Logic and Computer Design Fundamentals

Chapter 2 – Combinational Logic Circuits

Part I – Gate Circuits and Boolean Equations

Yueming Wang (王跃明)

ymingwang@zju.edu.cn

2018

College of Computer Science, Zhejiang University

Qiushi Academy for Advanced Studies, Zhejiang University

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

Binary Logic and Gates

- **Binary Logic**
  - **Binary variables**
    - take on one of two values, 1 and 0
      - A, B, y, z, or X_1 for now, RESET, START_IT, or ADD1 later
  - **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.
  - Basic circuits
- **Boolean Algebra**
  - a useful mathematical system for specifying and transforming logic functions.
- **We study Boolean algebra as a foundation for designing and analyzing digital systems!**

Logical Operations

- **The three basic logical operations are:**
  - **AND**
  - **OR**
  - **NOT**
- **AND is denoted by a dot (·), $\times$, $\wedge$, or even none.**
- **OR is denoted by a plus (+) or $\vee$, .**
- **NOT is denoted by an overbar (¯), a single quote mark (') after, or (~) before the variable.**

Logical Operations

- **AND:** $Z = X \times Y$ **is read "Z is equal to X AND Y."**

  Z = 1 if and only if X = 1 and Y = 1; otherwise Z = 0;

- **OR:** $Z = X + Y$ **is read "Z is equal to X OR Y."**

  Z = 1 if X = 1 or if Y = 1, or if both X =1 and Y = 1;
  Z=0 if and only is X =0 and Y = 0;

- **NOT:** $Z = \overline{X}$ **is read "Z is equal to NOT X."**

  If X =1, Z= 0; but if X =0, then Z =1

Operator Definitions

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

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

- Note: The statement:

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

  is not the same as

  1 + 1 = 1 (read "1 or 1 equals 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	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

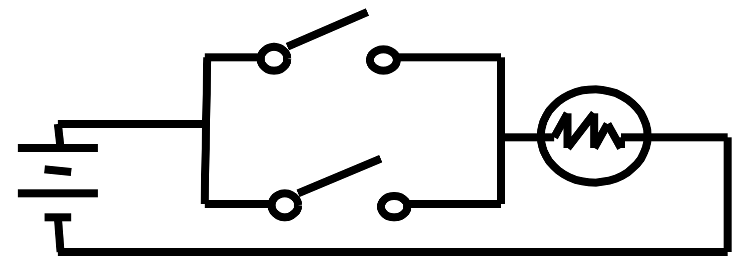
OR		
X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

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

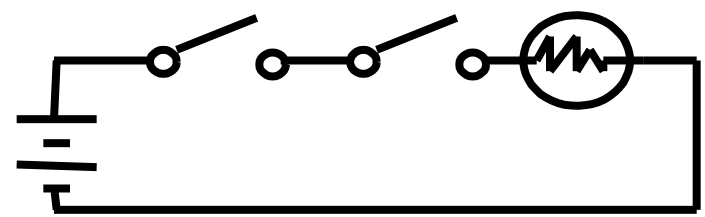
Logic Function Implementation

- 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

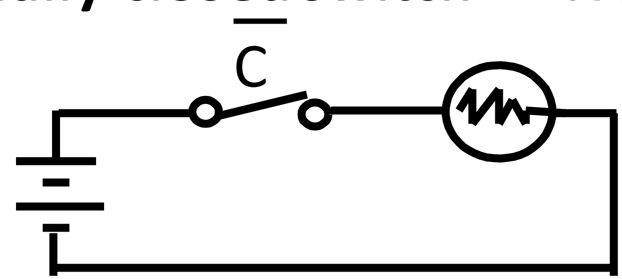
Switches in parallel => OR



Switches in series => AND

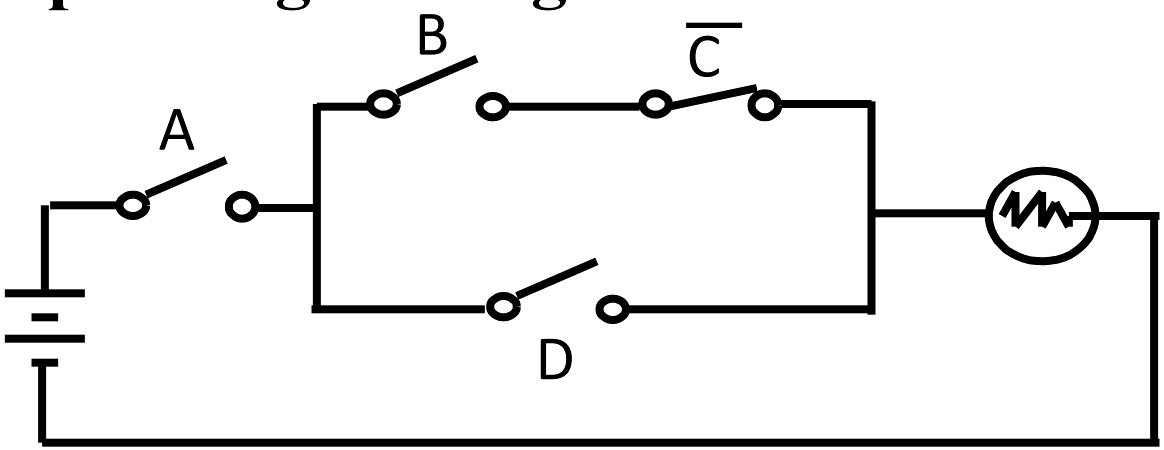


Normally-closed switch => NOT



Logic Function Implementation (Continued)

- **Example: Logic Using Switches**



- **Light is on (L = 1) for**

  $L(A, B, C, D) = A((B\overline{C}) + D) = A B \overline{C} + A D$

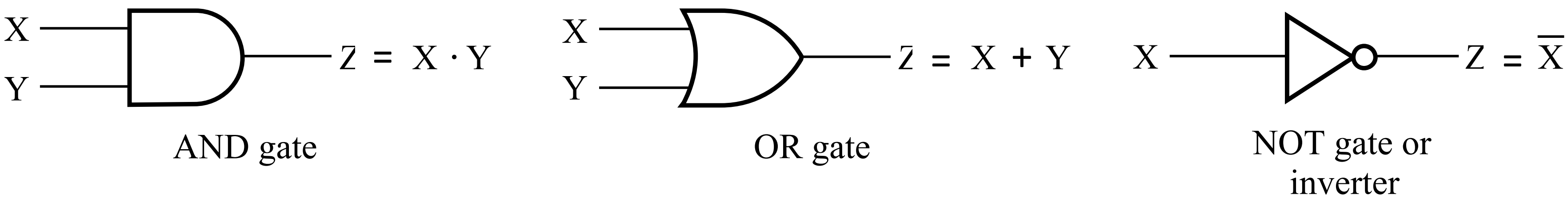
  and off (L = 0), otherwise.

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 electronically replaced relays.**
- **Today, *transistors* are used as electronic switches that open and close current paths.**
- **Optional: Chapter 6 – Part 1: The Design Space**

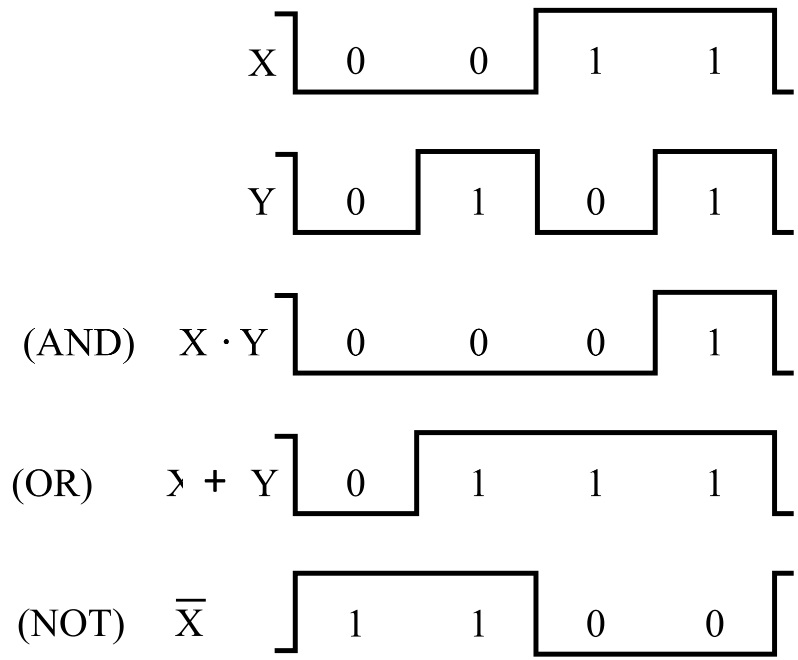
Logic Gate Symbols and Behavior

- **Logic gates have special symbols:**



(a) Graphic symbols

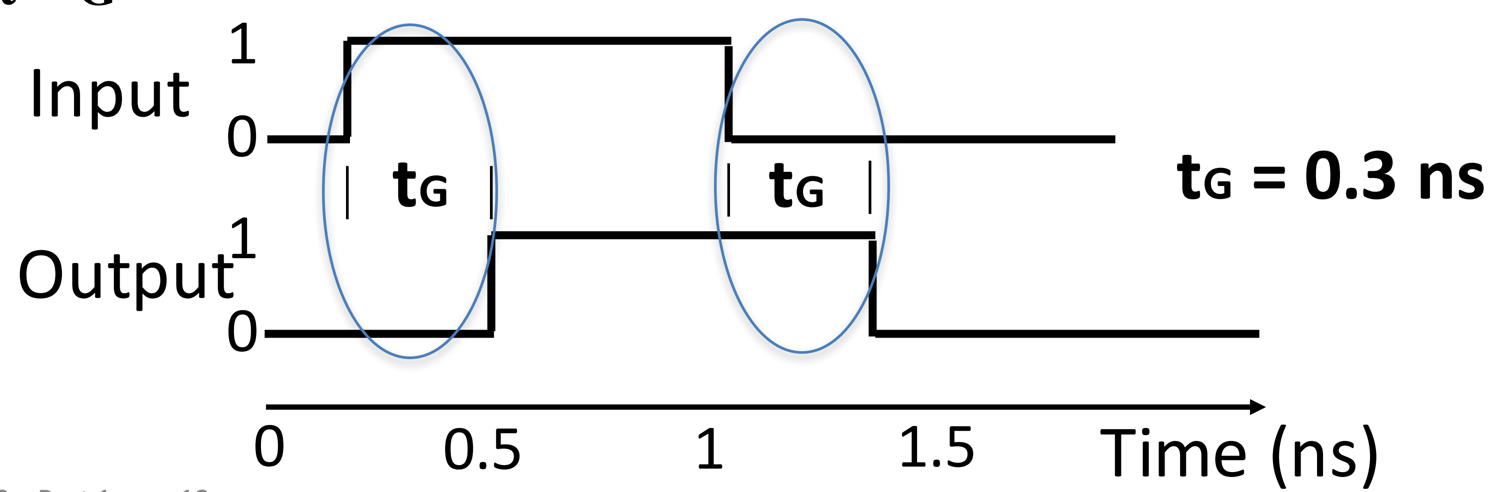
- **And waveform behavior in time as follows:**



(b) Timing diagram

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:



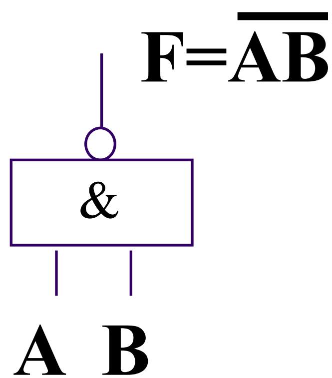
NOT AND (NAND)

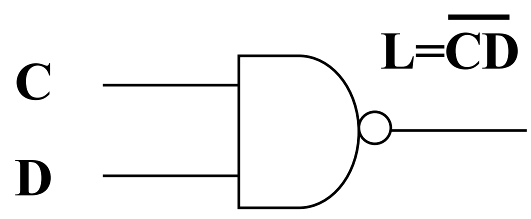
True and False

A	B	L
F	F	T
T	F	T
F	T	T
T	T	F

1 and 0

A	B	L
0	0	1
1	0	1
0	1	1
1	1	0





Graphic symbols

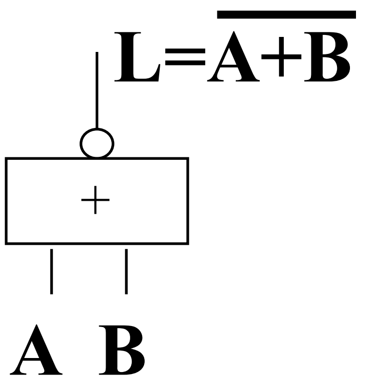
NOT OR (NOR)

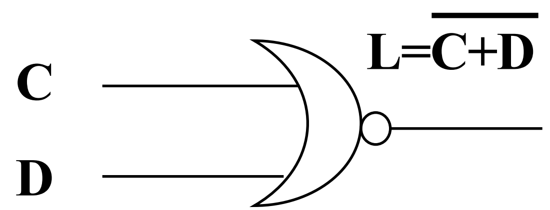
True and False

A	B	L
F	F	T
T	F	F
F	T	F
T	T	F

1 and 0

A	B	L
0	0	1
1	0	0
0	1	0
1	1	0

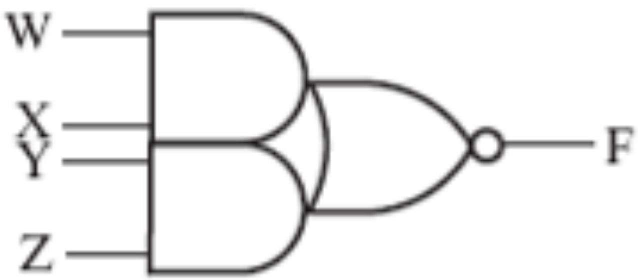




Graphic symbols

AND-OR-INVERT (AOI)

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



Graphic symbols

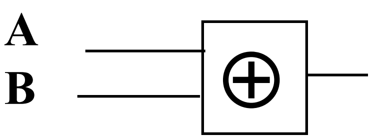
Exclusive-OR (XOR)

True and False

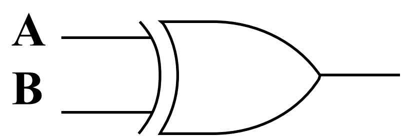
A	B	L
F	F	F
T	F	T
F	T	T
T	T	F

1 and 0

A	B	L
0	0	0
1	0	1
0	1	1
1	1	0



$L = A \oplus B$



$L = A \oplus B$

Graphic symbols

Eight Identities of XOR

$$X \oplus 0 = X \qquad X \oplus 1 = \overline{X}$$

$$X \oplus X = 0 \qquad X \oplus \overline{X} = 1$$

$$X \oplus \overline{Y} = \overline{X \oplus Y} \qquad \overline{X} \oplus Y = \overline{X \oplus Y}$$

$$X \oplus Y = Y \oplus X$$

$$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$$

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

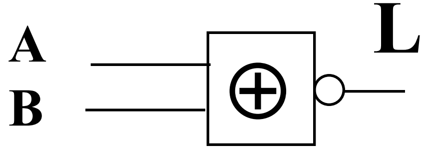
Exclusive-NOR (XNOR)

True and False

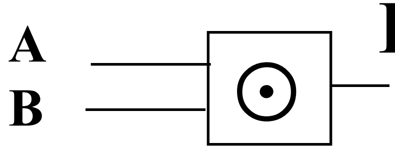
A	B	L
F	F	T
T	F	F
F	T	F
T	T	T

1 and 0

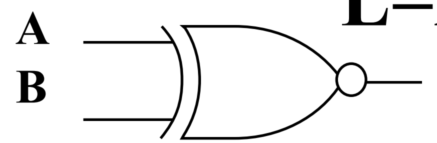
A	B	L
0	0	1
1	0	0
0	1	0
1	1	1



$$L=\overline{A\oplus B}$$



$$L=A\odot B$$

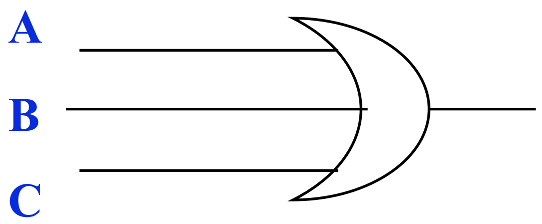


$$L=\overline{A\oplus B}$$

Graphic symbols

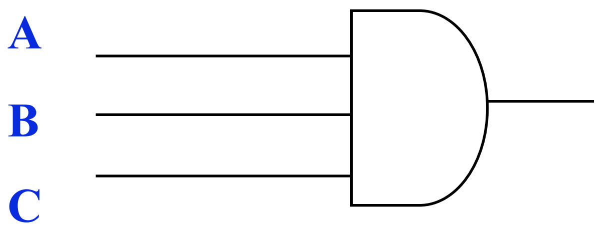
Multiple-input gates

OR:



F=A+B+C

AND:



F=ABC

Boolean Algebra

- deals with binary variables and logic operations (AND, OR, NOT etc.).
- A Boolean expression is an algebraic expression formed by (Example: **$(A + C) \cdot B + 0$**)
  - binary variables
  - Constants 1 or 0
  - Logic operation symbols
- A Boolean function is a Boolean equation consisting of Example: **$F = (A + C) \cdot B + 0$ or $F(A, B, C) = (A + C) \cdot B + 0$**
  - a binary variables
  - a “=“
  - a Boolean expression

Boolean Function/Equation

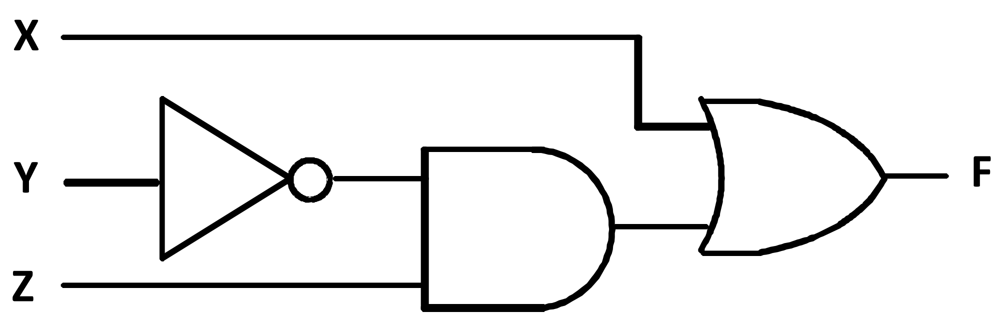
Truth Table

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

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

Logic Diagram



- The output of Boolean functions can be single or multiple.
- 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.

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:** $F = A(B + C)(\overline{C} + D)$

Boolean Algebra

- **An algebraic structure defined on a set of at least two elements, together with three binary operators (denoted +, · and $\overline{\ }$) that satisfies the following basic identities:**

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.	$(XY) Z = X(Y Z)$	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

Property of Duality of Boolean algebra

- The dual of an algebraic expression is obtained by interchanging + and $\cdot$, and replacing 0s by 1s and 1 by 0.

  Example: $F = (A + C) \cdot B + 0$

  dual $F = (A \cdot C + B) \cdot 1 = A \cdot C + B$

  Example: $G = X \cdot Y + (\overline{W + Z})$

  dual $G = ((X+Y) \cdot (\overline{W \cdot Z})) = ((X+Y) \cdot (W + Z)$

- Note the order of evaluation, add () if necessary
- If F is the dual of G, then G is also the dual of F
- If F=G, then their duals are also equal.

Property of Duality of Boolean algebra

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

  Example: $H = A \cdot B + A \cdot C + B \cdot C$

  dual H = $(A + B)(A + C)(B + C)$

  $= (A + BC)\,(B + C) = AB + AC + BC.$

  So H is self-dual !

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 = $(\overline{a} + bc)\overline{d} + e$

  $\overline{G} = ((a\,(\overline{b} + \overline{c})) + d\,)\,\overline{e} = (a\,(\overline{b} + \overline{c}) + d)\,\overline{e}$
- Difference between it and Dual?
  - Without complementing variable in dual

Other Useful Theorems

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

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

Proof of Absorption Theorem

- $A + A \cdot B = A$ (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)
$= A \cdot 1$	$1 + X = 1$
$= A$	$X \cdot 1 = X$

- Compute the dual on both sides, the equation is equal either

$$A \cdot (A+B) = A$$

Proof of Minimization Theorem

- $AB + A\overline{B} = A$ (Minimization Theorem)

Proof Steps Justification (identity or theorem)

$AB + A\overline{B}$

$= A(B + \overline{B})$ $X \cdot Y + X \cdot Z = X \cdot (Y + Z)$ (Distributive Law)

$= A \cdot 1$ $1 + X = 1$

$= A$

- Compute the dual on both sides, the equation is equal either

$$(A+B) \cdot (A+\overline{B}) = A$$

Proof of Simplification Theorem

- $A + \overline{A}B = A + B$ **(Simplification Theorem)**

Proof Steps **Justification (identity or theorem)**

$A + \overline{A}B$

$= (A + \overline{A})\,(A + B)$ $\quad X + Y \cdot Z = (X+Y) \cdot (X + Z)$ **(Distributive Law)**

$= 1 \cdot (A+B)$ $\quad X + X = 1$

$= A + B$

- **Compute the dual on both sides, the equation is equal either**

$$A(\overline{A}+B) = AB$$

Proof of Consensus Theorem

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

Proof Steps **Justification (identity or theorem)**

$AB + \overline{A}C + BC$

$= AB + \overline{A}C + 1 \cdot BC$? $1 \cdot X = X$

$= AB + \overline{A}C + (A + \overline{A}) \cdot BC$? $X + X' = 1$

$= AB + A'C + ABC + A'BC$ $X(Y + Z) = XY + XZ$ (Distributive Law)

$= AB + ABC + A'C + A'BC$ $X + Y = Y + X$ (Commutative Law)

$= AB \cdot 1 + ABC + A'C \cdot 1 + A'C \cdot B$

$X \cdot 1 = X$, $X \cdot Y = Y \cdot X$ (Commutative Law)

$= AB\,(1 + C) + A'C\,(1 + B)$ $X(Y + Z) = XY + XZ$ (Distributive Law)

$= AB \cdot 1 + A'C \cdot 1 = AB + A'C$ $X \cdot 1 = X$

Proof of the 15th basic identity

- Using the absorption theorem

  $X + XY = X$

- $X + YZ = (X+Y)(X+Z)$
  - From the right side

$$
\begin{aligned}
(X+Y)(X+Z) &= X+XY+XZ+YZ \\
&= X + XZ + YZ \\
&= X + YZ
\end{aligned}
$$

Proof of DeMorgan's Laws

$$\overline{x + y} = \overline{x} \times \overline{y} \qquad \overline{x \times y} = \overline{x} + \overline{y}$$

To show this we need to show that A + B= 1 and A·B= 0 with A = x + y and B = x'· y'. This proves that x'· y' = (x + y)'.

Part 1: Show x + y + x'· y' = 1.

x + y + x'· y'	
= (x + y + x') (x + y + y')	X + YZ = (X + Y)(X + Z) (Distributive Law)
= (x + x' + y) (x + y + y')	X + Y = Y + X (Commutative Law)
= (1 + y)(x + 1)	X + X' = 1
= 1 · 1	1 + X = 1
= 1	1 · X = 1

Proof of DeMorgan's Laws

$$\overline{x + y} = \bar{x} \times \bar{y} \qquad\qquad \overline{x \times y} = \bar{x} + \bar{y}$$

To show this we need to show that A + A' = 1 and A·A' = 0 with A = x + y and A' = x'· y'. This proves that x'· y' = (x + y)'.

Part 2: Show (x + y) · x'· y' = 0.

(x + y) · x'· y'

= (x · x'· y' + y · x'· y') X (Y + Z) = XY + XZ (Distributive Law)

= (x · x'· y' + y · y' · x') XY = YX (Commutative Law)

= (0 · y' + 0 · x') X · X' = 0

= (0 + 0) 0 · X = 0

= 0 X + 0 = X (With X = 0)

Based on the above two parts, x'y' = (x + y)'

The second one is proved by duality.

Example: Boolean Algebraic Proofs

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

Proof Steps **Justification (identity or theorem)**

$(\overline{X + Y})Z + X\overline{Y}$

Proof Steps	Justification (identity or theorem)
$= X' Y' Z + X Y'$	$(A + B)' = A' \cdot B'$ (DeMorgan's Law)
$= Y' X' Z + Y' X$	$A \cdot B = B \cdot A$ (Commutative Law)
$= Y' (X' Z + X)$	$A(B + C) = AB + AC$ (Distributive Law)
$= Y' (X' + X)(Z + X)$	$A + BC = (A + B)(A + C)$ (Distributive Law)
$= Y' \cdot 1 \cdot (Z + X)$	$A + A' = 1$
$= Y' (X + Z)$	$1 \cdot A = A$, $A + B = B + A$ (Commutative Law)

Expression Simplification

- Target: the smallest number of literals and the smallest number of terms (complemented and uncomplemented variables):
- AND-OR forms

$$A\ B + \overline{A}\ C\ D + \overline{A}\ B\ D + \overline{A}\ C\ \overline{D} + A\ B\ C\ D$$

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

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

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

$$= B\ (A + D) + \overline{A}\ C \quad \text{5 literals}$$

Expression Simplification

- OR-AND forms

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

- Dual

$$F' = \overline{A}\,\overline{B} + \overline{A}\,\overline{C}D + AC + B\overline{C}$$

$$= (\overline{A}\,\overline{B} + B\overline{C} + \overline{A}\,\overline{C}D) + AC$$

$$= \overline{A}\,\overline{B} + B\overline{C} + AC$$

- Dual of F' => F

$$F = F'' = (\overline{A} + \overline{B})(B + \overline{C})(A + C)$$

Example: Simplify Expression

$$L = AB + A\overline{C} + \overline{B}C + \overline{C}B + \overline{B}D + \overline{D}B + ADE(F + G)$$

$$L = A\overline{\overline{B}C} + \overline{B}C + \overline{C}B + \overline{B}D + \overline{D}B + ADE(F + G)$$ DeMorgan Laws

$$= A + \overline{B}C + \overline{C}B + \overline{B}D + \overline{D}B + ADE(F + G)$$ $A + \overline{A}B = A + B$

$$= A + \overline{B}C + \overline{C}B + \overline{B}D + \overline{D}B$$ $A+AB=A$

$$= A + \overline{B}C(D + \overline{D}) + \overline{C}B + \overline{B}D + \overline{D}B(C + \overline{C})$$ $A + \overline{A} = 1$

$$= A + \overline{B}CD + \overline{B}C\overline{D} + \overline{C}B + \overline{B}D + \overline{D}BC + \overline{D}B\overline{C}$$ Distributive Laws

$$= A + \overline{B}C\overline{D} + \overline{C}B + \overline{B}D + \overline{D}BC$$ $A+AB=A$

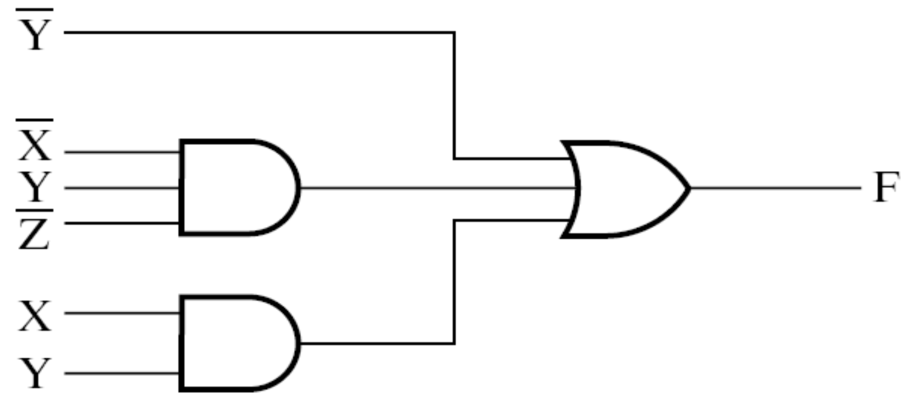
$$= A + C\overline{D}(\overline{B} + B) + \overline{C}B + \overline{B}D$$

$$= A + C\overline{D} + \overline{C}B + \overline{B}D$$ $A + \overline{A} = 1$

Sum of Products, Products of Sum

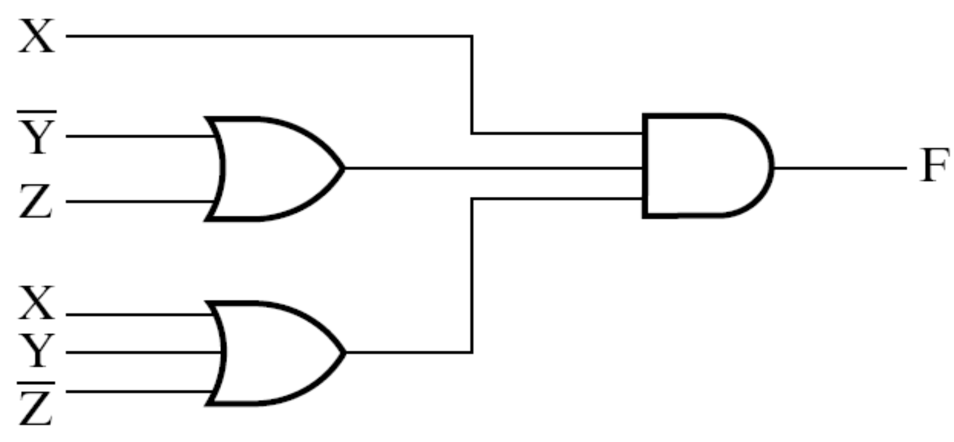
- Sum of products

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



- Product of sums

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



Overview – Canonical Forms

- What are Canonical 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)**
  - **Any function can be expressed to the canonical form**

Minterms

- **Minterms are AND terms with every variable present in either true or complemented form.**
- **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.**

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

Minterms

- **Complemented or uncomplemented? A literal is**
  - **complemented if the corresponding bit is 0**
  - **Uncomplemented if the corresponding bit 1**
- **The <u>index</u>:** the decimal equivalent of the binary combination corresponding to the Minterm

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

Maxterms

- **<u>Maxterms</u> are OR terms with every variable in true or complemented form.**
- **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.**

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

Maxterms

- **Complemented or uncomplemented? A literal is**
  - **complemented if the corresponding bit is 1**
  - **Uncomplemented if the corresponding bit is 0**
- **The <u>index</u>:** the decimal equivalent of the binary combination corresponding to the Minterm

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:**

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

Thus M_i is the complement of m_i.

Index	Minterm	Maxterm
0	$\overline{x}\,\overline{y}$	$x + y$
1	$\overline{x}\,y$	$x + \overline{y}$
2	$x\,\overline{y}$	$\overline{x} + y$
3	$x\,y$	$\overline{x} + \overline{y}$

Observations

- **In the function tables:**
  - **Each <u>min</u>term has one and only one 1 present in the 2^n terms (a <u>minimum</u> of 1s). All other entries are 0.**
  - **Each <u>max</u>term has one and only one 0 present in the 2^n terms All other entries are 1 (a <u>max</u>imum of 1s).**
- **We can implement any function by "ORing" the minterms corresponding to "1" entries in the function table. These are called the minterms of the function.**
- **We can implement any function by "ANDing" the maxterms corresponding to "0" entries in the function table. These are called the maxterms of the function.**
- **This gives us two <u>canonical forms</u>:**
  - **<u>Sum of Minterms (SOM)</u>**
  - **<u>Product of Maxterms (POM)</u>**

for stating any Boolean function.

Sum of Minterms (SOM)

- **Example: Find $F_1 = m_1 + m_4 + m_7$**
- **$F1 = \overline{x}\ \overline{y}\ z + x\ \overline{y}\ \overline{z} + x\ y\ z$**

x y z	index	m1	+	m4	+	m7	= F1
0 0 0	0	0	+	0	+	0	= 0
0 0 1	1	1	+	0	+	0	= 1
0 1 0	2	0	+	0	+	0	= 0
0 1 1	3	0	+	0	+	0	= 0
1 0 0	4	0	+	1	+	0	= 1
1 0 1	5	0	+	0	+	0	= 0
1 1 0	6	0	+	0	+	0	= 0
1 1 1	7	0	+	0	+	1	= 1

Product of Maxterms (POM)

- **Example: Implement F1 in maxterms:**

$$F_1 = M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$$

$$F_1 = (x + y + z) \cdot (x + \overline{y} + z) \cdot (x + \overline{y} + \overline{z}) \cdot (\overline{x} + y + \overline{z}) \cdot (\overline{x} + \overline{y} + z)$$

x y z	i	$M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$	= F1
0 0 0	0	0 · 1 · 1 · 1 · 1	= 0
0 0 1	1	1 · 1 · 1 · 1 · 1	= 1
0 1 0	2	1 · 0 · 1 · 1 · 1	= 0
0 1 1	3	1 · 1 · 0 · 1 · 1	= 0
1 0 0	4	1 · 1 · 1 · 1 · 1	= 1
1 0 1	5	1 · 1 · 1 · 0 · 1	= 0
1 1 0	6	1 · 1 · 1 · 1 · 0	= 0
1 1 1	7	1 · 1 · 1 · 1 · 1	= 1

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 ($v + \overline{v}$).**
- **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: $f = m_3 + m_2 + m_0 = \Sigma_m(0, 2, 3)$

Canonical Product of Maxterms

- **Any Boolean Function can be expressed as a <u>Product of Maxterms (POM)</u>.**
  - **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 $v \times \bar{v}$ and then applying the distributive law again.**
- **Example: Convert to product of maxterms:**

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

Apply the distributive law:

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

Add missing variable z:

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

Express as POM: $f = M_2 \cdot M_3 = \Pi M(2, 3)$

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** $F(x, y, z) = \Sigma_m(1,3,5,7)$

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

$$\overline{F}(x, y, z) = \Pi_M(1,3,5,7)$$

Standard Forms

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

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.**

Standard Sum-of-Products (SOP)

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

$$F = \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + AB\overline{C} + ABC$$

- **Simplifying:**

$$\begin{aligned} F &= A'\,B'\,C + A\,(B'\,C' + B\,C' + B'\,C + B\,C) \\ &= A'\,B'\,C + A\,(B' + B)\,(C' + C) \\ &= A'\,B'\,C + A\cdot 1\cdot 1 \\ &= A'\,B'\,C + A \\ &= B'C + A \end{aligned}$$

- **Simplified F contains 3 literals compared with 15 in minterm F**

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.

Home Assignment

2-1; 2-2a; 2-3a, c; 2-6b, d; 2-10a, c; 2-11a, b, d; 2-12b; 2-13a, c;