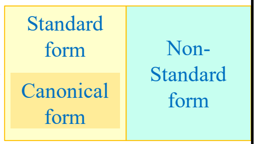
Chapter 2

Boolean Algebra
and
Logic Gates

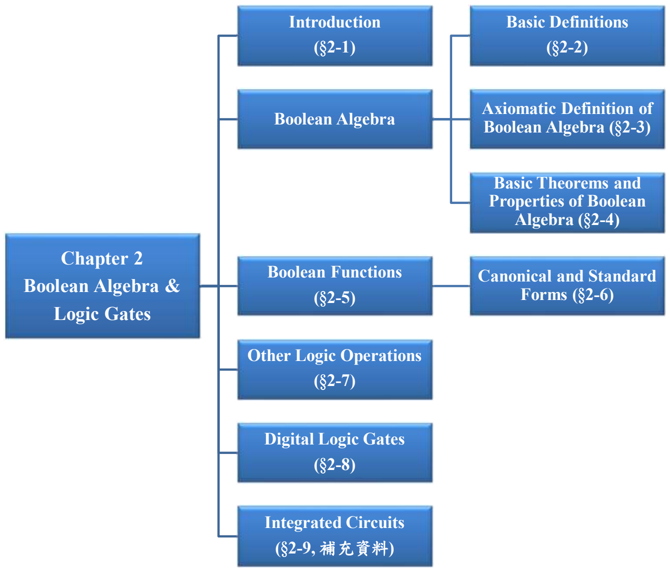
J.J. Shann

■1

Chapter Overview



■2



■3

Exercises in Textbook (6th ed.)

Sections	Exercises	Typical Ones
§2-3	2.1	2.1(a)
§2-4	2.2~2.4, 2.24	2.3, 2.4
§2-5	2.5~2.9, 2-13	2.6
§2-6	2.10, 2.11, 2.15~2.23, 2.27, 2.29~2.31	2.17(a)*, 2.20, 2.22*, 2.30
§2-7	2.12, 2.14, 2.25	2.14, 2.25
§2-8	2.26, 2.28, 2.32, 2.33	2.28, 2.32*

* : Answers to problems appear at the end of the text.

■4

2-1

Introduction

J.J. Shann

Introduction

- Important factor of digital circuit design:
  - the cost of the circuit
  - ⇒ Find simpler and cheaper, but equivalent, realizations of a circuit.
- **Boolean algebra**:
  - Mathematical methods that simplify circuits rely primarily on Boolean algebra.

2-2

Basic Definitions

J.J. Shann

Basic Definitions

- A deductive mathematical system may be defined with
  - a set of elements: S
  - a set of operators: binary/unary operator
  - a number of unproven axioms or postulates:
    - Form the basic assumptions from which it is possible to deduce the rules, theorems, and properties of the system

Common Postulates

- elements: $\in S$
- operators
- axioms or postulates

- For algebraic structures:
  1. Closure
     - E.g.: the set of natural numbers $N = \{1, 2, 3, 4, \ldots\}$ is closed w.r.t. "+" (arithmetic addition), but not to "–" (arithmetic subtraction)
  2. Associative law: $(x * y) * z = x * (y * z)$
  3. Commutative law: $x * y = y * x$
  4. Identity element: e, $e * x = x * e = x$
     - E.g.: 0 is an identity element w.r.t. "+" on the set of integers $I = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
  5. Inverse: $x * y = e$
     - E.g.: In the set of integers, I, and the operator +, with $e = 0$, the inverse of an element a is $(-a)$.
  6. Distributive law: $x * (y \bullet z) = (x * y) \bullet (x * z)$

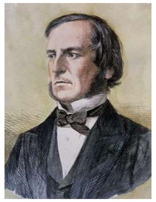
2-3

Axiomatic Definition of Boolean Algebra

J.J. Shann

Axiomatic Definition of Boolean Algebra

- 1854 George Boole: Boolean algebra
- 1904 E. V. Huntington: postulates https://en.wikipedia.org/wiki/Edward_Vermilye_Huntington
- 1938 C. E. Shannon: 2-valued Boolean algebra (switching algebra, binary logic)



George Boole
(1815~1864)
https://en.wikipedia.org/wiki/George_Boole

C.E. Shannon
(1916~2001)
https://en.wikipedia.org/wiki/Claude_Shannon

Boolean Algebra

- Boolean algebra: an algebra structure
  - a set of elements: ***B***
  - a set of operators: 2 binary operators: +, •
  - (Huntington) postulates

Huntington Postulates

- (Huntington) postulates:
  1. (a) Closure w.r.t the operator +.
     (b) Closure w.r.t the operator •.
  2. (a) An identity element w.r.t. +: 0, $x + 0 = 0 + x = x$
     (b) An identity element w.r.t. •: 1, $x \bullet 1 = 1 \bullet x = x$
  3. (a) Commutative w.r.t. +: $x + y = y + x$
     (b) Commutative w.r.t. • : $x \bullet y = y \bullet x$
  4. (a) • is distributive over +: $x \bullet (y + z) = (x \bullet y) + (x \bullet z)$
     (b) + is distributive over • : $x + (y \bullet z) = (x + y) \bullet (x + z)$
  5. Complement: x', (a) $x + x' = 1$ (b) $x \bullet x' = 0$
  6. There exists at least two elements $x, y \in \boldsymbol{B}$ s.t. $x \neq y$.
- ❖ Associative law: can be derived from the other postulates

J.J. Shann 2-13

Two-Valued Boolean Algebra

- Set of elements: $\boldsymbol{B} = \{0, 1\}$
- Set of operators: 2 binary operators: +, •

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

AND

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

OR

x	x'
0	1
1	0

NOT
(Postulate 5)

- (Huntington) postulates:

J.J. Shann 2-14

$x\ y$	$x \cdot y$
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

- (Huntington) postulates:
  1. Closure
  2. Identity elements: two; 0 for +, 1 for •
     $x + 0 = 0 + x = x$, $x \bullet 1 = 1 \bullet x = x$
  3. Commutative laws: $x + y = y + x$, $x \bullet y = y \bullet x$
  4. (a) • is distributive over +: $x \bullet (y + z) = (x \bullet y) + (x \bullet z)$
     (b) + is distributive over • : $x + (y \bullet z) = (x + y) \bullet (x + z)$
  5. Complement:
     (a) $x + x' = 1$: $0 + 0' = 0 + 1 = 1$, $1 + 1' = 1 + 0 = 1$
     (b) $x \bullet x' = 0$: $0 \bullet 0' = 0 \bullet 1 = 0$, $1 \bullet 1' = 1 \bullet 0 = 0$
  6. $\boldsymbol{B} = \{0,1\}$, $0 \neq 1$.

J.J. Shann 2-15

Proof by Truth Table

*** Prove 4(b)!**

<Proof> 4(a) $x \bullet (y + z) = (x \bullet y) + (x \bullet z)$

$x\ y\ z$	$y + z$	$x \bullet (y + z)$	$x \bullet y$	$x \bullet z$	$(x \bullet y) + (x \bullet z)$
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

2-4

Basic Theorems and Properties of Boolean Algebra

J.J. Shann

Basic Theorems and Properties of Boolean Algebra

- Duality:
  - Every algebraic expression deducible from the postulates of Boolean algebra (or truth table) remains valid
    if the operators and identity elements are interchanged
    (OR ↔ AND, 0 ↔ 1)

    * Positive vs. Negative Logic
    (p.2-86~2-88)
  - E.g.:

$$\left.\begin{array}{l} 4.(a)\ x \bullet (y+z) = (x \bullet y) + (x \bullet z) \\ \quad (b)\ x + (y \bullet z) = (x+y) \bullet (x+z) \end{array}\right]\ \text{dual}$$

$$\left.\begin{array}{l} 5.(a)\ x + x' = 1 \\ \quad (b)\ x \bullet x' = 0 \end{array}\right]\ \text{dual}$$

A. Basic Theorems

- Postulates and theorems of Boolean algebra:
  - Postulates: basic axioms, need no proof
  - Theorems: must be proven
    1. From the postulates and proven theorems
    2. From truth table

dual

Postulate/Theorem	(a)	(b)
Postulate 2, identity element	$x + 0 = x$	$x \cdot 1 = x$
Postulate 5, complement	$x + x' = 1$	$x \cdot x' = 0$
Postulate 3, commutative	$x + y = y + x$	$x\,y = y\,x$
Postulate 4, distributive	$x\,(y + z) = xy + xz$	$x + (y\,z) = (x + y)\,(x + z)$
Theorem 1	$x + x = x$	$x \cdot x = x$
Theorem 2	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Theorem 4, associative	$x + (y + z) = (x + y) + z$	$x\,(y\,z) = (x\,y)\,z$
Theorem 5, DeMorgan	$(x + y)' = x'\,y'$	$(xy)' = x' + y'$
Theorem 6, absorption	$x + xy = x$	$x\,(x + y) = x$
Theorem *, consensus	$xy + x'z + yz = xy + x'z$	$(x + y)\,(x' + z)\,(y + z) = (x + y)\,(x' + z)$

dual

Postulate/Theorem	(a)	(b)
Postulate 2, identity element	$x + 0 = x$	$x \cdot 1 = x$
Postulate 5, complement	$x + x' = 1$	$x \cdot x' = 0$
Theorem 1	$x + x = x$	$x \cdot x = x$
Theorem 2	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Postulate 3, commutative	$x + y = y + x$	$x y = y x$
Theorem 4, associative	$x + (y + z) = (x + y) + z$	$x (y z) = (x y) z$
Postulate 4, distributive	$x (y + z) = xy + xz$	$x + (y z) = (x + y) (x + z)$
Theorem 5, DeMorgan	$(x + y)' = x' y'$	$(xy)' = x' + y'$
Theorem 6, absorption	$x + xy = x$	$x (x + y) = x$
Theorem *, consensus	$xy + x'z + yz = xy + x'z$	$(x + y) (x' + z) (y + z) = (x + y) (x' + z)$

J.J. Shann 2-21

21

Theorem 1

- 1(a): $x + x = x$

$$x + x = (x + x) \bullet 1 \quad \text{by postulate: } 2(b)$$
$$= (x + x)(x + x') \quad 5(a)$$
$$= x + xx' \quad 4(b)$$
$$= x + 0 \quad 5(b)$$
$$= x \quad 2(a)$$

x	$x + x$
0	
1	

- 1(b): $x \bullet x = x$

$$x \bullet x = x x + 0 \quad \text{by postulate: } 2(a)$$
$$= x x + x x' \quad 5(b)$$
$$= x (x + x') \quad 4(a)$$
$$= x \bullet 1 \quad 5(a)$$
$$= x \quad 2(b)$$

x	$x \cdot x$
0	$0 \cdot 0 = 0$
1	$1 \cdot 1 = 1$

J.J. Shann 2-22

22

Theorem 2

- 2(a): $x + 1 = 1$

$$x + 1 = 1 \bullet (x + 1) \quad \text{by postulate: } 2(b), 3(b)$$
$$= (x + x')(x + 1) \quad 5(a)$$
$$= x + x' \bullet 1 \quad 4(b)$$
$$= x + x' \quad 2(b)$$
$$= 1 \quad 5(a)$$

x	$x + 1$
0	
1	

- 2(b): $x \bullet 0 = 0$

$x + 1 = 1$ is proven

by duality principle (OR ↔ AND, 0 ↔ 1)

$x \bullet 0 = 0$

J.J. Shann 2-23

23

Theorem 3: Involution

- $(x')' = x$
  - Postulate 5 defines the complement of x:

    $x + x' = 1$ and $x x' = 0$
  - The complement of x' is x and is also $(x')'$.

    Since the complement is unique $\Rightarrow (x')' = x$.

x	x'	$(x')'$
0		
1		

J.J. Shann 2-24

24

Theorem 5: DeMorgan

Postulate/Theorem	(a)	(b)
Postulate 2, identity element	$x + 0 = x$	$x \cdot 1 = x$
Postulate 5, complement	$x + x' = 1$	$x \cdot x' = 0$
Postulate 3, commutative	$x + y = y + x$	$x y = y x$
Postulate 4, distributive	$x (y + z) = xy + xz$	$x + (y z) = (x + y)(x + z)$
Theorem 1	$x + x = x$	$x \cdot x = x$
Theorem 2	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Theorem 4, associative	$x + (y + z) = (x + y) + z$	$x (y z) = (x y) z$
Theorem 5, DeMorgan	$(x + y)' = x' y'$	$(xy)' = x' + y'$
Theorem 6, absorption	$x + xy = x$	$x (x + y) = x$
Theorem *, consensus	$xy + x'z + yz = xy + x'z$	$(x + y)(x' + z)(y + z) = (x + y)(x' + z)$

- 5(a): $(x + y)' = x' y'$

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

=

- 5(b): $(x\ y)' = x' + y'$

* Prove Theorem 5 by using Postulate 5!

- Two variables: $(x + y)' = x' y'$, $(x\ y)' = x' + y'$
- Three or more variables:

$(A + B + C)' = (A + X)'$ let $B + C = X$
$= A' X'$ by DeMorgan's
$= A' (B + C)'$ substitute $B + C = X$
$= A' (B' C')$ by DeMorgan's
$= A' B' C'$ associative

- Generalized form:

$(A + B + C + \ldots + G)' = A' B' C' \ldots G'$
$(A B C \ldots G)' = A' + B' + C' + \ldots + G'$
$\Rightarrow$ AND $\leftrightarrow$ OR and complement each literal

Theorem 6: Absorption

Postulate/Theorem	(a)	(b)
Postulate 2, identity element	$x + 0 = x$	$x \cdot 1 = x$
Postulate 5, complement	$x + x' = 1$	$x \cdot x' = 0$
Postulate 3, commutative	$x + y = y + x$	$x y = y x$
Postulate 4, distributive	$x (y + z) = xy + xz$	$x + (y z) = (x + y)(x + z)$
Theorem 1	$x + x = x$	$x \cdot x = x$
Theorem 2	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Theorem 4, associative	$x + (y + z) = (x + y) + z$	$x (y z) = (x y) z$
Theorem 5, DeMorgan	$(x + y)' = x' y'$	$(xy)' = x' + y'$
Theorem 6, absorption	$x + xy = x$	$x (x + y) = x$
Theorem *, consensus	$xy + x'z + yz = xy + x'z$	$(x + y)(x' + z)(y + z) = (x + y)(x' + z)$

- 6(a): $x + xy = x$

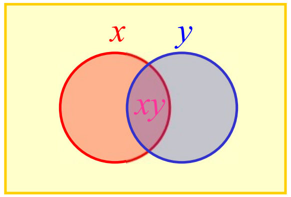
i. $x + xy = x \bullet 1 + xy$ by postulate: 2(b)
$= x (1 + y)$ 4(a)
$= x (y + 1)$ 3(a)
$= x \bullet 1$ (Theorem) 2(a)
$= x$ 2(b)

ii. Truth table:

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

=

- 6(a): $x + xy = x$
  - * Venn diagram:



Theorem * : Consensus

Postulate/Theorem	(a)	(b)
Postulate 2, identity element	$x + 0 = x$	$x \cdot 1 = x$
Postulate 5, complement	$x + x' = 1$	$x \cdot x' = 0$
Postulate 3, commutative	$x + y = y + x$	$x y = y x$
Postulate 4, distributive	$x (y + z) = xy + xz$	$x + (y z) = (x + y) (x + z)$
Theorem 1	$x + x = x$	$x \cdot x = x$
Theorem 2	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Theorem 4, associative	$x + (y + z) = (x + y) + z$	$x (y z) = (x y) z$
Theorem 5, DeMorgan	$(x + y)' = x' y'$	$(xy)' = x' + y'$
Theorem 6, absorption	$x + xy = x$	$x (x + y) = x$
Theorem *, consensus	$xy + x'z + yz = xy + x'z$	$(x + y) (x' + z) (y + z) = (x + y) (x' + z)$

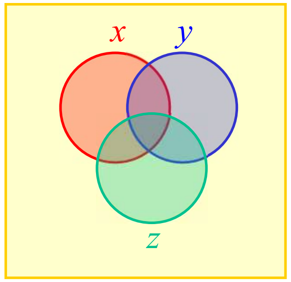
- $xy + x'z + yz = xy + x'z$
  $(x + y) (x' + z) (y + z) = (x + y) (x' + z)$

$xy + x'z + yz$
$= xy + x'z + (x + x') yz$ … P2(b), P3(b), P5(a)
$= xy + x'z + xyz + x'yz$ … P4(a)
$= xy (1 + z) + x'z (1 + y)$ … P3(a), P4(a)
$= xy + x'z$ … T2(a), P2(b)

- can be used to eliminate redundant terms from Boolean expressions.
- can be used to generate redundant terms for further simplification.

* Venn diagram

- $xy + x'z + yz = xy + x'z$
  - * Venn diagram:



B. Operator Precedence

- Operator precedence for evaluating Boolean expression:
  - parenthesis
  - NOT
  - AND
  - OR
- Example:
  $(x + y)' + x'y$

2-5

Boolean Functions

Boolean Functions

- Boolean algebra:
  - is an algebra dealing w/ binary variables and logic ops
    - binary variables: are designated by letters of the alphabet
    - logic ops: AND, OR, NOT
- Boolean expression:
  - an algebraic expression formed by using
    binary variables,
    the constants 0 and 1,
    the logic op symbols, and
    parentheses.
  - E.g.: $X \cdot (\overline{Y} + Z) + Z \cdot 1$

J.J. Shann 2-33

33

Boolean Function

- Boolean function:
  - can be described by
    a Boolean equation,
    a truth table, or
    a logic ckt diagram

J.J. Shann 2-34

34

Boolean Equation

- Boolean equation:
  - consists of a binary variable identifying the function followed by an equal sign and a Boolean expression.
  - expresses the logical relationship b/t binary variables
  - can be expressed in a variety of ways
    * Obtain a simpler expression for the same function.
  - E.g.:

$$F(W,X,Y,Z) = X\overline{YZ} + \overline{Y}Z + \overline{W}XYZ + WXY + \overline{W}XY\overline{Z}$$

$$= X + \overline{Y}Z$$

terms

J.J. Shann 2-35

35

Truth Table

- Truth table for a function: is unique
  - a list of all combinations of 1's and 0's that can be assigned to the binary variables and a list that shows the value of the function for each binary combination.
  - E.g.:

$$F(X,Y,Z) = X + \overline{Y}Z$$

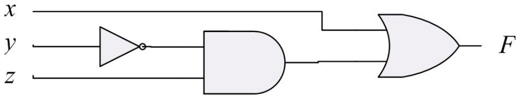
X Y Z	F
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	1

J.J. Shann 2-36

36

Logic Circuit Diagram

- Logic circuit diagram:
  - An algebraic expression for a Boolean function
    $\Rightarrow$ A ckt diagram composed of logic gates
  - Circuit gates are interconnected by wires that carry logic signals.
  - E.g.: $F(x,y,z) = x + \bar{y}z$



* Combinational logic circuits

J.J. Shann 2-37

■37

Example

Inputs $\nrightarrow$ Insurance Policy $\rightarrow$ Output

- Present a set of requirements under which an insurance policy will be issued:

  The applicant must be
  1. a married female 25 years old or over, or
  2. a female under 25, or
  3. a married male under 25 who has not been involved in a car accident, or
  4. a married male who has been involved in a car accident, or
  5. a married male 25 years or over who has not been involved in a car accident.

J.J. Shann 2-38

■38

<Ans.>

Inputs $\nrightarrow$ Insurance Policy $\rightarrow$ Output

- Define input variables: 4; w, x, y, z
  - $w = 1$ if applicant has been involved in a car accident
  - $x = 1$ if applicant is married
  - $y = 1$ if applicant is a male
  - $z = 1$ if applicant is under 25
- Find a Boolean expression which assumes the value 1 whenever the policy should be issued:

$x y' z'$	1. a married female 25 years old or over
$y' z$	2. a female under 25
$w' x y z$	3. a married male under 25 who has not been involved in a car accident
$w x y$	4. a married male who has been involved in a car accident
$w' x y z'$	5. a married male 25 years or over who has not been involved in a car accident

$\Rightarrow$ $F = x y' z' + y' z + w' x y z + w x y + w' x y z'$

J.J. Shann 2-39

■39

- Simplify the expression and suggest a simpler set of requirements:

$$F = x y' z' + y' z + \underbrace{w' x y z + w x y + w' x y z'}_{x y}$$
$$= x + y' z$$

Insurance policy

- $w = 1$ if applicant has been involved in a car accident
- $x = 1$ if applicant is married
- $y = 1$ if applicant is a male
- $z = 1$ if applicant is under 25

The applicant must be
1. married or
2. a female under 25.

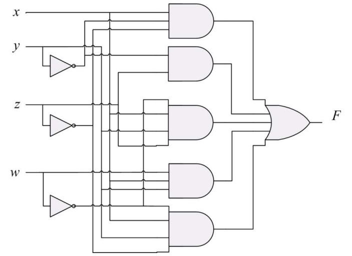
J.J. Shann 2-40

■40

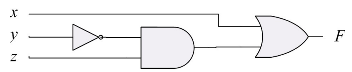
Inputs → Insurance Policy → Output

- Draw the logic circuit diagram:

  $F = x\,y'\,z' + y'\,z + w'\,x\,y\,z + w\,x\,y + w'\,x\,y\,z'$



$= x + y'\,z$



J.J. Shann 2-41

41

Simplification of Boolean Functions

- Boolean algebra is a useful tool for simplifying digital ckts.

Postulate/Theorem	(a)	(b)
Postulate 2, identity element	$x + 0 = x$	$x \cdot 1 = x$
Postulate 5, complement	$x + x' = 1$	$x \cdot x' = 0$
Postulate 3, commutative	$x + y = y + x$	$x\,y = y\,x$
Postulate 4, distributive	$x\,(y + z) = xy + xz$	$x + (y\,z) = (x + y)\,(x + z)$
Theorem 1	$x + x = x$	$x \cdot x = x$
Theorem 2	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Theorem 4, associative	$x + (y + z) = (x + y) + z$	$x\,(y\,z) = (x\,y)\,z$
Theorem 5, DeMorgan	$(x + y)' = x'\,y'$	$(xy)' = x' + y'$
Theorem 6, absorption	$x + xy = x$	$x\,(x + y) = x$
Theorem *, consensus	$xy + x'z + yz = xy + x'z$	$(x + y)\,(x' + z)\,(y + z) = (x + y)\,(x' + z)$

  - E.g.:

$$F_2 = x'y'z + x'yz + xy'$$
$$= x'z\,(y' + y) + xy' \quad \ldots \text{ Postulate 4(a)}$$
$$= x'z \cdot 1 + xy' \quad \ldots \text{ Postulate 5(a)}$$
$$= x'z + xy' \quad \ldots \text{ Postulate 2(b)}$$

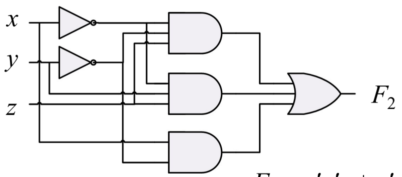
J.J. Shann 2-42

42

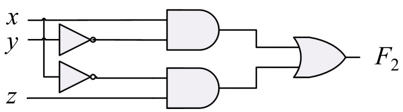
$F_2 = x'y'z + x'yz + xy'$ …… (a)

$= x'z + xy'$ …… (b)

$x\ y\ z$	F_2
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	0



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



$F_2 = x'z + xy'$

J.J. Shann 2-43

43

A. Algebraic Manipulation

- Implementation of Boolean function w/ gates:
  - each term is implemented w/ a gate
  - each literal designates an input to a gate
    - *literal*: a primed or unprimed variable
  - E.g.: $F_2 = x'y'z + x'yz + xy'$ … 3 terms & 8 literal
    $= x'z + xy'$ … 2 terms & 4 literals
- Criterion of equipment minimization:
  - Minimize the # of "literals"
  - Minimize the # of "terms"
- Algebraic manipulation: literal minimization
  - Method: cut-and-try (hard)
    - employ the postulates, basic theorem, ...

J.J. Shann 2-44

44

Example 2-1

- Simplify the following Boolean functions to a minimum # of literals:

1. $x(x' + y)$
   $= xx' + xy$
   $= 0 + xy$
   $= xy$ 3→2

2. $x + x'y$
   $= (x + x')(x + y)$
   $= 1\,(x + y)$
   $= x + y$ 3→2 (or by duality from 1)

Postulate/Theorem	(a)	(b)
Postulate 2, identity element	$x + 0 = x$	$x \cdot 1 = x$
Postulate 5, complement	$x + x' = 1$	$x \cdot x' = 0$
Postulate 3, commutative	$x + y = y + x$	$xy = yx$
Postulate 4, distributive	$x(y + z) = xy + xz$	$x + (yz) = (x + y)(x + z)$
Theorem 1	$x + x = x$	$x \cdot x = x$
Theorem 2	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Theorem 4, associative	$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$
Theorem 5, DeMorgan	$(x + y)' = x'y'$	$(xy)' = x' + y'$
Theorem 6, absorption	$x + xy = x$	$x(x + y) = x$
Theorem *, consensus	$xy + x'z + yz = xy + x'z$	$(x + y)(x' + z)(y + z) = (x + y)(x' + z)$

J.J. Shann 2-45

45

3

3. $(x + y)(x + y')$
   $= xx + xy + xy' + yy'$
   $= x + xy + xy' + 0$
   $= x(1 + y + y')$
   $= x$ 4→1

4. $xy + x'z + yz$
   $= xy + x'z + yz(x+x')$
   $= xy + x'z + xyz + x'yz$
   $= xy(1 + z) + x'z(1 + y)$
   $= xy + x'z$ 6→4

5. $(x + y)(x' + z)(y + z)$
   $= (x + y)(x' + z)$ 6→4 (duality from 4)

Postulate/Theorem	(a)	(b)
Postulate 2, identity element	$x + 0 = x$	$x \cdot 1 = x$
Postulate 5, complement	$x + x' = 1$	$x \cdot x' = 0$
Postulate 3, commutative	$x + y = y + x$	$xy = yx$
Postulate 4, distributive	$x(y + z) = xy + xz$	$x + (yz) = (x + y)(x + z)$
Theorem 1	$x + x = x$	$x \cdot x = x$
Theorem 2	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Theorem 4, associative	$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$
Theorem 5, DeMorgan	$(x + y)' = x'y'$	$(xy)' = x' + y'$
Theorem 6, absorption	$x + xy = x$	$x(x + y) = x$
Theorem *, consensus	$xy + x'z + yz = xy + x'z$	$(x + y)(x' + z)(y + z) = (x + y)(x' + z)$

* 4 & 5
→ Consensus theorem

J.J. Shann 2-46

46

B. Complement of a Function

- Complement of a function F:
  - E.g.: p.2-38 (Insurance Policy)
    F = 1 whenever the insurance policy should be issued
    ⇒ G = F′ = 1 whenever the insurance policy should ***not*** be issued
  - i. interchange of 1's to 0's and 0's to 1's for the values of F in the truth table
  - ii. can be derived algebraically by applying DeMorgan's theorem ⇒ interchange AND and OR ops and complement each variable and constant

    $\overline{X+Y} = \overline{X}\cdot\overline{Y}$ $\overline{X\cdot Y} = \overline{X}+\overline{Y}$

  - iii. take the dual (OR ↔ AND, 0 ↔ 1) of the function eq & complement each literal

J.J. Shann 2-47

47

Example 2.2 $\overline{X+Y} = \overline{X}\cdot\overline{Y}$ $\overline{X\cdot Y} = \overline{X}+\overline{Y}$

- Complementing functions by applying DeMorgan's theorem:
  - E.g.s:

$$F_1 = \overline{X}Y\overline{Z} + \overline{X}\,\overline{Y}Z \qquad \overline{F_1} = (\overline{X}Y\overline{Z} + \overline{X}\,\overline{Y}Z)'$$
$$= (\overline{X}Y\overline{Z})' \cdot (\overline{X}\,\overline{Y}Z)'$$
$$= (X + \overline{Y} + Z)\cdot(X + Y + \overline{Z})$$

$$F_2 = X(\overline{Y}\,\overline{Z} + YZ) \qquad \overline{F_2} = (X(\overline{Y}\,\overline{Z} + YZ))'$$
$$= \overline{X} + (\overline{Y}\,\overline{Z} + YZ)'$$
$$= \overline{X} + (\overline{Y}\,\overline{Z})' \cdot (YZ)'$$
$$= \overline{X} + (Y + Z)\cdot(\overline{Y} + \overline{Z})$$

. Shann 2-48

48

Example 2.3

- Complementing functions by using duals:
  - E.g.s: (dual: OR ↔ AND, 0 ↔ 1)

$$F_1 = \overline{X}Y\overline{Z} + \overline{X}\,\overline{Y}Z$$
$$\text{dual of } F_1 \Rightarrow (\overline{X}+Y+\overline{Z})\cdot(\overline{X}+\overline{Y}+Z)$$
$$\text{complement each literal} \Rightarrow (X+\overline{Y}+Z)\cdot(X+Y+\overline{Z})$$
$$= \overline{F_1}$$
$$F_2 = X(\overline{Y}\,\overline{Z}+YZ)$$
$$\text{dual of } F_2 \Rightarrow X+(\overline{Y}+\overline{Z})\cdot(Y+Z)$$
$$\text{complement each literal} \Rightarrow \overline{X}+(Y+Z)\cdot(\overline{Y}+\overline{Z})$$
$$= \overline{F_2}$$

2-6

Canonical & Standard Forms

Canonical & Standard Forms

- Boolean expressions:
  - an expression formed w/:
    - binary variables
    - constants 0 and 1
    - binary operators OR and AND
    - unary operator NOT
    - parentheses
    - equal sign
  - Special forms:
    - Canonical forms
    - Standard forms

Boolean expressions

Standard form

Canonical form

Non-Standard form

A. Canonical Forms

- Canonical forms:
  - special forms of Boolean expression being expressed directly from truth tables
  - Two types:

    i. *sum-of-minterms form*

    (*min*term) + … + (*min*term)

    ii. *product-of-maxterms form*

    (*max*term) • … • (*max*term)

$x\ y\ z$	F_2
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	1

minterms

- *minterm*: standard product, m_j
  - an AND term of the n variables, w/ each variable being primed if the corresponding bit is 0 and unprimed if a 1.
  - n variables $\rightarrow 2^n$ minterms
  - E.g.: minterms for 3 variables

$x\ y\ z$	minterm		m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0 0 0	$x'y'z'$	(m_0)	**1**	0	0	0	0	0	0	0
0 0 1	$x'y'z$	(m_1)	0	**1**	0	0	0	0	0	0
0 1 0	$x'yz'$	(m_2)	0	0	**1**	0	0	0	0	0
0 1 1	$x'yz$	(m_3)	0	0	0	**1**	0	0	0	0
1 0 0	$xy'z'$	(m_4)	0	0	0	0	**1**	0	0	0
1 0 1	$xy'z$	(m_5)	0	0	0	0	0	**1**	0	0
1 1 0	xyz'	(m_6)	0	0	0	0	0	0	**1**	0
1 1 1	xyz	(m_7)	0	0	0	0	0	0	0	**1**

53

Maxterms

- *Maxterm*: standard sum, M_j $(= m_j')$
  - an OR term of the n variables, w/ each variable being unprimed if the corresponding bit is 0 and primed if a 1.
  - n variables $\rightarrow 2^n$ Maxterms
  - E.g.: Maxterms for 3 variables

$x\ y\ z$	Maxterm		M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0 0 0	$x+y+z$	(M_0)	**0**	1	1	1	1	1	1	1
0 0 1	$x+y+z'$	(M_1)	1	**0**	1	1	1	1	1	1
0 1 0	$x+y'+z$	(M_2)	1	1	**0**	1	1	1	1	1
0 1 1	$x+y'+z'$	(M_3)	1	1	1	**0**	1	1	1	1
1 0 0	$x'+y+z$	(M_4)	1	1	1	1	**0**	1	1	1
1 0 1	$x'+y+z'$	(M_5)	1	1	1	1	1	**0**	1	1
1 1 0	$x'+y'+z$	(M_6)	1	1	1	1	1	1	**0**	1
1 1 1	$x'+y'+z'$	(M_7)	1	1	1	1	1	1	1	**0**

54

minterms vs. Maxterms

- E.g.: for 3 binary variables **(* $M_j = m_j'$)**

$x\ y\ z$	minterm		Maxterm	
000	$x'y'z'$	(m_0)	$x+y+z$	(M_0)
001	$x'y'z$	(m_1)	$x+y+z'$	(M_1)
010	$x'yz'$	(m_2)	$x+y'+z$	(M_2)
011	$x'yz$	(m_3)	$x+y'+z'$	(M_3)
100	$xy'z'$	(m_4)	$x'+y+z$	(M_4)
101	$xy'z$	(m_5)	$x'+y+z'$	(M_5)
110	xyz'	(m_6)	$x'+y'+z$	(M_6)
111	xyz	(m_7)	$x'+y'+z'$	(M_7)

  - E.g.: $\boxed{m_3 = \bar{x}yz} \Rightarrow$

$$\overline{m_3} = \overline{\bar{x}yz} = x + \bar{y} + \bar{z} = M_3$$

55

Canonical Forms

- Canonical forms:
  - A Boolean function can be expressed algebraically from a given truth table by

i. **Sum-of-minterms form**:

(*minterm*) + … + (*minterm*)

- form a minterm for each combination of the variables that produces a 1 in the function, and then take the OR of all those terms

ii. **Product-of-Maxterms form**:

(*maxterm*) • … • (*maxterm*)

- form a maxterm for each combination of the variables that produces a 0 in the function, and then take the AND of all those terms

56

Example

- Given the truth table of two functions, f_1 and f_2, express each of the functions as sum-of-minterms and product-of-maxterms forms.

x y z	f_1	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

<Ans.>

Sum of minterms:

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7 = \Sigma m(1, 4, 7)$$

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7 = \Sigma m(3, 5, 6, 7)$$

x y z	f_1	f_2
0 0 0	0	0
0 0 1	1	0
0 1 0	0	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1
1 1 1	1	1

$f_1 = x'y'z + xy'z' + xyz$
$= m_1 + m_4 + m_7 = \Sigma m(1,4,7)$
$f_2 = x'yz + xy'z + xyz' + xyz$
$= m_3 + m_5 + m_6 + m_7 = \Sigma m(3,5,6,7)$

Product of maxterms:

$$f_1' = m_0 + m_2 + m_3 + m_5 + m_6$$
$$= x'y'z' + x'yz' + x'yz + xy'z + xyz'$$
$$f_1 = (f_1')' = (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z)$$
$$= M_0 M_2 M_3 M_5 M_6 = \Pi M(0, 2, 3, 5, 6)$$
$$f_2 = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)$$
$$= M_0 M_1 M_2 M_4 = \Pi M(0, 1, 2, 4)$$

Conversion to Sum of Minterms Form

- Two methods of expressing a Boolean function in its sum-of-minterms form:
  i. Expand the expression into a sum of AND terms by using the distributive law, $x(y + z) = xy + xz$.
     Any missing variable x in each AND term is ANDed with $(x + x')$.
  ii. Obtain the truth table of the function directly from the algebraic expression and then read the minterms from the truth table.

Example: $F(A, B, C) = A + B'C$

<Ans.>

Approach 1:

$$A = A(B+B')(C+C')$$
$$= (AB + AB')(C+C')$$
$$= AB(C+C') + AB'(C+C')$$
$$= ABC + ABC' + AB'C + AB'C'$$

$$B'C = (A+A')B'C$$
$$= AB'C + A'B'C$$

$$F(A, B, C) = A'B'C + AB'C' + AB'C + ABC' + ABC$$
$$= m1 + m4 + m5 + m6 + m7$$
$$= \Sigma m(1, 4, 5, 6, 7)$$

Approach 2:

$F(A, B, C) = A + B'C$

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

$\Rightarrow$

$F(A, B, C)$
$= A'B'C + AB'C' + AB'C + ABC' + ABC$
$= \Sigma m(1, 4, 5, 6, 7)$

Conversion to Product-of-Maxterms Form

- Two methods of expressing a Boolean function in its product-of-maxterms form:
  i. Convert the function into OR terms by using the distributive law, $x + yz = (x + y)(x + z)$.
     Any missing variable x in each OR term is ORed with xx'.
  ii. Obtain the truth table of the function directly from the algebraic expression and then read the maxterms from the truth table.

Example: $F(x, y, z) = xy + x'z$

<Ans.>

Approach 1:

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

$\Rightarrow$

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

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

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

$\Downarrow$

$F = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')$
$= M_0 M_2 M_4 M_5$
$= \Pi M(0, 2, 4, 5)$

Approach 2:

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

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

$\Rightarrow$

$F(x, y, z)$
$= (x+y+z)(x+y'+z)$
$(x'+y+z)(x'+y+z')$
$= \Pi M(0, 2, 4, 5)$

Conversion b/t Canonical Forms

- To convert from one canonical form to another:
  - Interchange the symbols Σ and Π & list those numbers missing from the original form
  - E.g.: $F = xy + x'z$

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

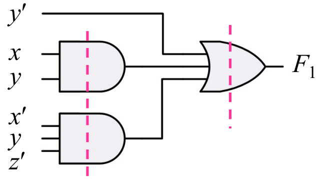
$F(x, y, z)$
$= \Sigma m(1, 3, 6, 7)$
$= ......$
$= \Pi M(0, 2, 4, 5)$
$= ......$

B. Standard Forms

- Canonical forms: special cases of standard forms
  - sum of ***minterms*** & product of ***maxterms***
  - the basic forms that one obtains from reading a function from the truth table.
  - Each terms must contain all the variables
    - Seldom w/ the least # of literals (not minimized)
- Standard forms:
  - Sum of ***products*** & product of ***sums***
    (***AND*** terms) (***OR*** terms)
  - The terms may contain any number of literals

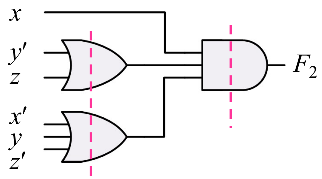
Sum of Products (SOP)

- SOP: (*AND term*) + … + (*AND term*)
  - is a Boolean expression containing ***AND*** terms (***product*** terms) of one or more literals each.
    There ***AND*** terms are ***OR***ing together.
    → 2-level gating structure
    (if the complement of the input variables are available)
  - E.g.: $F_1 = y' + xy + x'yz'$



Product of Sums (POS)

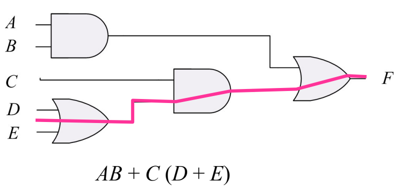
- POS: (*OR term*) • … • (*OR term*)
  - is a Boolean expression containing ***OR*** terms (***sum*** terms) of one or more literals each.
    These ***OR*** terms are ***AND***ing together.
    → 2-level gating structure
    (if the complement of the input variables are available)
  - E.g.: $F_2 = x(y'+z)(x'+y+z')$



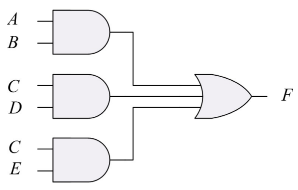
C. Nonstandard Form

- Nonstandard form:
  - E.g.: $F = AB + C(D + E)$ $\rightarrow$ 3-level
- Nonstandard form $\rightarrow$ Standard form:
  - By using the distributive laws
  - E.g.: $F = AB + C(D + E)$
    $= AB + CD + CE$ $\rightarrow$ 2-level

Standard form vs Nonstandard form



$AB + C(D + E)$



$AB + CD + CE$

J.J. Shann 2-69

■69

2-7

Other Logic Operations

J.J. Shann

■70

Other Logic Operations

- Basic logic operations: AND, OR, NOT
- All possible functions of n binary variables:
  - n binary variables $\rightarrow 2^n$ distinct minterms
  - $\rightarrow 2^{2^n}$ possible functions
- E.g.: $n = 2$ $\rightarrow$ 4 minterms $\rightarrow$ 16 possible functions

	$x\ y$	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}
m_0	0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
m_1	0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
m_2	1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
m_3	1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

J.J. Shann 2-71

■71

$x\ y$	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

$F_0 = 0$	Null	$F_{15} = 1$	Identity
$F_1 = xy$	AND, $x \cdot y$	$F_{14} = (xy)'$	NAND, $x \uparrow y$
$F_2 = xy'$	Inhibition, x/y	$F_{13} = x' + y$	Implication, $x \supset y$
$F_3 = x$	Transfer	$F_{12} = x'$	Complement, x'
$F_4 = x'y$	Inhibition, y/x	$F_{11} = x + y'$	Implication, $x \subset y$
$F_5 = y$	Transfer	$F_{10} = y'$	Complement, y'
$F_6 = xy' + x'y$	Exclusive-OR, $x \oplus y$	$F_9 = xy + x'y'$	Eauivalence, $(x \oplus y)'$
$F_7 = x + y$	OR, $x + y$	$F_8 = (x + y)'$	NOR, $x \downarrow y$

J.J. Shann 2-72

■72

$F_0 = 0$	Null	$F_{15} = 1$	Identity
$F_1 = xy$	AND, $x \cdot y$	$F_{14} = (xy)'$	NAND, $x \uparrow y$
$F_2 = xy'$	Inhibition, x/y	$F_{13} = x' + y$	Implication, $x \supset y$
$F_3 = x$	Transfer	$F_{12} = x'$	Complement, x'
$F_4 = x'y$	Inhibition, y/x	$F_{11} = x + y'$	Implication, $x \subset y$
$F_5 = y$	Transfer	$F_{10} = y'$	Complement, y'
$F_6 = xy' + x'y$	Exclusive-OR, $x \oplus y$	$F_9 = xy + x'y'$	Eauivalence, $(x \oplus y)'$
$F_7 = x + y$	OR, $x + y$	$F_8 = (x + y)'$	NOR, $x \downarrow y$

– The 16 functions can be subdivided into 3 categories:
1. 2 functions that produce a constant 0 or 1.
2. 4 functions w/ unary operations complement (NOT) and transfer.
3. 10 functions w/ binary operators that define eight different operations AND, OR, NAND, NOR, exclusive-OR, equivalence, inhibition, and implication.

2-8

Digital Logic Gates

Digital Logic Gates

- Boolean expression:
  - AND, OR and NOT operations
  - It is easier to implement a Boolean function in these types of gates.
- Factors in considering the construction of other types of logic gates:
  - the feasibility and economy of implementing the gate w/ electronic components
  - the possibility of extending the gate to more than two inputs
  - the basic properties of the binary operator
    - E.g.: commutativity, associativity, …

    * Why?
  - the ability of the gate to implement Boolean functions alone or in conjunction w/ other gates

    * Why?

$F_0 = 0$	Null	$F_{15} = 1$	Identity
$F_1 = xy$	AND, $x \cdot y$	$F_{14} = (xy)'$	NAND, $x \uparrow y$
$F_2 = xy'$	Inhibition, x/y	$F_{13} = x' + y$	Implication, $x \supset y$
$F_3 = x$	Transfer	$F_{12} = x'$	Complement, x'
$F_4 = x'y$	Inhibition, y/x	$F_{11} = x + y'$	Implication, $x \subset y$
$F_5 = y$	Transfer	$F_{10} = y'$	Complement, y'
$F_6 = xy' + x'y$	Exclusive-OR, $x \oplus y$	$F_9 = xy + x'y'$	Eauivalence, $(x \oplus y)'$
$F_7 = x + y$	OR, $x + y$	$F_8 = (x + y)'$	NOR, $x \downarrow y$

- Consider the 16 functions:
  - two are equal to a constant 0, 1
  - four are repeated twice Transfer, Complement, Inhibition, Implication
  - inhibition and implication are **not** commutative or associative (×)
  - complement, transfer, AND, OR, NAND, NOR, XOR, and equivalence (XNOR) are used as standard gates
    - complement: inverter, NOT
    - transfer: buffer (drive strength)

  * NAND and NOR gates are far more popular than the AND and OR gates.

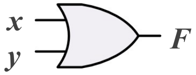
AND, OR

AND

$F = xy$

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

OR



$F = x + y$

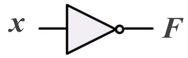
x y	F
0 0	0
0 1	1
1 0	1
1 1	1

J.J. Shann 2-77

77

Inverter, Buffer

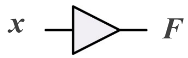
Inverter



$F = x'$

x	F
0	1
1	0

Buffer



$F = x$

x	F
0	0
1	1

J.J. Shann 2-78

78

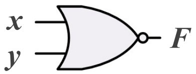
NAND, NOR

NAND

$F = (xy)'$

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

NOR



$F = (x + y)'$

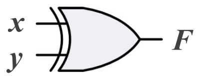
x y	F
0 0	1
0 1	0
1 0	0
1 1	0

J.J. Shann 2-79

79

XOR, NXOR

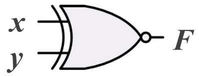
XOR



$F = xy' + x'y$
$= x \oplus y$

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

XNOR



$F = xy + x'y'$
$= (x \oplus y)'$

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

J.J. Shann 2-80

80

Extension to Multiple Inputs

- Extension to multiple inputs
  - A gate can be extended to multiple inputs if its binary operation is commutative and associative.
- AND, OR: commutative and associative:

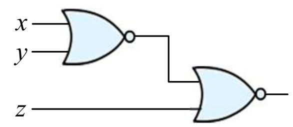
$$x + y = y + x$$
$$(x + y) + z = x + (y + z)$$
$$x \bullet y = y \bullet x$$
$$(x \bullet y) \bullet z = x \bullet (y \bullet z)$$

J.J. Shann 2-81

81

- NAND (↑), NOR (↓):
  - commutative but **not** associative ⇒ not extendable
  - E.g.: $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$

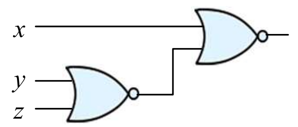
$$(x \downarrow y) \downarrow z = [(x + y)' + z]' = (x + y)\, z' = xz' + yz'$$
$$x \downarrow (y \downarrow z) = [x + (y + z)'\,]' = x'\,(y + z) = x'y + x'z$$



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

$\Sigma m(2, 4, 6)$

$\neq$



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

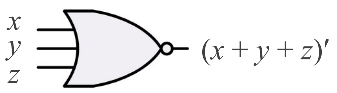
$\Sigma m(1, 2, 3)$

J.J. Shann 2-82

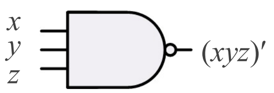
82

- Modification of the definition:
  - The multiple NOR (or NAND) gate is a complement OR (or AND) gate.

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



3-input NOR gate

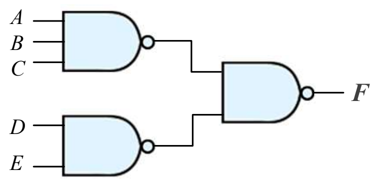


3-input NAND gate

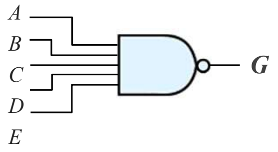
J.J. Shann 2-83

83

- In writing cascaded NOR and NAND operations:
  - Use the correct parentheses to signify the proper sequence of the gates.
  - E.g.: Cascaded NAND gates



$\neq$



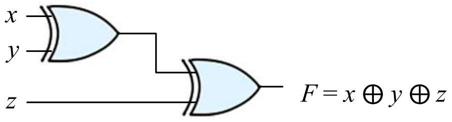
$F = [(ABC)' \cdot (DE)']' = ABC + DE$

$G = (ABCDE)'$
$= A' + B' + C' + D' + E'$

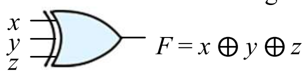
J.J. Shann 2-84

84

- Exclusive-OR (⊕) and Equivalence:
  - are commutative and associative
  - Modification of the definition of multiple-input XOR:
    - an odd function: = 1, if the input variables have an odd # of 1's



Using 2-input gates



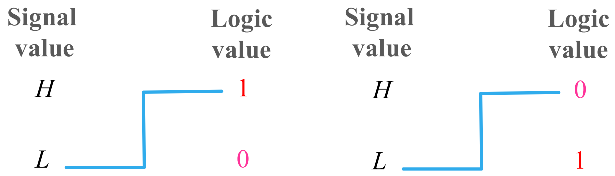
Using 3-input gate

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

  - Equivalence: an even function

Positive & Negative Logic

- Logic-value assignment: polarity assignment
  - Positive logic
  - Negative logic



Positive logic Negative logic

- Demonstration of positive & negative logic:

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

Truth table with *H* and *L*

H = 1
L = 0

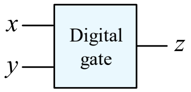
x	*y*	*z*
0	0	0
0	1	0
1	0	0
1	1	1

Truth table for positive logic

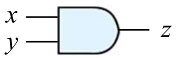
H = 0
L= 1

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

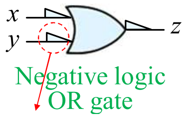
Truth table for negative logic



Gate block diagram



Positive logic AND gate



Negative logic OR gate

* Polarity indicator

- Conversion b/t positive and negative logic:
  - Change 1's to 0's and 0's to 1's in both inputs and output of a gate (truth table)

    ⇒ take the dual of a function

    (AND → OR, OR → AND, 0 → 1, 1 → 0)

*** Duality principle**

2-9

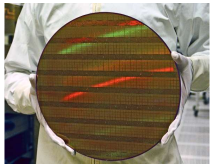
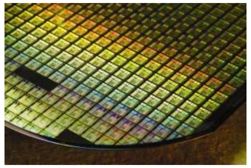
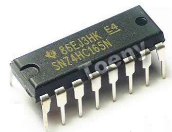
Integrated Circuits (ICs)

J.J. Shann

■89

Integrated Circuits (ICs)

- Digital ckts are constructed w/ integrated ckts.
- Integrated circuit (IC):
  - is a silicon semiconductor crystal (chip) containing the electronic components for the digital gates and storage elements.
  - The various components are interconnected on the chip.
  - The chip is mounted in a ceramic or plastic container, and connections are welded from the chip to the external pins.

J.J. Shann 2-90

■90

A. Levels of Integration

- Levels of integration: ckt complexity
  (# of logic gates in a single silicon chip)
  - SSI: small-scale integration, <10 gates
  - MSI: medium-scale integration, 10 ~ 100 (Ch 4~7)
    - performs specific elementary digital functions
    - e.g.: addition of 4 bits
  - LSI: large-scale integration: 100 ~ x000
    - eg.: small processors, small memories, programmable modules
  - VLSI: very large-scale integration, x000 ~
    - e.g.: complex microprocessors, digital signal processing chips

J.J. Shann 2-91

■91

B. Digital Logic Families

- Digital logic families: ckt technology
  - TTL: transistor-transistor logic
  - ECL: emitter-coupled logic (speed)
  - MOS: metal-oxide semiconductor (density)
  - CMOS: complementary MOS (power)
  - The basic ckt in each family is a **NAND**, **NOR**, or **inverter** gate. (Ch3)

J.J. Shann 2-92

■92

(Supplementary materials)

補充資料：Technology Parameters

- The most important parameters of digital logic families:
  - Fan-in
  - 一Fan-out
  - Noise margin
  - Propagation delay
  - Cost
  - Power dissipation

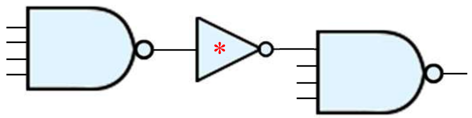
Reference:
M. Morris Mano & Charles R. Kime, *Logic and Computer Design Fundamentals*, 3rd Edition , 2004, Pearson Prentice Hall. (§3-2)

J.J. Shann 2-93

93

Fan-in

- ***Fan-in***: # of inputs available on a gate
  - For high-speed technologies, fan-in is often restricted on gate primitives to ≤ 4 or 5.
- E.g.: Implementation of a 7-input NAND gate using NAND gates w/ 4 or fewer inputs



* NAND is not associative!

J.J. Shann 2-96

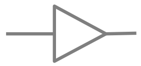
96

Fan-Out

- ***Fan-out***: # of standard loads driven by a gate output
  - *Max fan-out* : the fan-out that the output can drive w/o impairing gate performance
- For CMOS gates:
  - The load on the output of a gate determines the time required for the output of the gate to change from L to H and from H to L. → *transition time*

J.J. Shann 2-97

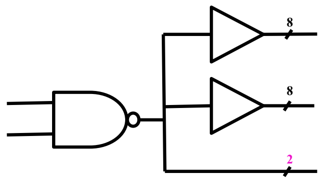
97



- Example:

An integrated circuit logic family has
NAND gates with a fan-out of 4 standard loads &
buffers with a fan-out of 8 standard loads.
Sketch a schematic showing how the output signal of a single NAND gate can be applied to 18 other gate inputs. Assume that each input is one standard load.

<Ans.>



J.J. Shann 2-98

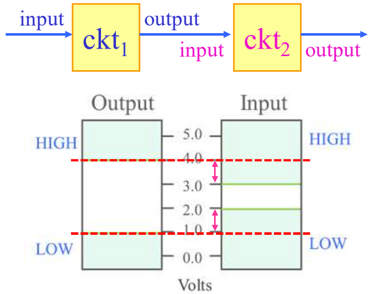
98

- Fan-in & Fan-out:
  - must be dealt w/ in the technology mapping step of the design process.
  - Gate w/ fan-ins larger than available ⇒ Multiple gates
  - Gate w/ fan-outs either exceed its max allowable fan-out or have too high a delay ⇒ Multiple gates or added buffers at its output

99

Noise Margin

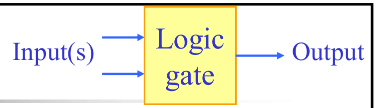
- ***Noise margin***: the max external noise voltage superimposed on a normal input value that will not cause an undesirable change in the ckt output



(p.1-8)

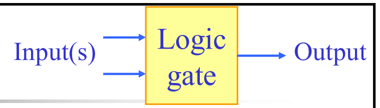
100

Propagation Delay

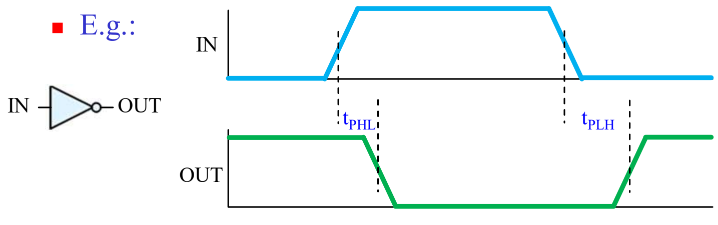


- ***Propagation delay***: the time required for a change in value of a signal to propagate from input to output
  - The operating speed of a ckt is inversely related to the longest propagation delays through the gates of the ckt.
- 3 propagation delay parameters:
  - high-to-low propagation time t_{PHL}
  - low-to-high propagation time t_{PLH}
  - propagation delay t_{pd} : $\max(t_{PHL}, t_{PLH})$

101



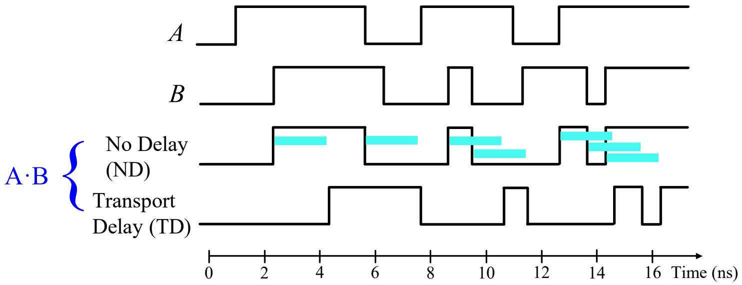
- 3 propagation delay parameters:
  - high-to-low propagation time t_{PHL}
  - low-to-high propagation time t_{PLH}
  - propagation delay t_{pd} : $\max(t_{PHL}, t_{PLH})$
- E.g.:



$t_{pd} = \max(t_{PHL}, t_{PLH})$

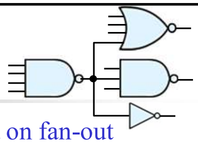
102

- E.g.: behavior of propagation delay of an AND gate



: propagation delay = 2 ns

- For CMOS gates:
  - The load on the output of a gate determines the time required for the output of the gate to change from L to H and from H to L. → *transition time*
- E.g.: (next page)



- E.g.: Calculation of gate delay based on fan-out

A 4-input NAND gate output is attached to the inputs of the following gates w/ the given # of standard loads representing their inputs:

4-input NOR gate-- 0.80 standard load

3-input NAND gate-- 1.00 standard load

Inverter-- 1.00 standard load

The formula for the delay of the 4-input NAND gate is

$$t_{pd} = 0.07 + 0.021 \times SL\ ns$$

SL: the sum of the standard loads driven by the gate

<Ans.> Ignoring the wiring delay

$$t_{pd} = 0.07 + 0.021 \times (0.80 + 1.00 + 1.00) = 0.129\ ns$$

Cost

- ***Cost*** for a gate:
  - is usually based on the area occupied by the layout cell (∝ the size of the transistors & the wiring in the gate layout)

Power dissipation

- ***Power dissipation***: the power drawn from the power supply and consumed by the gate
  - must be considered in relation to the operating temperature and cooling

J.J. Shann 2-107

107

C. Computer-Aided Design (CAD) of VLSI Circuits

- CAD tools:
  - software programs that support computer-based representation and aid in the development of digital hardware by automating the design process.
  - Schematic capture tools:
    - support the drawing of blocks and interconnections at all levels of the hierarchy.
    - At the level of primitives and functional blocks, libraries of graphics symbols are provided.
  - Logic simulator:
    - verify the behavior and the timing of the hierarchical blocks and the entire ckt
  - Logic synthesizer:
    - optimize design being generated automatically from HDL (hardware description language) specifications in physical area or delay

J.J. Shann 2-108

108

* Chapter Summary

- Primitive logic ops: AND, OR, NOT
- Boolean algebra
- Canonical form: minterm & maxterm standard forms
- Standard forms: SoP and PoS $\Rightarrow$ 2-level gate ckts
- Other logic gates:
  - NAND, NOR
  - XOR, XNOR
- Integrated circuits
  - Technology parameters

J.J. Shann 2-109

109