Computer Design fundamentals

Chapter 2 – Boolean algebra

浙江大学计算机学院　王总辉
zhwang@zju.edu.cn
http://course.zju.edu.cn
2023年9月

Overview

- Binary Logic and Gates
- Basic concepts of Boolean algebra
- Standard Forms
- Karnaugh map of Function
- Two-Level Optimization: Map Manipulation
- * Multi-level circuit optimization
- Other Gate Types
- * Properties of the Exclusive-OR operator
- High-Resistance output (tristate gate)

Binary Logic and Gates

Binary variables take on one of two values：0、1

- **Logical operators**
  - **operate on binary values and binary variables**
  - **Basic logical operators：AND, OR and NOT**

"And" operation symbols: “ • ”or “ ∧”

$$Z = X \bullet Y = XY = X \wedge Y$$

Verilog HDL：assign z = X && Y

“OR" operation symbols: “＋” or “ ∨”

$$Z = X + Y = X \vee Y$$

Verilog HDL：assign z = X || Y

“NOT” operation symbols: “—”or “ ¬ ”

$$Z = \overline{X} = \neg X$$

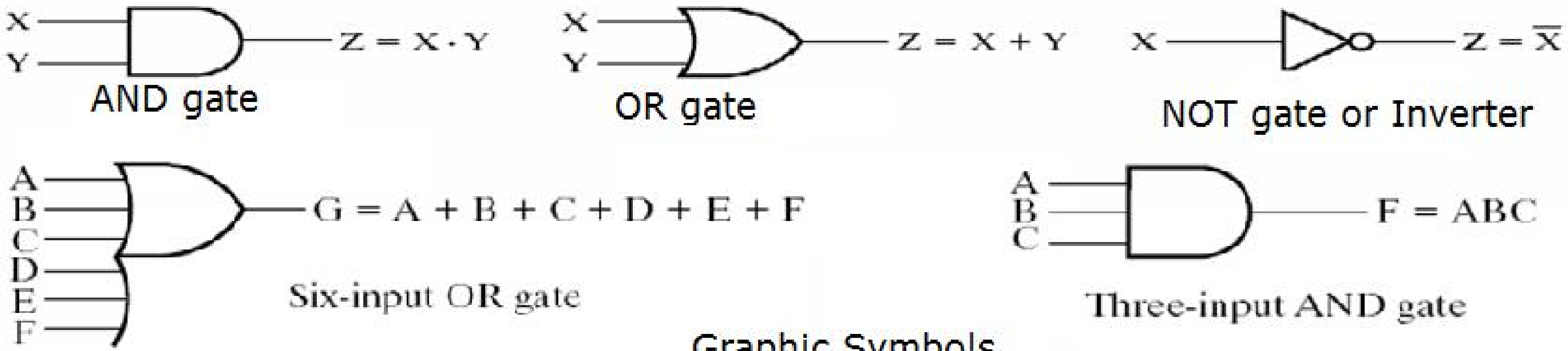
Verilog HDL：assign z = ～X

Basic logic gates

- Basic logic gates
  - With the three basic logic of operation of the corresponding circuit, the input can be the plurality (voltage or current) to generate an output signal, voltage or current may represent a binary logical variable “1” or “0”.
- Logic gate
  - The basic hardware components (circuit), does not focus the intrinsic electronic properties of each circuit, simply Note what the circuit external logical attributes are.
  - Middle region
  - Rising or Falling (edge)

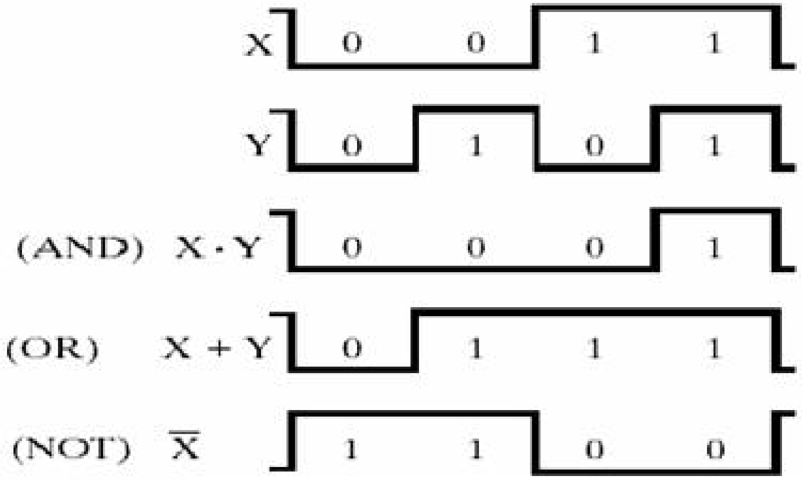
Logic Gate Symbols and Behavior

- **Logic gates have special symbols:**

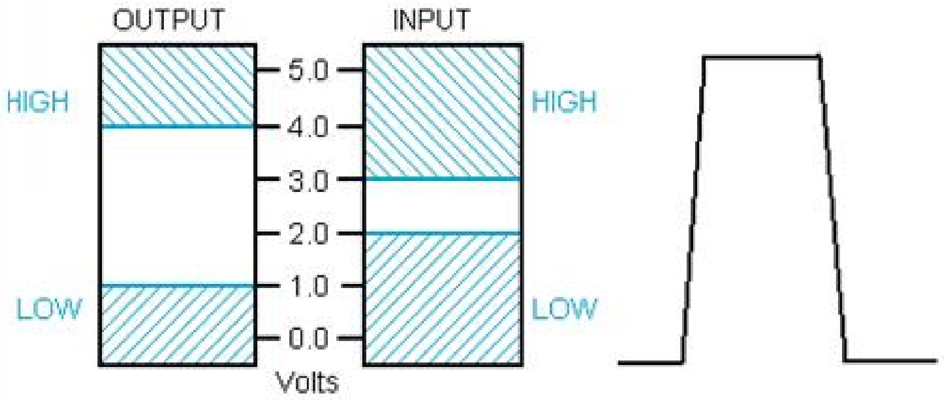


Graphic Symbols

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



Timing Diagram



Example of Voltage Ranges

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:**

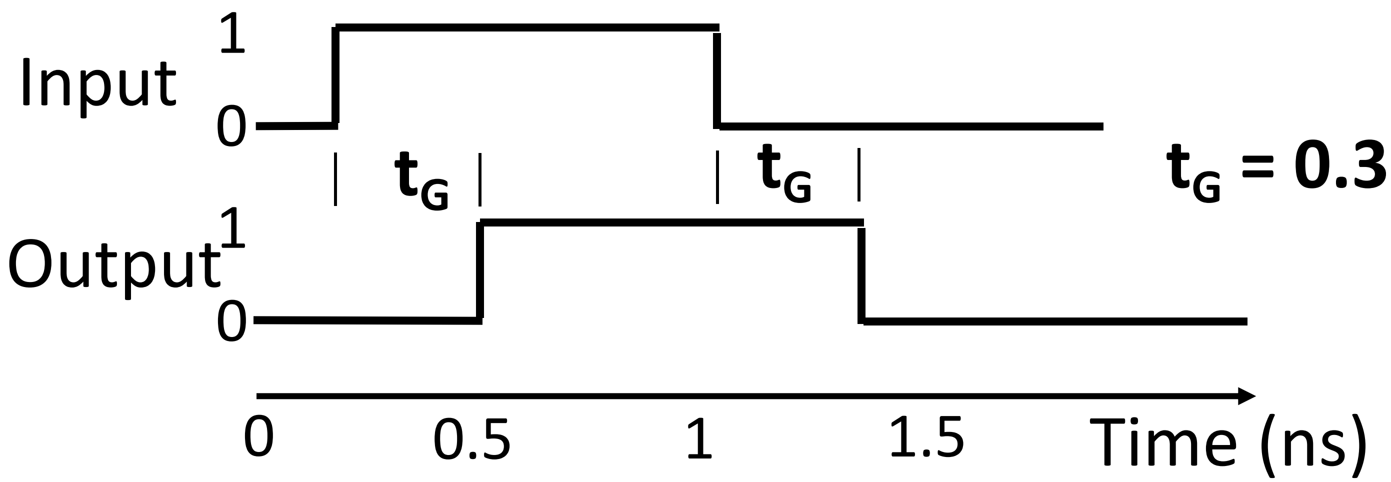
NOT	
X	$Z = \overline{X}$
0	1
1	0

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

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:**



"AND" operation

If decide that a certain event is truly then all conditions must have met, such a causal relationship is called logic "AND".

In logic algebra, logical "and" relationship is described with the "AND" operation. "And" operator, also known as logical multiplication operator with ordinary algebra.

$F = A \cdot B$; $F=AB$; $F=A \times B$; $F=A \cap B$

It means that if A & B are true (1), then F is true (1); otherwise F is false (0).

Truth table for Logic "and"

Expressed event with T、 F

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

Expressed event with 1、0

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

Operation rules：

F· F = F　　**T· F = F**

F · T =F　　**T · T = T**

or:

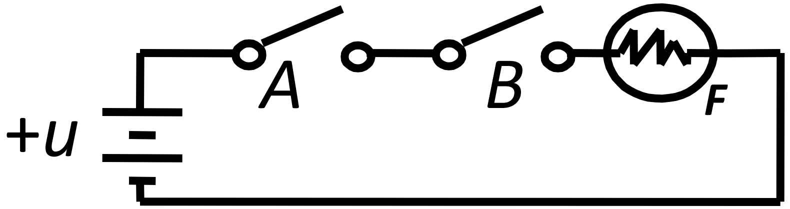
0· 0= 0　　**0· 1 = 0**

1· 0 = 0　　**1· 1 =1**

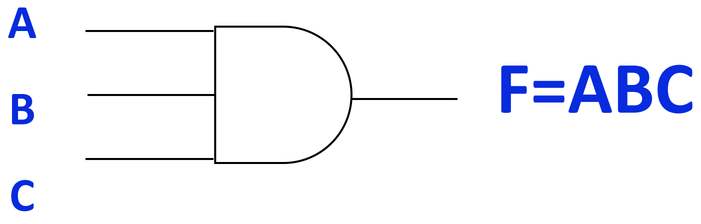
"And" arithmetic logic circuit is called the "and" gate in the digital system.

Logic Function Implementation and symbol

Switches in series => AND

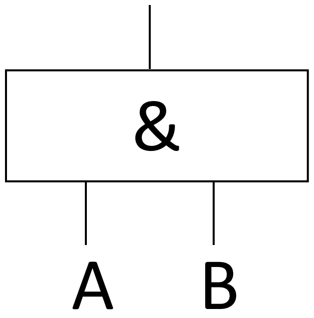


"AND" Logical event schematic diagram



General logic symbols

$F = A \cdot B$



GB logical symbols

"OR" operation

There are a multi-conditions that decide a particular event become truly, as long as there is one or more than one condition was established, the event can occur, this causal relationship is called "OR" logic.

In logic algebra, logical "OR" relationship is described with the "OR" operation. "OR" operator also known as logical add.

$$F=A+B \text{ or } F=A\cup B$$

It means that if A or B is true (1), then F is true (1); when only A & B all are false, F is false (0).

Truth table for Logic “OR”

“OR” Truth

A	B	F
F	F	**F**
F	T	**T**
T	F	**T**
T	T	**T**

A	B	F
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Operation rules： F+ F = F； T+ F = T

F + T =T； T+ T = T

or

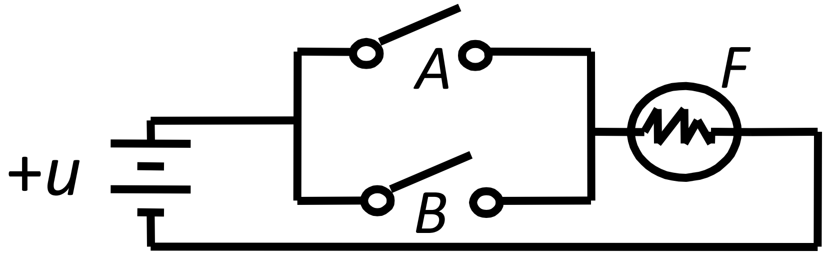
0+0= 0 0+1 = 1

1+0 = 1 1+1 = 1

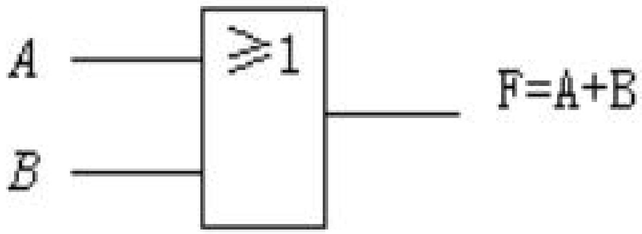
“OR" arithmetic logic circuit is called the "OR" gate in the digital system.

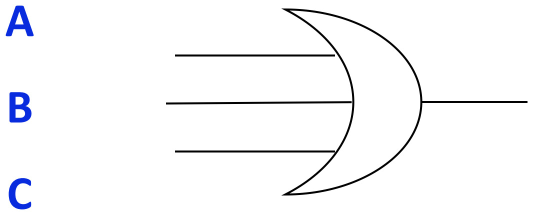
Logic Function Implementation and symbol

Switches in parallel => OR



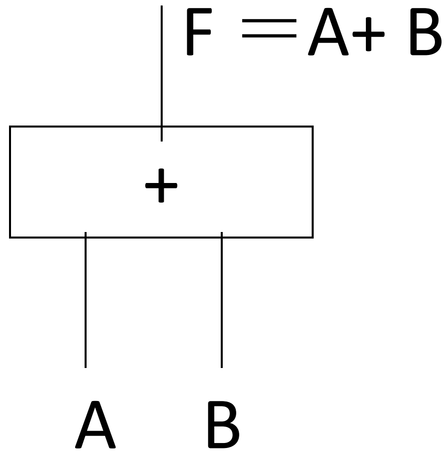
"OR" Logical event schematic diagram





F=A+B+C

General logic symbols



GB logical symbols

"NOT" operation

If the occurrence condition of an event depends on the negation of other event, the causal relationship called "NOT" logic. "Non-logic " with non-operation description. "Non" operation, also known as negated arithmetic operator.

$$F=\overline{A} \quad \text{or} \quad F=\neg A$$

Pronounced as "F equals to the A bar"

It Means if A = 0, then F is 1; Conversely, if A = 1, F is 0.

Truth table for Logic “NOT”

“NOT” Truth

A	F
F	T
T	F

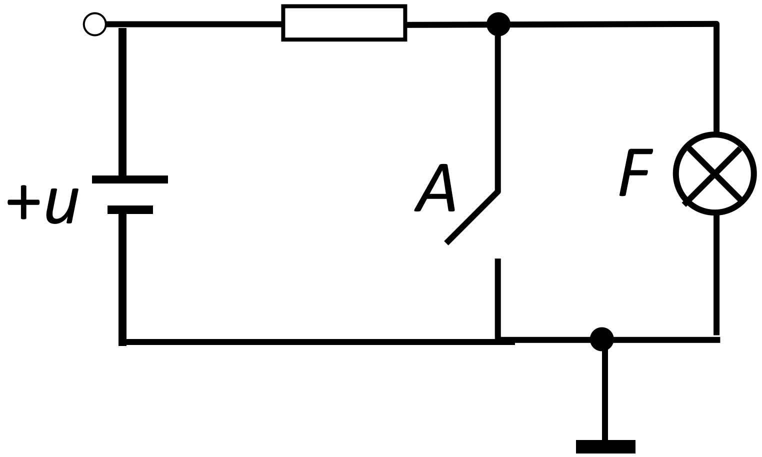
A	F
0	1
1	0

Operation rules： $\overline{0}=1$ $\overline{1}=0$

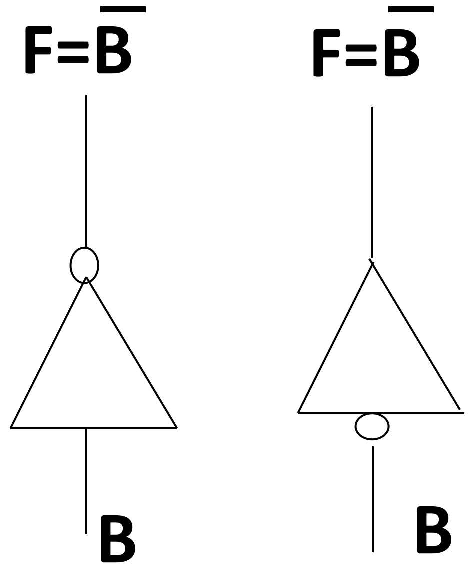
$\overline{F}=T$ $\overline{T}=F$

"NOT" arithmetic logic circuit is called the “NOT" gate in the digital system.

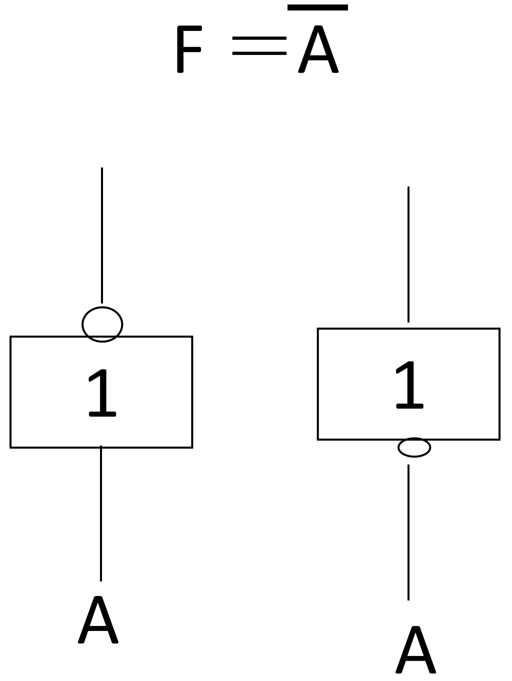
Logic Function Implementation and symbol



“NOT" Logical event schematic diagram

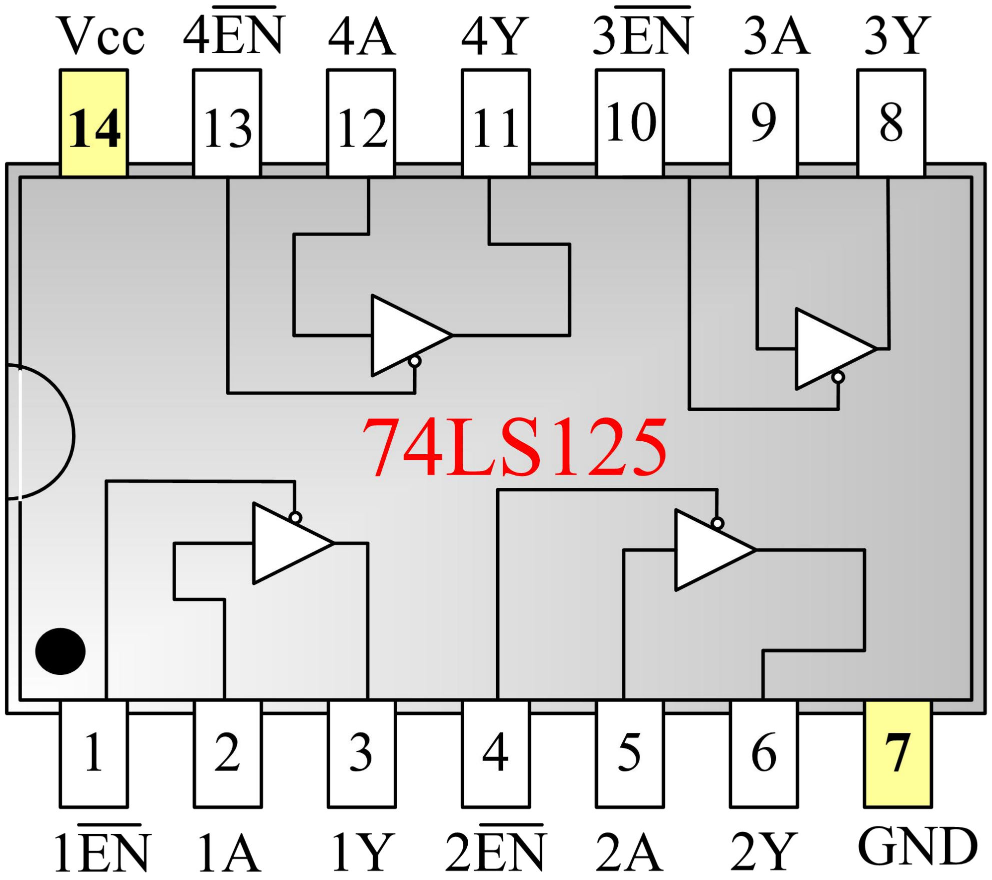


General logic symbols



GB logical symbols

Tristate gate



EN	*A*	*Y*
0	0	0
0	1	1
1	×	高阻态

Logic Diagrams and Expressions

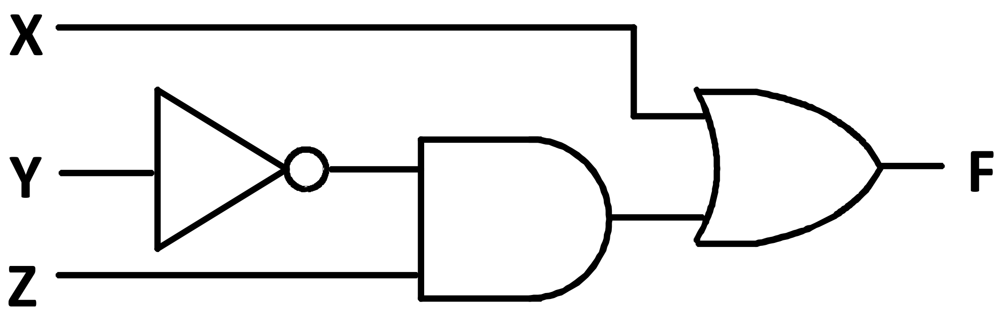
Truth Table

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



- **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.**

Common logic gate circuit

- **Basic logic gate circuit**
  - AND、OR、NOT
  - According to basic logical circuit, can form Commonly compound logic circuit
  - Example： “NAND”、 “NOR ” 、 “XOR” etc.

" NAND " operation

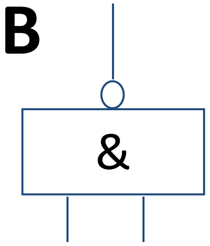
Expressed event with T、F

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

Expressed event with 1、0

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

$F=\overline{AB}$

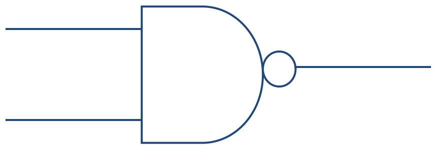


A B

Circuit symbols

C

D



$L=\overline{CD}$

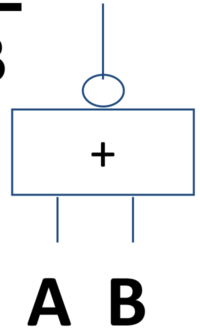
“NAnd" arithmetic logic circuit is called the “Nand" gate in the digital system

"NOR" operation

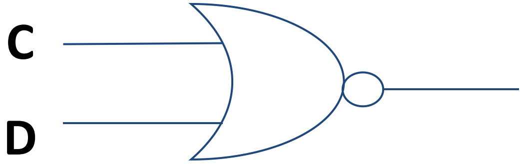
Expressed event with T、F		
A	*B*	***L***
F	F	T
F	T	F
T	F	F
T	T	F

Expressed event with 1、0		
A	*B*	***L***
0	0	1
0	1	0
1	0	0
1	1	0

$F=\overline{A+B}$



Circuit symbols



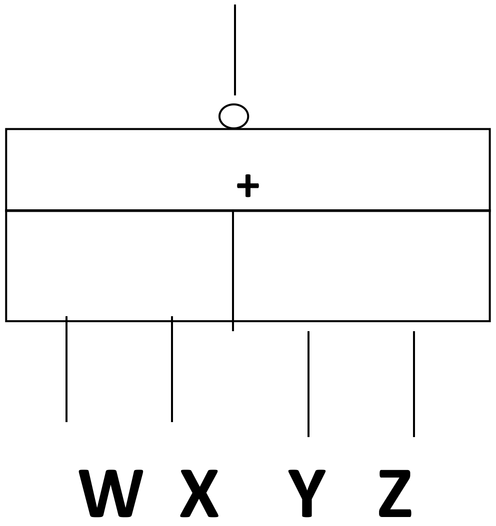
$L=\overline{C+D}$

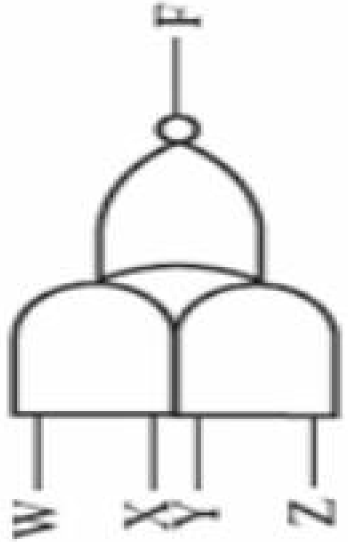
"NOR" arithmetic logic circuit is called the "NOR" gate in the digital system

"AND-OR-INVERT" operation

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

$$F=\overline{WX+YZ}$$





Circuit symbols

"AND-OR-INVERT " arithmetic logic circuit is called the "AND-OR-INVERT " gate in the digital system

Exclusive OR $L = A \oplus B$

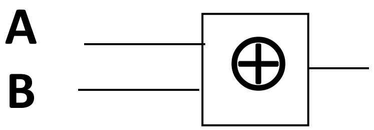
Expressed event with T、F

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

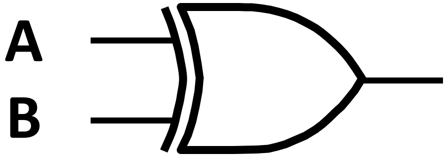
Expressed event with 1、0

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

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



$L = A \oplus B$



$L = A \oplus B$

Circuit symbols

The XOR identities： Eight properties

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

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

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

$X \oplus Y = Y \oplus X$

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

Uses for the XOR

1. The controllable source / inverted output

$$A \oplus 0 = A$$

$$A \oplus 1 = \overline{A}$$

2. Digital comparator

$$A \oplus \overline{A} = 0$$

$$A \oplus A = 1$$

3. half adder

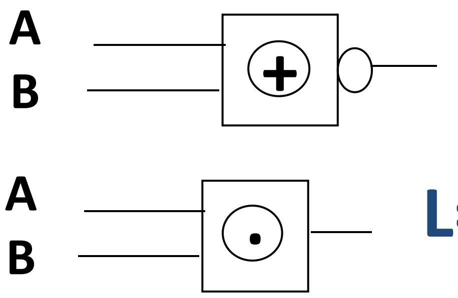
Truth table, without Carry

Exclusive NOR $L = A \odot B$ Also known as "XNOR operation

Truth table

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

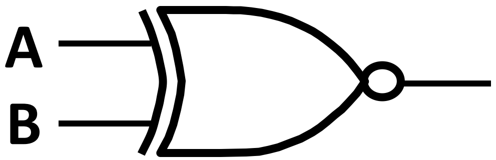
A	B	L
0	0	1
0	1	0
1	0	0
1	1	1



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

$L = A \odot B$

$$\overline{A \oplus B} = AB + \overline{A}\,\overline{B}$$



$L = A \odot B$

Circuit symbols

Basic concepts of Boolean algebra

- **Boolean Algebra:**
  - **also called Logic Algebra , useful mathematical system for specifying and transforming logic functions.**
- **We study Boolean algebra as a foundation for designing and analyzing digital systems!**
- **Logical variables**
  - Is a binary variable. Two logical values only take 0,1 (true, false).
  - The logic value not size
- **Three basic functions：**
  - **“AND”、 “OR”、 “NOT”**

Boolean function defined

The definition of the logic function!!!

Assume a (circuit) input logic variable for X1, X2, ..., Xn, and the output logic variable is F. When X_1, X_2, ..., X_n value is determined, the value of F is uniquely decided upon, then F is called to as X_1, X_2, ..., X_n, the logic function, denoted as

$$F=f\,(X_1, X_2, \ldots, X_n)$$

Note: Although X+Y=X+Z or XY=XZ ,

but Y = Z is impossible true

Boolean algebra representation

- Boolean algebra representation： The Expression constituted by the binary variables, constants 0 and 1, logical operators, and parentheses

  For example：

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

  – Function can be expressed in a variety of expressions, Function expressions isn‘t **unique**.
- **A single output** and **multi-output**
  – the latter require more than one function expression represents the output.
- Can also use a truth table
  – A Boolean function is **unique** with
  a Truth table

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

Basic properties of Boolean algebra

-- basic formula, rules and additional formula

Basic Equation

simplifying Boolean expressions with the properties

1.	$X + 0 = X$	2.	$X \cdot 1 = X$	**Identities**
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	**Null elements**
5.	$X + X = X$	6.	$X \cdot X = X$	**Idempotency**
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	**Complements**
9.	$\overline{\overline{X}} = X$			**Involution**
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**

18

$A(A+B) = A$	$A + AB = A$	**Covering**
$A(\overline{A}+B) = AB$	$A + \overline{A}B = A + B$	
$(A+B)(\overline{A}+C)(B+C) = (A+B)(\overline{A}+C)$	$AB + \overline{A}C + BC = AB + \overline{A}C$	**Consensus**

Boolean function equal

- **Assuming two logical function**

$$F_1=f_1\ (X_1, X_2, \ldots, X_n)$$

$$F_2=f_2\ (X_1, X_2, \ldots, X_n)$$

If corresponding to X_1, X_2, ..., X_n for any set of values, the value of F1 and F2 are the same, then the functions F1 is equal to the function F2.

$$F_1= F_2$$

Truth table is completely identical!!!

浙江大学
ZheJiang University

Complementing and Duality rules*

Complementing function

AND、 OR swap， 0、 1 swap
& Put variable Inverses

For logic function F, interchange **AND** and **OR** operators ; complement each **constant value** and **literal,** then obtained the new function is the inverse function of the original function is referred to as: $\overline{F}$

For example： known $F = \overline{A}B + C\overline{D}$

According to inversion Rules can be obtained :

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

Note the following two points:

① **The holding operation priority unchanged, if necessary, add brackets indicate.**

② **Within converting, public non-operation remains unchanged for several variables**

For example 1:

$$F = \overline{A} + \overline{B} \cdot (C + \overline{D}E), \text{则}$$

$$\overline{F} = A \cdot [B + \overline{C}(D + \overline{E})]$$

$$\overline{F} \neq A \cdot B + \overline{C} \cdot D + \overline{E}$$

Example 2:

$$L = A \cdot \overline{B + C + \overline{D}}$$

$$\overline{L} = \overline{A} + \overline{\overline{B} \cdot \overline{C} \cdot D}$$

Duality rules

AND、OR swap，0、1 swap let variable unchanged

For logic function F, all “AND” changed to “OR”、“OR” changed “AND”、“0” changed to “1”、 “1” is changed to “0”, then the resulting new logic function F‘ is F Duality function.

Variable don't Inverses!!!

Seek a function F’ Duality , the operation sequence keep as same as the original function 。

If F‘ is the F Duality, then F is also F’ of Duality. F and F‘ is mutually Duality formula .

If the two logical functions F and G are equal, then the Duality formula F' and G' are also equal.

Substitution rules

Any logical equation that contains a variable A, and if all occurrences of A's position are replaced with a logical function F, the equation still holds 。

Example： Assume $X(Y+Z)=XY+XZ$， if X + YZ Instead of X， then equation still holds：

$$(X+YZ)(Y+Z)=(X+YZ)B+(X+YZ)Z$$

- **Similarly equation (complementary)**

$$f(X_1, X_2, \ldots, X_n)+\overline{f}(X_1, X_2, \ldots, X_n)=1$$

Additional equation

Shannon formula

- **Equation 1：**
  - Assuming: Function F contained variables x、$\bar{x}$, at "x AND F" operation, variable x may be replaced by "1" , variable $\bar{x}$ can be replaced by "0". at "$\bar{x}$ AND F" operation, x can be "0", $\bar{x}$ can be replaced with "1" .

$$x\, f(x, \bar{x}, y \ldots, z_n) = x\ f(1, 0, y \ldots, z)$$

$$\bar{x}\, f(x, \bar{x}, y \ldots, z_n) = \bar{x}\ f(0, 1, y \ldots, z)$$

- This expression is actually expansion:

$$\mathbf{X} \cdot \bar{\mathbf{X}} = 0 \text{ and } \mathbf{X} \cdot \mathbf{X} = X$$

Similarly, based on A + $\bar{A}$ = 1, A + $\bar{A}$B = A + B, and A + AB = A, we can obtain the following equation

$$x + f(x, \bar{x}, y, \ldots, z) = x + f(0, 1, y, \ldots, z)$$

$$\bar{x} + f(x, \bar{x}, y, \ldots, z) = \bar{x} + f(1, 0, y, \ldots, z)$$

Additional **equation** is very useful for simplifying.

Example : $f = xy + \bar{x}z + (\bar{x} + \bar{y})(x + z)$，求$x \cdot f$

$$\begin{aligned} x \cdot f &= x \cdot [xy + \bar{x}z + (\bar{x} + \bar{y})(x + z)] \\ &= x \cdot [1 \cdot y + 0 \cdot z + (0 + \bar{y})(1 + z)] \\ &= x \cdot [y + \bar{y}] = x \end{aligned}$$

- **Equation 2： The function F contains the both of the variables x、$\bar{x}$， may be follow:**

$$f(X,\bar{X},Y\ldots Z)= x\cdot f（x,\bar{x},y,\ldots z)+\bar{x}f(x,\bar{x},y\ldots z)$$

$$= xf(1,0,y,\ldots z)+\bar{x}f(0,1,y\ldots z)$$

逻辑函数分解

(Problrm 3-2)

- The same duality formula:

$$F(x,\bar{x},y,..z)=[x+f(1,0,y\ldots z)]\cdot[\bar{x}+f(0,1,y,\ldots z)]$$

Shannon expansion

Equation 1and Equation 2 can be obtained:

$$f(x_1,x_2,x_3)=x_1x_2x_3f(1,1,1)+ x_1x_2\bar{x}_3f(1,1,0)$$

$$+ x_1\bar{x}_2x_3f(1,0,1)+ x_1\bar{x}_2\bar{x}_3f(1,0,0)$$

$$+\bar{x}_1x_2x_3f(0,1,1)+ \bar{x}_1x_2\bar{x}_3f(0,1,0)$$

$$+ \bar{x}_1\bar{x}_2x_3f(0,0,1)+ \bar{x}_1\bar{x}_2\bar{x}_3f(0,0,0)$$

Logic Functions Simplification

In general, the more simple expression of logic functions, circuit design is more simple. Simplify the expression simplest form is known as the logical function minimization, there are three commonly used methods, namely：

An application of Boolean algebra；

Karnaugh Maps (K-map)

Implicant Table（Q — M）

An application of Boolean algebra

Simplify to contain the smallest number of literals (complemented and uncomplemented variables):

The use of logic algebra formulas、theorems and rules of logic functions derived convert "AND-OR expression", and further simplified. No fixed steps you can follow, but can use Merged-items 、 absorption 、 eliminating-items、 allocating-items and so method.

Mainly depends on skilled flexibility in the use of formulas、theorems and rules.

The most simple expression is not unique!

AND-OR style simplification

The simplification should satisfy two conditions:

① "AND-item" is least in expression；

② The premise of is met ①, there are least number of variables for each "item"。

Example：$F = A\overline{C} + ABC + AC\overline{D} + CD$

Answer：$F = A(\overline{C} + BC) + C(A\overline{D} + D)$

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

$$= A\overline{C} + AB + AC + CD$$

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

$$= A(1 + B) + CD = A + CD$$

Several methods

(1) Merged AND-items Use of complementary Discipline formula

Extract the same head, the tail normalized

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

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

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

$A + \overline{A} = 1$

(2) **absorption** Absorption and Covering to reduce item

Example: $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$

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

$= \overline{X}Y + XZ$

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

$\left(A+B\right)\left(\overline{A}+C\right) = A\overline{A} + AC + \overline{A}B + BC$

Consensus

$= AC + \overline{A}B + BC = AC + \overline{A}B$

（3） allocating-items

Use of **A+$\bar{A}$=1、A$\bar{A}$=0，** Increase AND-term

Example:

$$L = AB + \bar{A}C + BCD = AB + \bar{A}C + BCD(A + \bar{A})$$

$$= AB + \bar{A}C + ABCD + \bar{A}BCD = AB + \bar{A}C$$

（4） eliminating-items

Using the absorption A+$\bar{A}$B=A+B, eliminating-items

Example：

$$L = \bar{A} + AB + \bar{B}E = \bar{A} + B + \bar{B}E = \bar{A} + B + E$$

Example： Simplifying logic functions: comprehensive

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

$$L = A\overline{\bar{B}C} + \bar{B}C + \bar{C}B + \bar{B}D + \bar{D}B + ADE(F+G)$$ （利用反演律 ）

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

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

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

$$= A + \bar{B}CD + \bar{B}C\bar{D} + \bar{C}B + \bar{B}D + \bar{D}BC + \bar{D}B\bar{C}$$

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

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

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

OR-AND style simplification*

The simplification should satisfy two conditions：

① “OR-item" is least in expression；

② The premise of is met ①, there are least number of variables for each "item" 。

✷ **methods：　Take advantage of Duality:**

"OR-AND" convert "AND-OR"

Simplification and then convert "OR-AND"

Example：

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

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

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

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

Duality again： "OR-AND" the most simplified:

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

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)**

Minterms

- **<u>Minterms</u> 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., x), there are 2^n minterms for *n* variables. denoted as m_i**

Minterm has the following properties：

1） only one set of variables value make to 1 for any one minterm .

2） any two minterms multiplied equal to 0 :$m_i \cdot m_j=0$ 。*i≠j*

3） Sum of all minterms equal to 1： $\Sigma m_i=1$。$i=0 \sim 2^n-1$

4) Any one minterm is not contained in the original function F , it can be seen as inAnti-function $\overline{F}$ 。

Note that the variable Order！

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

3 variables (X、Y、Z) Minterm

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

$$=m_2+m_3+m_6+m_7$$

$$=\Sigma\, m\,(2,3,6,7)$$

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., x), there are 2^n maxterms for *n* variables. denoted as M_i**

Maxterms has the following properties：

1） only one set of variables value make to 0 for any one Maxterm

2）sum of any two Maxterms equal to ： $M_i+M_j=1$ 。 $i \neq j$

3） Product of all Maxterms equal to 0 :

$$\prod_{i=0}^{2^n-1} M_i = 0$$

4) Any one Maxterm is not contained in the original function F , it can be seen as inAnti-function $\overline{F}$

3 variables (X、Y、Z)

Maxterm standard formula

Note that the variable Order ！

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

3 variables (X、Y、Z) Maxterm

Minterm and Maxterm Relationship

1) M_i and m_i is complement 。 $\overline{M_i}=m_i$; $\overline{m_i}=M_i$

Example： $m_3=\overline{A}\,B\,C$。 $M_3=A+\overline{B}+\overline{C}$

2) $F=\sum m_i=\overline{\prod M_i}$

Function Tables for Both

- Minterms of 2 variables

x y	m_0	m_1	m_2	m_3
0 0	1	0	0	0
0 1	0	1	0	0
1 0	0	0	1	0
1 1	0	0	0	1

Maxterms of 2 variables

x y	M_0	M_1	M_2	M_3
0 0	0	1	1	1
0 1	1	0	1	1
1 0	1	1	0	1
1 1	1	1	1	0

- **Each column in the maxterm function table is the complement of the column in the minterm function table since M_i is the complement of m_i.**

Function of the canonical forms

- 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 canonical forms:
  - Sum of Minterms (SOM)
  - Product of Maxterms (POM)

  for stating any Boolean function.

Minterm Function Example

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

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

Minterm Function Example

- $F(A, B, C, D, E) = m_2 + m_9 + m_{17} + m_{23}$
- $F(A, B, C, D, E) =$

Maxterm Function Example

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

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

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

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

Maxterm Function Example

- $F(A,B,C,D) = M_3 \cdot M_8 \cdot M_{11} \cdot M_{14}$
- F(A, B,C,D) =

Example: write the following minterm and Maxterm of the true table

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

minterm $F_{\min} = \overline{X}\,\overline{Y}\,\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}Z + XYZ$

Maxterm $F_{\max} = \left(X + Y + \overline{Z}\right)\left(X + \overline{Y} + \overline{Z}\right)\left(\overline{X} + Y + Z\right)\left(\overline{X} + \overline{Y} + Z\right)$

$$F_{\min}(X,Y,Z) = m_0 + m_2 + m_5 + m_7 = \sum(0,2,5,7)$$

$$F_{\max}(X,Y,Z) = M_1 \bullet M_3 \bullet M_4 \bullet M_6 = \prod(1,3,4,6)$$

$$\overline{F_{\min}(X,Y,Z)} = \overline{X}\,\overline{Y}Z + \overline{X}YZ + X\overline{Y}\,\overline{Z} + XY\overline{Z}$$

$$= m_1 + m_3 + m_4 + m_6 = \sum(1,3,4,6)$$

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: $A\,B\,C + \overline{A}\,\overline{B}\,C + B$
  - POS: $(A + B) \cdot (A + \overline{B} + \overline{C}) \cdot C$
- These "mixed" forms are neither SOP nor POS
  - $(A\,B + C)\,(A + C)$
  - $A\,B\,\overline{C} + A\,C\,(A + B)$

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:**

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

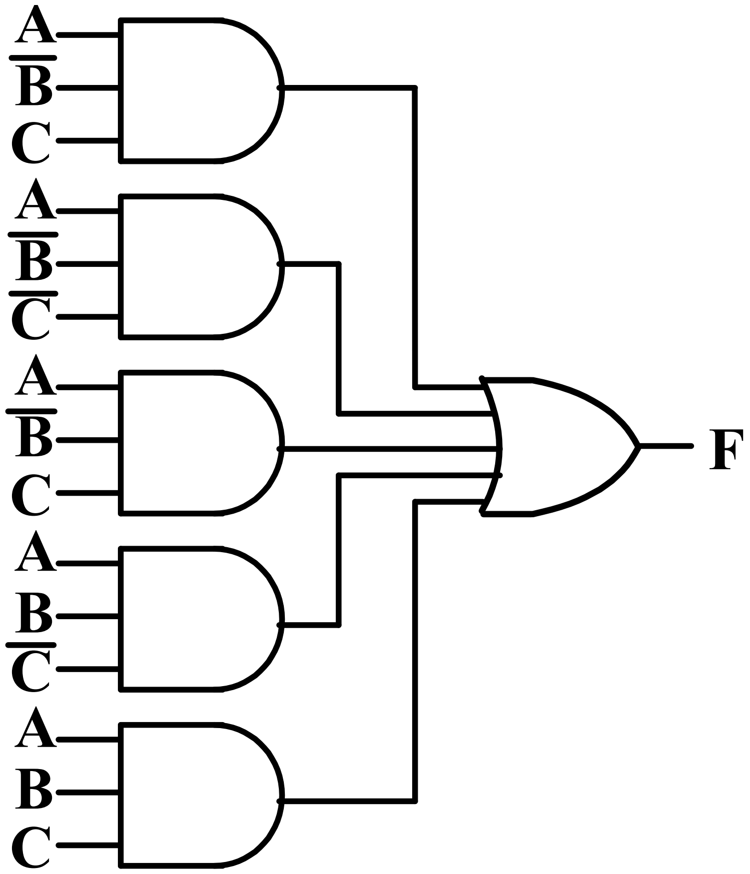
- **Simplifying:**

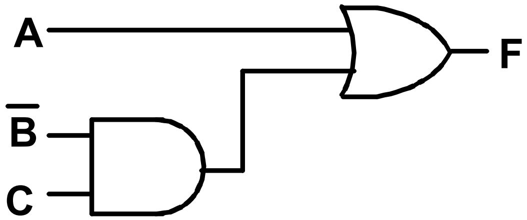
F =

- **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!**

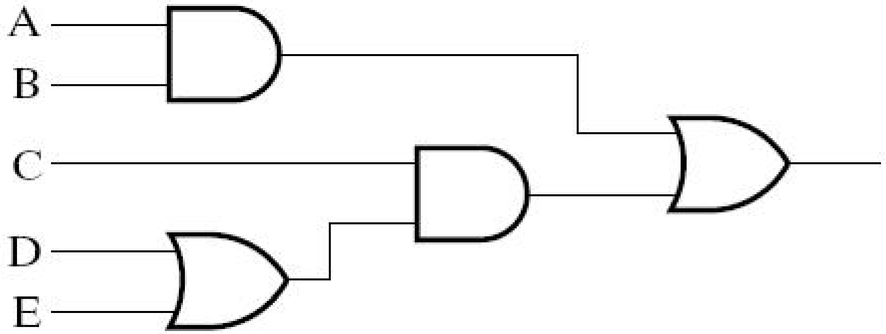




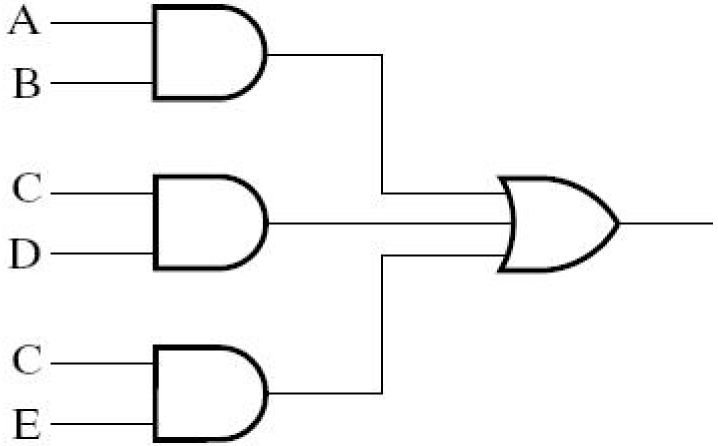
Different circuit implementation

Different Implementation: $F = AB + C\left(D + E\right)$

$$F = AB + C\left(D + E\right) = AB + CD + CE$$



(a) AB + C(D + E)

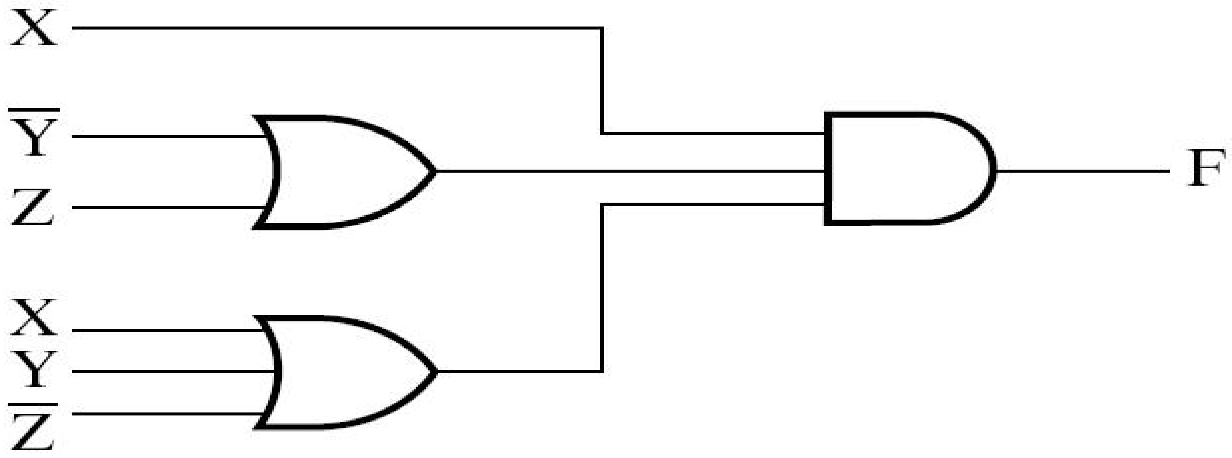


(b) AB + CD + CE

Two-level or three- Implementation

circuit implementation with POS

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



Function representation with Karnaugh Maps

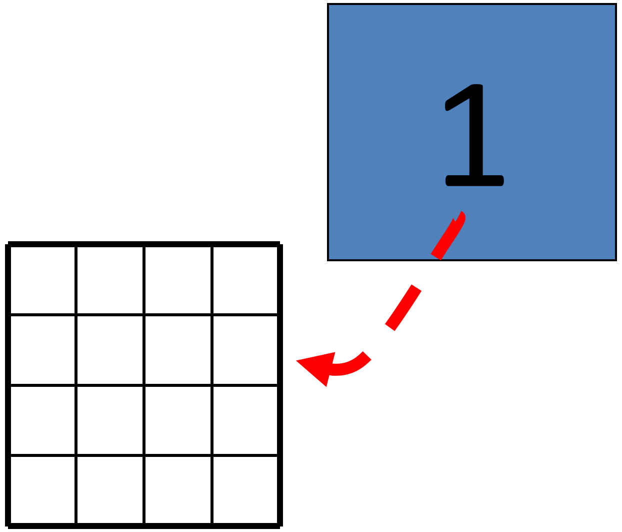
- The graphical Simplification is simple, intuitive and easy to grasp.
- It is very effective when variables is less than or equal to 5.

Karnaugh Maps (K-map)

- **A K-map is a collection of squares**
  - Each square represents a minterm
  - The collection of squares is a graphical representation of a Boolean function
  - Adjacent squares differ in the value of one variable
  - Alternative algebraic expressions for the same function are derived by recognizing patterns of squares
- **The K-map can be viewed as**
  - A reorganized version of the truth table
  - A topologically-warped Venn diagram as used to visualize sets in algebra of sets

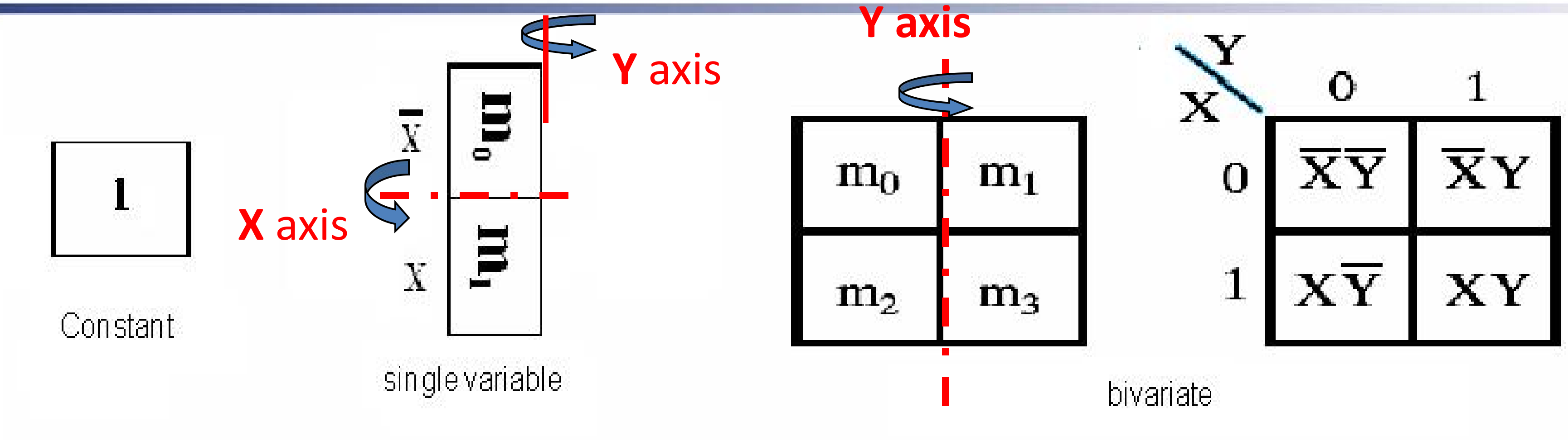
Boolean algebra geometry expressed: Karnaugh map

- **Any logic function can be expressed into SOM form**
  - **A function can expressed to a number of small squares (minterm) in the graphics.**
- **Function equal 1： *f(w,x,y,z)*=1**
  - **It means function contains all minterms**
  - **Fill in a large square "1" indicates**

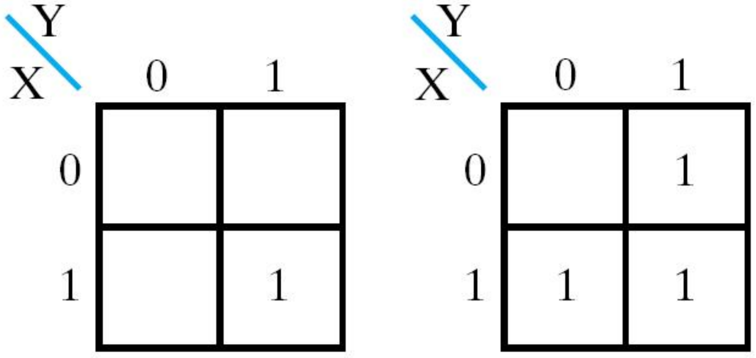


- **Function identically equal 1:**
  - **representative of 1 large square divided into 2^n small grid array**
  - **Each small square represents a minterm of function**
  - **All adjacent squares （minterm) can have only one different variable**

Constant, single variable and bivariate Karnaugh map



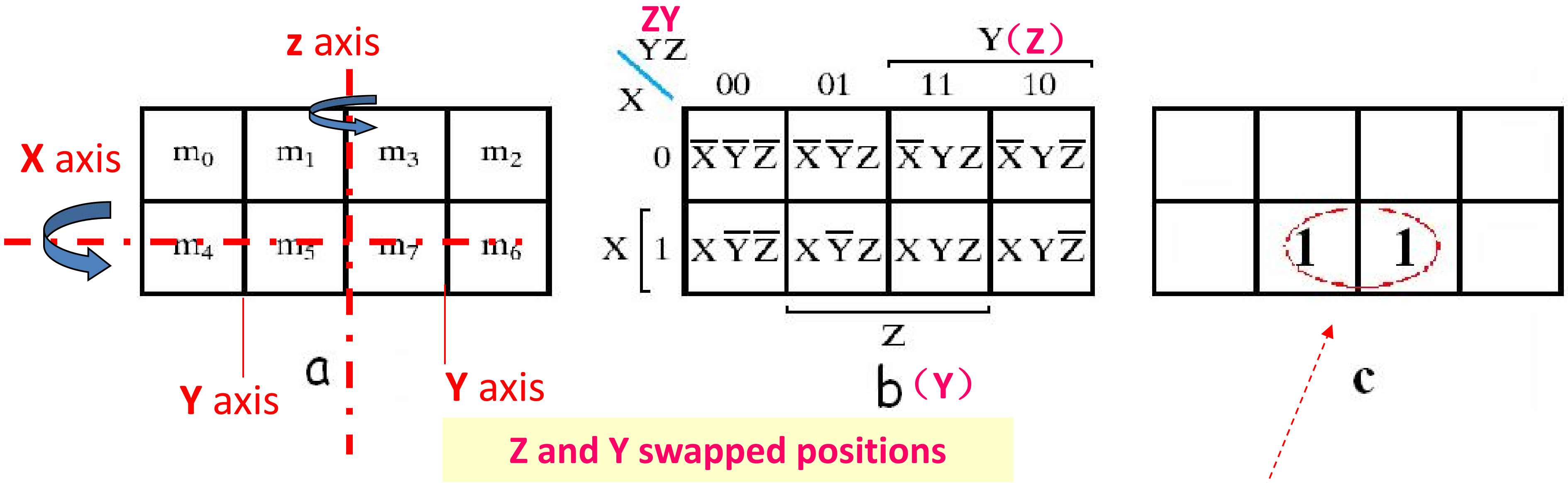
$$F(X,Y) = m_1 + m_2 + m_3 = \bar{X}Y + X\bar{Y} + XY = XY$$



(a) XY (b) X + Y

Three-variable Karnaugh map

there are 2^3 minterm, composed of 8 squares



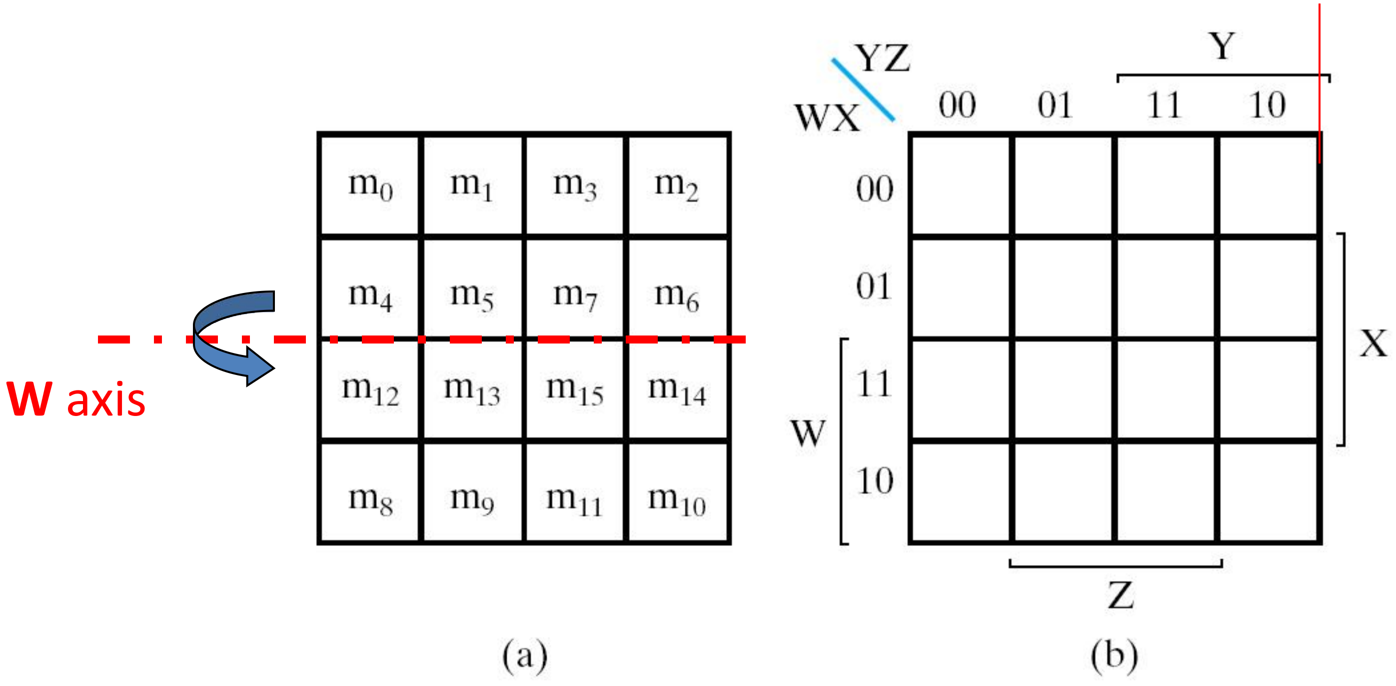
Contrast 2 variable Karnaugh map:

when y=1 Minterm+2; x = 1, minterm +4

e.g. $F(X,Y,Z) = m_5 + m_7 = X\overline{Y}Z + XYZ = XZ(\overline{Y} + Y) = XZ$

Four-variable Karnaugh map

there are 2^4 minterm, composed of 16 squares



Contrast 3 variable Karnaugh map: when w=1 Minterm+8

Five-variable Karnaugh map

there are 2^5 minterm, composed of 32 squares

Y axis X axis Y axis

VW \ XYZ	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

V axis

W axis

W axis

Z axis Z axis Z axis Z axis

- In summary, the Karnaugh map simplification function expression, an important step is to find out adjacent squares.
  - adjacent minterms including: squares adjacent squares overlap

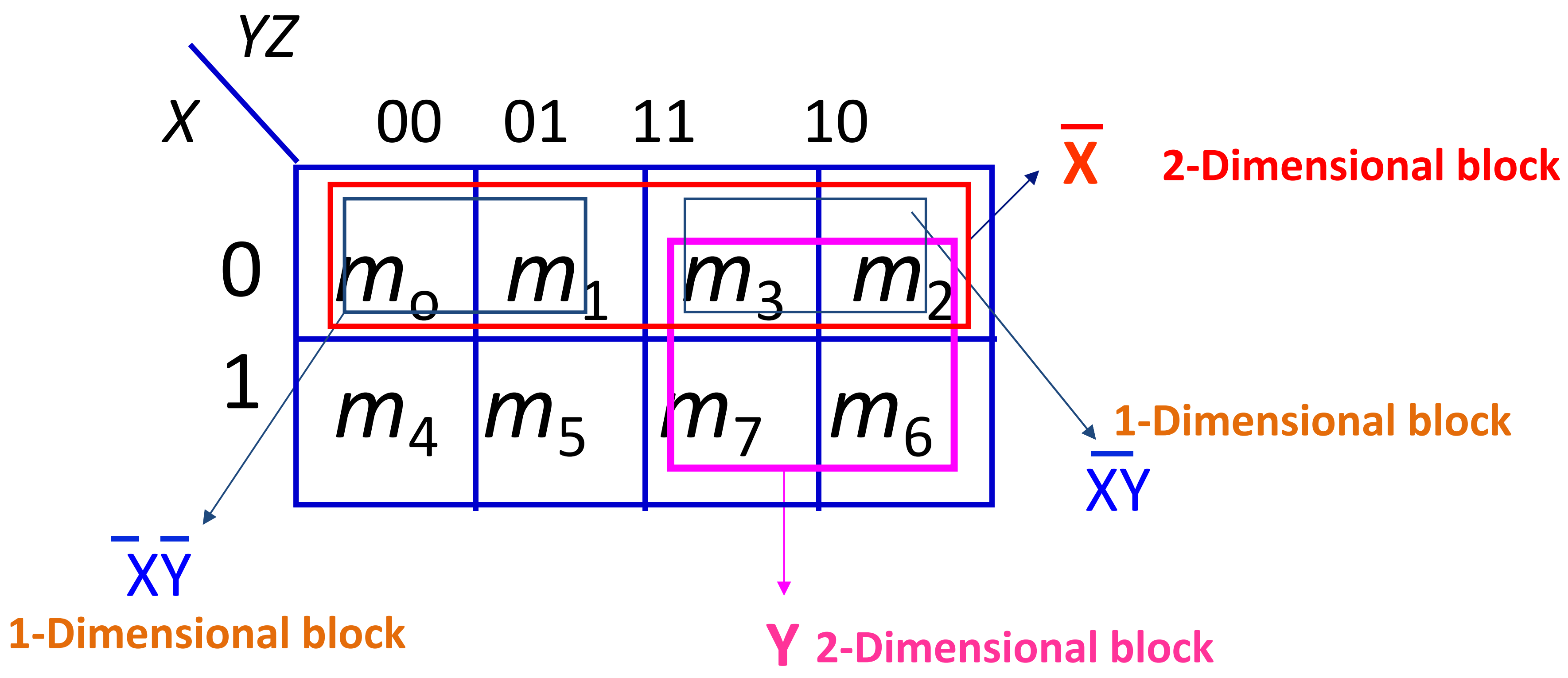
Graphical simple principle

- **Eliminating variables**
  - **According to Equation: $XXXB+XXX\overline{B}=XXX$,**
    - **Two adjacent AND-terms (minterms) can be combined into one AND-term**
    - **This can be eliminated one variable in two AND-item.**
- **Implicant**
  - **All squares in the map corresponding maximum possible number of adjacent minterms is combined together into a simple "AND-items"**
  - **It is a rectangle with the number of squares a power of 2**
  - **known as the "Karnaugh Circle" and also known as “Dimensional block"**

Adjacent and Dimensional block

Implicant

Three adjacent: connecting, symmetrical and overlap



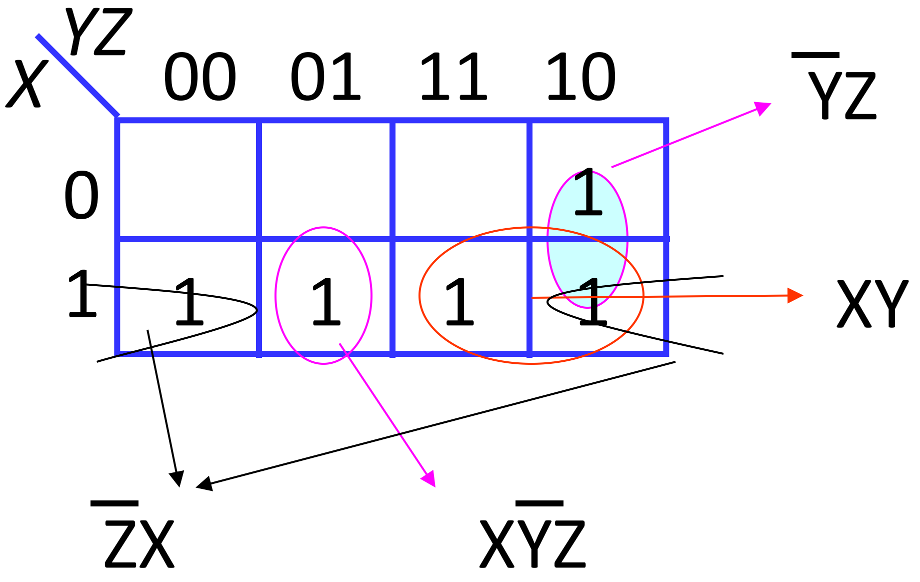
1-dimensional block can eliminate one variable,

and N-dimensional elimination of N variables

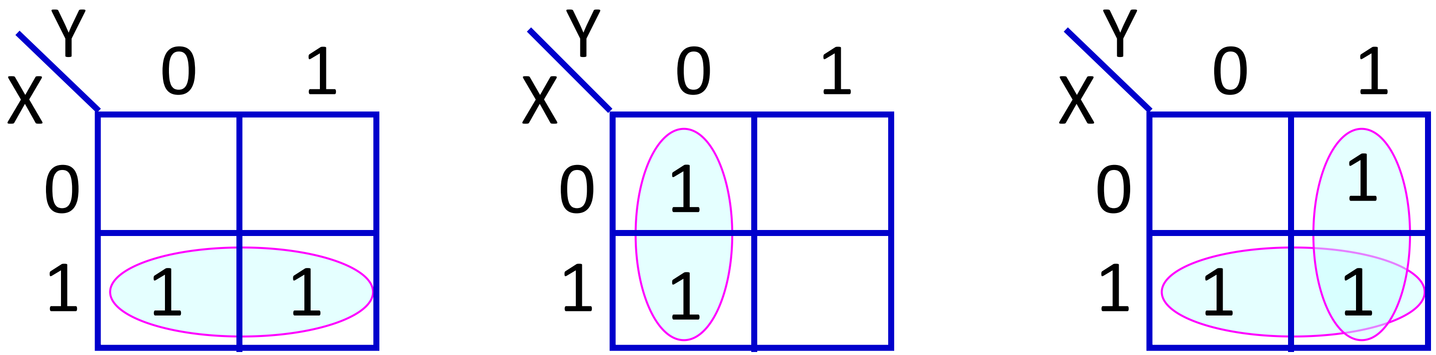
logic function representation with K-map

K-Map can be used to simplify Boolean functions.   Terms are selected to cover the"1s"in the map.

Example: Simplify $F(X,Y,Z)=\overline{Z}X+\overline{Z}Y+XY+X\overline{Y}Z$



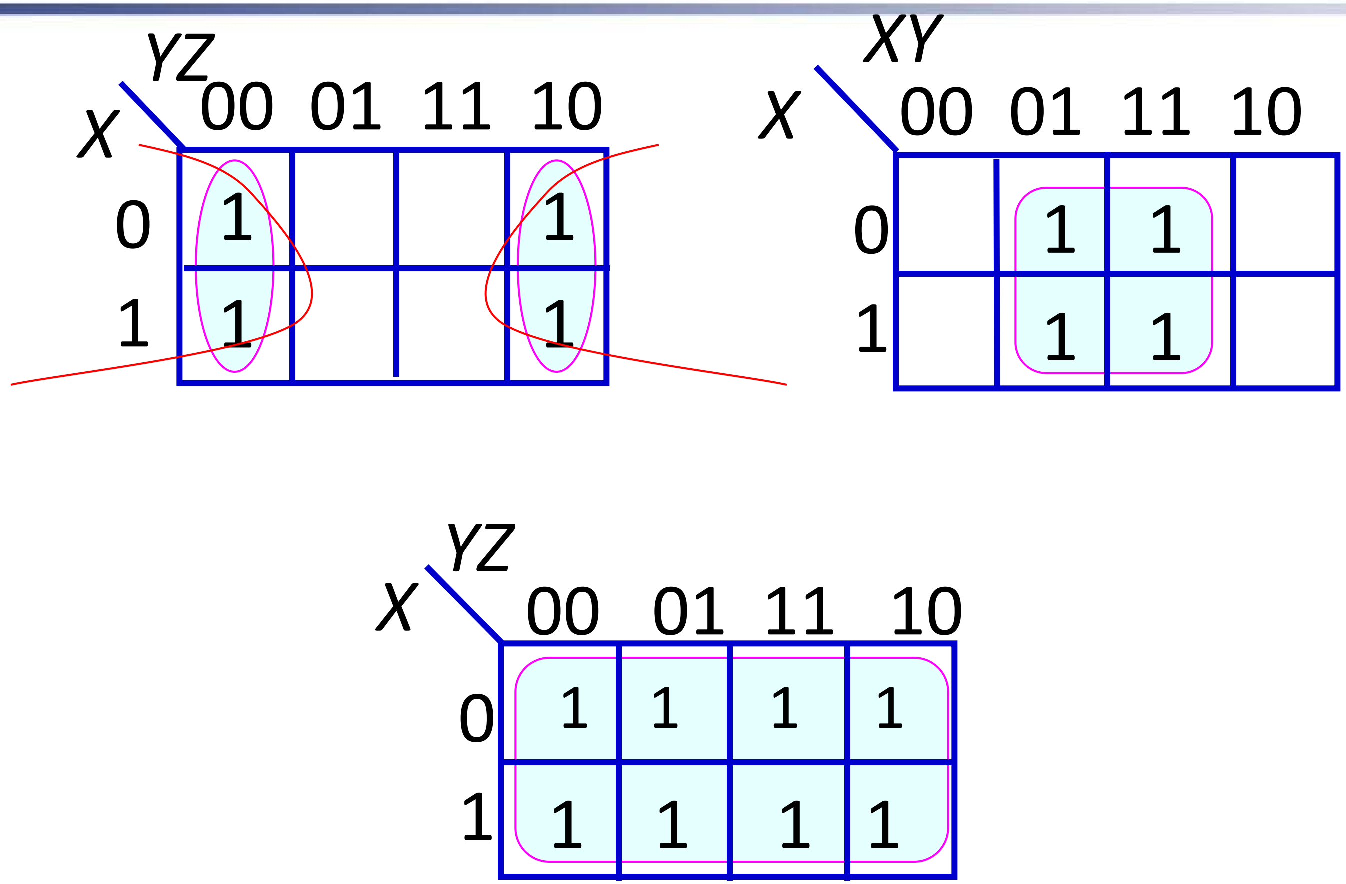
The merger of the two variable K-map



$\bar{Y}+Y=1$ $X+\bar{X}=1$ $(Y+\bar{Y})+(X+\bar{X})=1$

Typical two variable K-map merger

The merger of the three variable K-map



Typical three variable K-map merger

The merger of the four variable K-map

XY \ ZW	00	01	11	10
00	1			1
01		1	1	
11		1	1	
10	1			1

XY \ ZW	00	01	11	10
00		1	1	
01		1	1	
11	1	1	1	1
10		1	1	

XY \ ZW	00	01	11	10
00		1	1	
01	1			1
11	1			1
10		1	1	

Typical four variable K-map merger

A prime implicant

- **implicant： "AND- term” contains all Minterms"**
  - Dimensional Block constitute by the "AND-term" in the K-map (fill 1 grid rectangle)
- **A prime implicant**
  - is a product term obtained by combining the maximum possible number of adjacent squares in the map into a rectangle with the number of squares a power of 2.
- **Essential Prime Implicant**
  - it is the **only** prime implicant that covers (includes) one or more minterms.

e.g.: $F=\overline{A}D+B\overline{D}$

Essential
- **prime implicant： $\overline{A}B$**
- **prime implicant： $B\overline{D}$**
- **prime implicant： $\overline{A}D$**

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

K-Map Simplification

Step 1: write a function minterm;

- **Minterm expression**
- **Truth table**
- **Use AND-items directly filling K-map**

Step 2: make the function K-map ;

Step 3: Circle the largest Dimension-block of the function in K-map **Prime implicant**

- **Be sure to include one or more than one minterm, minterm can be used repeatedly (as much as possible to avoid)**
- **Must be sure to include the function all minterms (avoid missing grid).**

Essential Prime Implicant

Step 4: According to dimension-block simplification rules, elimination of redundant variables.

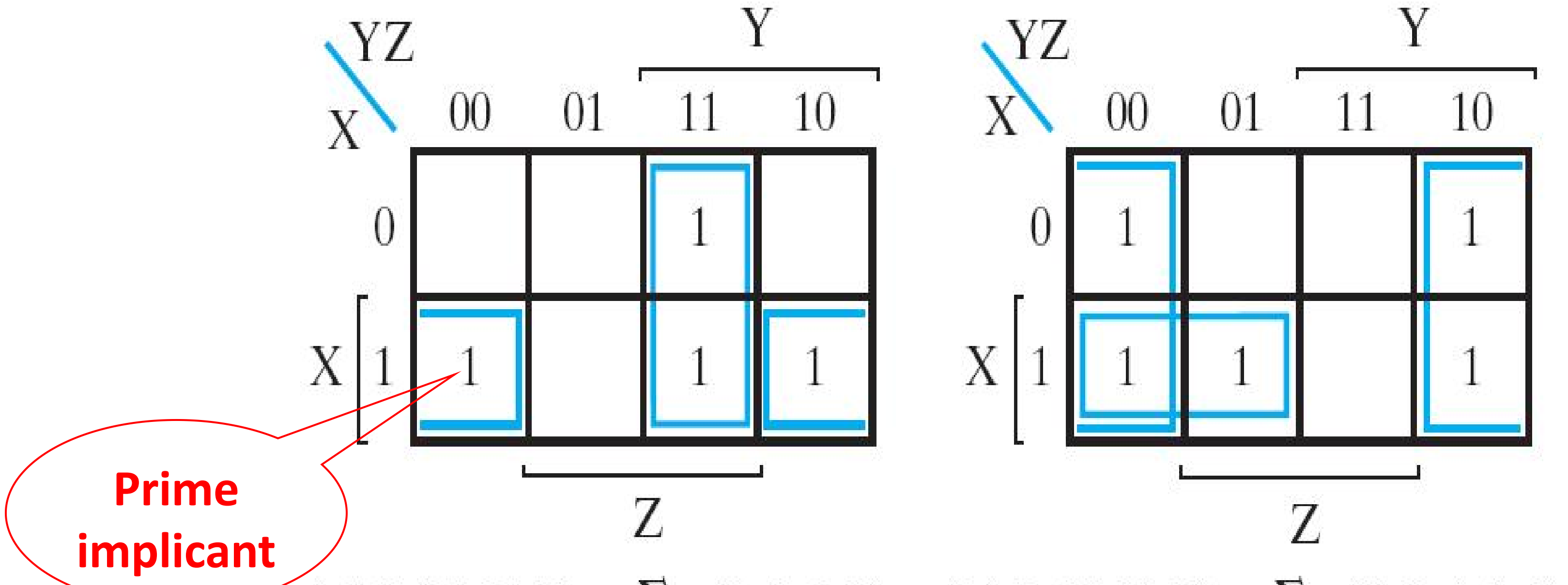
Simplification instance with K-map

Example 1： $F_1(X,Y,Z)=\sum m(3,4,6,7)$

$F_2(X,Y,Z)=\sum m(0,2,4,5,6)$

$$F_1(X,Y,Z)=m_3+m_4+m_6+m_7=\sum m(3,4,6,7)$$

$$F_2(X,Y,Z)=m_0+m_2+m_4+m_5+m_6=\sum m(0,2,4,5,6)$$



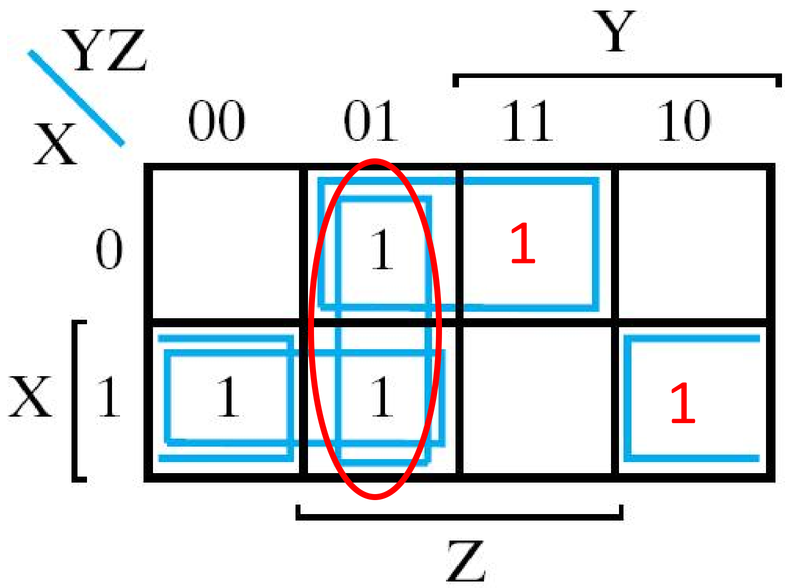
(a) $F_1(X,Y,Z)=\sum m(3,4,6,7)$
$= YZ + X\overline{Z}$

(b) $F_2(X,Y,Z)=\sum m(0,2,4,5,6)$
$= \overline{Z} + X\overline{Y}$

Example 2：

$$F_1 = F(X,Y,Z) = \sum m(1,3,4,5,6)$$

$$F_2 = F(X,Y,Z) = \sum m(1,2,3,5,7)$$



$$F(X,Y,Z) = \sum m(1,3,4,5,6)$$
$$= \overline{X}Z + X\overline{Y} + X\overline{Z}$$
$$= \overline{X}Z + \overline{Y}Z + X\overline{Z}$$

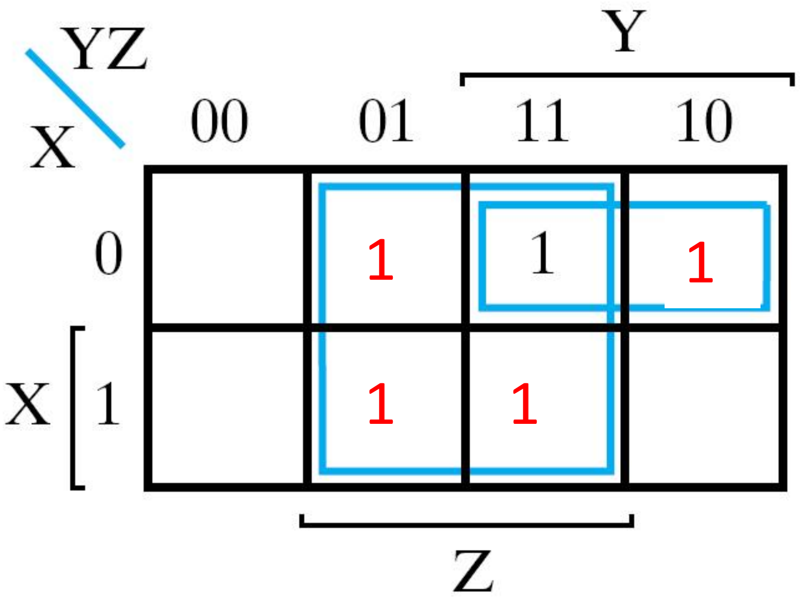


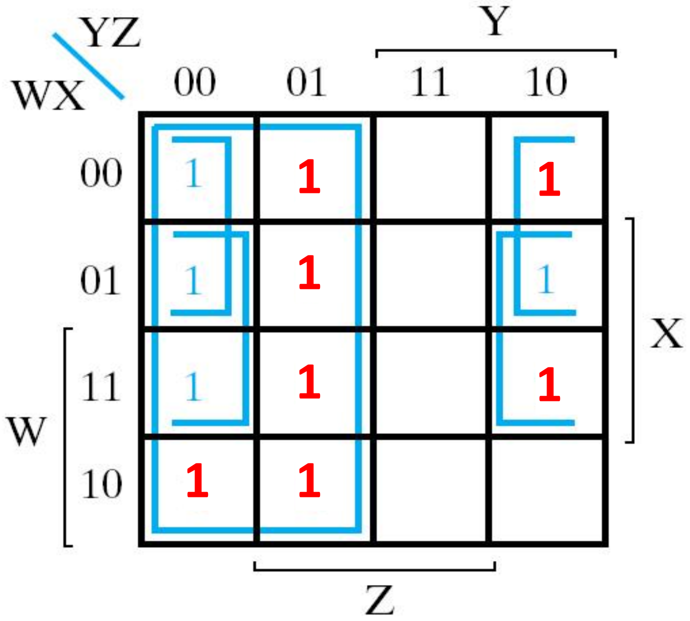
图2-16

$$F(X,Y,Z) = \sum m(1,2,3,5,7)$$
$$= Z + \overline{X}Y$$

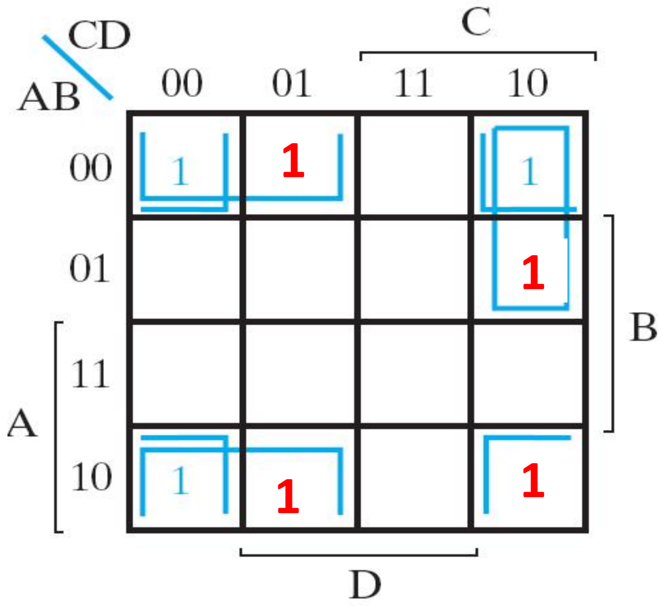
Example 3：

$$F(w,X,Y,Z)=\sum m(0,1,2,4,5,6,8,9,12,13,14)$$

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



F(W,X,Y,Z) K-map



F(A,B,C,D)K-map

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

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

Maxterm simplification with K-map

Method 1 for Maximum simplification：

Identifies the maxterm dimensional-block

Simplification SOP with Maxterm

$$F\left(A,B,C,D\right)=\sum m\left(0,1,2,5,8,9,10\right)$$

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(A,B,C,D) Maxterm K-map

$$\overline{F}=AB+CD+B\overline{D}$$

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

Method 2 for Maximum simplification：Complement/Inverse Function

Simplification POS: $F=\left(\overline{A}+\overline{B}+C\right)\left(B+D\right)$

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

Negated function：

DC \ BA	00	01	11	10
00	0	0	0	1
01	0	0	1	1
11	1	1	1	1
10	1	1	0	1

Merger "1"： $F=\overline{A}B+\overline{B}D+BC$

Merger "0"： $\overline{F}=\overline{B}\,\overline{D}+AB\overline{C}$

Result: $F=\left(\overline{A}+\overline{B}+C\right)\left(B+D\right)$

It has been the simplest.

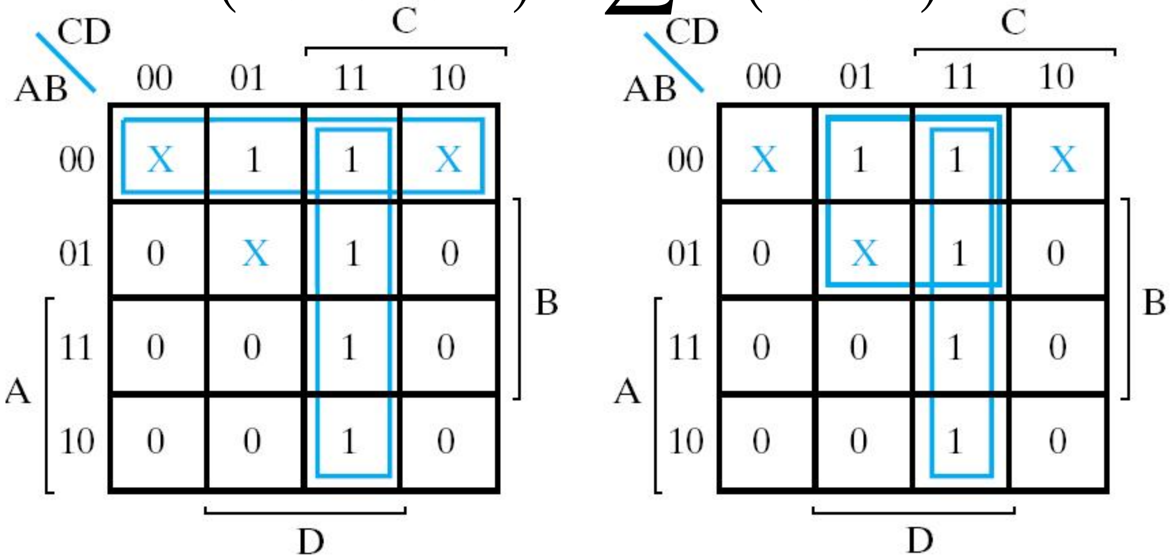
Incompletely Specified Function

- Don't-care terms： Uncertainty minterm in the function
  - Choose the don’t-care to be 1 or 0
- Two case： 1) Don’t appear in the input combinations;
  2) for some combination of inputs, there is an uncertain output
- using "×" or " Φ " indicates don't-care" in K-map.
- may be obtained more simply function expression, reasonably use of unrelated-terms in the K-map, "don't-care" only used to simplify.

Simplification with "Don't-care terms"

例：简化函数：$F\left(A,B,C,D\right)=\sum m\left(1,3,7,11,15\right)$

$$d\left(A,B,C,D\right)=\sum m\left(0,2,5\right)$$



(a) $F = CD + \overline{A}\,\overline{B}$ (b) $F = CD + \overline{A}D$

Using of unrelated conditions simplification function

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

Multi-Level Circuit Optimization*

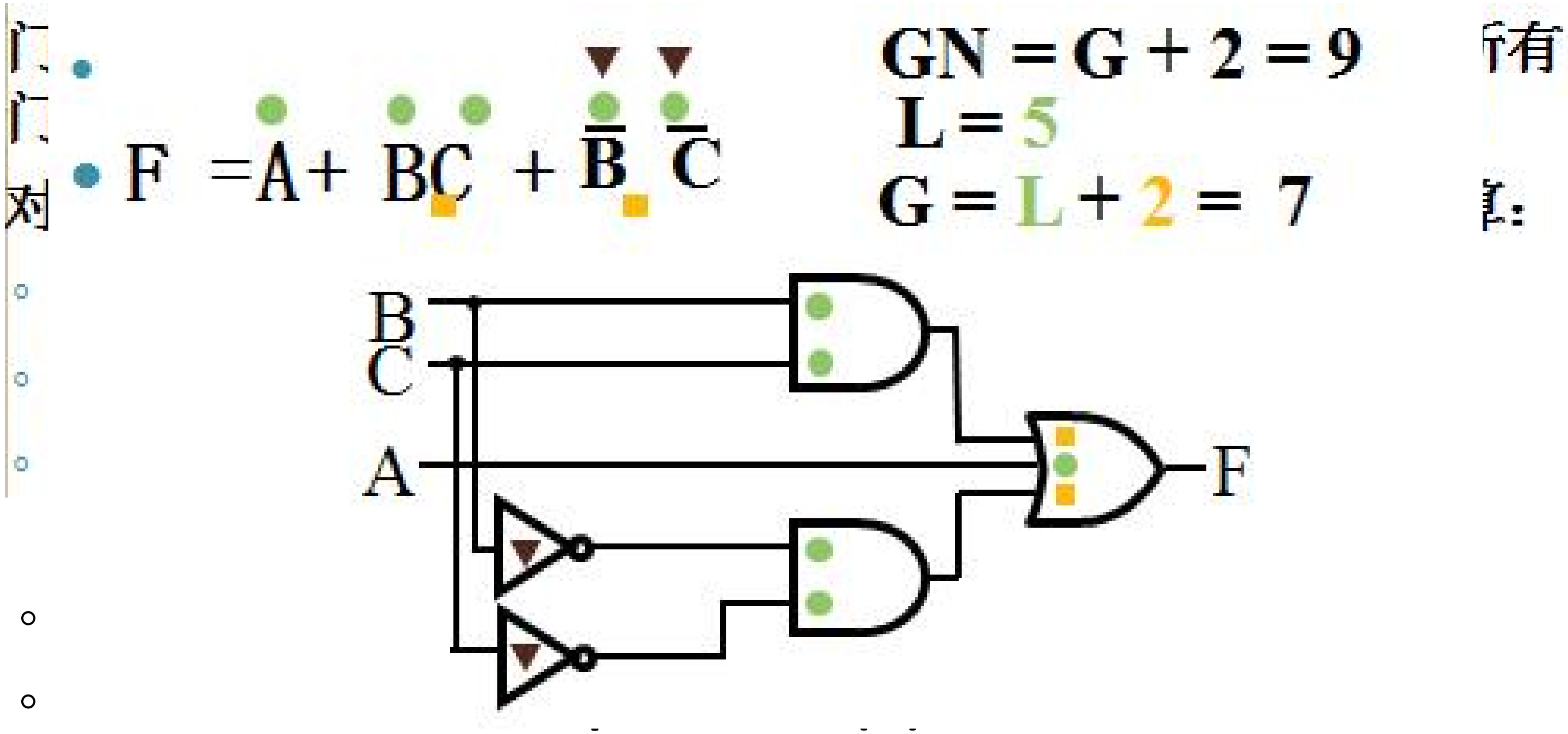
- **Multiple-level circuits**
  - **circuits that are not two-level (with or without input and/or output inverters)**
- **Multiple-level circuits can have reduced gate input cost compared to two-level (SOP and POS) circuits**
- **Multiple-level optimization is performed by applying transformations to circuits represented by equations while evaluating cost**

Transformations

- **Factoring** - finding a **factored form** from SOP or POS expression
  - Algebraic - No use of axioms specific to Boolean algebra such as complements or idempotence
  - Boolean - Uses axioms unique to Boolean algebra
- **Decomposition** - expression of a function as a set of new functions

Transformations (continued)

- **Substitution** of G into F - expression function F as a function of G and some or all of its original variables
- **Elimination** - Inverse of substitution
- **Extraction** - decomposition applied to multiple functions simultaneously



- 带非门的门输入成本（Gate input cost with NOTs） (GN)

Transformation Example 1

- **Algebraic Factoring**

  $F = \overline{A}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C} + ABC + AC\overline{D}$ $\quad G = 16$

  - **Factoring:**

  $F = \overline{A}(\overline{C}\,\overline{D} + B\overline{C}) + A(BC + C\overline{D})$ $\quad G = 16$

  - **Factoring again:**

  $F = \overline{A}\,\overline{C}(B + \overline{D}) + AC(B + \overline{D})$ $\quad G = 12$

  - **Factoring again:**

  $F = (\overline{A}\,\overline{C} + AC)(B + \overline{D})$ $\quad G = 10$

Transformation Example 2

$$G = ABC + ABD + E + ACF + ADF \qquad (a)$$

$$G = AB(C + D) + E + AF(C + D) \qquad (b)$$

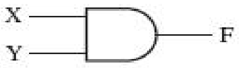
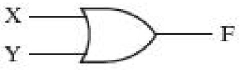
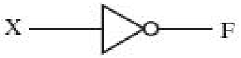
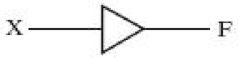
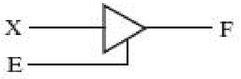
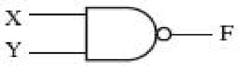
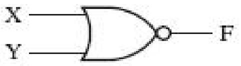
$$G = (AB + AF)(C + D) + E \qquad (c)$$

$$G = A(B + F)(C + D) + E \qquad (d)$$

Other Gate Types: Primitive

1. Primitive Digital Logic Gates

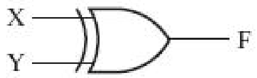
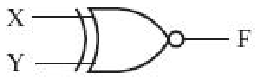
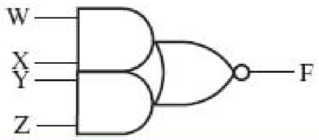
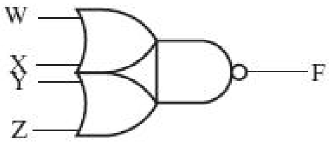
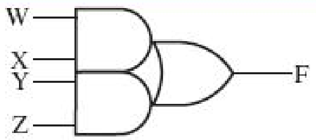
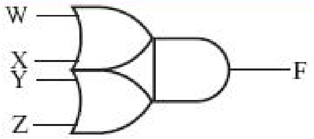
- **Primitive Logic Gates has the most simple, the most rapid function.**

Name	Distinctive shape	Algebraic equation	Truth table
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
NOT (inverter)		$F = \overline{X}$	X | F 0 | 1 1 | 0
Buffer		$F = X$	X | F 0 | 0 1 | 1
3-State Buffer			E X | F 0 0 | Hi-Z 0 1 | Hi-Z 1 0 | 0 1 1 | 1
NAND		$F = \overline{X \cdot Y}$	X Y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{X + Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0

Other Gate Types: Complex

2.Complex Digital Logic Gates

- the number of transistors needed is fewer than required by connecting together primitive gates
- potentially, the circuit delay is smaller, increasing the circuit operating speed

Name	Distinctive shape symbol	Algebraic equation	Truth table
Exclusive–OR (XOR)		$F = X\overline{Y} + \overline{X}Y$ $= X \oplus Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive–NOR (XNOR)		$F = XY + \overline{X}\,\overline{Y}$ $= \overline{X \oplus Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1
AND-OR-INVERT (AOI)		$F = \overline{WX + YZ}$	
OR-AND -INVERT (OAI)		$F = \overline{(W + X)(Y + Z)}$	
AND-OR (AO)		$F = WX + YZ$	
OR-AND (OA)		$F = (W + X)(Y + Z)$	

Hi-Impedance Outputs

- **Logic gates introduced thus far**
  - have 1 and 0 output values,
  - cannot have their outputs connected together, and
  - transmit signals on connections in only one direction.
- **Three-state logic adds a third logic value**
  - Hi-Impedance (Hi-Z), giving three states: **0, 1**, and Hi-Z on the outputs.
- **The presence of a Hi-Z state makes a gate output as described above behave quite differently:**
  - “1 and 0” become “1, 0, and Hi-Z”
  - “cannot” becomes “can,” and
  - “only one” becomes “two”

Hi-Impedance Outputs (continued)

- **What is a Hi-Z value?**
  - The Hi-Z value behaves as an open circuit
  - This means that, looking back into the circuit, the output appears to be disconnected.
  - It is as if a switch between the internal circuitry and the output has been opened.
- **Hi-Z may appear on the output of any gate, but we restrict gates to:**
  - a 3-state buffer, or
  - Optional: a transmission gate (See Reading Supplement: More on CMOS Circuit-Level Design),

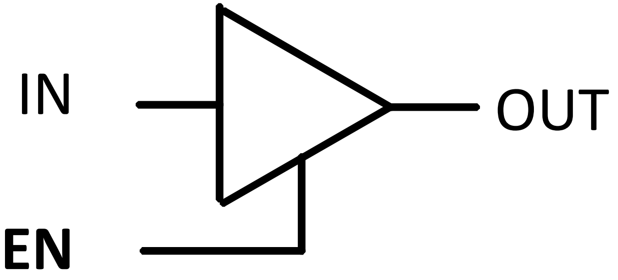
each of which has one data input and one control input.

The 3-State Buffer

- **For the symbol and truth table, IN is the data input, and EN, the control input.**
- **For EN = 0, regardless of the value on IN (denoted by X), the output value is Hi-Z.**
- **For EN = 1, the output value follows the input value.**
- **Variations:**
  - Data input, IN, can be inverted
  - Control input, EN, can be inverted

  by addition of "bubbles" to signals.

Symbol



Truth Table

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

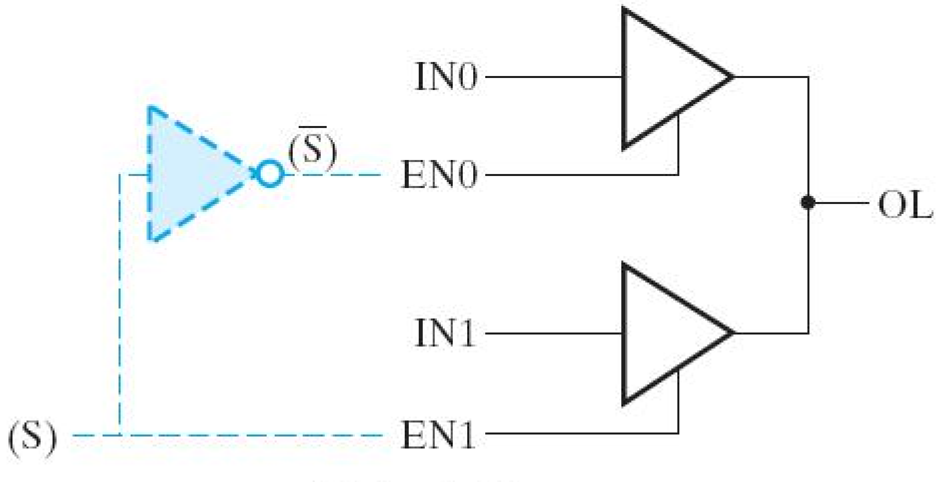
Resolving 3-State Values on a Connection

- **Connection of two** 3-state buffer outputs, B1 and B0, to a wire, OUT
- Assumption: Buffer data inputs can take on any combination of values 0 and 1
- **Resulting Rule**: At **least one** buffer output value must **be Hi-Z**. Why?
- How many valid buffer output combinations exist?
- What is the rule for *n* 3-state buffers connected to wire, OUT?
- How many valid buffer output combinations exist?

Resolution Table		
B1	B0	OUT
0	Hi-Z	0
1	Hi-Z	1
Hi-Z	0	0
Hi-Z	1	1
Hi-Z	Hi-Z	Hi-Z

Data Selection Function with 3-state buffers

- **Data Selection Function:**
  - If s = 0, OL = IN0, else OL = IN1
- **Performing data selection with 3-state buffers:**



(a) Logic Diagram

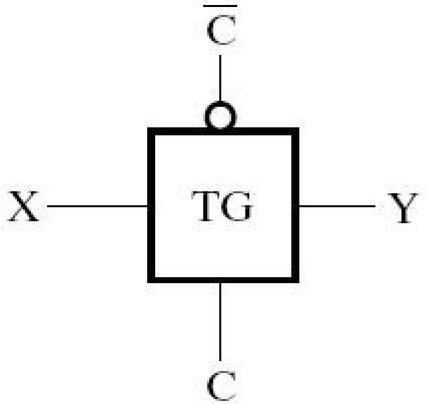
EN1	EN0	IN1	IN0	OL
0	0	X	X	Hi-Z
(S) 0	$(\overline{S})$ 1	X	0	0
0	1	X	1	1
1	0	0	X	0
1	0	1	X	1
1	1	0	0	0
1	1	1	1	1
1	1	0	1	
1	1	1	0	

(b) Truth table

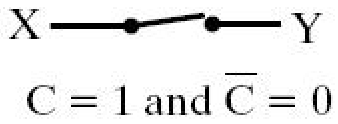
- **Since EN0 = $\overline{S}$ and EN1 = S, one of the two buffer outputs is always Hi-Z plus the last row of the table never occurs.**

Transmission 3-state gate

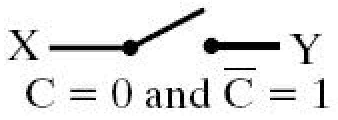
Transmission gate（TG）



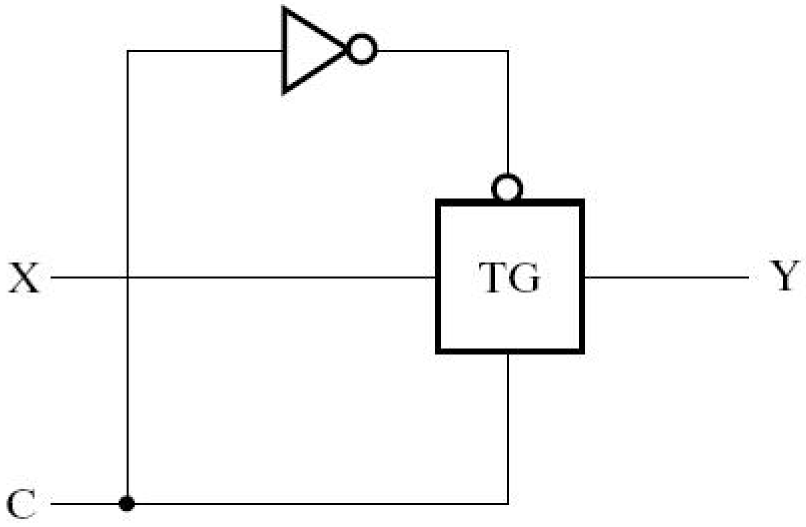
(a)



(b)

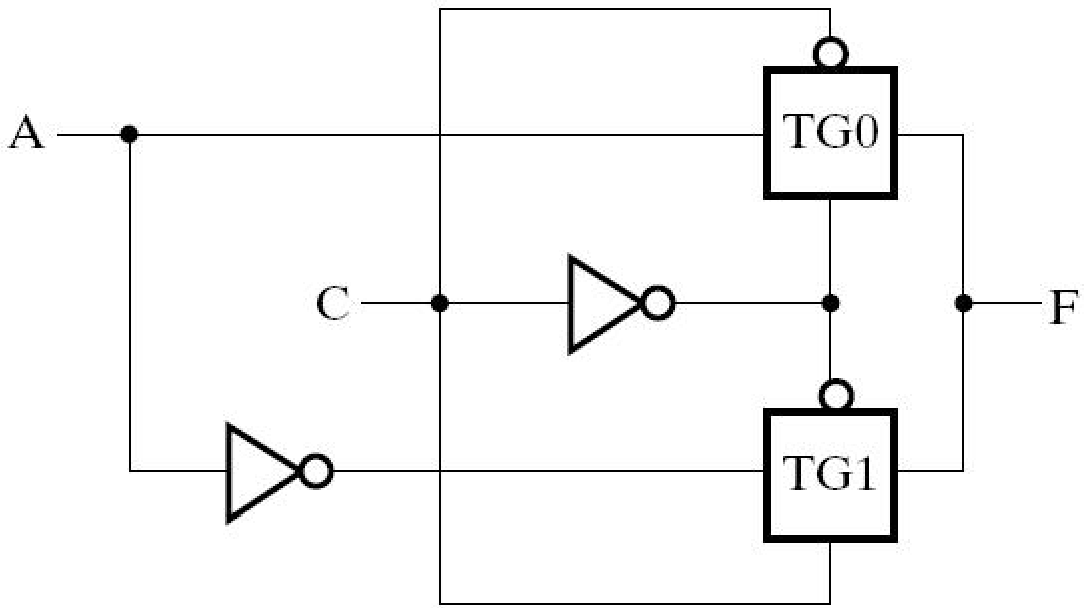


(c)



(d)

XOR Function with transmission gate



(a)

A	C	TG1	TG0	F
0	0	No path	Path	0
0	1	Path	No path	1
1	0	No path	Path	1
1	1	Path	No path	0

(b)

See Assignment Section of Website

Thank You！