Chapter 2
Boolean Algebra and Logic Gates

Definitions

Theorems

Functions

Canonical and Standard Forms

Operations

Gates

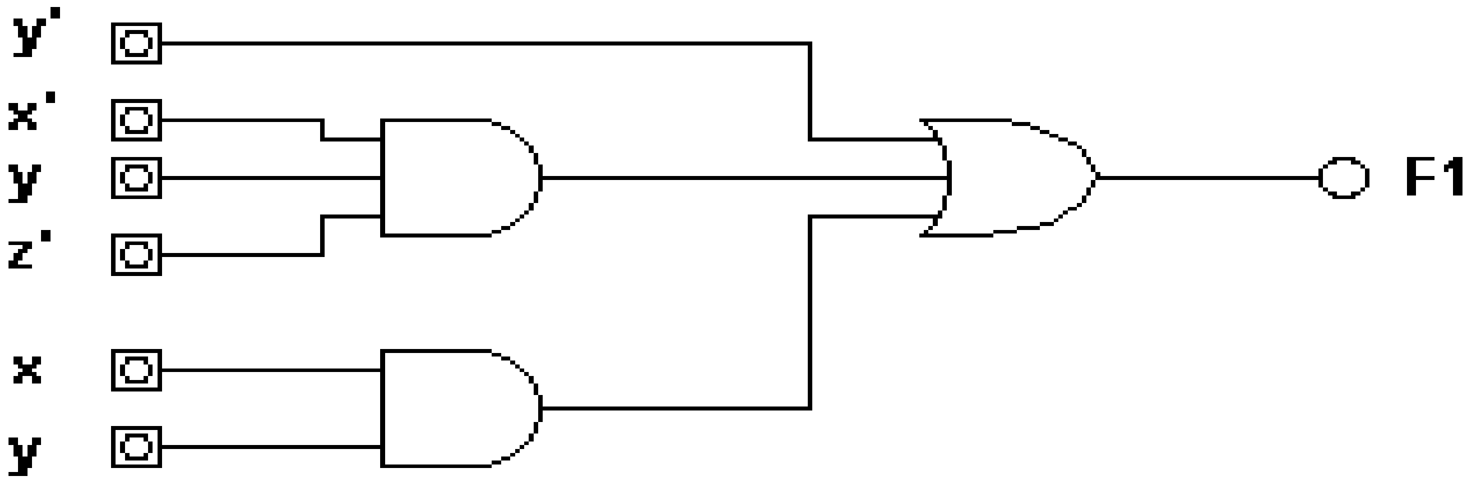
Integrated Circuits

Standard Forms

- **Standard forms are those forms that allow the terms forming the function to consist of any number of the variables.**
- **There are two standard forms:**
  - **sum of products (SOP)**
  - **product of sums (POS)**

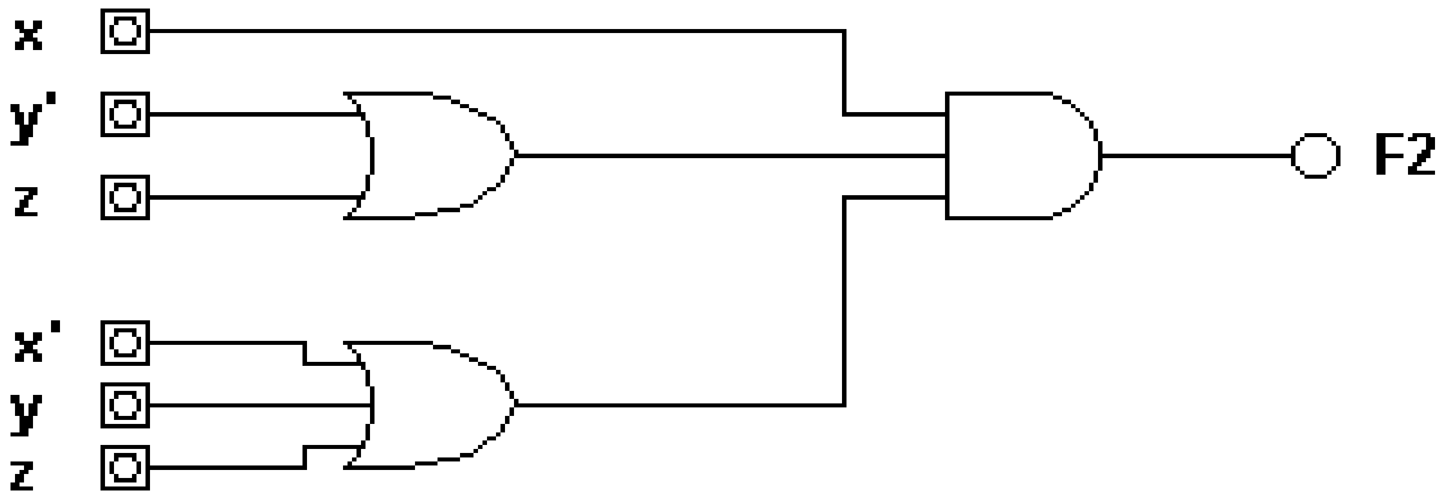
Sum of Products

- The Sum of Products (SOP) is a Boolean expression containing AND terms, called product terms, of one or more literals each.
  - $F_1 = y' + xy + x'yz'$



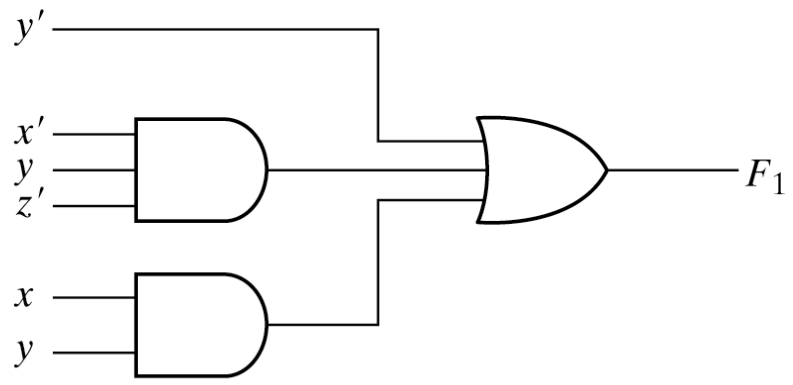
Product of Sums

- The Product of Sums (POS) is a Boolean expression containing OR terms, called sum terms, of one or more literals each.
  - $F_2 = x(y' + z)(x' + y + z')$

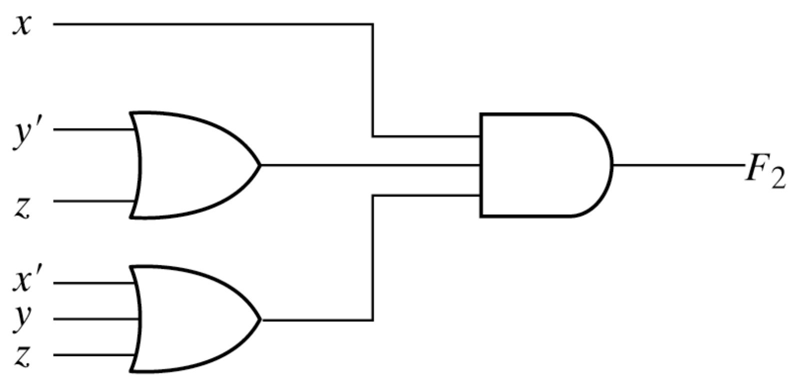


Two Level Implementation

- **The standard type of expression results in a two-level gating structure**



(a) Sum of Products



(b) Product of Sums

Fig. 2-3 Two-level implementation

Conversion from Nonstandard to Standard Form

- A Boolean function may be expressed in a nonstandard form (fig 2.4a shows a function that is neither in sum of products nor in product of sums). It has three levels of gating
- It can be converted to a standard form (Sum of product) by using distributive law to remove parenthesis
- Two-level implementation is preferred as it produces the least amount of delay

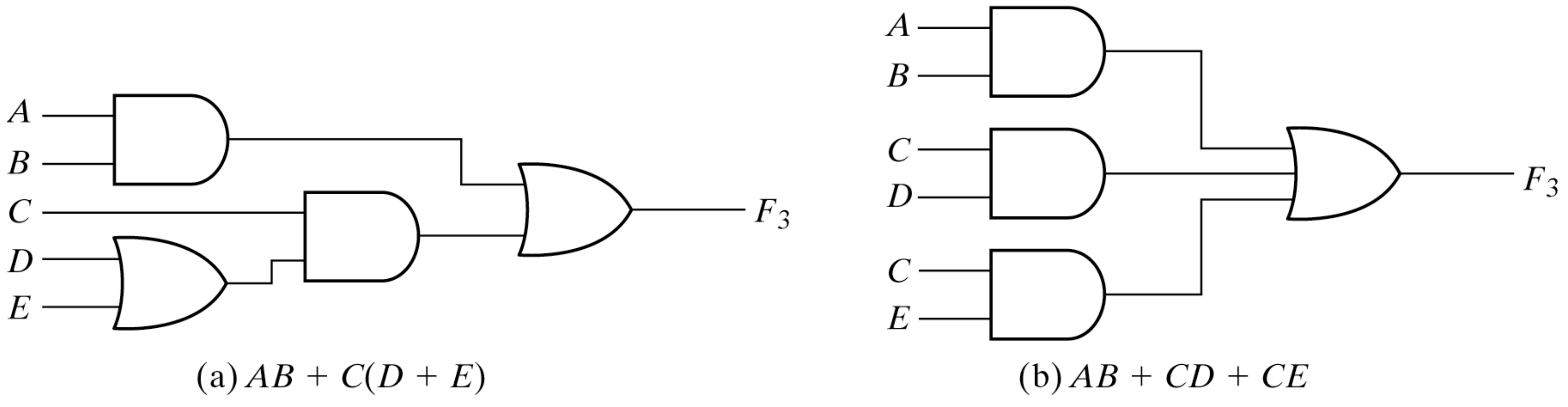


Fig. 2-4 Three- and Two-Level implementation

Other Logic Operations

- **Given two Boolean variables:**
  - When binary operators AND and OR are placed between two variables they form two Boolean functions x . y and x + y
  - there are 2^{2x2} = 16 combinations of the two variables as there are 2^{2n} possible functions for n binary variables (we will see the details of these 16 functions in next slides)
  - each combination of the variables can result in one of two values, 0 or 1, therefore there are 2^4=16 functions (combinations of 0's and 1's for the four combinations, 00,01,10,11)
- **AND and OR represent two of the 16 possible functions.**

Function Combinations

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

- **F_1 represents the AND Operation**
- **F_7 represents the OR Operation**
- **There are 14 other functions**

16 Two-Variable Functions

Boolean Function	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 (XNOR)	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

Function Gate Implementations

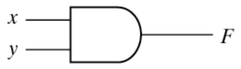
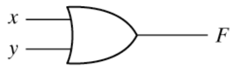
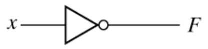
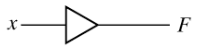
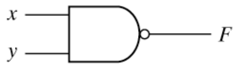
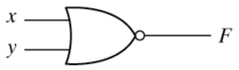
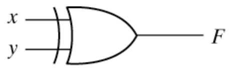
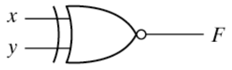
Name	Graphic symbol	Algebraic function	Truth table
AND		$F = xy$	x y F 0 0 0 0 1 0 1 0 0 1 1 1
OR		$F = x + y$	x y F 0 0 0 0 1 1 1 0 1 1 1 1
Inverter		$F = x'$	x F 0 1 1 0
Buffer		$F = x$	x F 0 0 1 1
NAND		$F = (xy)'$	x y F 0 0 1 0 1 1 1 0 1 1 1 0
NOR		$F = (x + y)'$	x y F 0 0 1 0 1 0 1 0 0 1 1 0
Exclusive-OR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

Fig. 2-5 Digital logic gates

Function Gate Implementations

- It is easier to implement a Boolean function with these types of gates (as seen on last slide)
- Inverter (Complement), Buffer (transfer), AND, OR, NAND, NOR, X-OR, and XNOR (equivalence) are used as standard gates in digital design
- NAND and NOR are extensively used logic gates and are more popular than AND and OR gates because these gates are easily constructed with transistor circuits and digital circuits are easily implemented with them

Multiple Inputs

- **All of the previous defined gates, with the exception of the inverter and the buffer, can have multiple inputs.**
  - **A gate can have multiple inputs provided it is a binary operation that is commutative ($x + y = y + x$ and $xy = yx$) and associative ($x + (y + z) = (x + y) + z$ and $x(yz) = (xy)z$)**
  - **NAND and NOR functions are commutative but not associative**
  - **To overcome this difficulty we define multiple NOR (or NAND) gate as a complemented OR (or AND) gate e.g., as $(x+y+z)'$ or $(xyz)'$**

Multiple Inputs (Nonassociative NOR operation)

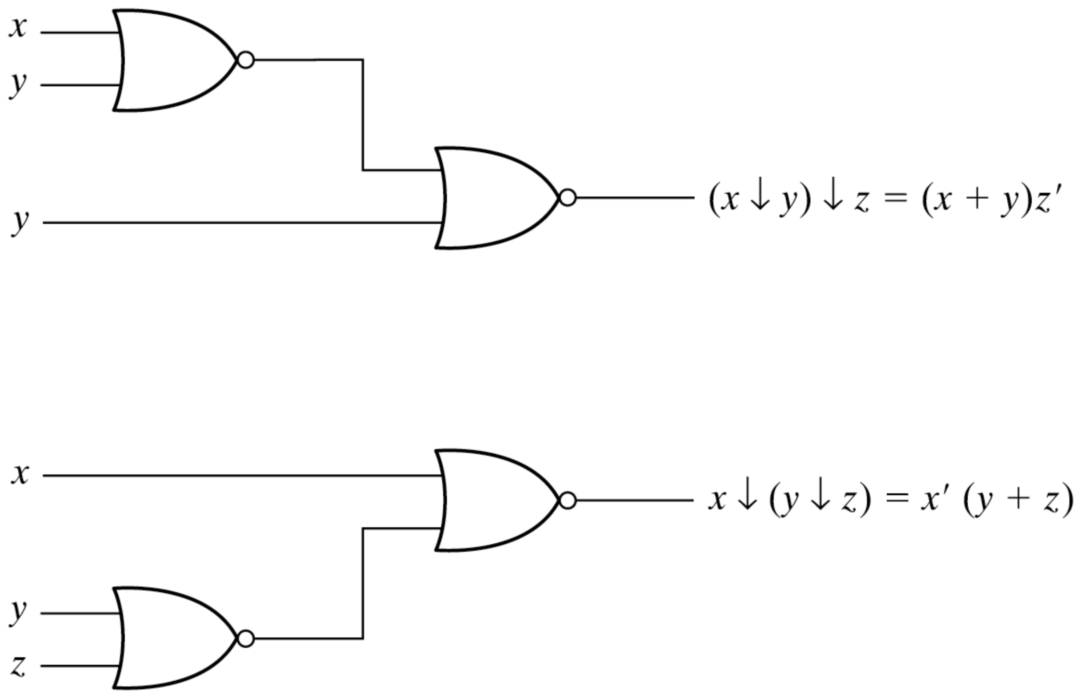
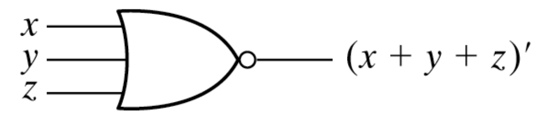
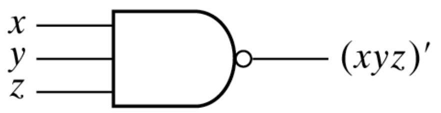

x
y
y
$(x \downarrow y) \downarrow z = (x + y)z'$
x
y
z
$x \downarrow (y \downarrow z) = x' (y + z)$

Fig. 2-6 Demonstrating the nonassociativity of the NOR operator; $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$

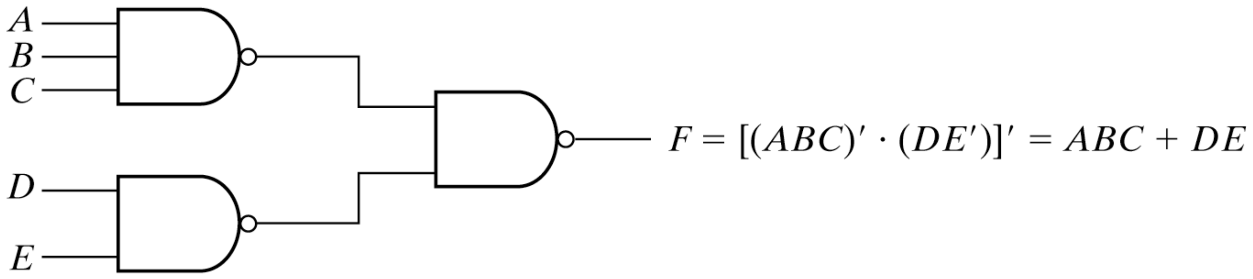
Multiple Inputs NOR and NAND gates



(a) 3-input NOR gate



(b) 3-input NAND gate



(c) Cascaded NAND gates

Fig. 2-7 Multiple-input and cascaded NOR and NAND gates