数字逻辑设计

王鸿鹏

计算机科学与技术学院

wanghp@hit.edu.cn

逻辑

- “事物发展有其内在的逻辑”
- “这个人为人处世，有他自己的逻辑”
- “说话、写文章都要讲逻辑”
- “按照对方辩友的逻辑，岂不是说 ……”
- 帝国主义者的逻辑和人民的逻辑是这样的不同。捣乱，失败，再捣乱，再失败，直至灭亡——这就是帝国主义和世界上一切反动派对待人民事业的逻辑，...斗争，失败，再斗争，再失败，再斗争，直至胜利——这就是人民的逻辑。

（毛泽东选集第四卷——丢掉幻想，准备斗争）

逻辑学

- 我们思考问题，分析问题，要从逻辑思维的高度进行，要独立思考判断，不能只是停留在感官认知层面以及习惯性接收和反应客观信息上。要跳出本能的习惯性反应，发挥大脑的逻辑思考能力，去思辨。这要求我们学好逻辑学。

- https://baike.baidu.com/item/逻辑学/85559

布尔代数的形式定义

- 布尔代数是一个集合A，其上定义了以下结构：
  - 二元运算∧: A×A→A （与、合取）
  - 二元运算∨: A×A→A （或、析取）
  - 一元运算 ': A→A （非、否定）
  - 零元运算（常数）0和1。
- 这些运算满足以下条件：∀a，b，c∈A

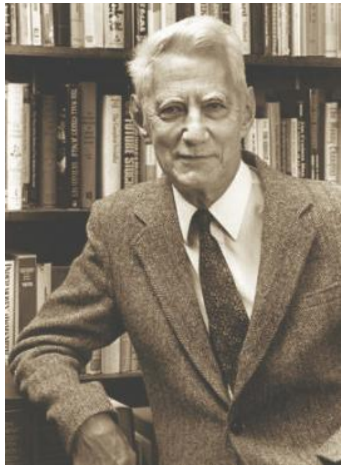
一般要求布尔集至少有两个不同的元素0和1，而且其元素对三种运算∧，∨，′都封闭

$a \vee (b \vee c) = (a \vee b) \vee c$	$a \wedge (b \wedge c) = (a \wedge b) \wedge c$	结合律
$a \vee b = b \vee a$	$a \wedge b = b \wedge a$	交换律
$a \vee (a \wedge b) = a$	$a \wedge (a \vee b) = a$	吸收律
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	分配律
$a \vee \neg a = 1$	$a \wedge \neg a = 0$	互补律

布尔代数——>开关代数/逻辑代数

George Boole

- 1854年，George Boole发表了《思维规律》，布尔代数问世了，数学史上树起了一座新的里程碑。
- 布尔代数不仅可以在数学领域内实现集合运算，更广泛应用于电子学、计算机硬件、计算机软件等领域的逻辑运算：当集合内只包含两个元素（1和0）时，分别对应{真}和{假}，可以用于实现对逻辑的判断。
- 1938年，就读于MIT的香农发表了他那篇著名的硕士论文《继电器与开关电路的符号分析》，奠定了数字电路的理论基础。



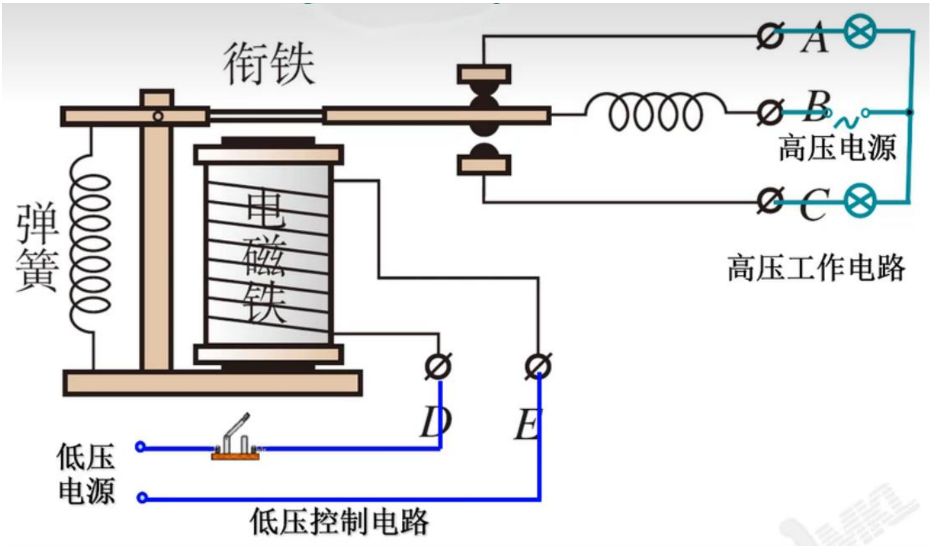
Claude E. Shannon

原文表述 《继电器与开关电路的符号分析》

- In the control and protective circuits of complex electrical systems it is frequently necessary to make intricate interconnections of relay contacts and switches. Examples of these circuits occur in automatic telephone exchanges, industrial motor-control equipment, and in almost any circuits designed to perform complex operations automatically. In this paper a **mathematical analysis** of certain of the properties of such networks will be made. Particular attention will be given to the problem of **network synthesis**. Given certain characteristics, it is required to find a circuit incorporating these characteristics. The solution of this type of problem is not unique and methods of finding those particular circuits requiring the least number of relay contacts and switch blades will be studied. Methods will also be described for **finding any number of circuits equivalent to a given circuit** in all operating characteristics. It will be shown that several of the well-known theorems on impedance networks have roughly analogous theorems in relay circuits. Notable among these are the delta-wye and star-mesh transformations, and the duality theorem.

背景与意义 《继电器与开关电路的符号分析》

- 在复杂电气系统的控制和保护电路中，经常要进行**继电器**触点和开关的复杂互连。
  - 自动电话交换机
  - 工业电机控制设备
  - 几乎所有自动执行复杂操作的电路。



- 研究需要最少数量的继电器触点和开关的特定电路的寻找方法。（电路简化）
- 研究与给定电路具有相同特性的任意电路的寻找方法。（电路等价）

电话交换机

原文表述 《继电器与开关电路的符号分析》

- The method of attack on these problems may be described briefly as follows: **any circuit is represented by a set of equations**(the terms of the equations corresponding to the various relays and switches in the circuit). A calculus is developed for manipulating these equations by simple mathematical processes, most of which **are similar to ordinary algebraic algorisms**. This calculus is shown to be exactly **analogous to the calculus of propositions used in the symbolic study of logic**. For the synthesis problem the desired characteristics are first written as a system of equations, and the equations are then manipulated into the form representing the simplest circuit. The circuit may then be immediately drawn from the equations. By this method it is always possible to find the simplest circuit containing only series and parallel connections, and in some cases the simplest circuit containing any type of connection.
- Our notation is taken chiefly from symbolic logic. Of the many systems in common use we have chosen the one which seems simplest and most suggestive for our interpretation. Some of our phraseology, such as node, mesh, delta, wye, etc., is borrowed from ordinary network theory for simple concepts in switching circuits.

实现方式 《继电器与开关电路的符号分析》

- 任何电路都可以由一组方程表示，通过简单的数学过程来演算这些方程。
  - 电路中各种继电器和开关对应为方程项

- 电路综合(synthesis)问题的解决过程：

期望特性（需求） → 方程组 → 特定形式 → 提取电路

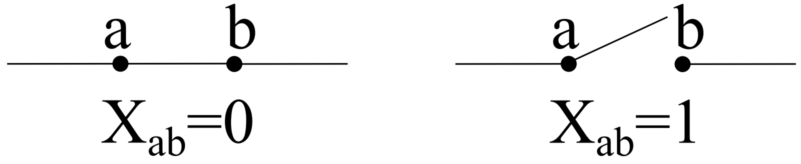
- 综合问题的解决结果：
  - 只包含串联和并联连接的最简单电路。
  - 包含任何类型连接的最简单电路。

Series-Parallel Two-Terminal Circuits Fundamental Definitions and Postulates

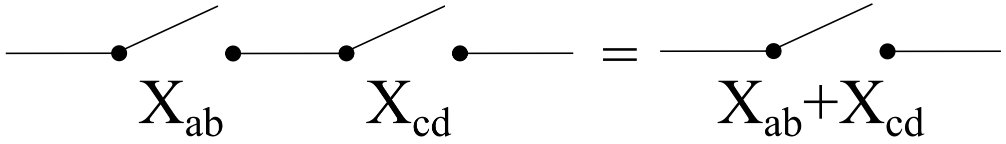
We shall limit our treatment of circuits containing only relay contacts and switches, and therefore at any given time the circuit between any two terminals must be either **open (infinite impedance)** or **closed (zero impedance)**. Let us associate a symbol X_{ab} or more simply X, with the terminals a and b. This variable, a function of time, will be called the hindrance of the two-terminal circuit a-b. The symbol 0(zero) will be used to represent the hindrance of a closed circuit, and the symbol 1 (unity) to represent the hindrance of an open circuit. Thus when the circuit a-b is open $X_{ab} = 1$ and when closed $X_{ab} = 0$. Two hindrances X_{ab} and X_{cd} will be said to be equal if whenever the circuit a-b is open, the circuit c-d is open, and whenever a-b is closed, c-d is closed. Now let the symbol +(plus) be defined to mean the series connection of the two-terminal circuits whose hindrances are added together. Thus $X_{ab} + X_{cd}$ is the hindrance of the circuit a-d when b and c are connected together. Similarly the product of two hindrances $X_{ab} \cdot X_{cd}$ or more briefly $X_{ab}X_{cd}$ will be defined to mean the hindrance of the circuit formed by connectingthe circuits a-b and c-d in parallel.

基本定义与假设《继电器与开关电路的符号分析》

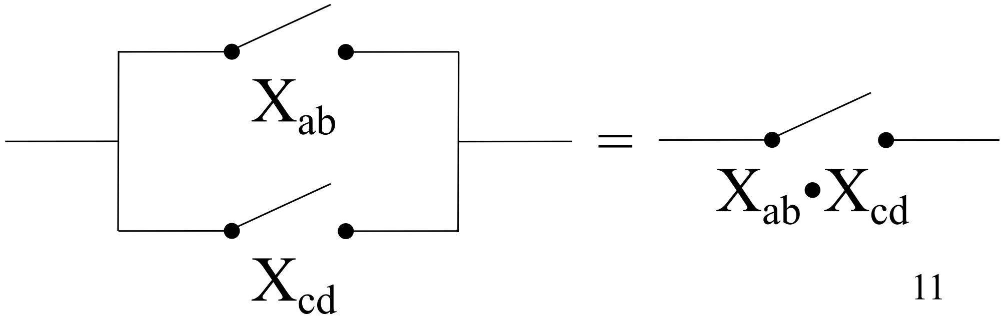
- 假设研究对象为只含有继电器触点和开关的电路。
- 电路只有两个状态：开，关
- X_{ab}：触点a和b之间的“阻抗”（Hindrance）
  - $X_{ab}=0$：闭合电路，阻抗为0
  - $X_{ab}=1$：断开电路，阻抗为无穷大



- 串联电路：$X_{ab}+X_{cd}$
  - 0+0=0，0+1=1，1+0=1，1+1=1



- 并联电路：$X_{ab}\bullet X_{cd}$
  - 0•0=0，0•1=0，1•0=0，1•1=1



布尔代数基础

- 逻辑运算
- 布尔表达式和真值表
- 逻辑代数定理及规则
- 代数化简法

各种逻辑运算

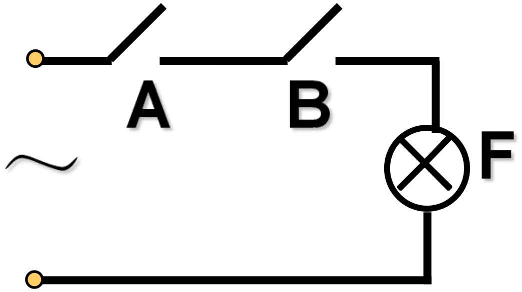
- **基本逻辑运算（Basic Operations）**
  - 与（AND）
  - 或（OR）
  - 非（NOT）
- **复合逻辑运算（Other Operations）**

基本运算——AND

1. AND（逻辑“与”/“乘”）

① $F = A \cdot B$

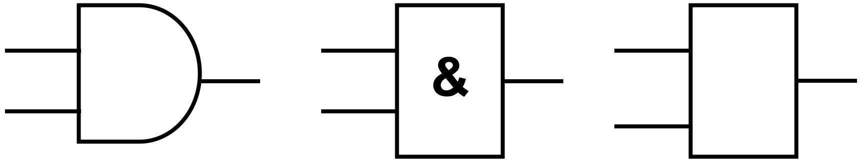
A/B=1:开关闭合
F=1 表示灯亮
注：此处与香农的假设不同



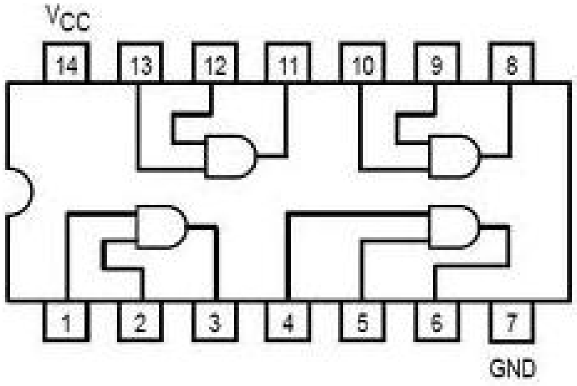
真值表

AB	F
0 0	0
0 1	0
1 0	0
1 1	1

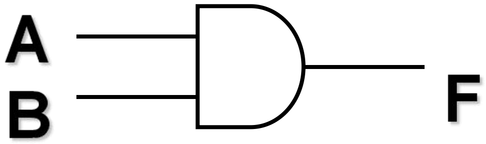
② AND gate（与门）

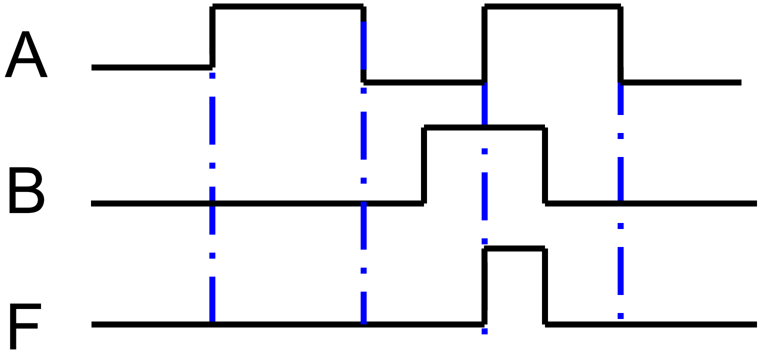


③ 常用芯片：74LS08



④逻辑符号

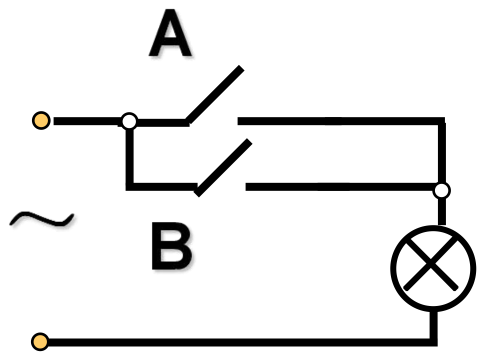




基本运算——OR

2. OR（逻辑“或”/“加”）

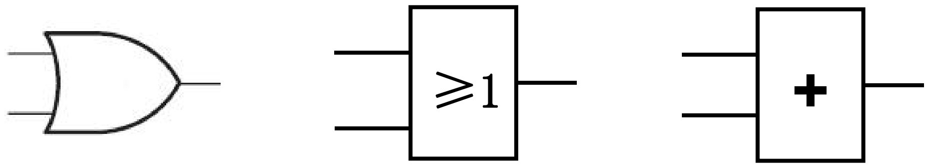
① $F = A+B$



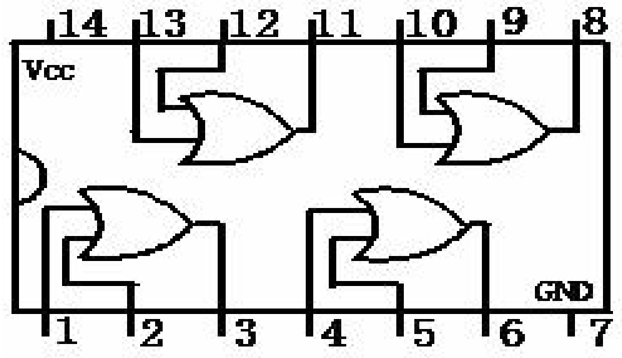
真值表

AB	F
0 0	0
0 1	1
1 0	1
1 1	1

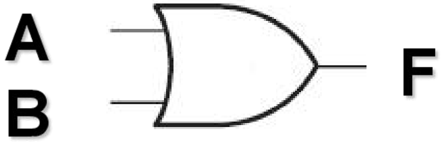
② OR gate（或门）

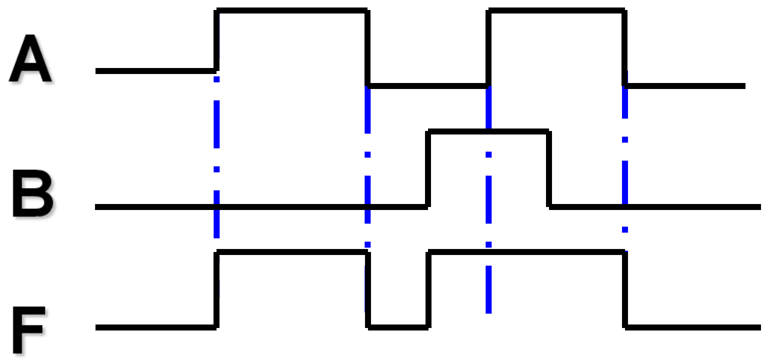


③ 常用芯片：74LS32



④

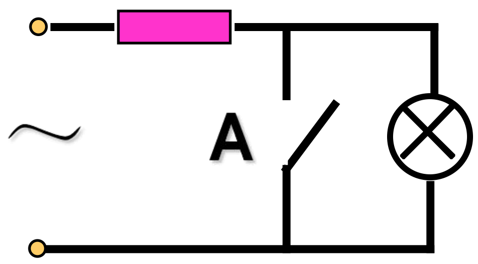




基本运算——NOT

3. NOT（逻辑“非”/反相器）

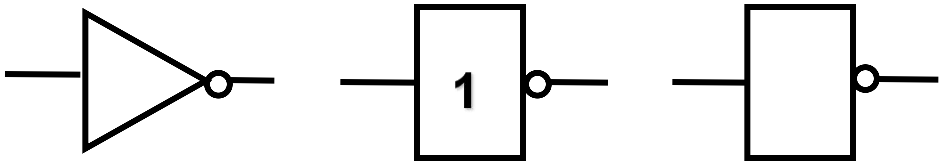
① $F=\overline{A}$ or $F=A'$



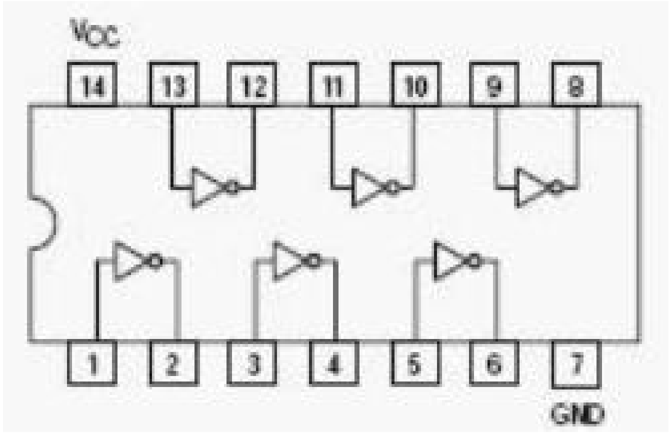
真值表

A	F
0	1
1	0

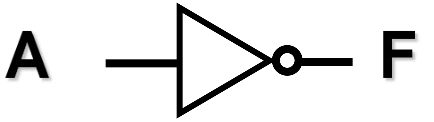
② NOT gate（非门）

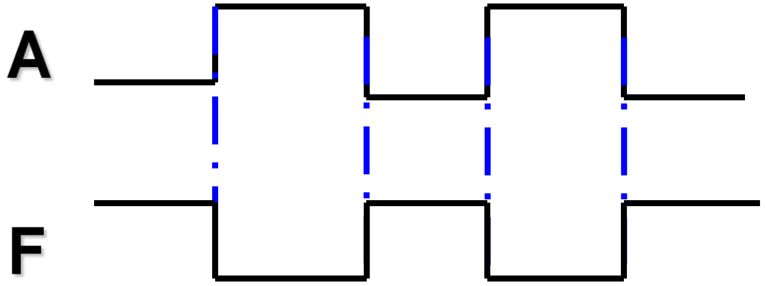


③ 常用芯片： **74LS04**



④





各种逻辑运算

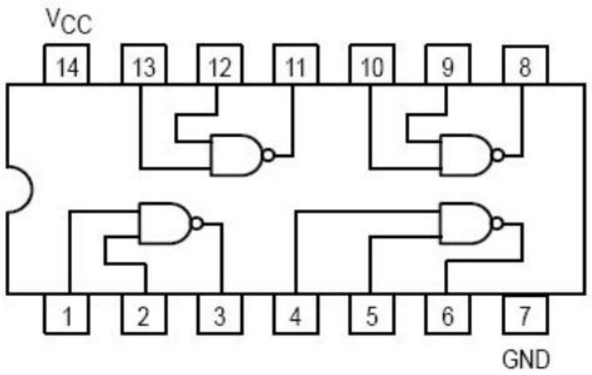
- 基本逻辑运算（**Basic Operations**）
  - 与（**AND**）
  - 或（**OR**）
  - 非（**NOT**）
- 复合逻辑运算（**Other Operations**）

复合逻辑运算——NAND

4. 与非

$$F=\overline{A \cdot B}$$

- 常用芯片： **74LS00**



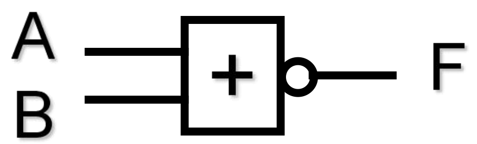
真值表

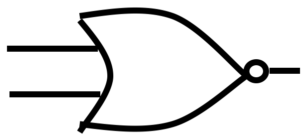
AB	F
0 0	1
0 1	1
1 0	1
1 1	0

复合逻辑运算——NOR

5. 或非（NOR）

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

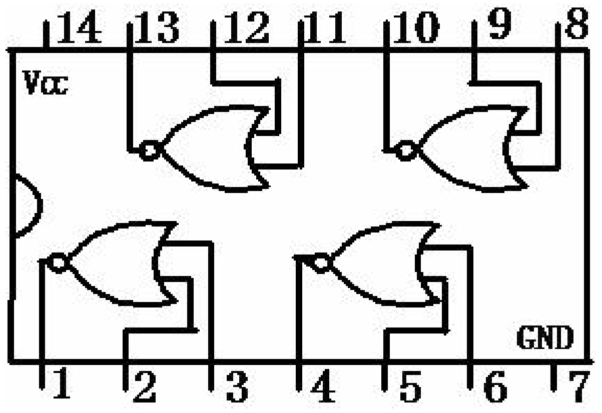




真值表

AB	F
0 0	1
0 1	0
1 0	0
1 1	0

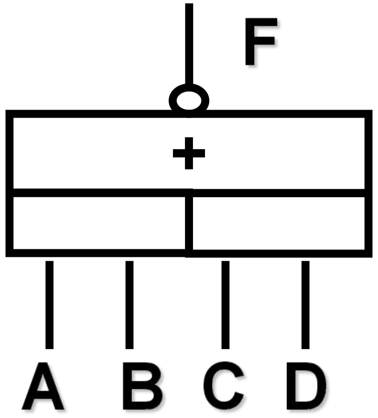
- 常用芯片： **74LS02**

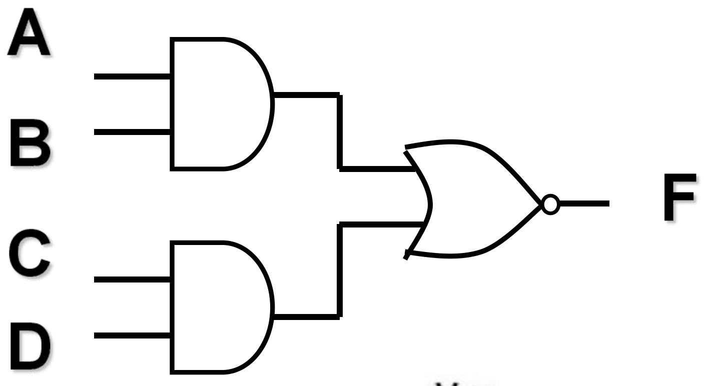


复合逻辑运算——NAND-OR-NOT

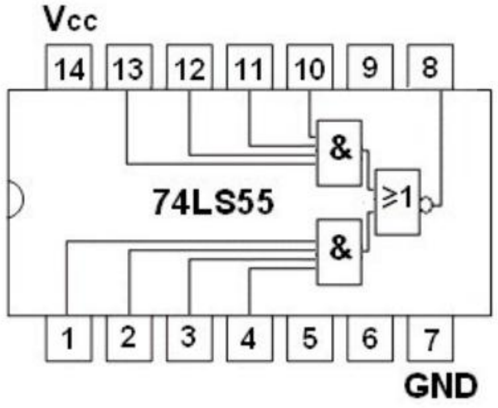
6. 与或非（AND-OR-NOT）

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





- 常用芯片： **74LS51,74LS55**



复合逻辑运算——异或（Exclusive-OR）

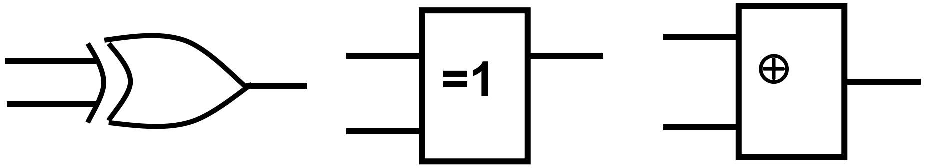
7. 异或 (Exclusive-OR)

① $F = A \oplus B = \bar{A}B + A\bar{B}$

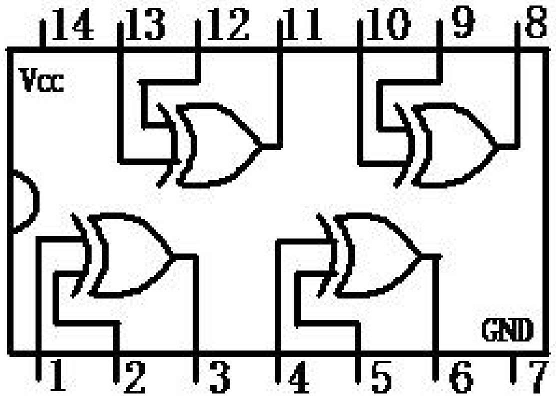
真值表

AB	F
0 0	0
0 1	1
1 0	1
1 1	0

② 逻辑符号



③ 常用芯片： 74LS86



复合逻辑运算——同或（Exclusive-NOR）

8. 同或 (Exclusive-NOR)

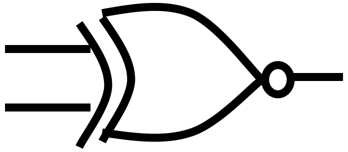
$F = A \equiv B$ or

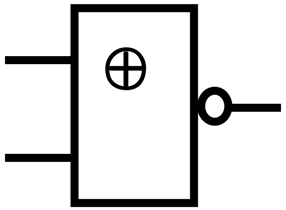
$F = A \odot B = \bar{A}\bar{B} + AB$

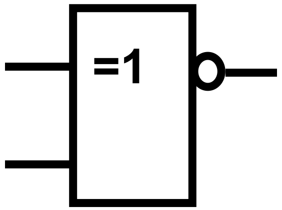
真值表

AB	F
0 0	1
0 1	0
1 0	0
1 1	1

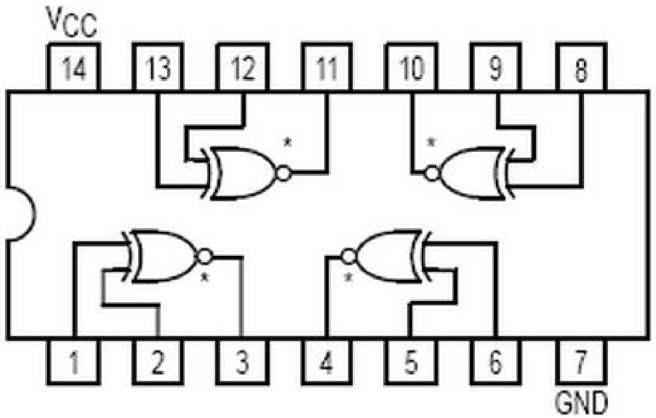
逻辑符号







常用芯片：74LS266



异或、同或的性质

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

$A \oplus 0 = A$

$A \oplus A = 0$

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

$A \odot 1 = A$

$A \odot 0 = \bar{A}$

$A \odot A = 1$

$A \odot \bar{A} = 0$

同或（Exclusive-NOR）的应用

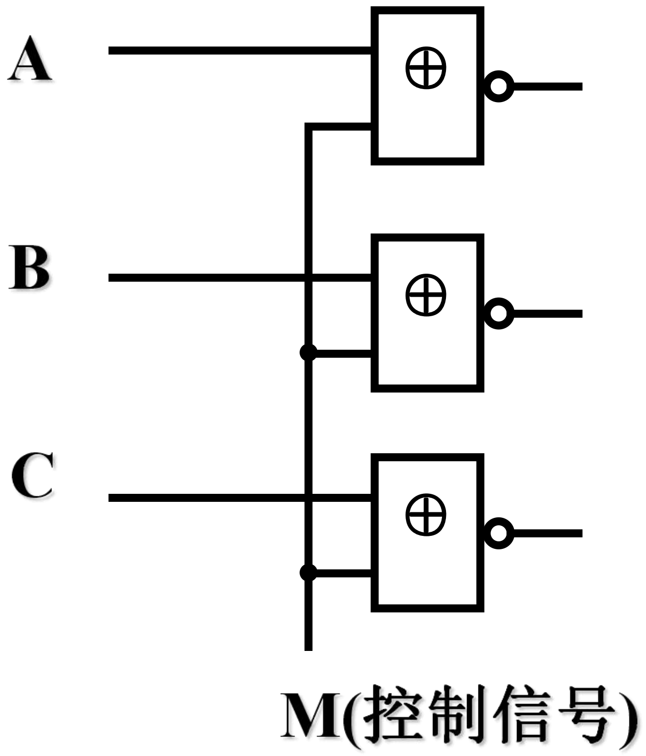
■根据控制信号不同输出原值或者原值的反

■M = 1，

输出为A，B，C

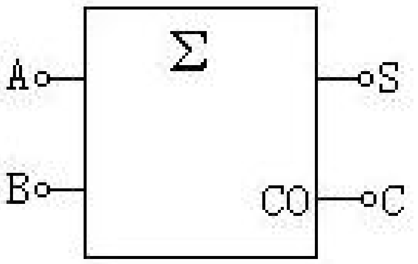
■M = 0，

输出为A反、B反、C反。



异或门的应用——加法器

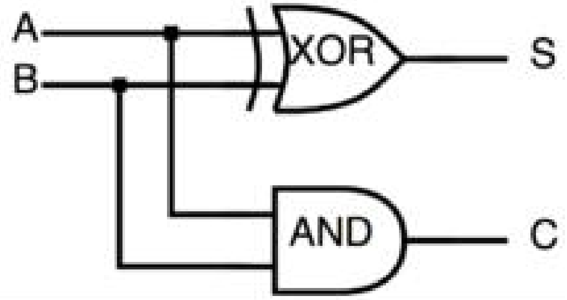
- 半加器（**Half-adder**）



半加器逻辑符号

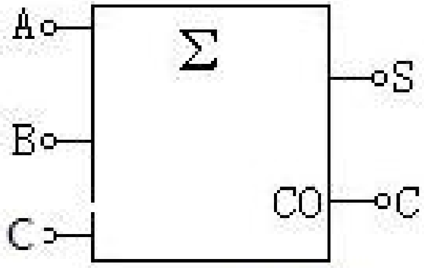
输入		输出	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

逻辑表达式：**S=A⊕B；C=A•B**



半加器的逻辑实现

- 全加器（**Full-adder**）

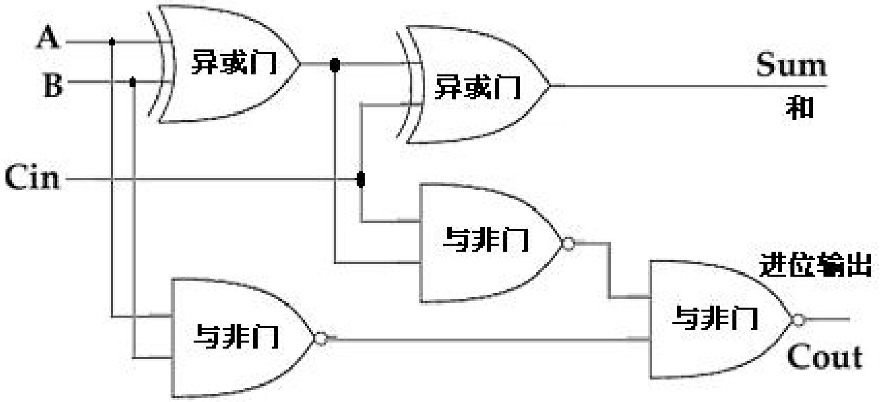


全加器逻辑符号

输入			输出	
C_{in}	A	B	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

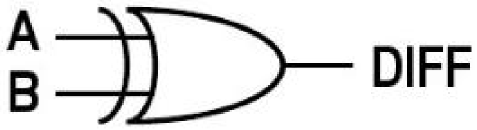
$$Sum = A \oplus B \oplus C_{in}$$

$$C_{out} = A \bullet B + C_{in} \bullet (A \oplus B)$$



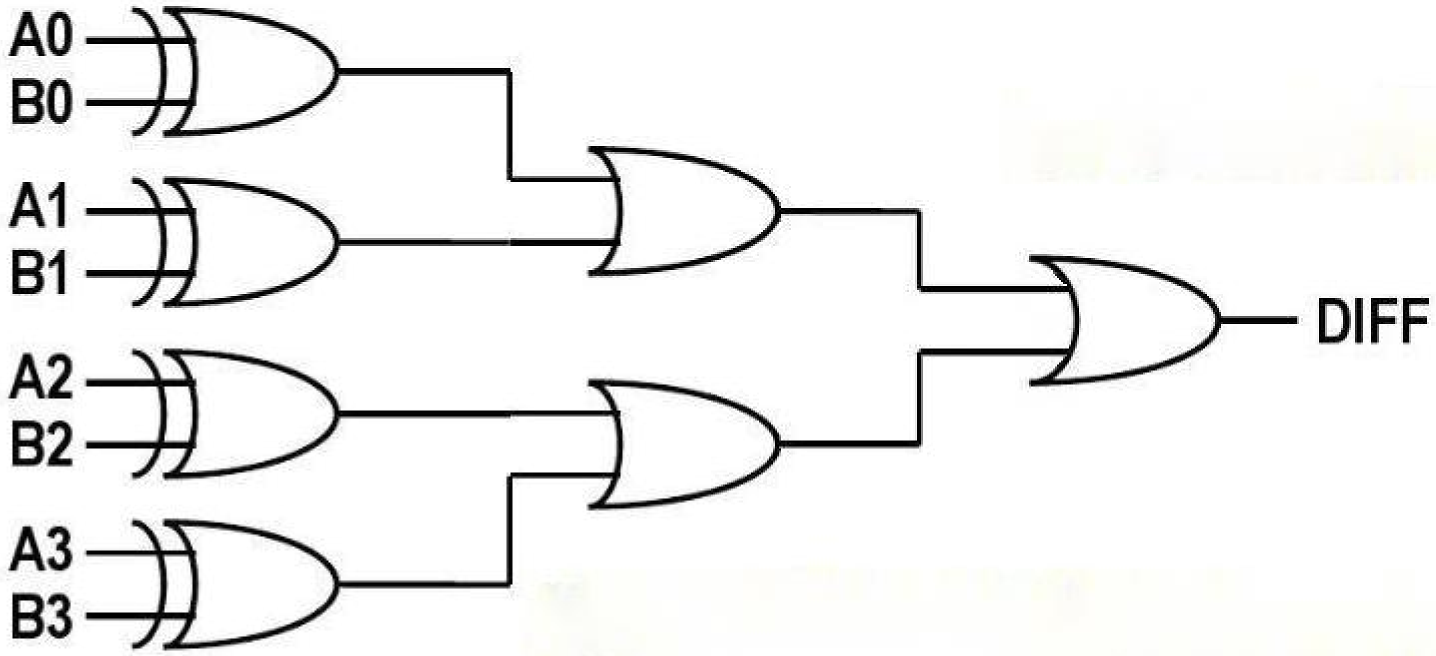
异或门的应用——等值比较器

如何构造1位等值比较器？



DIFF : different

如何实现4位等值比较器？



布尔代数基础

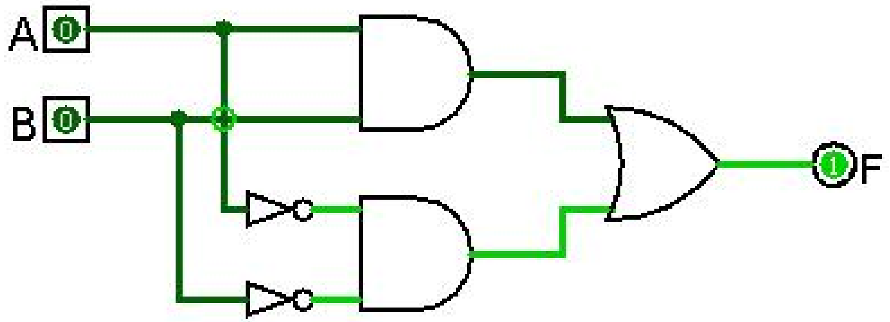
- 逻辑运算
- 布尔表达式和真值表
- 逻辑代数定理及规则
- 代数化简法

布尔表达式和真值表

布尔表达式（Boolean Expressions）：是由一个或者多个变量和逻辑运算符（与、或、非等）组成的式子。

$$F = AB + \bar{A}\bar{B}$$

逻辑图：



真值表

AB	F
0 0	1
0 1	0
1 0	0
1 1	1

$$F = [A(C+D)]' + BE$$

- **n** 个输入变量有 2^n 种取值组合

逻辑表达式如何判定相等？

- 如果两个逻辑表达式的真值表相同，则两个表达式相等.

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

A B C	$AB'+C$	$(A+C)(B'+C)$
0 0 0	0	0
0 0 1	1	1
0 1 0	0	0
0 1 1	1	1
1 0 0	1	1
1 0 1	1	1
1 1 0	0	0
1 1 1	1	1

适用情况：逻辑表达式简单，逻辑变量较少

布尔代数基础

- 逻辑运算
- 布尔表达式和真值表
- **布尔（逻辑）代数定理及规则**
- 代数化简法

布尔代数的公理和定理

公理（Axiom）

(A1) $0 \bullet 0 = 0$

(A1D) $0+0 = 0$

(A2) $0 \bullet 1 = 1 \bullet 0 = 0$

(A2D) $1+0 = 0+1=1$

(A3) $1 \bullet 1 = 1$

(A3D) $1+1 = 1$

(A4) $\overline{0}=1$

(A4D) $\overline{1} = 0$

(A5) **If** $A \neq 0$ **then** $A=1$

(A5D) **If** $A \neq 1$ **then** $A=0$

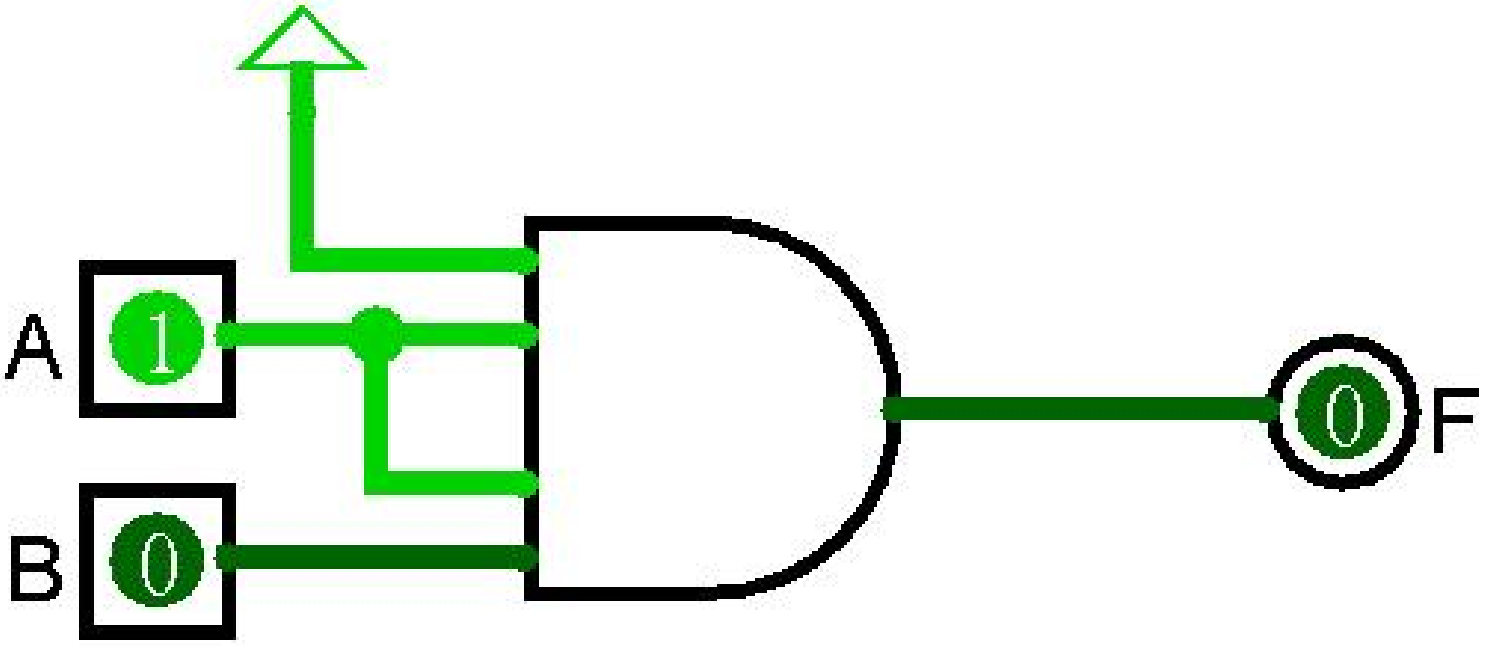
对偶规则（**Dual Rule**）：$+ \longrightarrow \bullet$,　　$\bullet \longrightarrow +$ ...

单变量定理（Theorem）

(T1)	$A+0=A$	(T1D)	$A \cdot 0 = 0$	0—1律
(T2)	$A+1=1$	(T2D)	$A \cdot 1 = A$	
(T3)	$A+\overline{A}=1$	(T3D)	$A \cdot \overline{A} = 0$	互补律
(T4)	$A+A=A$	(T4D)	$A \cdot A = A$	重叠律
(T5)	$\overline{\overline{A}} = A$			还原律

单变量定理应用

- 多输入逻辑门中不用的输入端怎么办？



二变量定理和三变量定理

交换律

(T6) $A+B=B+A$

(T6D) $A \bullet B= B \bullet A$

结合律

(T7) $(A+B)+C=A+(B+C)$

(T7D) $(A \bullet B)\bullet C= A \bullet (B \bullet C)$

分配律

(T8) $A \bullet (B+C)= AB+AC$

第二分配律

(T8D) $A+BC=(A+B) \bullet (A+C)$

普通代数不支持

特殊定理——对偶规则（Inference of Dual Rule）

变量：　　　不变

运算符：　$\cdot \longrightarrow +$

$+ \longrightarrow \cdot$

$\oplus \longrightarrow \odot$

$\odot \longrightarrow \oplus$

不能改变原来的优先级

- 求对偶式的另一种方法
  - 对整个表达式取反，然后再对每个变量取反。

对偶规则运用实例

$$F=A\cdot(B+C) \xrightarrow{\text{对偶}} (F)^D = A+B\cdot C$$

$$F=A\cdot\bar{B}+AC \xrightarrow{\text{对偶}} (F)^D = (A+\bar{B})\cdot(A+C)$$

$$F=\overline{\bar{A}\cdot\overline{B\cdot\bar{C}}} \xrightarrow{\text{对偶}} (F)^D = \overline{\bar{A}+\overline{B+\bar{C}}}$$

$F'=(A\cdot(B+C))'=A'+(B+C)'=A'+B'C'$ (对整个表达式取反)

$F^D=A+BC$ (对每个变量取反)

对偶规则的性质

① $F \xleftrightarrow{\text{Dual Rule}} (F)^D$

② 两个逻辑表达式相等，它们的对偶式也相等

$$A+BCD = (A+B)(A+C)(A+D)$$

$\Downarrow$ Dual Rule $\qquad\qquad$ $\Downarrow$ Dual Rule

$$A \cdot (B+C+D) = AB+AC+AD$$

二变量定理和三变量定理——续

(T9) $A + AB = A$ (T9D) $A(A + B) = A$ （吸收律）

(T10) $AB + A\overline{B} = A$ (T10D) $(A+B)(A+\overline{B}) = A$ （合并律）

(T11) $A + \overline{A}B = A + B$ （消除律）

$A+\overline{A}B$

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

$=A+B$

（德摩根定律）

$(X+Y)' = X'Y'$ $(XY)' = X'+Y'$

二变量定理和三变量定理——续

(T9) $A + AB = A$ (T9D) $A(A + B) = A$ （吸收律）

(T10) $AB + A\bar{B} = A$ (T10D) $(A+B)(A+\bar{B}) = A$ （合并律）

(T11) $A + \bar{A}B = A + B$ （消除律）

(T12) $AB + \bar{A}C + BC = AB + \bar{A}C$

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

$= AB + \bar{A}C + ABC + \bar{A}BC$

$= AB + \bar{A}C$

二变量定理和三变量定理——续

(T9) $A + AB = A$ (T9D) $A(A + B) = A$ （吸收律）

(T10) $AB + A\bar{B} = A$ (T10D) $(A+B)(A+\bar{B}) = A$ （合并律）

(T11) $A + \bar{A}B = A + B$ （消除律）

(T12) $AB + \bar{A}C + BC = AB + \bar{A}C$ （蕴含律）

(T12D) $(A+B)(B+C)(A'+C) = (A+B)(A'+C)$

(T13) $\overline{A\bar{B} + \bar{A}B} = \bar{A}\bar{B} + AB$

$$\overline{A\bar{B} + \bar{A}B}$$

$$= \overline{A\bar{B}} \cdot \overline{\bar{A}B}$$

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

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

n变量定理

(T14) $X+X+\ldots+X=X$ (T14D) $X\bullet X\bullet\ldots\bullet X=X$ （广义同一律）

(T15) $(X_1\bullet X_2\bullet\ldots\bullet X_n)'=X_1'+X_2'+\ldots+X_n'$

(T15) $(X_1+X_2+\ldots+X_n)'=X_1'\bullet X_2'\bullet\ldots\bullet X_n'$ （德·摩根定理）

(T16) $[F(X_1, X_2, \ldots, X_n, +, \bullet)]'=F(X_1', X_2', \ldots, X_n', \bullet, +)$ (广义德摩根定理)

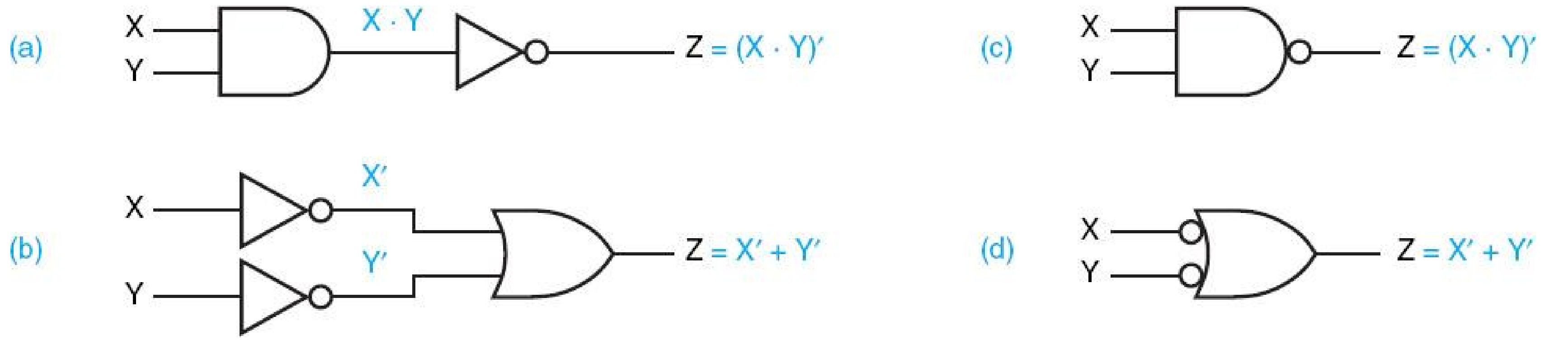
(T17) $F(X_1, X_2, \ldots, X_n)=X_1\bullet F(1, X_2, \ldots, X_n)+X_1'\bullet F(0, X_2, \ldots, X_n)$

(T17D) $F(X_1, X_2, \ldots, X_n)=[X_1+F(0, X_2, \ldots, X_n)]\bullet[X_1'+F(1, X_2, \ldots, X_n)]$

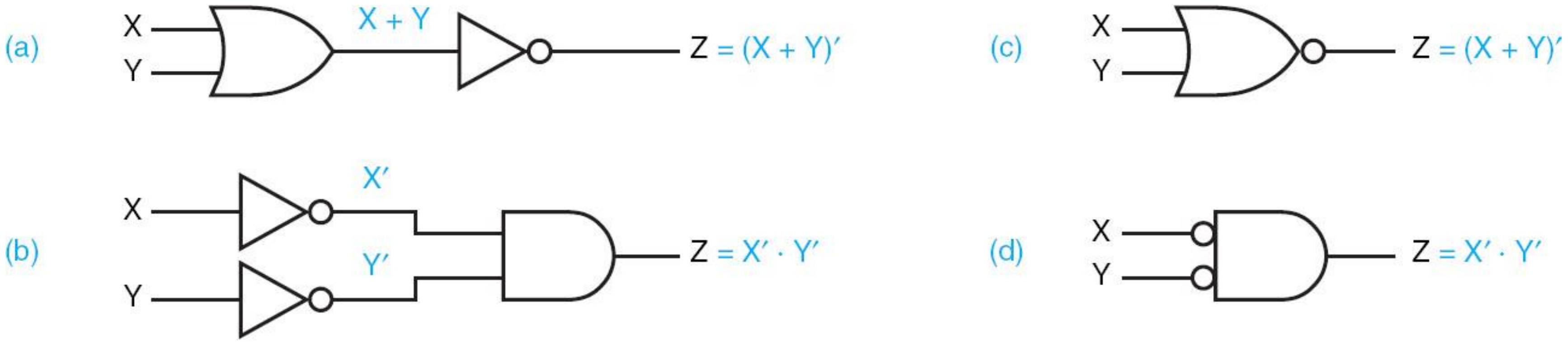
（香农展开定理）

根据德·摩根定理的等效电路

(T15) $(X_1 \bullet X_2 \bullet \cdots \bullet X_n)' = X_1' + X_2' + \cdots + X_n'$



(T15D) $(X_1 + X_2 + \cdots + X_n)' = X_1' \cdot X_2' \cdot \cdots \cdot X_n'$



布尔代数基础

- **逻辑运算**
- **布尔表达式和真值表**
- **逻辑代数定理及规则**
- **代数化简法**

代数化简

一个逻辑函数有多种不同的表达式

$F=AB+A\bar{C}$ ……与-或

$=\overline{\overline{AB+A\bar{C}}}$

$=\overline{\overline{AB}\cdot\overline{A\bar{C}}}$ ……与非-与非

$=\overline{(\bar{A}+\bar{B})\cdot(\bar{A}+C)}$ ……或-与非

$=\overline{(\bar{A}+\bar{B})}+\overline{(\bar{A}+C)}$ ……或非-或

$F=(A+B)\cdot(A+\bar{C})$ ……或-与

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

$=\overline{\overline{(A+B)}+\overline{(A+\bar{C})}}$ ……或非-或非

$=\overline{\bar{A}\cdot\bar{B}+\bar{A}\cdot C}$ ……与-或非

$=\overline{\bar{A}\,\bar{B}}\cdot\overline{\bar{A}C}$ ……与非-与

与或式——积之和式

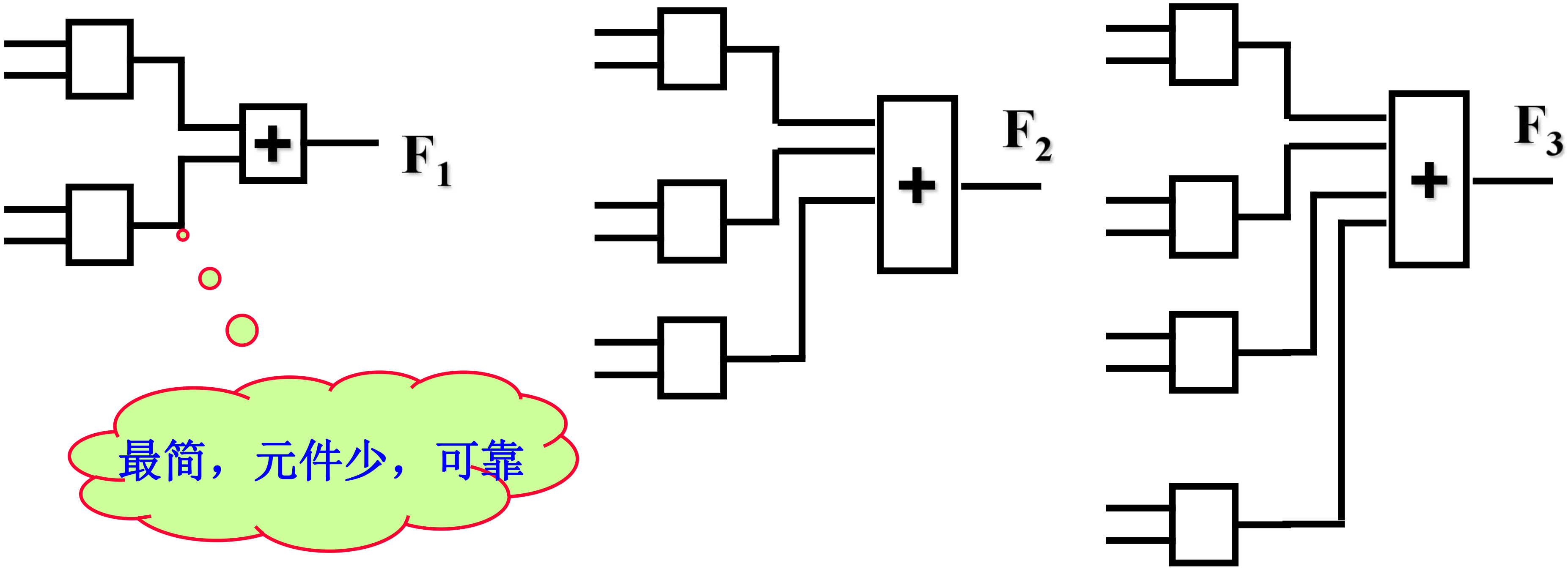
或与式——和之积式

同一类型的表达式也不是唯一的

$F=AB+\bar{A}C$ …………① F_1

$=AB+\bar{A}C+BC$ ………… ② F_2

$=ABC+AB\bar{C}+\bar{A}BC+\bar{A}\bar{B}C$ …………③ F_3



代数化简

最简（Minimum Expressions）？

① 与项（和项）的个数最少

② 每个与项（和项）中变量的个数最少

成本最低

① 逻辑门的数量最少

② 逻辑门的输入个数最少

目的：

- 降低成本
- 提高可靠性

方法：
- 代数法（Algebraic techniques）
- 卡诺图法（K. map method）

代数化简法例子－1

$$F= A+\underbrace{A\overline{B}\,\overline{C}}+\overline{A}CD+\overline{C}E+\overline{D}E$$

$$= A+\underbrace{\overline{A}CD}+\overline{C}E+\overline{D}E$$

$$= A+CD+\underbrace{\overline{C}E+\overline{D}E}$$

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

$$= A+\underbrace{CD+E\overline{CD}}$$

$$= A+CD+E$$

代数化简法例2

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

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

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

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

$$= A + \bar{B}C + B\bar{C} + \bar{B}D + \cancel{B\bar{D}} + C\bar{D}$$

$$= A + \cancel{\bar{B}C} + B\bar{C} + \bar{B}D + C\bar{D}$$

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

代数化简法例子—3,4

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

对偶规则

$$F^D=\bar{B}D+\bar{B}DAG+CE+\bar{C}G+AEG$$

$$=\bar{B}D+CE+\bar{C}G+\cancel{AEG}$$

$$=\bar{B}D+CE+\bar{C}G$$

对偶规则

$$F=(\bar{B}+D)(C+E)(\bar{C}+G)$$

$$=\ldots$$

代数化简法优缺点

- 优点——
  - 不受变量数目的约束
  - 对公理、定理和规则十分熟练时，化简较方便
- 缺点——
  - 技巧性强
  - 在很多情况下难以判断化简结果是否最简

*

小　结

- 各种逻辑运算
- 布尔表达式和真值表
- 逻辑代数定理及规则
- 代数化简法