EED2003 Digital Design

Presentation 2: Boolean Algebra

Asst. Prof.Dr. Ahmet ÖZKURT
Asst. Prof.Dr Hakkı T. YALAZAN

Based on the Lecture Notes by Jaeyoung Choi choi@comp.ssu.ac.kr Fall 2000

2.1 Binary Logic and Gates

- Digital circuits
  - hardware components that manipulate binary information
  - implemented using transistors and interconnections in IC
  - each basic circuit is called *logic gate*
    - performs a specific logical operation

- Boolean Algebra
  - mathematical notation to specify the operation of each gate
  - used to analyze and design circuits

2.1 Binary Logic and Gates

- Binary Logic
  - take on two discrete values &
    with the operations of mathematical logic
  - three logical operations

    1) AND Z = X • Y (Z is equal to X and Y)

    2) OR Z = X + Y (Z is equal to X or Y)

    3) NOT Z = X' (Z is equal to NOT X)

    -- complement operation (0 => 1 & 1 => 0)
  - AND/OR is similar to multiplication/addition

2.1 Binary Logic and Gates

- logical OR
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 1

logical AND
- 0 • 0 = 0
- 0 • 1 = 0
- 0 • 1 = 0
- 1 • 1 = 1

logical NOT
- 0' = 1
- 1' = 0

AND

X	Y	X · Y
0	0	0
0	1	0
1	0	0
1	1	1

OR

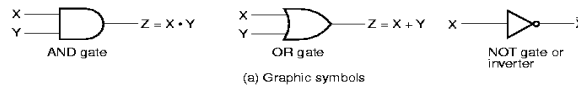
X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT

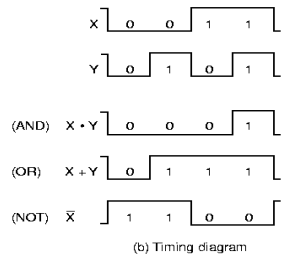
X	X'
0	1
1	0

2.1 Binary Logic and Gates

- Logic Gates
  - electronic circuits that operate on one or more input signals to produce an output signal
  - voltage operated circuits: logic 0 & logic 1
  - intermediate region is crossed during state *transition*



(a) Graphic symbols



(b) Timing diagram

2.2 Boolean Algebra

- deal with binary variables and logic operations
  - with three basic logic operations AND, OR, NOT
  - express logical relationship between binary variables

- Consider F = X + Y' Z
  - represented in a truth table
  - transformed from an algebraic expression into a circuit diagram composed of logic gates (Fig 2.3)

F = X + 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

2.2 Boolean Algebra

- Basic Identities of Boolean Algebra
  - most basic identities of Boolean algebra
  - *dual* - obtained by interchanging OR and AND, and replacing 1's by 0's and 0's by 1's

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

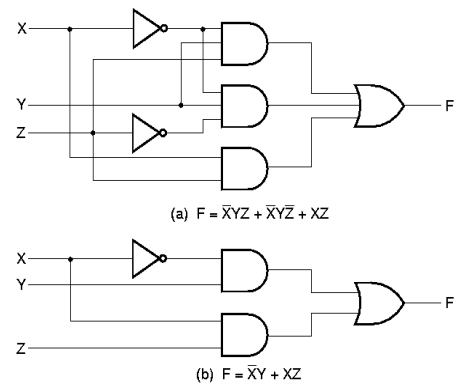
Table 2-3. Basic Identities of Boolean Algebra

2.2 Boolean Algebra

- AB + C + 1 = 1 (by identity 3)
- commutative laws -- the order doesn't affect the result
  X + Y = Y + X,
  X Y = Y X
- associative laws -- parentheses can be removed altogether
  X + (Y + Z) = (X + Y) + Z = X + Y + Z
  X (Y Z) = (X Y) Z = X Y Z
- distributive laws (dual)
  X + Y Z = (X + Y) (X + Z)
  (A + B) (A + CD) = ?
- DeMorgan's theorem -- obtain complement of an expression
  (X + Y)' = X' Y' (X Y)' = X' + Y'
  - can be extended to three or more variables
    (A + B + C + ...)' = A' B' C' ...

2.2 Boolean Algebra

- Algebraic Manipulation
  - Boolean algebra is a useful tool for simplifying digital circuits
  - F = X'YZ + X'YZ' + XZ
    = X'Y (Z+Z') + XZ
    = X'Y 1 + XZ
    = X'Y + XZ
  - compare two implementations in Fig 2.4



2.2 Boolean Algebra

- use truth table to verify two expressions (Table 2.5)
- manipulate Boolean algebra ==> obtain a simpler circuit
- popular tools
  1. X + XY = X (1 + Y) = X
  2. XY + XY' = X (Y + Y') = X
  3. X + X'Y = (X + X') (X + Y) = X + Y
  4. X (X + Y) = X + X Y = X (1 + Y) = X
  5. (X + Y)(X + Y') = X + YY' = X
  6. X (X' + Y) = XX' + XY = XY
- consensus theorem

  XY + X'Z + YZ = XY + X'Z (prove it!)

  dual (X+Y)(X'+Z)(Y+Z) = (X+Y)(X'+Z)

(Ex) (A+B)(A'+C) = AA' + AC + A'B + BC
= AC + A'B + BC = AC + A'B

2.2 Boolean Algebra

- Complement of a Function
  - obtained from an interchange of 1's to 0's and 0's to 1's
  - derived algebraically by applying DeMorgan's theorem

(Ex 2.1) Find the complement of F1 = X'YZ' + X'Y'Z

F1' = (X'YZ' + X'Y'Z)'
= (X'YZ')' (X'Y'Z)'
= (X + Y' + Z) (X + Y + Z')

(Ex 2.2) Find the complement of F1 = X'YZ' + X'Y'Z
by taking dual and complementing each literal

dual of F1	(X' + Y + Z') (X' + Y' + Z)
comp of each literal	(X + Y' + Z) (X + Y + Z')

2.3 Standard Forms

- facilitate the simplification procedures for Boolean expression
- contain product terms (XY'Z) and sum terms (X+Y+Z')

- Minterms & Maxterms
  - minterm (a product term) & maxterm (a sum term)
  - all the variables appear exactly once
  - show exactly one combination of the binary variables in a truth table
  - 2^n distinct terms for n variables

(Ex) 4 minterms for 2 variables X & Y

X'Y', X'Y, XY', & XY

2.3 Standard Forms

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

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

es

2.3 Standard Forms

- m_j (minterm) -- complemented if the bit is 0
  uncomplemented if the bit is 1
- M_j (maxterm) -- complemented if the bit is 1
  uncomplemented if the bit is 0
  - j denotes the binary number of the term
- minterm: having the minimum No of 1's in its truth table
  maxterm: having the maximum No of 1's in its truth table
- a minterm and maxterm with the same subscript are complements of each other ($M_j = m_j'$)

(Ex) $(m_3)' = (X' Y Z)' = X + Y' + Z' = M_3$

2.3 Standard Forms

- a Boolean function can be expressed by a *sum of minterms*

(Ex) Table 2-8(a)

F = X'Y'Z'+X'YZ'+XY'Z+XYZ = m0+m2+m5+m7

F(X,Y,Z) = Σ m(0,2,5,7) (Σ = logical sum, Boolean OR)

F' = X'Y'Z+X'YZ+XY'Z'+XYZ' = m1+m3+m4+m6

F(X,Y,Z)' = Σ m(1,3,4,6)

F = (m1+m3+m4+m6)' = m1' • m3' • m4' • m6'

= M1 • M3 • M4 • M6

= (X+Y+Z') • (X+Y'+Z') • (X'+Y+Z) • (X'+Y'+Z)

F(X,Y,Z) = Π M(1,3,4,6) (Π: logical product, Boolean AND)

- summary of minterms (p46)
- a function can be converted to the sum of minterms form by means of a truth table

2. 3 Standard Forms

X	Y	Z	F
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

(Ex) E = Y' + X'Z'

from the truth table,

E(X,Y,Z) = Σ m(0,1,2,4,5)

E(X,Y,Z)' = Σ m(3,6,7)

(the total number of minterms in E and E' is 8)

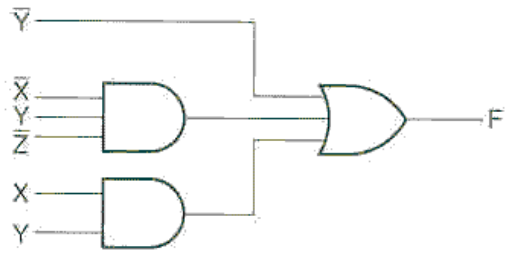
2.3 Standard Forms

- Sum of Products
  - a standard algebraic expression
  - obtained directly from a truth table (sum of minterms)
    & simplify the expression to *sum-of-products* form

(Ex) F = Y' + X'YZ' + XY

three product terms -- two AND gates, one OR gate

two-level implementation

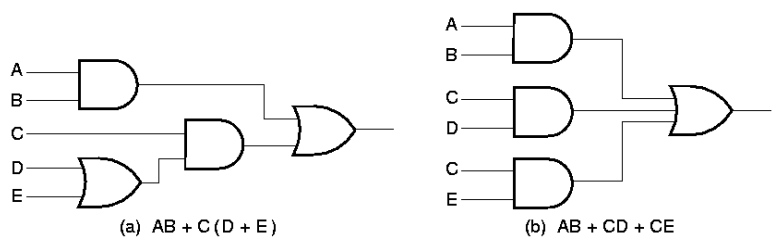


2.3 Standard Forms

(Ex) F = AB + C(D+E) (three-level)

=> AB + CD + DE (two-level)

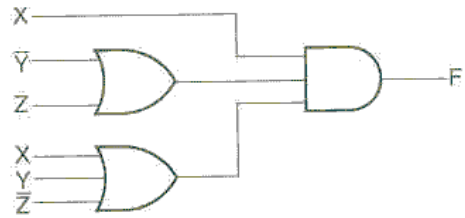
two-level implementation is preferred for its delay time



2.3 Standard Forms

- Product of Sums
  - another standard algebraic expression
  - obtained by forming a logical product of sum terms

(Ex) F = X(Y'+Z)(X+Y+Z')
needs 2 OR gates and one AND gates



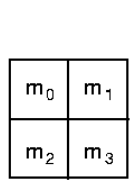
2.4 Map Simplification

- Karnough map (K-map)
  - to simplify Boolean functions of up to 4 variables

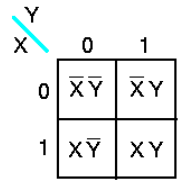
    (5 or 6 variables can be drawn, but cumbersome to use)
  - a diagram of squares, each representing one minterm
  - simplified expressions are in sum-of-products or product-of-sums
    - two-level implementations

2.4 Map Simplification

- Two-Variable Map
  - four minterms for a Boolean function with 2 variables

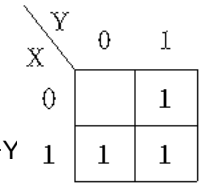


(a)



(b)

(Ex)



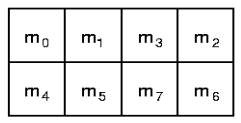
= X+Y

m1+m2+m3 = X'Y+XY'+XY = X+Y

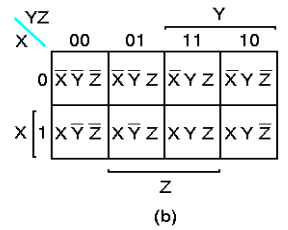
(by algebra) => X'Y+X(Y'+Y) = X'Y+Y

2.4 Map Simplification

- Three-variable Map
  - 8 minterms for 3 variables

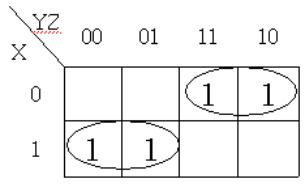


(a)



(b)

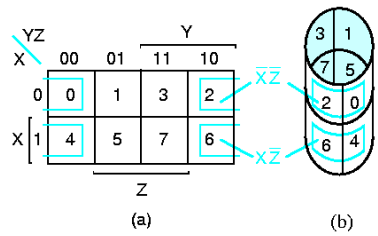
Ex2.3 F(X,Y,Z) = Σ m(2,3,4,5)



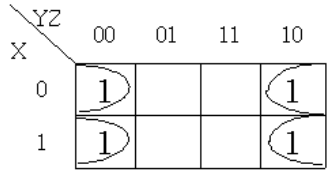
(from K-map) F = X'Y + XY'

2.4 Map Simplification

(Ex)



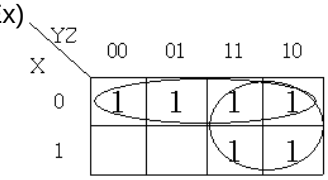
(Ex)



m0+m2+m4+m6 = Z'

= X' + Y

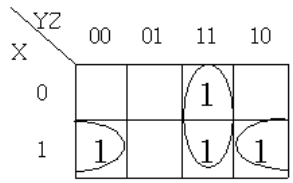
(Ex)



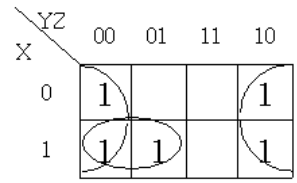
m0+m1+m2+m3+m6+m7

2.4 Map Simplification

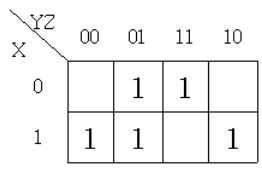
Ex2.4 F1(X,Y,Z) = Σ m(3,4,6,7)



F2(X,Y,Z) = Σ m(0,2,4,5,6)



(Ex)



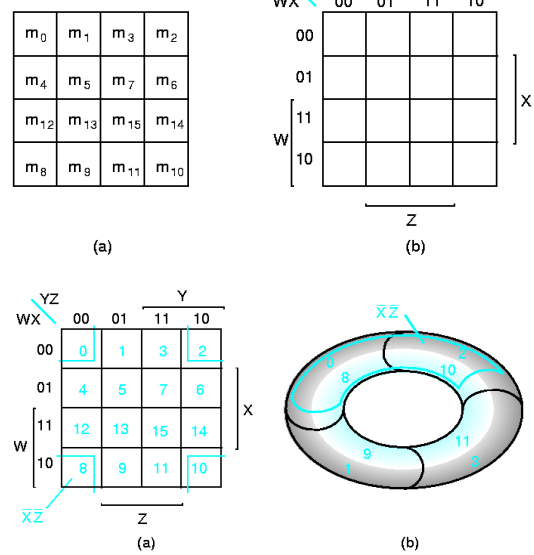
F2(X,Y,Z) = Σ m(1,3,4,5,6)

= X'Z + XZ' + XY'

or = X'Z + XZ' + Y'Z

2.4 Map Simplification

- Four-variable Map
  - 16 minterms for 4 variables



2.4 Map Simplification

Ex2.5 F(W,X,Y,Z)
= Σ m(0,1,2,4,5,6,8,9,12,13,14)

WX \ YZ	00	01	11	10
00	1	1		1
01	1	1		1
11	1	1		1
10	1	1		

F = Y' + W'Z' + XZ'

Ex2.6 F = A'B'C' + B'CD' + A'BCD' + AB'C'

AB \ CD	00	01	11	10
00	1	1		1
01				1
11				
10	1	1		1

F = B'D' + B'C' + A'CD'

2.5 Map Manipulation

- ensure that all minterms of the functions are included
- necessary to minimize the number of terms

- Essential Prime Implicants
  - *implicant*
    - if the function has the value 1 for all minterms of the product term
  - *prime implicant*
    - if the removal of any literal from an implicant results in a product term that is not an implicant
    - a product term obtained by combining the maximum possible number of adjacent squares in the map
  - *essential prime implicant*
    - if a minterm of a function is included in only one prime implicant

2.5 Map Manipulation

- To find the simplified expression from the map,
  1) first determine all prime implicant
  2) simplified expression
     - all the essential prime implicant + other prime implicant

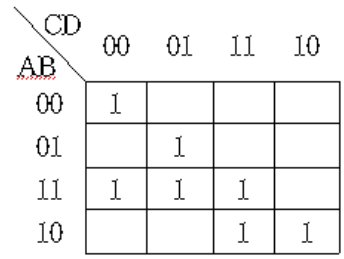
(Ex) Fig 2.21

AB \ CD	00	01	11	10
00		1	1	
01	1	1	1	1
11	1			1
10				

A'D and BD': essential prime implicants
A'B: not essential

2.5 Map Manipulation

(Ex) Fig 2.22



AB \ CD	00	01	11	10
00	1			
01		1		
11	1	1	1	
10			1	1

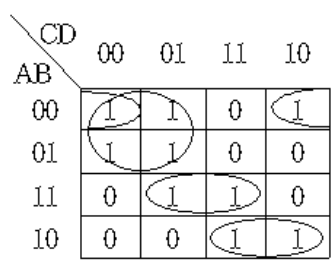
A'B'C'D', BC'D, ABC', AB'C: essential prime implicants
ACD or ABD: not essential

F = A'B'C'D' + BC'D + ABC' + AB'C + ACD or ABD

2.5 Map Manipulation

- Nonessential Prime Implicant
  - Selection Rule
    - minimize the overlap among prime implicant as much as possible

Ex2.7 F(A,B,C,D) = Σ m(0,1,2,4,5,10,11,13,15)



AB \ CD	00	01	11	10
00	1	1	0	1
01	1	1	0	0
11	0	1	1	0
10	0	0	1	1

F' = A'C' + ABD + AB'C + A'B'D'

2.5 Map Manipulation

- Product-of-Sums Simplification
  - from sum-of-products to product-of-sums
  - complement the function (taking a dual)

  1) Combine the squares marked with 0's

  2) change the function,

  which is expressed in product of sums to sum of products

Ex2.8 F(A,B,C,D) = Σ m(0,1,2,5,8,9,10)

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

F' = AB + CD + BD'

F = (A'+B')(C'+D')(B'+D)

2.5 Map Manipulation

Ex) F = (A'+B'+C) (B + D)

1) plot the map by taking its complement

F' = ABC' + B'D'

2) marking 0's in the squares to represent F'

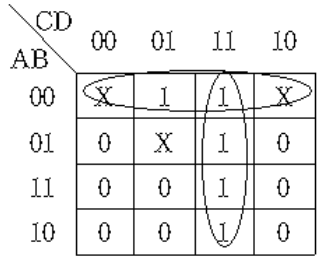
remaining squares are marked with 1's

3) combine the 0's and then complement the function

- Don't Care Conditions
  - unspecified minterms of a function
  - ex) 4-bit binary code for the decimal digits
  - marked with cross (X)
  - provide the further simplification of the function

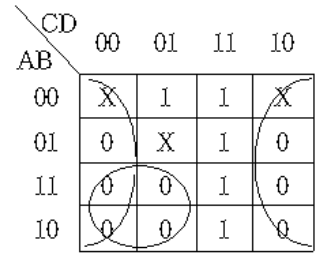
2.5 Map Manipulation

Ex) F(A,B,C,D) = Σ m(1,3,7,11,15)
d(A,B,C,D) = Σ m(0,2,5)



F = CD + A'B' = CD + A'D

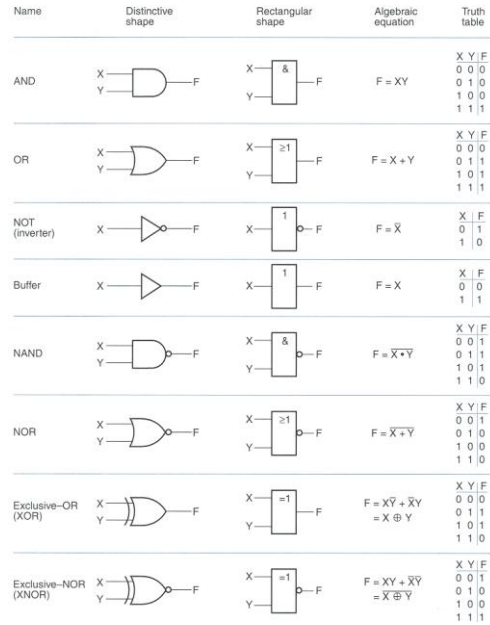
obtain a simplified product-



F' = Z' + WY'
F = Z(W' + Y)

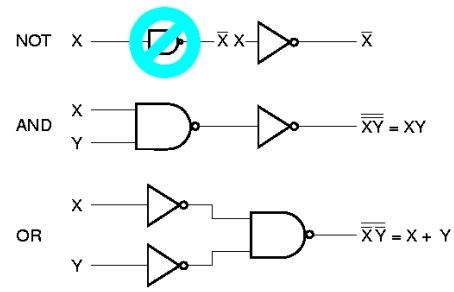
2.6 NAND and NOR Gates

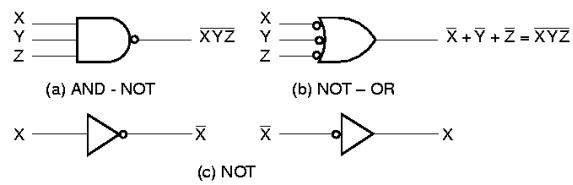
- Boolean functions are expressed in terms of AND, OR, NOT
- straight forward to implement the function with these gates
- Other useful logic gates



2.6 NAND and NOR Gates

- NAND gate
  - a universal gate
    - because any digital system can be implemented with it
  - Implementation of NOT (inverter), AND, OR
  - 2 graphic symbols: AND-invert & invert-OR

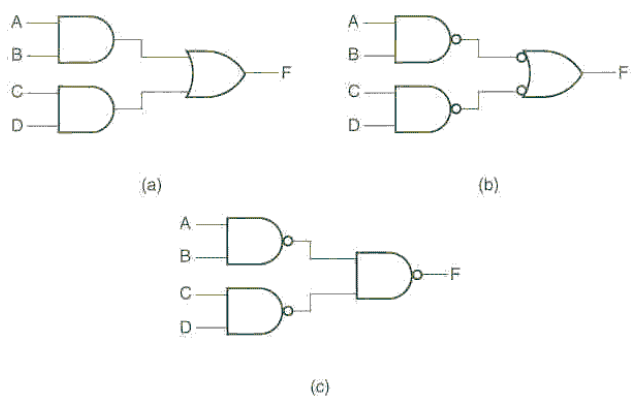




2.6 NAND and NOR Gates

- Two-Level Implementation
  - easy to implement with NAND gates, if the function is in sum of products form

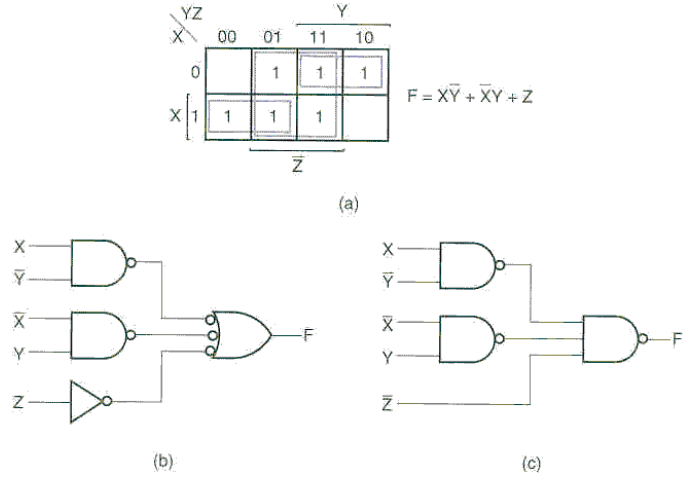
(Ex) F = AB + CD



F = ((AB)' (CD)')'
= AB + CD

2.6 NAND and NOR Gates

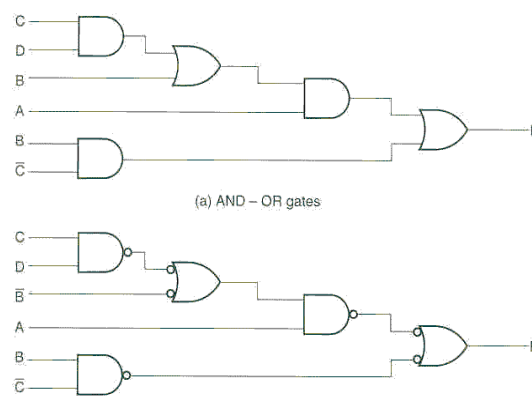
Ex2.9) F(X,Y,Z) = Σ m(1,2,3,4,5,7)



2.6 NAND and NOR Gates

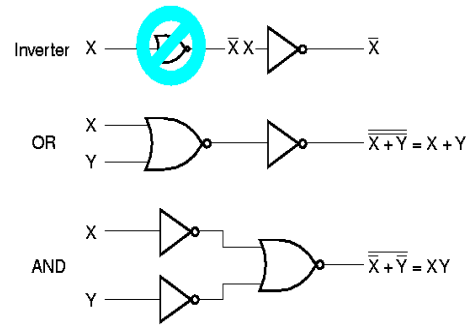
- Multilevel NAND Circuits
  - with three or more levels
    1) convert all AND gates to NAND gates w/ AND-invert
    2) convert all OR gates to NAND gates w/ invert-OR
    3) convert rest small circles to inverters

Ex) F = A (CD + B) + BC'

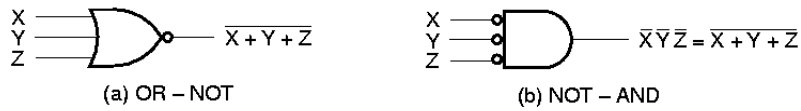


2.6 NAND and NOR Gates

- NOR gate
  - dual of the NAND operation
  - another universal gate
  - implementation of NOT (inverter), AND, OR



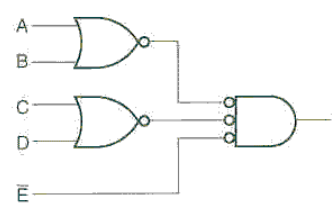
  - two graphic symbol for NOR gate



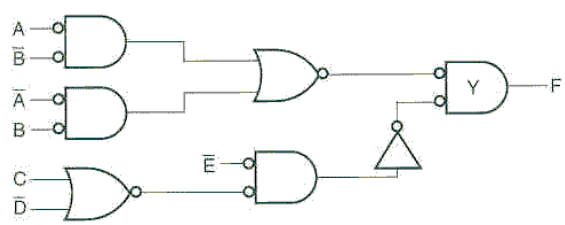
2.6 NAND and NOR Gates

- Two-Level Implementation
  - easy to implement with NOR gates, if the function is in product of sums form

(Ex) F = (A + B) (C + D) E



(Ex) F = (AB' + A'B) E (C + D')



2.7 Exclusive-OR Gate

- exclusive-OR (XOR) gate
  - $X \oplus Y = X Y' + X' Y$
  - 1 if only one variable is equal to 1, but not both
- exclusive-NOR gate
  - $(X \oplus Y)' = X Y + X' Y'$
  - 1 if both are equal to 1 or both are equal to 0
- they are to be the complement of each other

2.7 Exclusive-OR Gate

- properties

  $X \oplus 0 = X$ $\quad$ $X \oplus 1 = X'$

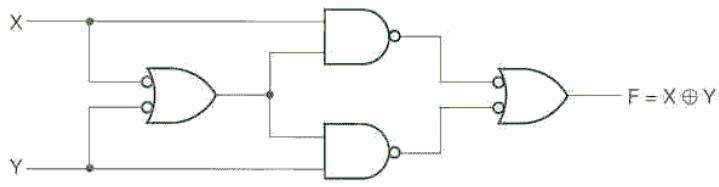
  $X \oplus X = 0$ $\quad$ $X \oplus X' = 1$

  $X \oplus Y' = (X \oplus Y)'$ $\quad$ $X' \oplus Y = (X \oplus Y)'$

  $A \oplus B = B \oplus A$

  $(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$

- implementation with NAND gates



2.8 Integrated Circuits

- Integrated Circuits (IC)
  - small silicon semiconductor crystal, called a chip
  - contains electronic components for the digital gates

- Levels of Integration
  - SSI (small scale integration), $\leq$ 10 gates
  - MSI, 10 ~ 100 gates
  - LSI, 100 ~ 1000s
  - VLSI, > 1000s

2.8 Integrated Circuits

- Digital Logic Families
  - RTL, DTL - earliest logic families
  - TTL - widespread, considered as standard
  - ECL - high speed operation
  - MOS - high component density
  - CMOS - low power consumption
  - BiCMOS - CMOS + TTL, used selectively

- Positive and Negative Logic
  - Normal Convention:
    - Positive Logic/Active High

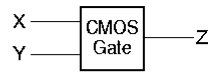
      Low Voltage = 0; High Voltage = 1
  - Alternative Convention sometimes used:
    - Negative Logic/Active Low

      Low Voltage = 1; High Voltage = 0

2.8 Integrated Circuits

X	Y	Z
L	L	L
L	H	L
H	L	L
H	H	H

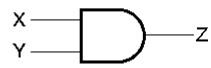
(a) Truth table with H and L



(b) Gate block diagram

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

(c) Truth table for positive logic



(d) Positive-logic AND gate

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

(e) Truth table for negative logic

(f) Negative-logic OR gate

(f) polarity indicator: small triangles in I/O