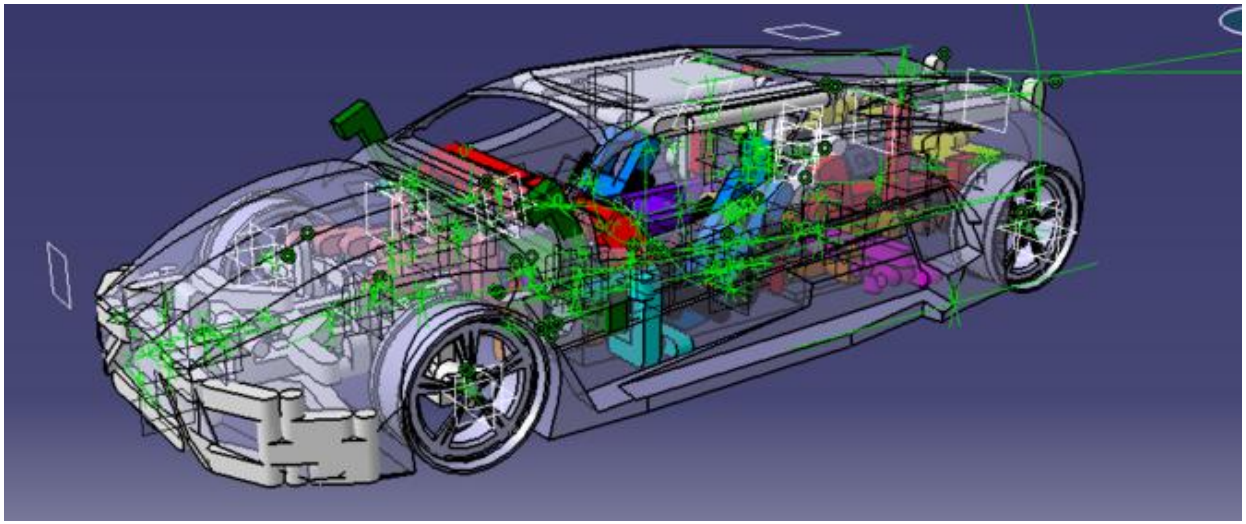
Chap 2
Boolean Algebra and Logic Gates

Hanwool Jeong

hwjeong@kw.ac.kr

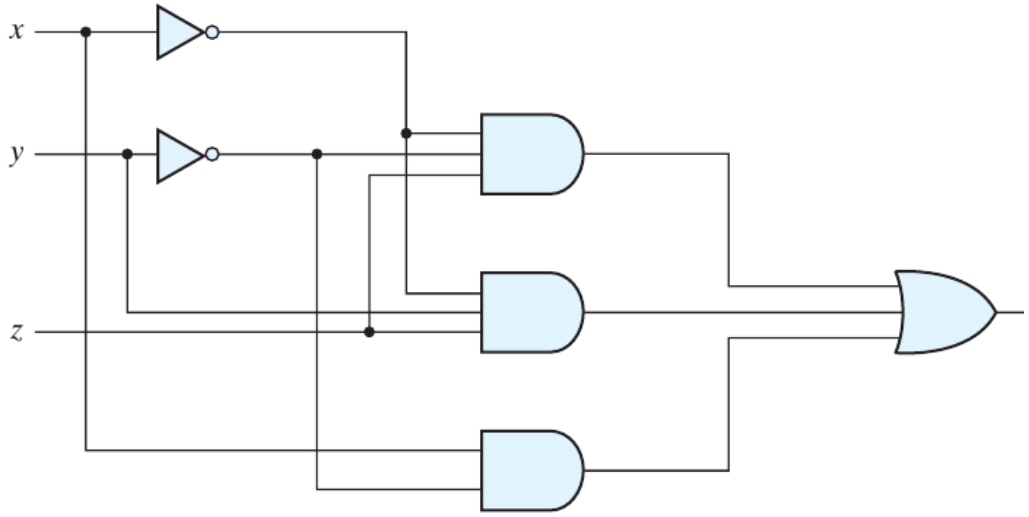
Roles of Math in Engineering

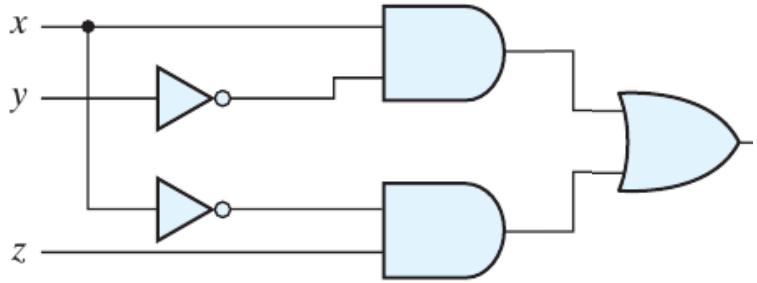
- It dramatically improves the efficiency in “communication” and “problem solving”.
- Consider you are an engineer designing a car.



Boolean “Algebra”? Why Math in Circuit?

- To find simpler and cheaper realizations of a circuit
- Mathematical methods that simplify circuits rely primarily on Boolean algebra.





Defining an Algebraic System

- Generally, algebraic system is composed on a set S, with operators, e.g. a *field* of real numbers
- In 1854, George Boole developed an algebraic system now called Boolean algebra.
- In 1938, Claude E. Shannon introduced a two-valued Boolean algebra called *switching algebra* that represented the properties of bistable electrical switching circuits.
- How can we formally defining Boolean algebra?
  ➔ With postulates formulated by E. V. Huntington in 1904

Two-Valued Boolean Algebra

- Two-valued Boolean algebra is defined on a set of B = {0, 1}, with rules for the two binary operators + and • as

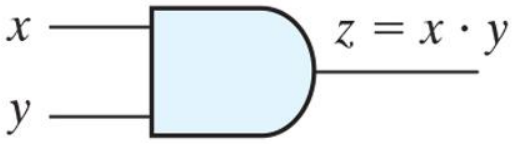
x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

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

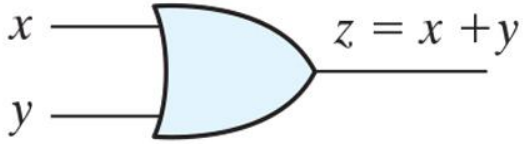
x	x'
0	1
1	0

- You may easily understand and feel the operation.

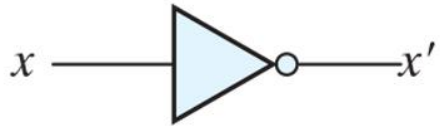
Revisiting Logic Operation



(a) Two-input AND gate



(b) Two-input OR gate



(c) NOT gate or inverter

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		

Postulates and Theorems

- Generalization of binary variables through x, y, and z.
- The postulates need no proof, while theorems must be proven from the postulates.

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)	$x(x + y) = x$

Let's Focus on Postulates First

- Identity & complement

(a) $x + 0 = x$ (b) $x \cdot 1 = x$

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

➔ Can you see duality?

- Commutative

(a) $x + y = y + x$ (b) $xy = yx$

- Distributive

(a) $x(y + z) = xy + xz$ (b) $x + yz = (x + y)(x + z)$

Proof of Theorem 1

THEOREM 1(a): $x + x = x.$

THEOREM 1(b): $x \cdot x = x.$

Proof of Theorem 2

THEOREM 2(a): $x + 1 = 1.$

THEOREM 2(b): $x \cdot 0 = 0$

Proof of Theorem 3 & Theorem 6

THEOREM 3: $(x')' = x.$

THEOREM 6(a): $x + xy = x.$

THEOREM 6(b): $x(x + y) = x$

Proof Through Truth Table

THEOREM 6(a): $x + xy = x.$

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

- You can easily prove theorem 3 & 4 with truth tables

Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$

Boolean Functions

- Consider

$$F_1 = x + y'z$$

- You can consider this F_1 as

"F_1 is equal to 1 if x is equal to 1 or if both y' and z are equal to 1. F_1 is equal to 0 otherwise."

Table Representation of F_1

$$F_1 = x + y'z$$



x	y	z	F_1
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

Gate Implementation of F_1

$$F_1 = x + y'z$$

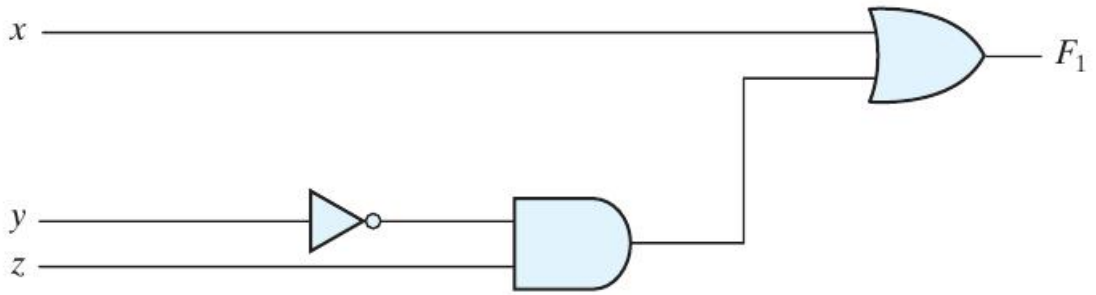


FIGURE 2.1
Gate implementation of $F_1 = x + y'z$

Practice

- $F = x_1x_2 + z'(y_1'+y_2)$

Various Ways of Expressing F_2

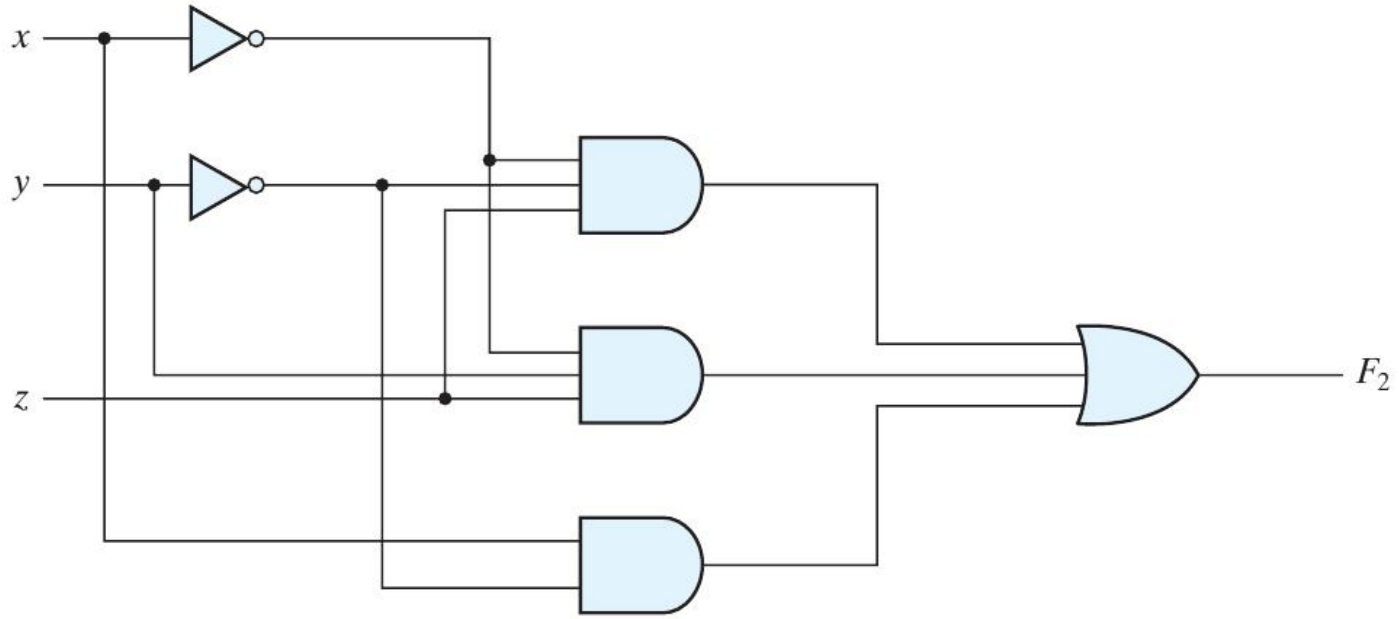
- Consider F_2 given by truth table shown

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

- Can you derive Boolean function of F_2 in algebraic expression? You can do it by examining “1” cases.

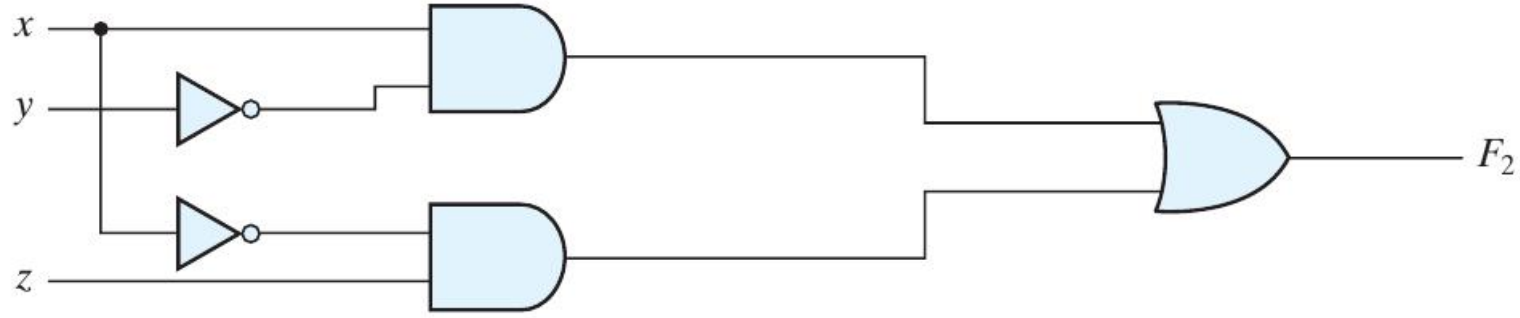
Gate Implementation of F_2

You Can Simplify It!



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

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



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

Algebraic Manipulation

- 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.

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

- Exposed to human error: More systematic and robust tool will be learned in the next chapter
- Still it becomes base and framework for the tool

Simplify Followings!

1. $x(x' + y) =$
2. $x + x'y =$
3. $(x + y)(x + y') =$
4. $xy + x'z + yz =$

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

Complement of Functions

- F_1 = x'y'z + x'y'z. Derive F_1'

- F_2 = x(y'z'+yz). F_2' = ?

Deriving Complementary Function Using Duality

- Take the dual of the function and complement each literal.

- F_1 = x'y'z + x'y'z. Derive F_1‘

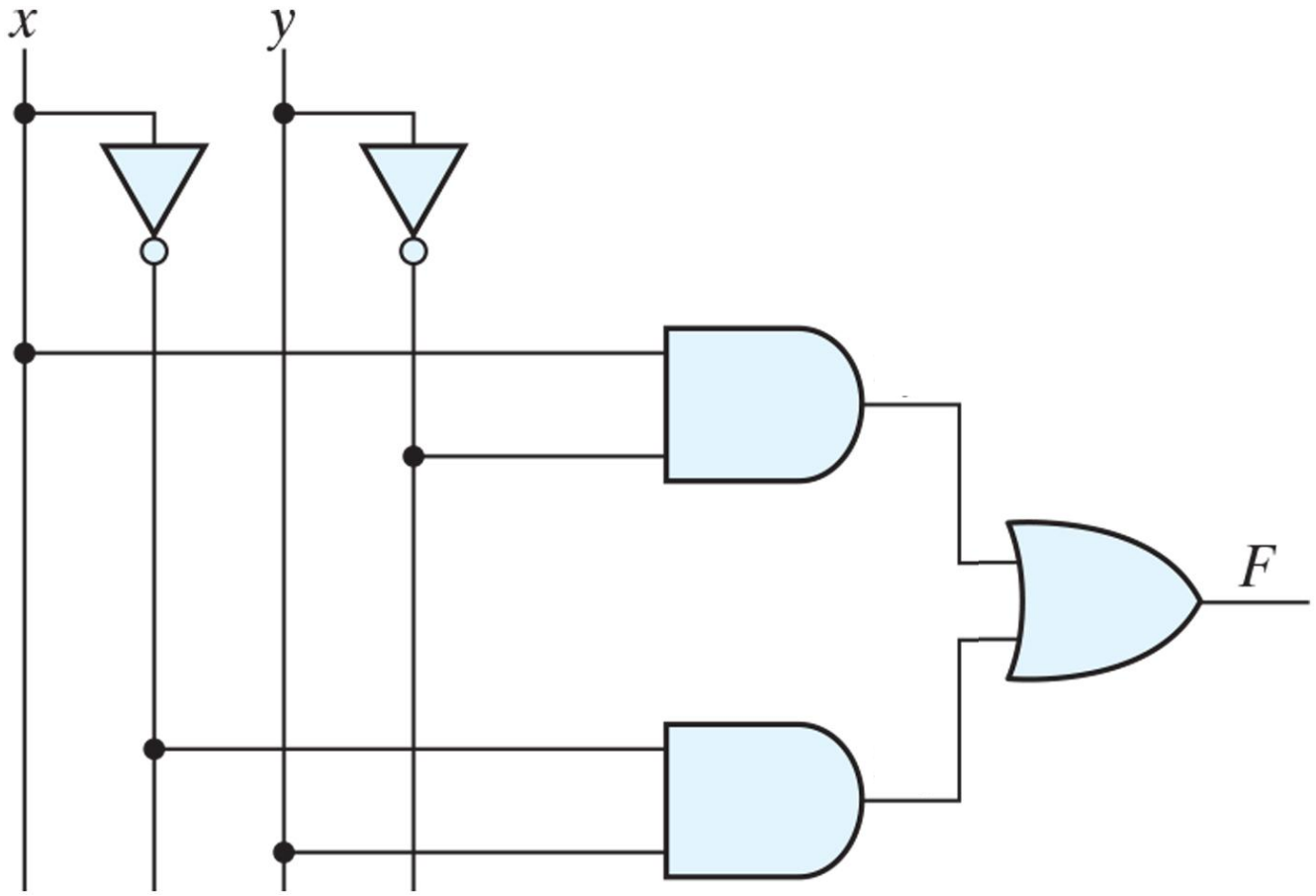
- F_2 = x(y'z'+yz). F_2' = ?

Practice2.3

- Draw a logic diagram for the Boolean function F = x'y+xy'.

Practice 2.4

- Derive the Boolean expression of the following implementation.



Minterms and Maxterms

			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 that 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 ➔ What is the relation of m_j & M_j?

Function Expression With Minterms

- Express f_1 and f_2 using sum of minterms

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

Function Expression With Maxterms

- Express f_1 and f_2 as product of maxterms
  (You can start from f_1' & and f_2')

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

Canonical Form

1) Form a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those minterms.
2) 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

Obtaining Sum of Minterms

EXAMPLE 2.4

Express the Boolean function $F = A + B'C$ as a sum of minterms.

- Brief notation using Σ:

Obtaining Sum of Minterms From Truth Table

- Truth table for F = A + B'C

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

Obtaining Sum of Maxterms

EXAMPLE 2.5

Express the Boolean function $F = xy + x'z$ as a product of maxterms.

- Brief notation using Π:

Conversion b/w Canonical Forms

- Consider the function of

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

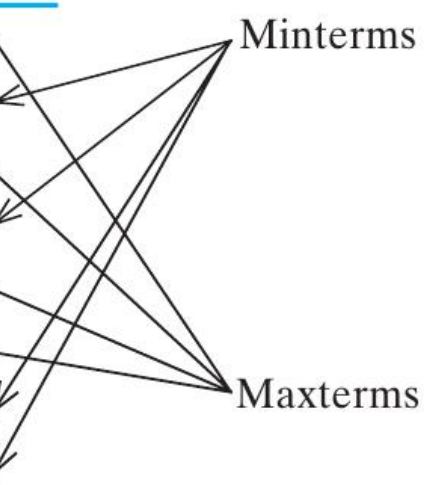
- Complementary F':

- With a complement operation again, F can be expressed as sum of maxterms

Example

- F = xy + x'z

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 =
- F' =

Practice 2.7 & 2.8

- Find a product of maxterms expression of F(x,y,z) = Σ(1,2,3,5,7)

- Find a sum of minterms expression of F = Π(1,3,4,6)

Practice 2.9

- Identify the minterms and maxterms of the truth table for F shown below.

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

Standard Forms

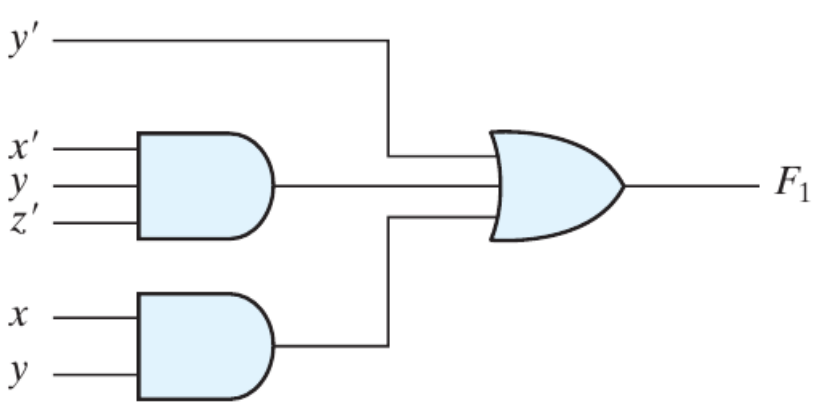
- Two types of standard forms: **the sum of products** and **the products of sums**
- Sum of product:

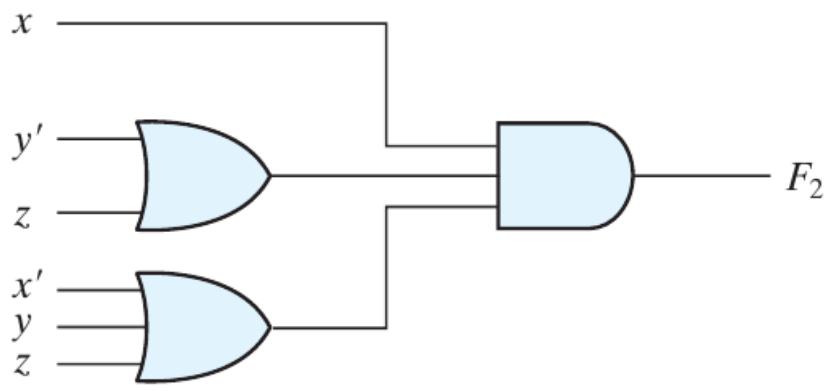
$$F_1 = y' + xy + x'yz'$$

- Product of sum:

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

- The standard type results in a two-level structure of gates.





Nonstandard Form

- Example, $F_3 = AB + C(D + E)$

- Can be changed into standard form:

$$F_3 = AB + C(D + E) = AB + CD + CE$$

Other Logic Operations

- There are so many possible functions: 2^{2^n} function for n variables

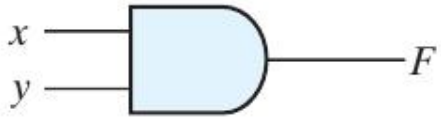
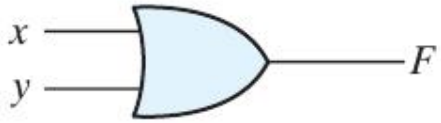
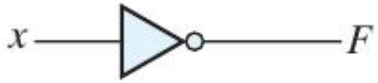
x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

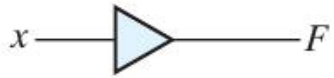
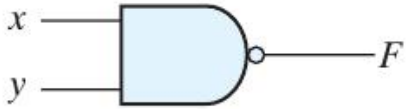
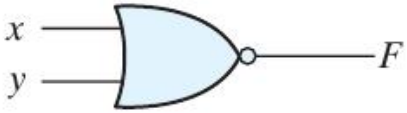
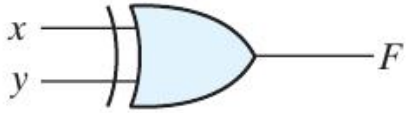
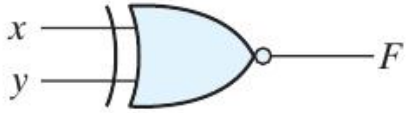
- There is no reason one cannot assign special operator symbols for expressing the other functions.
- Of course, still, each function can be expressed in terms of the Boolean operators AND, OR, and NOT.

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

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

Digital Logic Gates (cont'd)

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	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

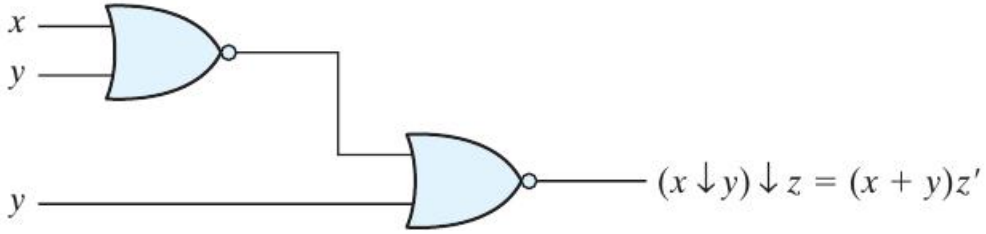
Buffer		$F = x$	x | F 0 | 0 1 | 1
NAND		$F = (xy)'$	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = (x + y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

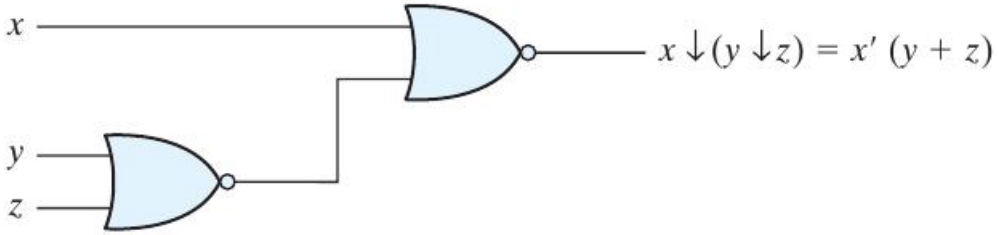
Extension to Multiple Inputs

- For OR,

$$(x + y) + z = x + (y + z) = x + y + z \quad \text{(associative)}$$

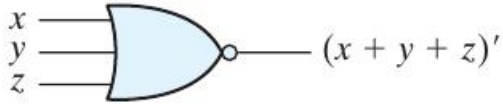
- For NOR or NAND, they are non associative



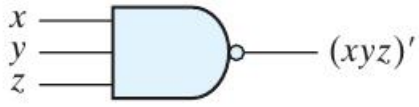


- Thus, multiple inputs are defined as

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

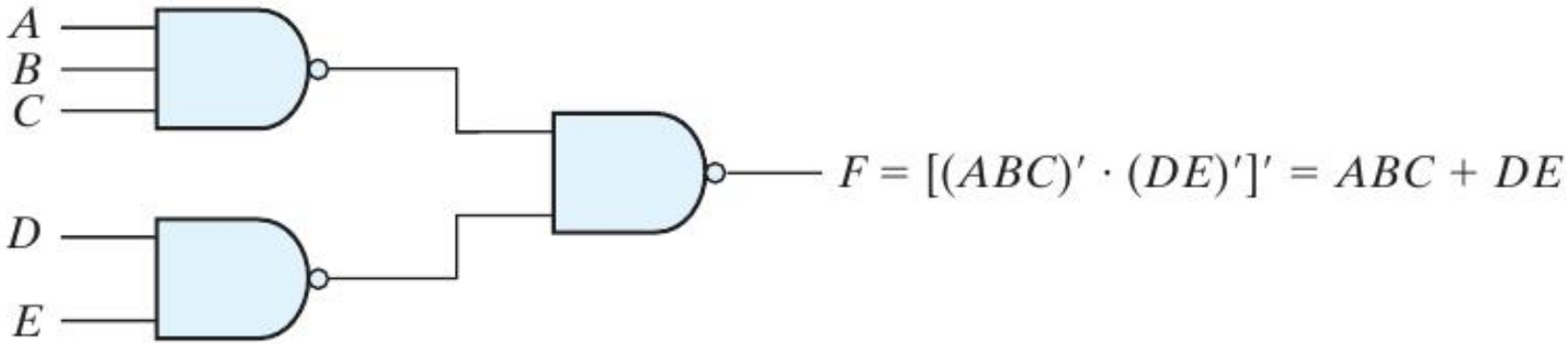


(a) 3-input NOR gate



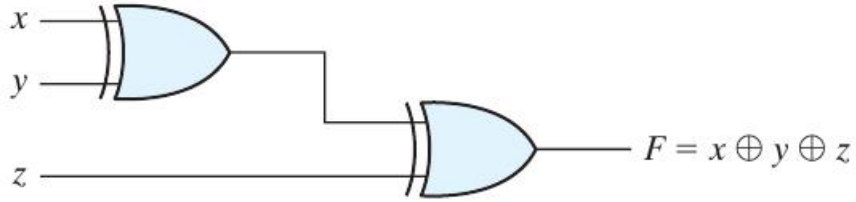
(b) 3-input NAND gate

Sum of Product Using Cascading NAND Gates

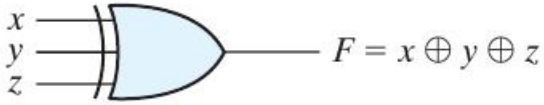


Multiple Inputs to XOR

- XOR is an odd function



(a) Using 2-input gates



(b) 3-input gate

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

(c) Truth table