Chapter 2.
Combinational Logic Circuits

Mar., 2008

INDEX

2

Goal

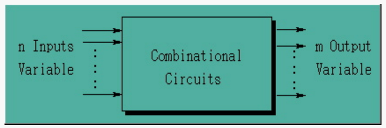
- 기본적인 논리요소인 게이트에 대한 이해
- 회로 설계를 위한 수학적 기법과 회로를 효율적으로 설계하는 방법을 학습
- 최적화 방법과 카노맵에 대한 이해
- 논리 게이트 특성 이해
- exclusive OR와 exclusive NOR 게이트 및 대수적 기법 소개

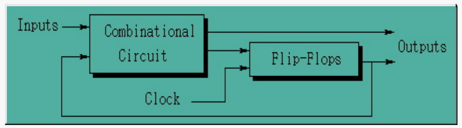
Overview

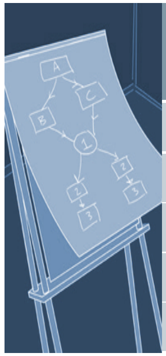
- Part 1 – Gate Circuits and Boolean Equations
  - Binary Logic and Gates
  - Boolean Algebra
  - Standard Forms
- Part 2 – Circuit Optimization
  - Two-Level Optimization
  - Map Manipulation
  - Practical Optimization (Espresso)
  - Multi-Level Circuit Optimization
- Part 3 – Additional Gates and Circuits
  - Other Gate Types
  - Exclusive-OR Operator and Gates
  - High-Impedance Outputs

Combinational Logic Circuit

- In digital circuit theory, *Combinational Logic* is a type of logic circuit whose output is a pure function of the present input only. This is in contrast to *Sequential Logic*, in which the output depends not only on the present input but also on the history of the input.
- In other words, Sequential Logic has *memory* while Combinational Logic does not.
- Combinational Logic is used in computer circuits to do Boolean algebra on input signals and on stored data. ***Practical computer circuits normally contain a mixture of combinational and sequential logic***.
  - For example, the part of an arithmetic logic unit, or ALU, that does mathematical calculations is constructed in accord with combinational logic, although the ALU is controlled by a sequencer that is constructed in accord with sequential logic.







1. 2진 논리와 게이트

2. Boolean Algebra

3. Standard Forms

4.

5.

6.

2.1 Binary Logic and Gate

- Definition
  - ***Binary Logic*** is processing based on the binary numbering system
  - *Binary variables* take on one of two values
  - *Logical operators* operate on binary values and binary variables
  - Basic logical operators are the *logic functions* AND, OR and NOT
  - ***Logic gates*** implement logic functions
  - *Boolean Algebra* is a useful mathematical system for specifying and transforming logic functions
  - We study Boolean algebra as a foundation for designing and analyzing digital systems!

Binary Variables

- Recall that the two *binary values* have different names:
  - True/False
  - On/Off
  - Yes/No
  - 1/0
- We use **1** and **0** to denote the two values.
- Variable identifier examples:
  - A, B, y, z, or X_1 for now
  - RESET, START_IT, or ADD1 later

Logical Operations

- The three basic logical operations are:
  - **AND** is denoted by a dot (·).
  - **OR** is denoted by a plus (+).
  - **NOT** is denoted by an overbar (‾), a single quote mark (') after, or (~) before the variable.

Notation Examples

- Examples:
  - Y = A × B is read "Y is equal to A AND B"
  - z = x + y is read "z is equal to x OR y"
  - X = $\overline{A}$ is read "X is equal to NOT A"
  - Note: The statement;

    1+1=2(10) read "one plus one equals two"

    is not the same as

    1+1=1 read "1 or 1 equals 1"

Operator Definitions

- Operations are defined on the values "0" and "1" for each operator;

AND	OR	NOT
0 · 0 = 0	0 + 0 = 0	$\overline{0}$ = 1
0 · 1 = 0	0 + 1 = 1	$\overline{1}$ = 0
1 · 0 = 0	1 + 0 = 1	
1 · 1 = 1	1 + 1 = 1	

Truth Tables

- *Truth table* – a tabular listing of the values of a function for all possible combinations of values on its arguments
- Example: Truth tables for the basic logic operations:

AND		
X	Y	Z=X · Y
0	0	0
0	1	0
1	0	0
1	1	1

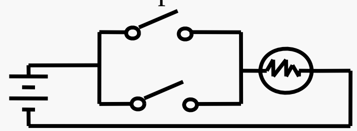
AND		
X	Y	Z=X + Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT	
X	Z=$\overline{X}$
0	1
1	0

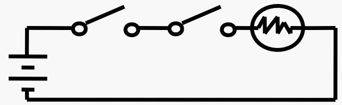
Logic Function Implementation

- Using Switches
  - For inputs:
    - logic 1 is switch closed
    - logic 0 is switch open
  - For outputs:
    - logic 1 is light on
    - logic 0 is light off
  - NOT uses a switch such that:
    - logic 1 is switch open
    - logic 0 is switch closed

Switches in parallel → OR

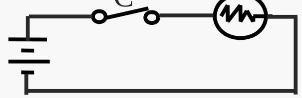


Switches in series → AND



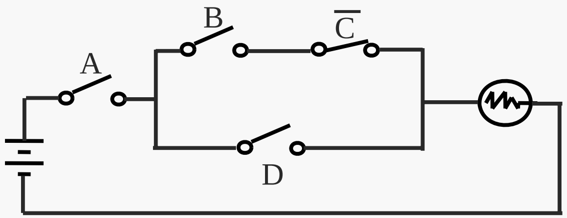
Normally-closed switch → NOT

$\overline{C}$



Logic Function Implementation (Continued)

- Example: Logic Using Switches



- Light is on (L = 1) for

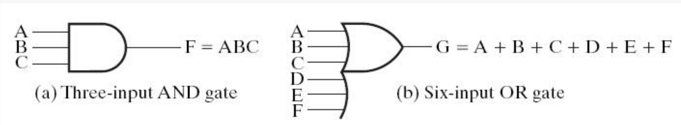
  $L(A, B, C, D) = A \cdot ((B \cdot \bar{C}) + D) = AB\bar{C} + AD$

  and off (L = 0), otherwise.
- Useful model for relay circuits and for CMOS gate circuits, the foundation of current digital logic technology

Logic Gates

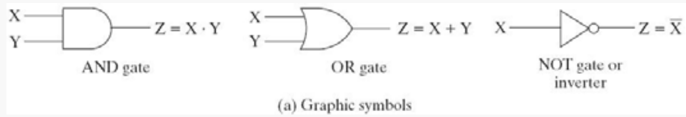
- In the earliest computers, switches were opened and closed by magnetic fields produced by energizing coils in *relays*. The switches in turn opened and closed the current paths.
- Later, *vacuum tubes* that open and close current paths lectronically replaced relays.
- Today, *transistors* are used as electronic switches that open and close current paths
- A *logic gate* performs a logical operation on one or more logic inputs and produces a single logic output. Because the output is also a logic-level value, an output of one logic gate can connect to the input of one or more other logic gates. The logic normally performed is Boolean logic and is most commonly found in digital circuits. Logic gates are primarily implemented electronically using diodes or transistors
- Optional: Chapter 6 – Part 1: The Design Space

- Example of Logic gate

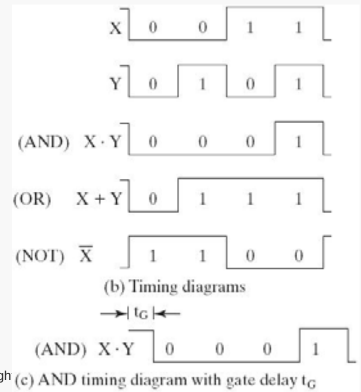


Logic Gate Symbols and Behavior

- Logic gates have special symbols:

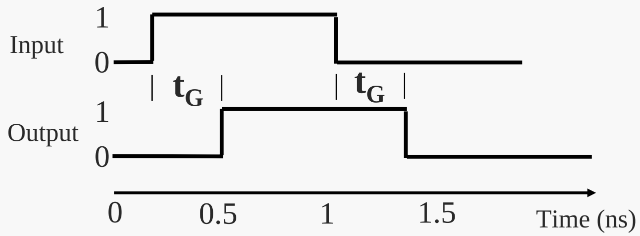


- And waveform behavior in time as follows:



Gate Delay

- In actual physical gates, if one or more input changes causes the output to change, the output change does not occur instantaneously.
- The delay between an input change(s) and the resulting output change is the *gate delay* denoted by t_G:



Logic Diagrams and Expressions

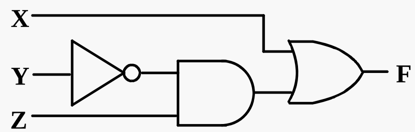
Truth Table

X Y Z	$F = X + \bar{Y} \times Z$
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

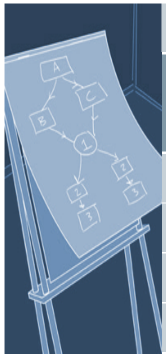
Equation

$$F = X + \bar{Y} Z$$

Logic Diagram



- *Boolean equations, truth tables and logic diagrams describe the same function*!
- Truth tables are unique; expressions and logic diagrams are not. This gives flexibility in implementing functions.



2.2 Boolean Algebra

- *Boolean Algebra* is an algebra dealing with binary variables and logic operations
  - Variables are designated by letters of the alphabet
  - 3 basic logic operations are AND, OR and NOT (complementaiton)
- *Boolean Expression* is an algebraic expression formed by using binary variable, constants 0 and 1, the logic operation symbols and parenthese
- *Boolean Function* can be described by a Boolean equation consisting of a binary variable identifying the function followed by an equals sign and a Boolean expression (함수를 나타내는 2진 출력변수, 그 다음에 등호, 등호 다음에는 2진 입력 변수 0,1을 사용하여 형태를 이루는 대수적 표현으로 구성된다 ?)
  - Boolean function is a function of the form $f : B^k \rightarrow B$, where $B = \{0, 1\}$ is a *boolean domain* and k is a nonnegative integer

2.2 Boolean Algebra

- An algebraic structure defined on a set of at least two elements, B, together with three binary operators (denoted +, · and ‾)
- Most basic identities of Boolean algebra

TABLE 2-3
Basic Identities of Boolean Algebra

1.	$X + 0 = X$	2.	$X \cdot 1 = X$	
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	
9.	$\overline{\overline{X}} = X$			
10.	$X + Y = Y + X$	11.	$XY = YX$	Commutative
12.	$X + (Y + Z) = (X + Y) + Z$	13.	$X(YZ) = (XY)Z$	Associative
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributive
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan's

Some Properties of Identities & the Algebra

- If the meaning is unambiguous, we leave out the symbol "·"
- The identities above are organized into pairs. These pairs have names as follows:

1-4 Existence of 0 and 1	5-6 Idempotence
7-8 Existence of complement	9 Involution
10-11 Commutative Laws	12-13 Associative Laws
14-15 Distributive Laws	16-17 DeMorgan's Laws

- The ***dual*** *of an algebraic expression* is obtained by interchanging + and · and interchanging 0's and 1's.
- The identities appear in dual pairs. When there is only one identity on a line the identity is *self-dual*, i. e., the dual expression = the original expression.
- DeMorgan's theorem can be illustrated by means of truth tables that assign all the possible binary values to X and Y (see Page 61, Table2-4).
- DeMorgan's theorem can be extended to three or more variables (see Page 62).

Some Properties of Identities & the Algebra (Continued)

- Unless it happens to be self-dual, the dual of an expression does not equal the expression itself.

- Example: F = $(A + \overline{C}) \cdot B + 0$
  dual F = $(A \cdot \overline{C}) + B \cdot 1 = A \cdot \overline{C} + B$
- Example: G = $X \cdot Y + \overline{(W + Z)}$
  dual G =
- Example: H = $A \cdot B + A \cdot C + B \cdot C$
  dual H =

- Are any of these functions self-dual?

Boolean Operator Precedence

- The order of evaluation in a Boolean expression is:
  1. Parentheses
  2. NOT
  3. AND
  4. OR
- Consequence: Parentheses appear around OR expressions
- Example: F = $A(B + C)(C + \overline{D})$

- $A + A \cdot B = A$ (Absorption Theorem)

Proof)

Steps	Justification (identity or theorem)
$A + A \cdot B$	
$= A \cdot 1 + A \cdot B$	$X = X \cdot 1$
$= A \cdot (1 + B)$	$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$ (Distributive Law)
$= A \cdot 1$	$1 + X = 1$
$= A$	$X \cdot 1 = X$

- Our primary reason for doing proofs is to learn:
  - Careful and efficient use of the identities and theorems of Boolean algebra, and
  - How to choose the appropriate identity or theorem to apply to make forward progress, irrespective of the application.

- $AB + \overline{A}C + BC = AB + \overline{A}C$ (Consensus Theorem)

Proof)

Steps	Justification (identity or theorem)
$AB + \overline{A}C + BC$	
$= AB + \overline{A}C + 1 \cdot BC$	?
$= AB + \overline{A}C + (A + \overline{A}) \cdot BC$	?
=	

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

Proof

Steps	Justification (identity or theorem)
$\mathbf{\overline{(X+Y)}\, Z + X\, \overline{Y}}$	

=

Useful Theorems

- $x \times y + \overline{x} \times y = y$ $\quad (x+y)(\overline{x}+y) = y$ $\quad$ Minimization
- $x + x \cdot y = x$ $\quad x \cdot (x+y) = x$ $\quad$ Absorption
- $x + \overline{x} \times y = x + y$ $\quad x \times (\overline{x} + y) = x \times y$ Simplification
- $x \times y + \overline{x} \cdot z + y \cdot z = x \cdot y + \overline{x} \cdot z$ $\quad$ Consensus

  $(x+y) \cdot (\overline{x}+z) \cdot (y+z) = (x+y) \cdot (\overline{x}+z)$
- $\overline{x+y} = \overline{x} \cdot \overline{y}$ $\quad \overline{x \cdot y} = \overline{x} + \overline{y}$ $\quad$ DeMorgan's Law

Proof of Simplification

- $x \times y + \bar{x} \times y = y$ $\quad (x+y)(\bar{x}+y) = y$

Proof of DeMorgan's Laws

- $\overline{x+y} = \bar{x} \cdot \bar{y}$ $\qquad \overline{x \cdot y} = \bar{x} + \bar{y}$

Boolean Function Evaluation

$$F1 = xy\bar{z}$$
$$F2 = x + \bar{y}z$$
$$F3 = \bar{x}\bar{y}\bar{z} + \bar{x}\,y\,z + x\bar{y}$$
$$F4 = x\bar{y} + \bar{x}\,z$$

x	y	z	F1	F2	F3	F4
0	0	0	0	0		
0	0	1	0	1		
0	1	0	0	0		
0	1	1	0	0		
1	0	0	0	1		
1	0	1	0	1		
1	1	0	1	1		
1	1	1	0	1		

Expression Simplification

- An application of Boolean algebra
- *Simplify to contain the smallest number of literals* (complemented and uncomplemented variables):

$$\mathbf{A\,B + \bar{A}\,C\,D + \bar{A}\,B\,D + \bar{A}\,C\,\bar{D} + A\,B\,C\,D}$$

$= AB + ABCD + \bar{A}\,C\,D + \bar{A}\,C\,\bar{D} + \bar{A}\,B\,D$

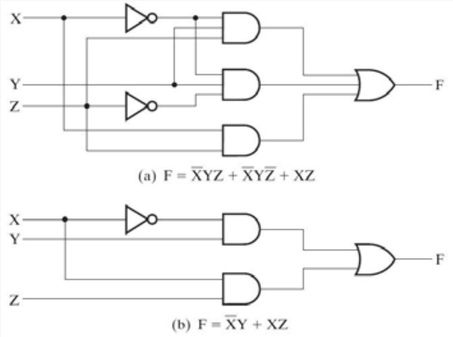
$= AB + AB(CD) + \bar{A}\,C\,(D + \bar{D}) + \bar{A}\,B\,D$

$= AB + \bar{A}\,C + \bar{A}\,B\,D$

$= B(A + \bar{A}D) + \bar{A}C$

$= B\,(A + D) + \bar{A}\,C$ $\quad$: 5 literals

- Simpler expression reduces both the number of gates in the circuit and the numbers of inputs to the gates.
  - Boolean algebra is a useful tool for simplifying digital circuits
- Two circuits implement the same function, but the one with fewer gates and/or fewer gate inputs is preferable because it requires fewer components.



(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$

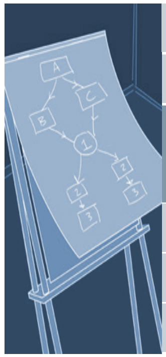
(b) $F = \overline{X}Y + XZ$

3 terms (항)
8 literals (문자)

2 terms
4 literals

Complementing Functions

- Use DeMorgan's Theorem to complement a function:
  1. Interchange AND and OR operators
  2. Complement each constant value and literal
- Example: Complement $F = \overline{x}y\overline{z} + x\overline{y}\overline{z}$
  $\overline{F} = (x + \overline{y} + z)(\overline{x} + y + z)$
- Example: Complement $G = (\overline{a} + bc)\overline{d} + e$
  $\overline{G} =$



1. Binary Logic and Gate
2. Boolean Algebra

3. 표준 형태

4.
5.
6.

Overview – Canonical Forms

- What are Canonical (규범적인, 표준적인) Forms?
- Minterms and Maxterms
- Index Representation of Minterms and Maxterms
- Sum-of-Minterm (SOM) Representations
- Product-of-Maxterm (POM) Representations
- Representation of Complements of Functions
- Conversions between Representations

Canonical Forms

- It is useful to specify Boolean functions in a form that:
  - Allows comparison for equality.
  - Has a correspondence to the truth tables
- Canonical Forms in common usage:
  - Sum of Minterms (SOM)
  - Product of Maxterms (POM)

  - ***Canonical form*** (정규형)은 함수의 항이 최소항의 합(sum of minterm)이나 최대항의 곱(product of maxterm)으로 표현되는 식
  - ***Standard form*** (표준형) 은 함수의 각 항이 곱의합(sum of product, SOP)이나 합의 곱(product of sum, POS) 형태로 표현되는 식

- In Boolean algebra, any Boolean function can be expressed in a *canonical form* using the dual concepts of *minterms* and *maxterms*. All logical functions are expressible in canonical form, both as a "sum of minterms" and as a "product of maxterms". This allows for greater analysis into the simplification of these functions, which is of great importance in the minimization of digital circuits.
- Generally, in mathematics, a ***canonical form*** (often called normal form or ***standard form***) of an object is a standard way of presenting that object.

- A Boolean function expressed as a disjunction (OR) of minterms is commonly known as a "*sum of products*" or "SoP". Thus it is a disjunctive normal form in which only minterms are allowed as summands. Its De Morgan dual is a "*product of sums*" or "PoS" , which is a function expressed as a conjunction (AND) of maxterms.

Standard Forms

Page. 66

- Standard Forms
  - facilitate the simplification procedures for Boolean expressions
  - in some cases, may result in more desirable expressions for implementing logic circuits

    (표준형태는 부울표현식에 대한 단순화의 절차를 쉽게 하고, 경우에 따라 논리회로 구현을 위한 바람직한 표현식을 만들어 낼수도 있다)

  - contains product terms and sun terms
    - Product term; $XY\overline{Z}$
    - Sum term $X+\overline{Y}+Z$

    - In Boolean algebra, the words "product" and "sum" do not imply arithmetic operations; instead, they specify the logical operations AND and OR, respectively

Minterm

- *Minterm* is AND terms with every variable present in either true or complemented form.

  (모든 변수가 보수나 보수가 아닌 상태로 정확히 한번 나타나는 논리곱항)
- Given that each binary variable may appear normal (e.g., x) or complemented (e.g., $\overline{x}$), there are 2^n minterms for n variables.
- Example: Two variables (X and Y) produce 2 x 2 = 4 combinations: (2개의 변수 X, Y에 대한 4개의 최소항)

  XY **(both normal)**

  $\mathbf{X\overline{Y}}$ **(X normal, Y complemented)**

  $\mathbf{\overline{X}Y}$ **(X complemented, Y normal)**

  $\mathbf{\overline{X}\,\overline{Y}}$ **(both complemented)**

- Thus, there are four minterms of two variables.
- Example, 8 minterms for 3 variables (Page 67, Table2-6).

Maxterm

- *Maxterm* is OR terms with every variable in true or complemented form.
  (보수나 보수가 아닌 상태의 모든 변수를 포함하는 논리합항)
- Given that each binary variable may appear normal (e.g., x) or complemented (e.g., $\overline{x}$), there are 2^n maxterms for n variables.
- Example: Two variables (X and Y) produce
  2 x 2 = 4 combinations:

  $X + Y$ (both normal)

  $X + \overline{Y}$ (x normal, y complemented)

  $\overline{X} + Y$ (x complemented, y normal)

  $\overline{X} + \overline{Y}$ (both complemented)

- Example: 8 maxterms for 3 variables (Page 68, Table 2-7)

Maxterms and Minterms

- Examples: Two variable minterms and maxterms.

M_{Index}	Minterm	Maxterm
m_0 (00)	$\overline{x}\,\overline{y}$	$x + y$
m_1 (01)	$\overline{x}\,y$	$x + \overline{y}$
m_2 (10)	$x\,\overline{y}$	$\overline{x} + y$
m_3 (11)	$x\,y$	$\overline{x} + \overline{y}$

- The index above is important for describing which variables in the terms are true and which are complemented.

Purpose of the Index

- The index for the minterm or maxterm, expressed as a binary number, is used to determine whether the variable is shown in the true form or complemented form.
- For Minterms:
  - "1" means the variable is "Not Complemented" and
  - "0" means the variable is "Complemented"
- For Maxterms:
  - "0" means the variable is "Not Complemented" and
  - "1" means the variable is "Complemented"

- Minterms and Maxterms for 3 variables (Page, 67 and 68)

TABLE 2-6
Minterms for Three Variables

X	Y	Z	Product Term	Symbol	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	$\overline{X}\overline{Y}\overline{Z}$	m_0	1	0	0	0	0	0	0	0
0	0	1	$\overline{X}\overline{Y}Z$	m_1	0	1	0	0	0	0	0	0
0	1	0	$\overline{X}Y\overline{Z}$	m_2	0	0	1	0	0	0	0	0
0	1	1	$\overline{X}YZ$	m_3	0	0	0	1	0	0	0	0
1	0	0	$X\overline{Y}\overline{Z}$	m_4	0	0	0	0	1	0	0	0
1	0	1	$X\overline{Y}Z$	m_5	0	0	0	0	0	1	0	0
1	1	0	$XY\overline{Z}$	m_6	0	0	0	0	0	0	1	0
1	1	1	XYZ	m_7	0	0	0	0	0	0	0	1

TABLE 2-7
Maxterms for Three Variables

X	Y	Z	Sum Term	Symbol	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	$X+Y+\overline{Z}$	M_1	1	0	1	1	1	1	1	1
0	1	0	$X+\overline{Y}+Z$	M_2	1	1	0	1	1	1	1	1
0	1	1	$X+\overline{Y}+\overline{Z}$	M_3	1	1	1	0	1	1	1	1
1	0	0	$\overline{X}+Y+Z$	M_4	1	1	1	1	0	1	1	1
1	0	1	$\overline{X}+Y+\overline{Z}$	M_5	1	1	1	1	1	0	1	1
1	1	0	$\overline{X}+\overline{Y}+Z$	M_6	1	1	1	1	1	1	0	1
1	1	1	$\overline{X}+\overline{Y}+\overline{Z}$	M_7	1	1	1	1	1	1	1	0

Minterm and Maxterm Relationship

- Review: DeMorgan's Theorem

  $\overline{x \cdot y} = \overline{x} + \overline{y}$ and $\overline{x + y} = \overline{x} \times \overline{y}$

- Two-variable example:

  $M_2 = \overline{x} + y$ and $m_2 = x \cdot \overline{y}$

  Thus, M_2 is the complement of m_2 and vice-versa.

- Since DeMorgan's Theorem holds for n variables, the above holds for terms of n variables
- giving:

  $M_i = \overline{m_i}$ and $m_i = \overline{M_i}$

  Thus, M_i is the complement of m_i.

- 최소항은 진리표에서 항상 0값이 아니면서 1의 개수가 최소인 함수이고, 최대항은 항상 1값이 아니면서 1의 개수가 최대인 함수이다.

  (같은 첨자에서 최소항과 최대항은 보수 관계)

Canonical Sum of Minterms

- Any Boolean function can be expressed as a <u>*Sum of Minterms*</u>.
  - For the function table, the <u>minterms</u> used are the terms corresponding to the 1's
  - For expressions, <u>expand</u> all terms first to explicitly list all minterms. Do this by "ANDing" any term missing a variable v with a term ($v + \overline{v}$).
  - 부울함수는 함수에서 1이되는 모든 최소항의 논리합을 형성하여 주어진 진리표에서 수학적으로 표현이 가능 → 최소항의 합

- Example: Implement $f = x + \overline{x}\ \overline{y}$ as a sum of minterms.

  First expand terms: $f = x(y + \overline{y}) + \overline{x}\ \overline{y}$

  Then distribute terms: $f = xy + x\overline{y} + \overline{x}\ \overline{y}$

  Express as sum of minterms: $f = m_3 + m_2 + m_0$

- In Table 2-8(a),

  the Boolean function F is equal to 1 for 000, 010, 101 and 111.

  These combinations correspond to minterms 0, 2, 5 and 7

  → By examining Table 2-8 and the truth table for these minterms in Table 2-6,

  The function F can be expressed algebraically as the logical sum of the stated minterms

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

  $= m_0 + m_2 + m_5 + m_7$

  $F(X,Y,Z) = \sum m(0,2,5,7)$

  where, $\sum$ stands for the logical sum (Boolean OR) of the minterms

TABLE 2-8
Boolean Functions of Three Variables

(a) X	Y	Z	F	$\overline{F}$
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

- Consider the complement of a Boolean function F,

  Binary values of $\overline{F}$ in Table 2-8(a) are obtained by chnging 1s to 0s, and 0s and 1s in the values of F

  $\overline{F} = \overline{X}\,\overline{Y}Z + \overline{X}YZ + X\overline{Y}\,\overline{Z} + XY\overline{Z}$

  $= m_1 + m_3 + m_4 + m_6$

  $\overline{F}(X,Y,Z) = \sum m(1,3,4,6)$

  * Minterms numbers for $\overline{F}$ are the ones missing from the list of the minterm numbers of F.

  Taking the complement of $\overline{F}$ to obtin F; see Page 69.

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

  → This shows the procedure for expressing a Boolean function as a *product of maxterms*.

  $F(X,Y,Z) = \prod M(1,3,4,6)$,

  where $\prod$ denotes the logical product (Boolean AND) of the maxterms whose numbers are listed in parentheses

- A function that is NOT in the sum-of-minterms form can be converted to that form using truth table.

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

$E(X,Y,Z) = \sum m(0,1,2,4,5)$

$\overline{E}(X,Y,Z) = \sum m(3,6,7)$

TABLE 2-8
Boolean Functions of Three Variables

(b) X	Y	Z	E
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Canonical Product of Maxterms

- Any Boolean Function can be expressed as a Product of Maxterms (POM).
  - For the function table, the maxterms used are the terms corresponding to the 0's.
  - For an expression, expand all terms first to explicitly list all maxterms. Do this by first applying the second distributive law , "ORing" terms missing variable v with a term equal to $v \times \overline{v}$ and then applying the distributive law again.
- Example: Convert to product of maxterms:

$$f(x, y, z) = x + \overline{x}\,\overline{y}$$

Apply the distributive law:

$$x + \overline{x}\,\overline{y} = (x + \overline{x})(x + \overline{y}) = 1 \times (x + \overline{y}) = x + \overline{y}$$

Add missing variable z:

$$x + \overline{y} + z \times \overline{z} = (x + \overline{y} + z)(x + \overline{y} + \overline{z})$$

Express as POM: $f = M_2 \cdot M_3$

Function Complements

- The complement of a function expressed as a sum of minterms is constructed by selecting the minterms missing in the sum-of-minterms canonical forms.
- Alternatively, the complement of a function expressed by a Sum of Minterms form is simply the Product of Maxterms with the same indices.
- Example: Given $F(x,y,z) = \sum m(1,3,5,7)$

  $\overline{F}(x,y,z) = \sum m(0,2,4,6)$

  $F(x,y,z) = \prod M(1,3,5,7)$

Conversion Between Forms

- To convert between sum-of-minterms and product-of-maxterms form (or vice-versa) we follow these steps:
  - Find the function complement by swapping terms in the list with terms not in the list.
  - Change from products to sums, or vice versa.
- Example:Given F as before: $F(x,y,z) = \sum m(1,3,5,7)$
- Form the Complement: $\overline{F}(x,y,z) = \sum m(0,2,4,6)$
- Then use the other form with the same indices – this forms the complement again, giving the other form of the original function:

  $F(x,y,z) = \prod M(0,2,4,6)$

Standard Forms

- Standard Sum-of-Products (SOP) form: equations are written as an OR of AND terms
- Standard Product-of-Sums (POS) form: equations are written as an AND of OR terms
- Examples:
  - SOP: $\mathbf{A\ B\ C + \overline{A}\ \overline{B}\ C + B}$
  - POS: $\mathbf{(A + B) \cdot (A + \overline{B} + \overline{C}) \cdot C}$
- These "mixed" forms are neither SOP nor POS
  - $\mathbf{(A\ \ B + C)\ (A\ + C)}$
  - $\mathbf{A\ B\ \overline{C} + A\ C\ (A\ + B)}$

Standard Sum-of-Products (SOP)

- A sum of minterms form for *n* variables can be written down directly from a truth table.
  - Implementation of this form is a two-level network of gates such that:
  - The first level consists of *n*-input AND gates, and
  - The second level is a single OR gate (with fewer than 2^n inputs)
- This form often can be simplified so that the corresponding circuit is simpler.

- A Simplification Example:

  $F(A,B,C) = \sum m(1,4,5,6,7)$
- Writing the minterm expression:

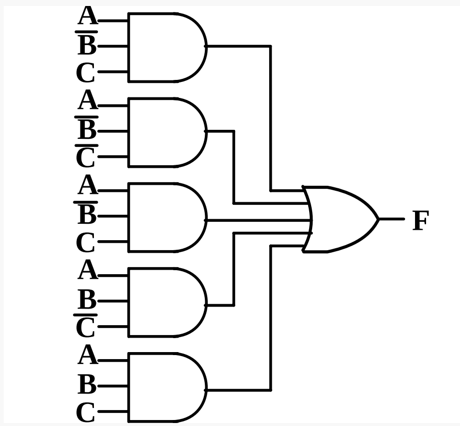
  $F = \overline{A}\ \overline{B}\ C + A\ \overline{B}\ \overline{C} + A\ \overline{B}\ C + AB\overline{C} + ABC$
- Simplifying:

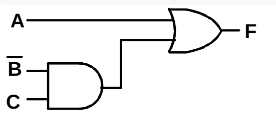
  F =

  $= \overline{B}C + A$
- Simplified F contains 3 literals compared to 15 in minterm F

AND/OR Two-level Implementation of SOP Expression

- The two implementations for F are shown below – it is quite apparent which is simpler!





SOP and POS Observations

- The previous examples show that:
  - Canonical Forms (Sum-of-minterms, Product-of-Maxterms), or other standard forms (SOP, POS) differ in complexity
  - Boolean algebra can be used to manipulate equations into simpler forms.
  - Simpler equations lead to simpler two-level implementations
- Questions:
  - How can we attain a "simplest" expression?
  - Is there only one minimum cost circuit?
  - The next part will deal with these issues.

Terms of Use

- All (or portions) of this material © 2008 by Pearson Education, Inc.
- Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.
- These materials or adaptations thereof are not to be sold or otherwise offered for consideration.
- This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.