be converted from an algebraic expression to a product of maxterms by means of a truth table and the canonical conversion procedure.
- Ex: ***F = xy + x'z***
- The function expressed as a **sum of minterms is $F(x, y, z) = \sum (1, 3, 6, 7)$**
- The function expressed as a product of maxterms is $F(x, y, z) = \prod(0, 2, 4, 5)$

X	y	z	x'	xy	x'z	F= *xy + x'z*	
0	0	0	1	0	0	0	Maxterm
0	0	1	1	0	1	1	Minterm
0	1	0	1	0	0	0	Maxterm
0	1	1	1	0	1	1	Minterm
1	0	0	0	0	0	0	Maxterm
1	0	1	0	0	0	0	Maxterm
1	1	0	0	1	0	1	Minterm
1	1	1	0	1	0	1	Minterm

Standard Forms

- Another way to express Boolean functions is in *standard* form.
- In this configuration, the terms that form the function may contain one, two, or any number of literals.
- There are two types of standard forms:

  sum of products and **products of sums.**
- The *sum of products* is a Boolean expression containing AND terms, called *product terms*, with one or more literals each.
- The *sum* denotes the ORing of these terms.
- **Ex:** $F_1 = y' + xy + x'yz'$
- A *product of sums* is a Boolean expression containing OR terms, called *sum terms*.
- Each term may have any number of literals.
- The *product* denotes the ANDing of these terms.
- **Example :** $F_2 = x(y' + z)(x' + y + z')$

Non-Standard Forms

- A Boolean function may be expressed in a nonstandard form.
- Example: the function $F_3 = AB + C(D + E)$
  is neither in sum-of-products nor in product-of-sums form.

Other Logic Operations

- There are 2^{2n} functions for n binary variables. Thus, for two variables, $n = 2$, and the number of possible Boolean functions is 16.
- The functions are determined from the 16 binary combinations that can be assigned to F.
- The 16 functions can be expressed algebraically by means of Boolean functions.
- The Boolean expressions listed are simplified to their minimum number of literals.

Truth Tables for the 16 Functions of Two Binary Variables

x	y	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

- Although each function can be expressed in terms of the Boolean operators AND, OR, and NOT, there is no reason one cannot assign special operator symbols for expressing the other functions.
- The 16 functions listed can be subdivided into three 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, exclusive-OR, equivalence, inhibition, and implication.
- Two (inhibition and implication) are used by logicians, but are seldom used in computer logic.
-

- The NOR function is the complement of the OR function, and its name is an abbreviation of *not-OR*.
- Similarly, NAND is the complement of AND and is an abbreviation of *not-AND*.
- The exclusive-OR, abbreviated XOR, is similar to OR, but excludes the combination of *both x* and *y* being equal to 1; it holds only when *x* and *y* differ in value. (It is sometimes referred to as the binary difference operator.)
- Equivalence is a function that is 1 when the two binary variables are equal (i.e., when both are 0 or both are 1). The exclusive-OR and equivalence functions are the complements of each other.
- The equivalence function is called exclusive-NOR, abbreviated XNOR.

Boolean Expressions for the 16 Functions of Two Variables

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	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

- **Note:** A Boolean function can be expressed algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those terms.
- Ex: $f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$

  $f_2 = xy'z' + x'yz' + x'y'z + x'y'z' = m_3 + m_5 + m_6 + m_7$

x	y	z	x'	y'	z'	x'y'z	xy'z'	xyz	x'yz'	x'y'z'			f_1	f_2
0	0	0	1	1	1	0	0	0	0	1		m_0	0	0
0	0	1	1	1	0	1	0	0	0	0		m_1	1	0
0	1	0	1	0	1	0	0	0	1	0		m_2	0	0
0	1	1	1	0	0	0	0	0	0	0		m_3	0	1
1	0	0	0	1	1	0	1	0	0	0		m_4	1	0
1	0	1	0	1	0	0	0	0	0	0		m_5	0	1
1	1	0	0	0	1	0	0	0	0	0		m_6	0	1
1	1	1	0	0	0	0	0	1	0	0		m_7	1	1