LOGIC DESIGN

Chapter 2

Boolean Algebra and Logic Gates

OUTLINE OF CHAPTER 2

- Basic Definition
- Axiomatic Definition of Boolean Algebra
- Basic Theorems and Properties of Boolean Algebra
- Boolean Functions
- Canonical and Standard Forms
- Other Logic Operations
- Digital Logic Gates
- Integrated Circuits

2.1 BASIC DEFINITIONS

BASIC DEFINITIONS

- What is an algebra?
  - Mathematical system consisting of
    - Set of elements
    - Set of operators
    - Axioms or postulates
- Why is it important?
  - Defines rules of “calculations”

BASIC DEFINITIONS

- Example: arithmetic on natural numbers
  - Set of elements: $N = \{1,2,3,4,\ldots\}$
  - Operator: +, –, *
  - Axioms: associativity, distributivity, closure, identity elements, etc.
- Note: operators with two inputs are called ***binary***
  - Does not mean they are restricted to binary numbers!
  - Operator(s) with one input are called ***unary***

BASIC DEFINITION

- A **set** is collection of having the same property.
  - ***S*** : set, ***x*** and ***y*** : element or event
  - For example: $S = \{1, 2, 3, 4\}$
    - If $x = 2$, then $x \in S$.
    - If $y = 5$, then $y \notin S$.

BASIC DEFINITIONS

- A ***binary operator*** defines on a set ***S*** of elements is a rule that assigns, to each pair of elements from ***S***, a unique element from ***S***.
  - For example: given a set ***S***, consider $x*y = z$.
    - $*$ is a binary operator.
  - If (x, y) through $*$ get z and $x, y, z \in S$, then
    - $*$ is a binary operator of ***S***.
  - if $*$ is not a binary operator of ***S*** and $x, y \in S$, then
    - $z \notin S$.

BASIC DEFINITION

The most common postulates used to formulate various algebraic structures are as follows:

1. **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***.
   - For example, natural numbers **N={1,2,3,…}** is **closed** with respect to the binary operator **+** by the rule of arithmetic addition, since, for any **x**, **y**$\in$**N**, there is a unique **z**$\in$**N** such that
     - $x + y = z$
   - But operator **–** is not closed for ***N***, because **2-3 = -1** and **2**, **3**$\in$***N***, but
     - **(-1)**$\notin$***N***.

BASIC DEFINITIONS

2. **Associative law.** A binary operator $*$ on a set S is said to be associative whenever
   - $(x * y) * z = x * (y * z)$ for all $x, y, z \in S$
     - $(x+y)+z = x+(y+z)$
3. **Commutative law.** A binary operator $*$ on a set S is said to be commutative whenever
   - $x * y = y * x$ for all $x, y \in S$
     - $x+y = y+x$

BASIC DEFINITIONS

4. **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 $e \in S$ with the property that
   - $e * x = x * e = x$ for every $x \in S$
     - $0 + x = x + 0 = x$ for every $x \in I$. $I = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$.
     - $1 * x = x * 1 = x$ for every $x \in I$. $I = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$.

BASIC DEFINITION

5. **Inverse.** A set **S** is having the identity element **e** with respect to the binary operator to have an inverse whenever, for every $x \in S$, there exists an element $y \in S$ such that

   – $x * y = e$

     - In the set of integers, ***I***, the operator **+** over ***I*** with $e = 0$, the inverse of an element x is $(-x)$ since $x+(-x) = 0$.

BASIC DEFINITION

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

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

BASIC DEFINITION

- The field of real numbers is the basis for arithmetic and ordinary algebra. The operators and postulates have the following meanings:
  - The binary operator **+** defines addition.
  - The additive identity is **0**.
  - The additive inverse defines subtraction.
  - The binary operator **·** defines multiplication.
  - The multiplicative identity is **1**.
  - The multiplicative inverse of **x = 1/x** defines division, **i.e., x · 1/x = 1**.
  - The only distributive law applicable is that of **·** over **+**:
    - **x · (y+z) = (x · y) + (x · z)**

2.2 AXIOMATIC DEFINITION OF BOOLEAN ALGEBRA

AXIOMATIC DEFINITION OF BOOLEAN ALGEBRA

- We need to define algebra for binary values
  - Developed by George Boole in 1854
- Huntington postulates for Boolean algebra (1904):
- $B = \{0, 1\}$ and two binary operations, **+** and **·**
  1. Closure with respect to operator **+** and operator **·**
  2. Identity element **0** for operator **+** and **1** for operator **·**
  3. Commutativity with respect to **+** and **·**

$$x+y = y+x, \quad x \cdot y = y \cdot x$$

AXIOMATIC DEFINITION OF BOOLEAN ALGEBRA

4. Distributivity of $\cdot$ over **+**, and **+** over $\cdot$
   - $x\cdot(y+z) = (x\cdot y)+(x\cdot z)$ and $x+(y\cdot z) = (x+y)\cdot(x+z)$
5. Complement for every element **x** is **x'** with $x+x'=1$, $x\cdot x'=0$
6. There are at least two elements $x,y \in B$ such that $x \neq y$

- Terminology:
  - **Literal:** A variable or its complement
  - **Product term:** literals connected by $\cdot$
  - **Sum term:** literals connected by **+**

AXIOMATIC DEFINITION OF 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 **•**
  - The rules of operations: AND、OR and NOT

AND

X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

X	Y	X + Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT

X	$\bar{X}$
1	0
0	1

AXIOMATIC DEFINITION OF BOOLEAN ALGEBRA

- Huntington postulates for the set $B = \{0,1\}$ and the two binary operators defined before.

1. Closure law
   - **$1,0 \in B$**
2. Identity laws
   - **0 for + and 1 for ·**
3. Commutative laws
   - **$x+y = y+x$, $x \cdot y = y \cdot x$**
4. Distributive laws
   - **$x \cdot (y+z) = (x \cdot y) + (x \cdot z)$**

AXIOMATIC DEFINITION OF BOOLEAN ALGEBRA

Distributive laws

x	y	z	y + z	x · (y + z)	x · y	x · z	(x · y) + (x · z)
0	0	0	0	**0**	0	0	**0**
0	0	1	1	**0**	0	0	**0**
0	1	0	1	**0**	0	0	**0**
0	1	1	1	**0**	0	0	**0**
1	0	0	0	**0**	0	0	**0**
1	0	1	1	**1**	0	1	**1**
1	1	0	1	**1**	1	0	**1**
1	1	1	1	**1**	1	1	**1**

AXIOMATIC DEFINITION OF BOOLEAN ALGEBRA

5. Complement
   - **$x+x' = 1 \rightarrow 0+0' = 0+1 = 1; 1+1' = 1+0 = 1$**
6. Has two distinct elements
   - **1 and 0, with $0 \neq 1$**

- Note
  - A set of two elements
  - $+$: OR operation; $\cdot$: AND operation
  - A complement operator: NOT operation
  - Binary logic is a two-valued Boolean algebra

2.3 BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

- **Duality Principle** says that:
  - if an expression is valid in Boolean algebra, then
  - the dual of that expression is also valid.
- To form the dual of an expression:
  - replace all **+** operators with **·** operators,
  - all **·** operators with **+** operators,
  - all **1's** with **0's**, and all **0's** with **1's**.

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

- Form the dual of the expression

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

- Following the replacement rules…

  $x(y + z) = xy + xz$

- Take care not to alter the location of the parentheses if they are present.

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

Postulates and Theorems of Boolean Algebra		
Postulate 2, identity	(a) $x + 0 = x$	(b) $x \cdot 1 = x$
Postulate 5, complementarity	(a) $x + x' = 1$	(b) $x \cdot x' = 0$
Theorem 1, idempotent	(a) $x + x = x$	(b) $x \cdot x = x$
Theorem 2, involution	(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) $x(x+y) = x$

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

- Need more rules to modify algebraic expressions
  - Theorems that are derived from postulates
- What is a theorem?
  - A formula or statement that is derived from postulates (or other proven theorems)
- Basic theorems of Boolean algebra
  - Theorem 1 (a): $x + x = x$ (b): $x \cdot x = x$
  - Looks straightforward, but needs to be proven!

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

- Theorem 1 (a) show that $x+x = x$.
- $x+x = (x+x) \cdot 1$ by 2(b)

  $= (x+x)(x+x')$ by 5(a)

  $= x+xx'$ by 4(b)

  $= x+0$ by 5(b)

  $= x$ by 2(a)

- Post. 2: (a) $x+0=x$, (b) $x \cdot 1=x$
- Post. 3: (a) $x+y=y+x$, (b) $x \cdot y=y \cdot x$
- Post. 4: (a) $x(y+z) = xy+xz$,

  (b) $x+yz = (x+y)(x+z)$
- Post. 5: (a) $x+x'=1$, (b) $x \cdot x'=0$

We can now use Theorem 1(a) in future proofs

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

- Theorem 1 (b) show that $x \cdot x = x$.
- $x \cdot x$ $= xx+0$ by 2(a)
  $= xx+xx'$ by 5(b)
  $= x(x+x')$ by 4(a)
  $= x \cdot 1$ by 5(a)
  $= x$ by 2(b)

- Post. 2: (a) $x+0=x$, (b) $x \cdot 1=x$
- Post. 3: (a) $x+y=y+x$, (b) $x \cdot y=y \cdot x$
- Post. 4: (a) $x(y+z) = xy+xz$,
  (b) $x+yz = (x+y)(x+z)$
- Post. 5: (a) $x+x'=1$, (b) $x \cdot x'=0$
- Th. 1: (a) $x+x=x$,

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

- Theorem 2 (a) show that $x+1 = 1$.
- $x + 1 = 1 \cdot (x + 1)$ by 2(b)

  $=(x + x')(x + 1)$ by 5(a)

  $= x + x' 1$ by 4(b)

  $= x + x'$ by 2(b)

  $= 1$ by 5(a)

- Post. 2: (a) $x+0=x$, (b) $x\cdot 1=x$
- Post. 3: (a) $x+y=y+x$, (b) $x\cdot y=y\cdot x$
- Post. 4: (a) $x(y+z) = xy+xz$,

  (b) $x+yz = (x+y)(x+z)$
- Post. 5: (a) $x+x'=1$, (b) $x\cdot x'=0$
- Th. 1: (a) $x+x=x$, (b) $x.x=x$

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

- Theorem 2(b): $x \cdot 0 = 0$ by duality
- Theorem 3: $(x')' = x$
  - Postulate 5 defines the complement of x,
    - $x + x' = 1$
    - $x \cdot x' = 0$
  - The complement of x' is x is also (x')'

- Post. 2: (a) $x+0=x$, (b) $x \cdot 1=x$
- Post. 3: (a) $x+y=y+x$, (b) $x \cdot y=y \cdot x$
- Post. 4: (a) $x(y+z) = xy+xz$,
  (b) $x+yz = (x+y)(x+z)$
- Post. 5: (a) $x+x'=1$, (b) $x \cdot x'=0$
- Th. 1: (a) $x+x=x$, (b) $x.x=x$
- Th. 2: (a) $x+1=1$

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

- Absorption Property (Covering)
  Theorem 6(a): $x + xy = x$
- $x + xy = x \cdot 1 + xy$ by 2(b)
  $= x\,(1 + y)$ by 4(a)
  $= x\,(y + 1)$ by 3(a)
  $= x \cdot 1$ by Th. 2(a)
  $= x$ by 2(b)
- Theorem 6(b): $x\,(x + y) = x$ by duality
- By means of truth table (another way to proof)

- Post. 2: (a) $x+0=x$, (b) $x\cdot 1=x$
- Post. 3: (a) $x+y=y+x$, (b) $x\cdot y=y\cdot x$
- Post. 4: (a) $x(y+z) = xy+xz$, (b) $x+yz = (x+y)(x+z)$
- Post. 5: (a) $x+x'=1$, (b) $x\cdot x'=0$
- Th. 1: (a) $x+x=x$, (b) $x\cdot x=x$
- Th. 2: (a) $x+1=1$, (b) $x\cdot 0=0$

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

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

- DeMorgan's Theorem
- Theorem 5(a): $(x + y)' = x'y'$
- Theorem 5(b): $(xy)' = x' + y'$

By means of truth table

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

Consensus Theorem

1. $xy + x'z + yz = xy + x'z$
2. $(x + y) \bullet (x' + z) \bullet (y + z) = (x + y) \bullet (x' + z)$ -- (dual)

- **Proof:**

$$
\begin{aligned}
xy + x'z + yz &= xy + x'z + (x + x')\, yz \\
&= xy + x'z + xyz + x'yz \\
&= (xy + xyz) + (x'z + x'zy) \\
&= xy + x'z
\end{aligned}
$$

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

- The operator precedence for evaluating Boolean Expression is
  - Parentheses
  - NOT
  - AND
  - OR
- Examples
  - $x\,y' + z$
  - $(x\,y + z)'$

2.4 BOOLEAN FUNCTIONS

BOOLEAN FUNCTIONS

- A Boolean function consists of:
  - Binary variables
  - Binary operators OR and AND
  - Unary operator NOT
  - Parentheses
  - The constants 0 and 1
- Examples
  - $F_1 = x\,y\,z'$
  - $F_2 = x + y'z$
  - $F_3 = x'\,y'\,z + x'\,y\,z + x\,y'$
  - $F_4 = x\,y' + x'\,z$

BOOLEAN FUNCTIONS

- For a given value of the binary variables, the function can be equal to either 1 or 0.
- Consider as an example the following Boolean function:
  - $F_1 = x + y' z$
  - The function $F_1 = 1$, if $x = 1$ **OR** if $y' = 1$ and $z = 1$.
  - $F_1 = 0$ otherwise.
- The complement operation dictates that when $y' = 1$ then $y = 0$.
- Therefore, we can say that
  - $F1 = 1$ if $x = 1$ OR if $y = 0$ and $z = 1$.

BOOLEAN FUNCTIONS

- A Boolean function expresses the logical relationship between binary variables.
- It 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.

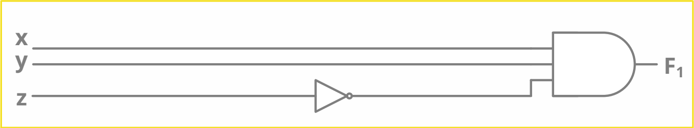
BOOLEAN FUNCTIONS

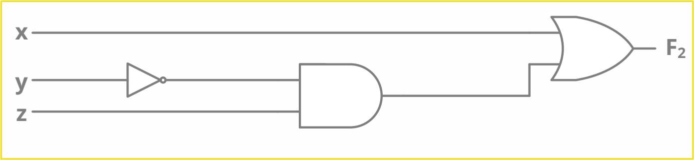
- The truth table of 2^n entries
- $F_1 = x\,y\,z'$, $F_2 = x + y'z$, $F_3 = x'\,y'\,z + x'\,y\,z + x\,y'$, $F_4 = x\,y' + x'\,z$
- Two Boolean expressions may specify the same function $F_3 = F_4$

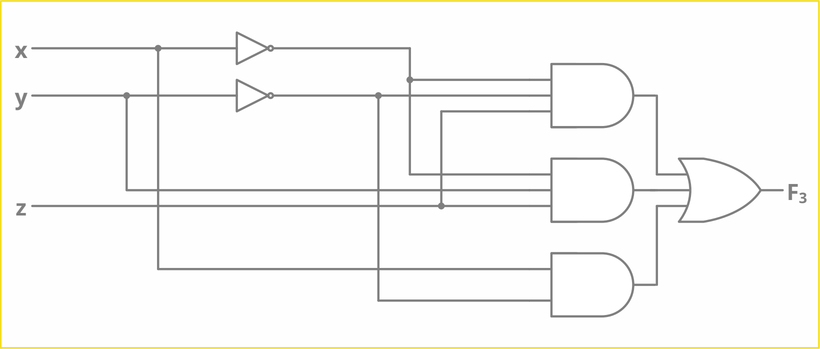
x	y	z	x'	y'	z'	y'z	x'y'z	x'yz	xy'	x'z	F1	F2	F3	F4
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	1	1	0	0	1	0	1	1	1
0	1	0	1	0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	0	1	0	0	1	1
1	0	0	0	1	1	0	0	0	1	0	0	1	1	1
1	0	1	0	1	0	1	0	0	1	0	0	1	1	1
1	1	0	0	0	1	0	0	0	0	0	1	1	0	0
1	1	1	0	0	0	0	0	0	0	0	0	1	0	0

BOOLEAN FUNCTION

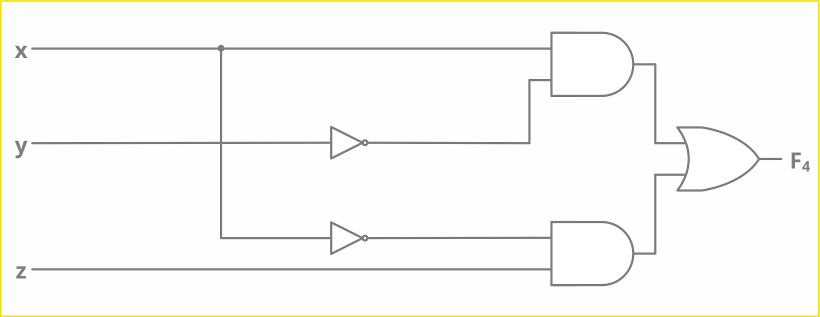
- Implementation with logic gates







- F_4 is more economical



BOOLEAN FUNCTION

Algebraic Manipulation

- When a Boolean expression is implemented with logic gates,
  - Each **term** requires a gate
  - Each **variable** within the term designates an input to the gate
  - **Literal** is a single variable within a term that may be complemented or not.

BOOLEAN FUNCTION

- Example:
  - $F_1 = x\,y\,z'$ — **1** term and **3** literals
  - $F_2 = x + y'z$ — **2** terms and **3** literals
  - $F_3 = x'\,y'\,z + x'\,y\,z + x\,y'$ — **3** terms and **8** literals
  - $F_4 = x\,y' + x'\,z$ — **2** terms and **4** literals
- Manipulation of Boolean algebra consists mostly of reducing an expression by reducing the number of **terms**, the number of **literals**, or **both** in a Boolean expression, it is often possible to obtain a **simpler, less area, cheaper circuit.**

BOOLEAN FUNCTION

Simplify the following functions to a minimum number of literals.

- $x(x'+y)$
  - I. $= (x\,x') + (x\,y)$ by post (4a)
  - II. $= 0 + x\,y$ by post (5b)
  - III. $= x\,y$ by post (2a)
- $x+x'y$
  - I. $= (x + x')\,(x + y)$ by post (4b)
  - II. $= 1\,(x + y)$ by post (5a)
  - III. $= x + y$ by post (2b)
- $(x + y)\,(x + y')$
  - I. $= x + y\,y'$ post (4b)
  - II. $= x + 0$ post (5b)
  - III. $= x$ post (2a)
- $(x + y)\,(x' + z)\,(y + z) = (x + y)\,(x' + z)$
  - consesus theorem with duality.

BOOLEAN FUNCTION

Simplify the following functions to a minimum number of literals.

- $x\,y + x'\,z + y\,z$

I.	$= x\,y + x'\,z + 1\,y\,z$	
II.	$= x\,y + x'\,z + (x + x')\,y\,z$	post (5a)
III.	$= x\,y + x'\,z + x\,y\,z + x'\,y\,z$	post (4a)
IV.	$= x\,y + x\,y\,z + x'\,z + x'\,y\,z$	post (3a)
V.	$= x\,y\,(1 + z) + x'z\,(1 + y)$	post (4a)
VI.	$= x\,y\,1 + x'\,z\,1$	Theo(2a)
VII.	$= x\,y + x'\,z$	post (2b)

BOOLEAN 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 De Morgan's theorem.
  - De Morgan's theorem can be extended to three or more variables.

BOOLEAN FUNCTION

- $(A + B + C)' = (A + x)'$ let $B + C = x$

  $= A' x'$ by theorem (5a) De Morgan

  $= A' (B + C)'$ substitute $B+C = x$

  $= A' (B' C')$ by theorem (5a) De Morgan

  $= A' B' C'$ by theorem (4b) (associative)

BOOLEAN FUNCTION

- Generalizations:
  - Function is obtained by interchanging AND and OR operators and complementing each literal.
    - $(A+B+C+D+ \ldots +F)' = A'B'C'D' \ldots F'$
    - $(ABCD \ldots F)' = A'+ B'+C'+D' \ldots +F'$

BOOLEAN FUNCTION

Find the complement of the function F_1 by applying De Morgan's theorem as many times as necessary

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

- $F_2' = x (y' z' + y z)$.
- $= [x(y' z' + y z)]'$
- $= x' + (y' z' + y z)'$
- $= x + (y' z')' (y z)'$
- $= x' (y+ z) (y' + z')$

BOOLEAN FUNCTION

- Simpler procedure:
  - Take the dual of the function and complement each literal
- Find the complement of the function F_1
  - $F_1 = (x' y z' + x' y' z)'$.
  - $= (x' y z')' (x' y' z)'$
  - The dual of $F_1 = (x' + y + z') (x' + y' z)$.
  - Complement each literal $= (x + y' + z) (x + y + z') = F_1'$

BOOLEAN FUNCTION

- Find the complement of the function F_2
  - $F_2 = x\,(y'\,z' + y\,z)$.
  - $= x\,(y'\,z') + (y\,z)$
  - The dual of $F_2 = x + (y' + z')\,(y + z)$
  - Complement each literal $= x' + (y + z)\,(y' + z') = F_2'$

2.5 CANONICAL AND STANDARD FORMS

CANONICAL AND STANDARD FORMS

- A binary variable may appear either in its normal for **(x)** or in its complement form **(x')**.
- Consider :
  - **x AND y**.
- Each variable may appear in either form, there are four possible combinations:
  - x' y'
  - x' y
  - x y'
  - x y
- Each of these four AND term is called a **minterm** or a **standard product**.
- **n** variables can be combined to form **2^n minterms**.

CANONICAL AND STANDARD FORMS

- In a similar fashion,
  - **n** variables forming and **OR** term,
  - with each variable being primed or unprimed,
  - provide **2^n** possible combinations,
  - called **maxterms** or **standard sums**.
- Each **maxterm** is the **complement** of its corresponding **minterm**, and vice versa.

CANONICAL AND STANDARD FORMS

			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′ y z′	m_2	x + y′ + z	M_2
0	1	1	x′ y z	m_3	x + y′ + z′	M_3
1	0	0	x y′ z′	m_4	x′ + y + z	M_4
1	0	1	x y′ z	m_5	x′ + y + z′	M_5
1	1	0	x y z′	m_6	x′ + y′ + z	M_6
1	1	1	x y z	m_7	x′ + y′ + z′	M_7

CANONICAL AND STANDARD FORMS

- A Boolean function can be expressed algebraically by
  - A truth table
  - Sum of **minterms**
  - Each of these **minterms** results in $f_1 = 1$.

CANONICAL AND STANDARD FORMS

x	y	z	f1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$x'y'z$

$+ xy'z'$

$+ xyz$

$f_1 = \quad = m_1 + m_4 + m_7$ (Minterms)

CANONICAL AND STANDARD FORMS

x	y	z	f2	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$x'yz$
1	0	0	0	
1	0	1	1	$+\ xy'z$
1	1	0	1	$+\ xyz'$
1	1	1	1	$+\ xyz$

$f_2 =$ $= m_3 + m_5 + m_6 + m_7$ (Minterms)

CANONICAL AND STANDARD FORMS

- These examples demonstrate an important property of Boolean algebra:
  - Any Boolean function can be expressed as a sum (**OR**ing) of **minterms**.
- Consider the complement of a Boolean function.
  - From the truth table
    - A **minterm** for each combination that produces a **0**
    - Then **OR**ing those terms.

CANONICAL AND STANDARD FORMS

- The complement of f_1:
  - $f_1' = m_0 + m_2 + m_3 + m_5 + m_6$
  - $= x'y'z' + x'yz' + x'yz + xy'z + xyz'$
  - If we take the complement of f_1', then we obtain f_1
  - $(f_1')' = (x'y'z' + x'yz' + x'yz + xy'z + xyz')'$
  - $f_1 = (x + y + z)(x + y' + z)(x + y' + z')(x' + y' + z)$
  - $= M_0 M_2 M_3 M_5 M_6$

x	y	z	f1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

CANONICAL AND STANDARD FORMS

- The complement of f_2:
  - $f_2' = m_0 + m_1 + m_2 + m_4$
  - $= x'y'z' + x'y'z + x'yz' + xy'z'$
  - If we take the complement of f_2', then we obtain f_2
  - $(f_2')' = (x'y'z' + x'y'z + x'yz' + xy'z')'$
  - $f_2 = (x + y + z)(x + y + z')(x + y' + z)(x' + y + z)$
  - $= M_0 M_1 M_2 M_4$

x	y	z	f2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

CANONICAL AND STANDARD FORMS

- These examples demonstrate a second property of Boolean algebra:
  - Any Boolean function can be expressed as a product of (**AND**ing) of maxterms.
- Consider the complement of a Boolean function.
  - From the truth table
    - A **maxterm** for each combination that produces a **0**
    - Then **AND**ing those terms.
- Boolean functions expressed as a sum of **minterms** or product of **maxterms** are said to be in **canonical form**.

CANONICAL AND STANDARD FORMS

- **Sum of minterms**:
  - There are 2^n minterms and
  - 2^{2n} combinations of function with **n** Boolean variables.
  - It is sometimes convenient to express the Boolean function in its sum of minterms form.
  - If 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.

CANONICAL AND STANDARD FORMS

- Example: express $F = A+B'C$ in a sum of minterms.
  - Step1: A is missing two variable B and C!
  - 1^{st} include B
  - $A = A(B + B')$
  - $\quad = AB + AB'$
  - 2^{nd} include C
  - $A = AB(C + C') + AB'(C+C')$
  - $\quad = ABC + ABC' + AB'C + AB'C'$

CANONICAL AND STANDARD FORMS

- Step2: B'C is missing one variable A!
- Include A
- B'C = B'C (A + A')

  = AB'C + A'B'C
- Step3: Combine all terms
- F = ABC + ABC' + $\underbrace{\boxed{\text{AB'C}} + \text{AB'C'} + \boxed{\text{AB'C}}}$ + A'B'C

  according to Theorem 1 (x + x = x),

  it is possible to remove one of them
- F = ABC + ABC' + AB'C' + AB'C + A'B'C = $m_1 + m_4 + m_5 + m_6 + m_7$

CANONICAL AND STANDARD FORMS

- F = ABC + ABC′ + AB′C′ + AB′C + A′B′C
- $= m_1 + m_4 + m_5 + m_6 + m_7$

- When in its sum of minterms the Boolean function can be expressed as:
  - F(A, B, C)= ∑(1, 4, 5, 6, 7)

A	B	C	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

CANONICAL AND STANDARD FORMS

- **Product of maxterms**:
  - 2^{2n} functions of **n** binary variables can be also expressed as product of maxterms.
  - To express the Boolean functions 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 x x'.

CANONICAL AND STANDARD FORMS

- Example: express $F = x\,y + x'\,z$ in a product of maxterm.
- Step1: using distributive law
  - $= (xy + x')\,(xy + z)$
- Step2: using distributive law
  - $= (x + x')\,(y + x')\,(x + z)\,(y + z)$
  - $= 1\,(x' + y)\,(x + z)\,(y + z)$
  - $= (x' + y)\,(x + z)\,(y + z)$

CANONICAL AND STANDARD FORMS

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

- Function has three variables x, y and z. Each OR term is missing one variable.
  - $(x' + y) = x' + y + zz' = \boxed{(x' + y + z)}(x' + y + z')$
  - $(x + z) = x + z + yy' = \boxed{(x + y + z)}(x + y' + z)$
  - $(y + z) = y + z + xx' = \boxed{(x + y + z)}\boxed{(x' + y + z)}$

- Combine all the terms and remove the ones that appear twice
  - $F = (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z')$

CANONICAL AND STANDARD FORMS

- F = x'yz' + x'yz + xy'z + xyz
- $= M_0 \, M_2 \, M_4 \, M_5$

• Convenient way to express this function is:

- F(x, y, z)= $\prod(0, 2, 4, 5)$

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

CANONICAL AND STANDARD FORMS

- The complement of a function expressed as:
  - The sum of minterms = the sum of minterms missing from the original function
  - Because the original function is expressed by those minterms that make the function equal to 1, where its complement is 0.
  - Example:
    - F(A, B, C)= $\sum(1, 4, 5, 6, 7)$
    - F'(A, B, C)= $\sum(0, 2, 3) = m_0 + m_2 + m_3$

CANONICAL AND STANDARD FORMS

- Example:
  - F(A, B, C)= ∑(1, 4, 5, 6, 7)
  - Thus F'(A, B, C)= ∑(0, 2, 3) = $m_0 + m_2 + m_3$
  - Complement of F' by DeMorgan's theorem
    - F = $(m_0 + m_2 + m_3)' = m_0'\ m_2'\ m_3' = M_0\ M_2\ M_3 = \prod(0, 2, 3)$
  - $m_0' = M_j$

CANONICAL AND STANDARD FORMS

- Sum of minterms = product of maxterms
- Interchange the symbols Σ and Π and list those numbers missing from the original form
  - Σ of 1's
  - Π of 0's

CANONICAL AND STANDARD FORMS

- Example: F = x y + x' z
  - x y = x y (z + z') = x y z + x y z'
  - x'z = x'z (y + y') = x' y z + x' y' z
  - F = x y z + x y z' + x' y z + x' y' z
  - F(x, y, z)= $\sum(1, 3, 6, 7)$
    - with 1's
  - F(x, y, z)= $\prod(0, 2, 4, 5)$
    - with 0's

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

CANONICAL AND STANDARD FORMS

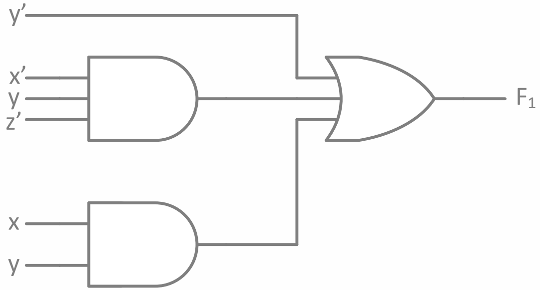
- The two canonical forms of Boolean algebra are basic forms that one obtains from reading a function from the truth table.
- These forms are very seldom the ones with the least number of literals, because each minterm or maxterm must contain, by definition, all the variables either complemented or uncomplemented.
- **Standard form**, the terms that form the function may contain one, two, or any number of literals.
  - **Sum of products and products of sum**

CANONICAL AND STANDARD FORMS

- **Sum of products**
  - Contains AND terms (product terms) of one or more literals each.
  - The sum denotes the ORing of these terms.
  - Example:
    - $F_1 = y' + x\,y + x'\,y\,z'$
      - 3 product terms of one, two and three literals.
      - Their sum is in effect an OR operation.

CANONICAL AND STANDARD FORMS

- $F_1 = y' + x\,y + x'\,y\,z'$
- Logic diagram of sum-of-products
  - Each product term requires and AND gate except for a term with a single literal.
  - The logic sum is formed with an OR gate.

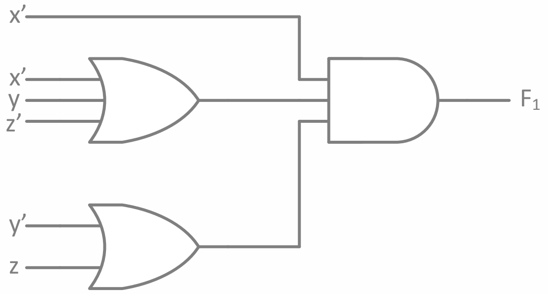


CANONICAL AND STANDARD FORMS

- **Product of sums**
  - Contains OR terms (sum terms) of one or more literals each.
  - The product denotes the ANDing of these terms.
  - Example:
    - $F_2 = x\,(y' + z)\,(x' + y + z')$
      - 3 sum terms of one, two and three literals.
      - Their product is in effect an AND operation.

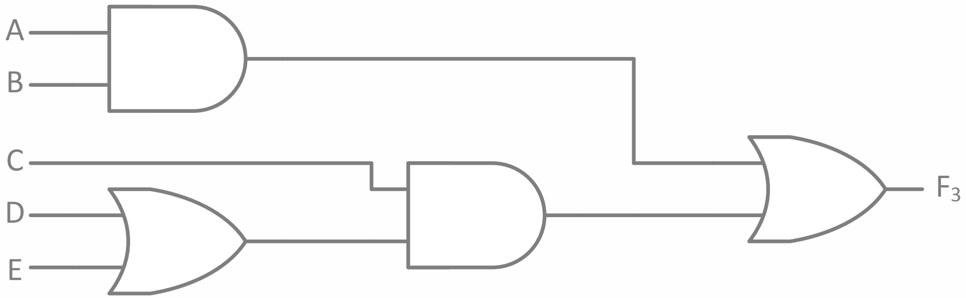
CANONICAL AND STANDARD FORMS

- $F_2 = x\,(y' + z)\,(x' + y + z')$
- Logic diagram of product of sums
  - Each product term requires and AND gate except for a term with a single literal.
  - The logic sum is formed with an OR gate.



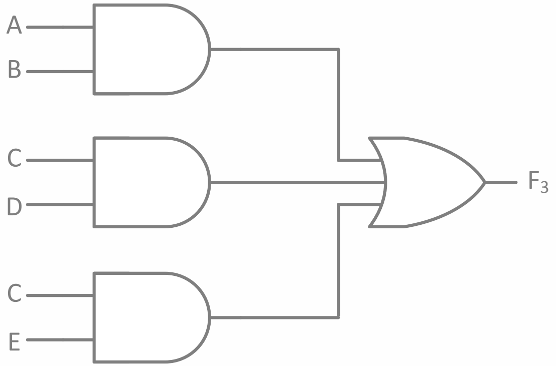
CANONICAL AND STANDARD FORMS

- Multi-level implementation
  - $F_3 = AB + C(D + E)$
  - Neither in sum of products nor in products of sums.



CANONICAL AND STANDARD FORMS

- Change it to a standard form by using the distributive law
  - $F_3 = AB + C(D + E) = AB + CD + CE$



2.6 OTHER LOGIC OPERATIONS

OTHER LOGIC OPERATIONS

- 2^n rows in the truth table of n binary variables.
- 2^{2n} functions for n binary variables.
- **n = 2**, and the number of possible Boolean functions is **16**.

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

OTHER LOGIC OPERATIONS

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary Constant 0
$F_1 = x\,y$	$x \cdot y$	AND	x and y
$F_2 = x\,y'$	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
$F_6 = x\,y' + x'\,y$	$x \oplus y$	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
$F_9 = (x\,y + x'\,y')$	$(x \oplus y)'$	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

OTHER LOGIC OPERATIONS

- 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.
- All the new symbols shown, except for the exclusive-OR symbol $\oplus$ are not in common use by digital designers.

2.7 DIGITAL LOGIC GATES

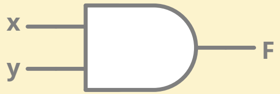
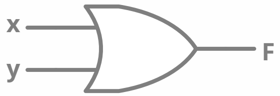
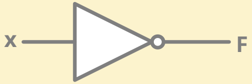
DIGITAL LOGIC GATES

- Boolean expression: AND, OR and NOT operations
- Constructing gates of other logic operations
  1. The feasibility and economy;
  2. The possibility of extending gate's inputs;
  3. The basic properties of the binary operations (commutative and associative);
  4. The ability of the gate to implement Boolean functions.

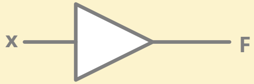
DIGITAL LOGIC GATES

- Consider the 16 functions in Table (**slide 82**)
  - Two are equal to a constant (F_0 and F_{15}).
  - Four are repeated twice (F_4, F_5, F_{10} and F_{11}).
  - Inhibition (F_2) and implication (F_{13}) are not commutative or associative.
  - The other eight: **complement (F_{12}), transfer (F_3), AND (F_1), OR (F_7), NAND (F_{14}), NOR (F_8), XOR (F_6), and equivalence (XNOR) (F_9)** are used as standard gates.
  - Complement: inverter.
  - Transfer: buffer (increasing drive strength).
  - Equivalence: XNOR.

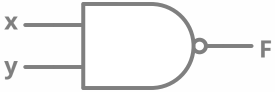
DIGITAL LOGIC GATES

<table>
<tr><th>Name</th><th>Graphic Symbol</th><th>Algebraic Function</th><th>Truth Table</th></tr>
<tr><td>AND</td><td>
</td><td>F = x y</td><td><table><tr><td>x</td><td>y</td><td>F</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>OR</td><td>
</td><td>F = x + y</td><td><table><tr><td>x</td><td>y</td><td>F</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>INVERTER</td><td>
</td><td>F = x ’</td><td><table><tr><td>x</td><td>y</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table></td></tr>
</table>

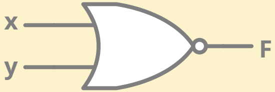
DIGITAL LOGIC GATES

<table>
<tr><th>Name</th><th>Graphic Symbol</th><th>Algebraic Function</th><th>Truth Table</th></tr>
<tr><td>BUFFER</td><td>
</td><td>F = x</td><td>

x	y
0	0
1	1

</td></tr>
<tr><td>NAND</td><td>
</td><td>F = (x y) ’</td><td>

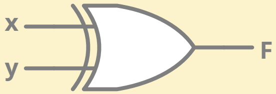
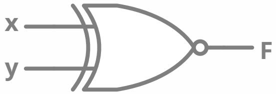
x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

</td></tr>
<tr><td>NOR</td><td>
</td><td>F = (x + y) ‘</td><td>

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

</td></tr>
</table>

DIGITAL LOGIC GATES

Name	Graphic Symbol	Algebraic Function	Truth Table		
			x	y	F
XOR		$F = x \oplus y$	0	0	0
			0	1	1
			1	0	1
			1	1	0
			x	y	F
XNOR		$F = (x \oplus y)'$	0	0	1
			0	1	0
			1	0	0
			1	1	1

DIGITAL LOGIC GATES

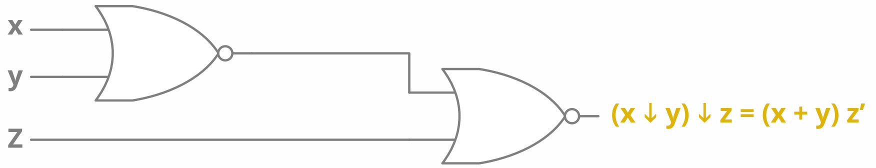
- Extension to multiple inputs
  - A gate can be extended to multiple inputs.
    - If its binary operation is commutative and associative.
  - AND and OR are commutative and associative.
    - OR
      - $x + y = y + x$
      - $(x + y) + z = x + (y + z) = x + y + z$
    - AND
      - $x\,y = y\,x$
      - $(x\,y)\,z = x\,(y\,z) = x\,y\,z$

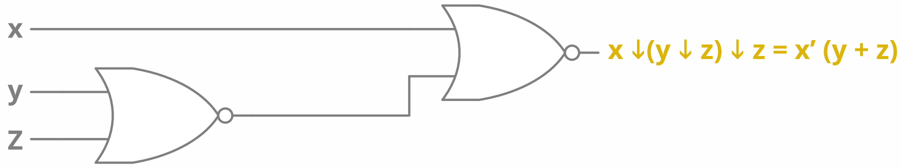
DIGITAL LOGIC GATES

- The NAND and NOR functions are commutative, but they are not associative
  - $(x \downarrow y) \downarrow z \neq x\ (y \downarrow z)$
    - $(x \downarrow y) \downarrow z = [\ (x + y)' + z]'$

      $= (x + y)\ z' = x\ z' + y\ z'$
    - $x\ (y \downarrow z) = [\ x + (y + z)'\ ]'$

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



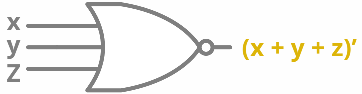


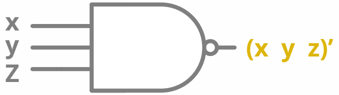
DIGITAL LOGIC GATES

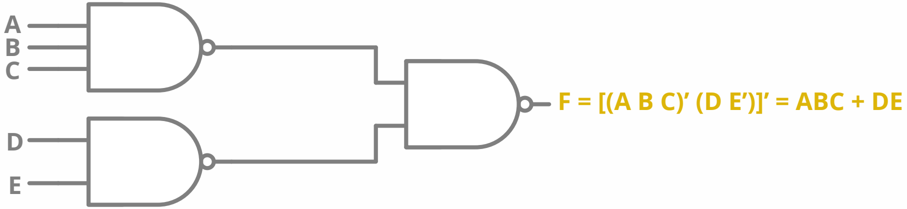
- The NAND and NOR functions are commutative, and their gates can be extended to have more than two inputs, provided that the definition of the operation is slightly modified.
- The difficulty is that the NAND and NOR operators are not associative
  - **$(x \downarrow y) \downarrow z \neq x\ (y \downarrow z)$**
    - **$(x \downarrow y) \downarrow z = [\,(x + y)' + z]' = (x + y)\, z' = x\, z' + y\, z'$**
    - **$x\ (y \downarrow z) = [\, x + (y + z)'\,]' = x'\, (y + z) = x'\, y + x'\, z$**
- To overcome this;
  - Define the multiple NOR (or NAND) gate as complemented OR (or AND) gate.
    - **$x \downarrow y \downarrow z = (x + y + z)'$**
    - **$x \uparrow y \uparrow z = (x\ \ y\ \ z)'$**

DIGITAL LOGIC GATES

- Multiple NOR = a complement of OR gate,
- Multiple NAND = a complement of AND.
- The cascaded NAND operations = sum of products.
- The cascaded NOR operations = product of sums.

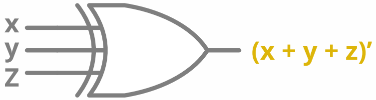


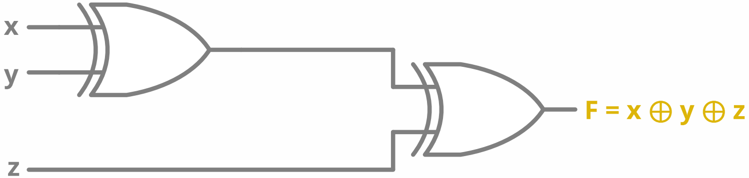




DIGITAL LOGIC GATES

- The XOR and XNOR gates are commutative and associative.
- Multiple-input XOR gates are uncommon?
- XOR is an odd function: it is equal to 1 if the inputs variables have an odd number of 1's.



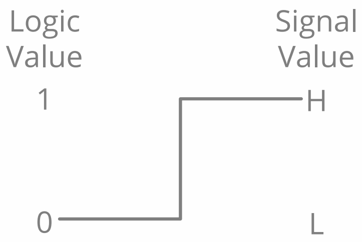


x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

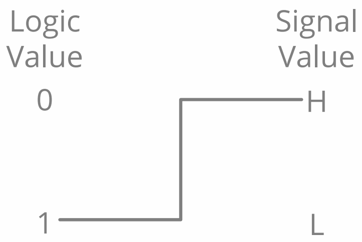
DIGITAL LOGIC GATES

- The binary signals at the inputs and outputs of any gate has one of two values, except during transition.
  - One signal value represents logic-1
  - The other logic-0.
- Two signal values are assigned to two logic values
- There exist two different assignments of signal level to logic value.

DIGITAL LOGIC GATES



Positive Logic



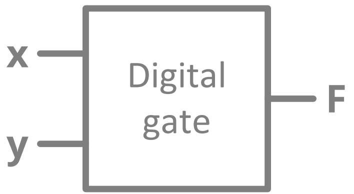
Negative Logic

- The higher signal level is designated by H and the lower signal level by L.
- Choosing the high-level H to represent logic-1 defines a positive logic system.
- Choosing the low-level L to represent logic-0 defines a negative logic system.

DIGITAL LOGIC GATES

x	y	F
L	L	L
L	H	L
H	L	L
H	H	H

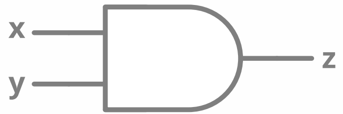
(a) Truth table with H and L



(b) Gate block diagram

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

(c) Truth table for positive logic



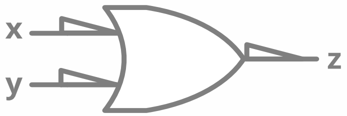
(d) Positive logic AND gate

- The physical behaviour of the gate when H is 3 volts and L is 0 volts.
- Truth table of (c) assumes positive logic assignment, with H = 1 and L = 0.

DIGITAL LOGIC GATES

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

(e) Truth table for negative logic



(f) Negative logic OR gate

- Truth table of (e) assumes positive logic assignment, with H = 0 and L = 1.

2.8 INTEGRATED CIRCUITS

INTEGRATED CIRCUITS

Level of Integration

- An IC (a chip)
- Examples:
  - Small-scale Integration (SSI): < 10 gates
  - Medium-scale Integration (MSI): 10 ~ 100 gates
  - Large-scale Integration (LSI): 100 ~ xk gates
  - Very Large-scale Integration (VLSI): > xk gates

INTEGRATED CIRCUITS

- VLSI
  - Small size (compact size)
  - Low cost
  - Low power consumption
  - High reliability
  - High speed

INTEGRATED CIRCUITS

- Digital logic families: circuit technology
  - TTL: transistor-transistor logic (dying?)
  - ECL: emitter-coupled logic (high speed, high power consumption)
  - MOS: metal-oxide semiconductor (NMOS, high density)
  - CMOS: complementary MOS (low power)
  - BiCMOS: high speed, high density

INTEGRATED CIRCUITS

- The characteristics of digital logic families
  - Fan-out: the number of standard loads that the output of a typical gate can drive.
  - Power dissipation.
  - Propagation delay: the average transition delay time for the signal to propagate from input to output.
  - Noise margin: the minimum of external noise voltage that caused an undesirable change in the circuit output.

INTEGRATED CIRCUITS

- CAD – Computer-Aided Design
  - Millions of transistors
  - Computer-based representation and aid
  - Automatic the design process
  - Design entry
    - Schematic capture
    - HDL – Hardware Description Language
      - Verilog, VHDL
  - Simulation
  - Physical realization
    - ASIC, FPGA, PLD

INTEGRATED CIRCUITS

- Why is it better to have more gates on a single chip?
  - Easier to build systems
  - Lower power consumption
  - Higher clock frequencies
- What are the drawbacks of large circuits?
  - Complex to design
  - Chips have design constraints
  - Hard to test

INTEGRATED CIRCUITS

- Need tools to help develop integrated circuits
  - Computer Aided Design (CAD) tools
  - Automate tedious steps of design process
  - Hardware description language (HDL) describe circuits
  - VHDL (see the lab) is one such system