



计算机系统核心课程群

《计算机硬件基础》

第二章 逻辑代数基础



本章知识要点

- 01 基本概念**
- 02 基本定理和规则**
- 03 逻辑函数的表示形式**
- 04 逻辑函数的化简**



1847年，英国数学家乔治·布尔(G. Boole)提出了用数学分析方法表示命题陈述的逻辑结构，并将形式逻辑归结为一种代数演，从而诞生了著名的“布尔代数”。

1938年，克劳德·香农(C. E. Shannon)将布尔代数应用于电话继电器的开关电路，提出了“开关代数”。

随着电子技术的发展，集成电路逻辑门已经取代了机械触点开关，故人们更习惯于把开关代数叫做逻辑代数。

逻辑代数是数字系统逻辑设计的理论基础和重要数学工具！

2.1 逻辑代数的基本概念

逻辑代数 L 是一个封闭的代数系统，它由一个逻辑变量集 K ，常量0和1以及“或”、“与”、“非”三种基本运算所构成，记为 $L=\{K, +, \cdot, -, 0, 1\}$ 。该系统应满足下列公理。

公理 1 交换律

对于任意逻辑变量 A 、 B ，有

$$A + B = B + A ; \quad A \cdot B = B \cdot A$$

公理 2 结合律

对于任意的逻辑变量 A 、 B 、 C ，有

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

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

公理 3 分配律

对于任意的逻辑变量A、B、C，有

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

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

公理 4 0—1 律

对于任意逻辑变量A，有

$$A + 0 = A ; \quad A \cdot 1 = A$$

$$A + 1 = 1 ; \quad A \cdot 0 = 0$$

公理 5 互补律

对于任意逻辑变量A，存在唯一的 \bar{A} ，使得：

$$\bar{A} + A = 1 \quad \bar{A} \cdot A = 0$$

公理是一个代数系统的基本出发点，无需加以证明。

2.1.1 逻辑变量及基本逻辑运算

一、变量

逻辑代数和普通代数一样，是用字母表示其值可以变化的量，即变量。所不同的是：

1. 任何逻辑变量的取值只有两种可能性——取值0或取值1。
2. 逻辑值0和1是用来表征矛盾的双方和判断事件真伪的形式符号，无大小、正负之分。

二、基本逻辑运算

描述一个数字系统，必须反映一个复杂系统中各开关元件之间的联系，这种相互联系反映到数学上就是几种运算关系。

逻辑代数中定义了“或”、“与”、“非”三种基本运算。

1.“或”运算

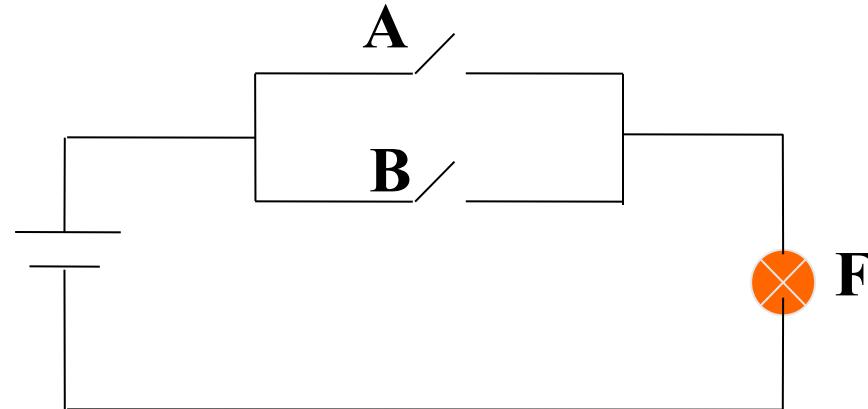
如果决定某一事件是否发生的多个条件中，只要有一个或一个以上条件成立，事件便可发生，则这种因果关系称之为“或”逻辑。

例如，用两个开关并联控制一个灯的照明控制电路。

逻辑代数基础



例如，下图所示电路。



并联开关电路

电路中，开关A和B并联控制灯F。可以看出，当开关A、B中有一个闭合或者两个均闭合时，灯F即亮。因此，灯F与开关A、B之间的关系是“或”逻辑关系。可表示为

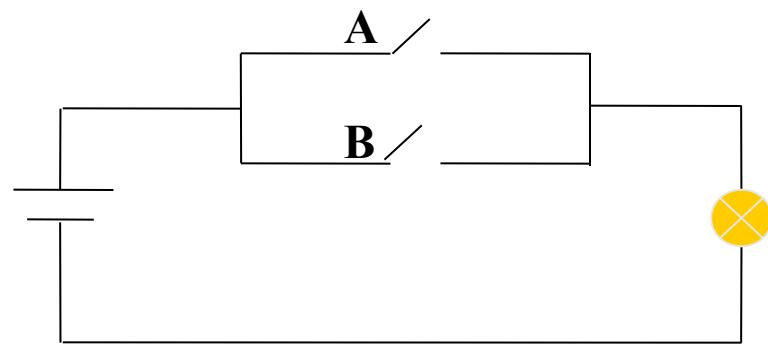
$$F = A + B \quad \text{或者} \quad F = A \vee B, \text{ 读作 “F等于A或B”。}$$



逻辑代数基础

假定开关断开用0表示，开关闭合用1表示；灯灭用0表示，灯亮用1表示，则灯F与开关A、B的关系如下表所示。

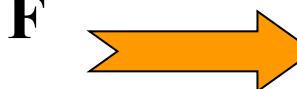
即：A、B中只要有一个为1，则F为1；仅当A、B均为0时，F才为0。



并联开关电路

“或”运算表

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



“或”运算的运算法则：

$$0 + 0 = 0 \quad 1 + 0 = 1$$

$$0 + 1 = 1 \quad 1 + 1 = 1$$

实现“或”运算关系的逻辑电路称为“或”门。

2. “与” 运算

如果决定某一事件发生的多个条件必须同时具备，事件才能发生，则这种因果关系称之为“与”逻辑。

在逻辑代数中，“与”逻辑关系用“与”运算描述。两变量“与”运算关系可表示为

$$F = A \cdot B \quad \text{或者} \quad F = A \wedge B$$

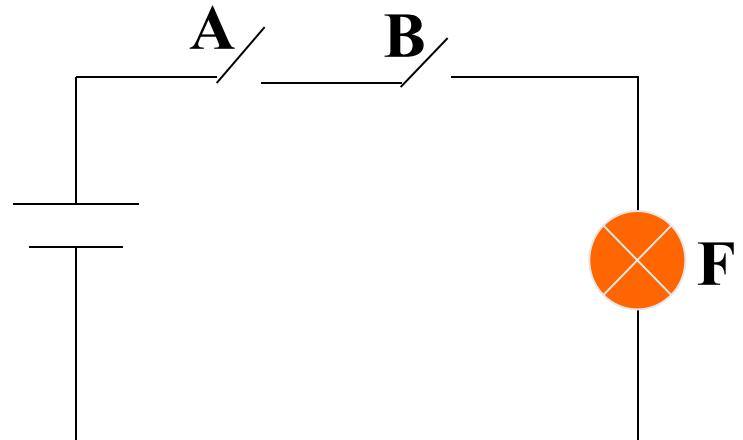
即：若A、B均为1，则F为1；否则，F为0。

“与”运算表

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

例如，两个开关串联控制同一个灯。显然，仅当两个开关均闭合时，灯才能亮，否则，灯灭。

假定开关闭合状态用1表示，断开状态用0表示，灯亮用1表示，灯灭用0表示，则F和A、B之间的关系“与”运算关系。



串联开关电路

“与”运算的运算法则：

$$0 \cdot 0 = 0 \quad 1 \cdot 0 = 0$$

$$0 \cdot 1 = 0 \quad 1 \cdot 1 = 1$$

数字系统中，实现“与”运算关系的逻辑电路称为“与”门。

3. “非” 运算

如果某一事件的发生取决于条件的否定，即事件与事件发生的条件之间构成矛盾，则这种因果关系称为“非”逻辑。

在逻辑代数中，“非”逻辑用“非”运算描述。其运算符号为“ \neg ”，有时也用“ \overline{A} ”表示。“非”运算的逻辑关系可表示为 $F = \overline{A}$ 或者 $F = \neg A$ 读作“F等于A非”。

即：若A为0，则F为1；若A为1，则F为0。

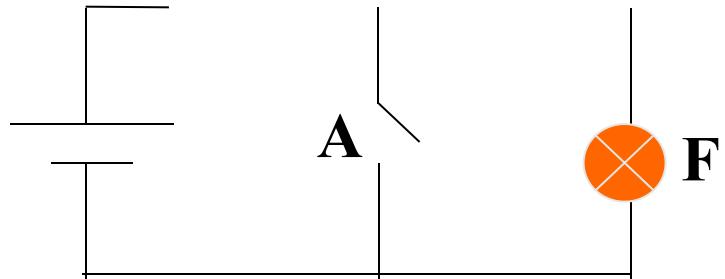
“非”运算表

A	F
0	1
1	0

逻辑代数基础



例如，下面开关与灯并联的电路中，仅当开关断开时，灯亮；一旦开关闭合，则灯灭。令开关断开用0表示，开关闭合用1表示，灯亮用1表示，灯灭用0表示，则电路中灯F与开关A的关系即为上表所示“非”运算关系。



开关与灯并联电路

“非”运算的运算法则：

$$\bar{0} = 1 ; \quad \bar{1} = 0$$

数字系统中实现“非”运算功能的逻辑电路称为“非”门，有时又称为“反相器”。

2.1.1 逻辑函数及逻辑函数间的相等

一、逻辑函数的定义

逻辑代数中函数的定义与普通代数中函数的定义类似，即随自变量变化的因变量。但和普通代数中函数的概念相比，逻辑函数具有如下特点：

1. 逻辑函数和逻辑变量一样，取值只有0和1两种可能；
2. 函数和变量之间的关系是由“或”、“与”、“非”三种基本运算决定的。

逻辑代数基础



设某一逻辑电路的输入逻辑变量为 A_1, A_2, \dots, A_n , 输出逻辑变量为 F , 如下图所示。



图中, F 被称为 A_1, A_2, \dots, A_n 的逻辑函数, 记为

$$F = f(A_1, A_2, \dots, A_n)$$

逻辑电路输出函数的取值是由逻辑变量的取值和电路本身的结构决定的。

逻辑函数和普通代数中的函数一样存在**是否相等**的问题。设有两个相同变量的逻辑函数

$$F_1 = f_1(A_1, A_2, \dots, A_n)$$

$$F_2 = f_2(A_1, A_2, \dots, A_n)$$

若对应于逻辑变量 A_1, A_2, \dots, A_n 的任何一组取值， F_1 和 F_2 的值都相同，则称函数 F_1 和 F_2 相等，记作 $F_1 = F_2$ 。

如何判断两个逻辑函数是否相等？

通常有两种方法：真值表法，代数法。



逻辑代数基础

2.1.3 逻辑函数的表示法

如何对逻辑功能进行描述？

常用的方法有逻辑表达式、真值表、卡诺图3种。

一、逻辑表达式

逻辑表达式是由逻辑变量和“或”、“与”、“非”3种运算符以及括号所构成的式子。例如

$$F = f(A, B) = \overline{A}B + A\overline{B}$$

函数F和变量A、B的关系是：

当变量A和B取值不同时，函数F的值为“1”；取值相同时，函数F的值为“0”。



逻辑代数基础

逻辑表达式的简写：

1. “非”运算符下可不加括号，如 $\overline{A \cdot B}$, $\overline{A + B}$ 等。
2. “与”运算符一般可省略，如 $A \cdot B$ 可写成 AB 。
3. 在一个表达式中，如果既有“与”运算又有“或”运算，则按先“与”后“或”的规则进行运算，可省去括号，如 $(A \cdot B) + (C \cdot D)$ 可写为 $AB+CD$ 。

注意： $(A+B) \cdot (C+D)$ 不能省略括号，即不能写成 $A+B \cdot C+D$ ！

运算优先法则： () → — → • → ⊕ → +

高 → 低

4. $(A+B)+C$ 或者 $A+(B+C)$ 可用 $A+B+C$ 代替； $(AB)C$ 或者 $A(BC)$ 可用 ABC 代替。



逻辑代数基础

二、真值表

依次列出一个逻辑函数的所有输入变量取值组合及其相应函数值的表格称为真值表。

一个 n 个变量的逻辑函数，其真值表有 2^n 行。例如，

函数 $F = A\bar{B} + \bar{A}C$ 的真值表

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

真值表由两部分组成：

左边一栏列出变量的所有取值组合，为了不发生遗漏，通常各变量取值组合按二进制数码顺序给出；右边一栏为逻辑函数值。

练习题：

求 $F = \bar{A}\bar{B} + BC$ 的真值表

三、卡诺图

卡诺图是由表示逻辑变量所有取值组合的小方格所构成的平面图。

这种用图形描述逻辑函数的方法，在逻辑函数化简中十分有用，将在后面结合函数化简问题进行详细介绍。

描述逻辑函数的3种方法可用于不同场合。但针对某个具体问题而言，它们仅仅是同一问题的不同描述形式，相互之间可以很方便地进行变换。

2.2 逻辑代数的基本定理和规则

2.2.1 基本定理

常用的 8 组定理：

定理1

$$0 + 0 = 0 \quad 1 + 0 = 1 \quad 0 \cdot 0 = 0 \quad 1 \cdot 0 = 0$$

$$0 + 1 = 1 \quad 1 + 1 = 1 \quad 0 \cdot 1 = 0 \quad 1 \cdot 1 = 1$$

证明：在公理4中，A表示集合K中的任意元素，因而可以是0或1。用0和1代入公理4中的A，即可得到上述关系。

如果以1和0代替公理5中的A，则可得到如下推论：

$$\bar{1} = 0 \qquad \bar{0} = 1$$

逻辑代数基础



定理2 $A + A = A$; $A \cdot A = A$

证明 $A+A = (A+A) \cdot 1$ 公理4

$$= (A+A) \cdot (A+\bar{A}) \quad \text{公理5}$$

$$= A+(A \cdot \bar{A}) \quad \text{公理3}$$

$$= A+0 \quad \text{公理5}$$

$$= A \quad \text{公理4}$$

定理3 $A + A \cdot B = A$; $A \cdot (A + B) = A$

证明 $A+A \cdot B = A \cdot 1 + A \cdot B$ 公理4

$$= A \cdot (1+B) \quad \text{公理3}$$

$$= A \cdot (B+1) \quad \text{公理1}$$

$$= A \cdot 1 \quad \text{公理4}$$

$$= A \quad \text{公理4}$$



逻辑代数基础

定理4 $A + \overline{A} \cdot B = A + B ; A \cdot (\overline{A} + B) = A \cdot B$

证明 $A + \overline{A} \cdot B = (A + \overline{A}) \cdot (A + B)$ 公理3

$$= 1 \cdot (A + B) \quad \text{公理5}$$

$$= A + B \quad \text{公理4}$$

定理5 $\overline{\overline{A}} = A$

证明 令 $A = \overline{X}$

$$\text{因而 } X \cdot A = \overline{0} \quad X + \overline{A} = 1 \quad \text{公理5}$$

$$\text{但是 } A \cdot A = \overline{0} \quad A + \overline{A} = 1 \quad \text{公理5}$$

由于 X 和 A 都满足公理5, 根据公理5的唯一性, 得到: $A = X$

由于 $\overline{\overline{A}} = X$, 所以 $\overline{\overline{A}} = A$



定理6 $\overline{A+B} = \overline{A} \cdot \overline{B}$ $\overline{A \cdot B} = \overline{A} + \overline{B}$

证明 由于 $(\bar{A}, \bar{B}) + (A + B) = (\bar{A}, \bar{B} + A) + B$ 公理2

$$= (\bar{B} + A) + B \quad \text{定理4}$$

$$= A + (B + \bar{B}) \quad \text{公理1,2}$$

=A+1 公理5

=1 公理4

而且 $(\bar{A} \cdot \bar{B}) \cdot (A + B) = \bar{A} \cdot \bar{B} \cdot A + \bar{A} \cdot \bar{B} \cdot B$ 公理3

$$= 0 + 0 \quad \text{公理1,5}$$

定理1

所以，根据公理5的唯一性可得 $\overline{A + B} = \overline{A} \cdot \overline{B}$

定理7 $A \cdot B + A \cdot \bar{B} = A$

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

证明
$$\begin{aligned} A \cdot B + A \cdot \bar{B} &= A \cdot (B + \bar{B}) && \text{公理3} \\ &= A \cdot 1 && \text{公理5} \\ &= A && \text{公理4} \end{aligned}$$



第二章 逻辑代数基础

定理8

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

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

证明

$$A \cdot B + \bar{A} \cdot C + B \cdot C$$

$$= A \cdot B + \bar{A} \cdot C + B \cdot C \cdot (A + \bar{A}) \quad \text{公理5}$$

$$= A \cdot B + \bar{A} \cdot C + B \cdot C \cdot A + B \cdot C \cdot \bar{A} \quad \text{公理3}$$

$$= A \cdot B + A \cdot B \cdot C + \bar{A} \cdot C + \bar{A} \cdot B \cdot C \quad \text{公理1}$$

$$= A \cdot B \cdot (1+C) + \bar{A} \cdot C \cdot (1+B) \quad \text{公理3}$$

$$= A \cdot B \cdot (C+1) + \bar{A} \cdot C \cdot (B+1) \quad \text{公理1}$$

$$= A \cdot B + \bar{A} \cdot C \quad \text{公理4}$$



逻辑代数基础

2. 2. 2 重要规则

逻辑代数有3条重要规则。

一、代入规则

任何一个含有变量A的逻辑等式, 如果将所有出现A的位置都代之以同一个逻辑函数F, 则等式仍然成立。这个规则称为代入规则。

例如, 将逻辑等式 $A(B+C)=AB+AC$ 中的C都用 $(C+D)$ 代替, 该逻辑等式仍然成立, 即

$$A [B+(C+D)] = AB+A(C+D)$$

代入规则的正确性是显然的, 因为任何逻辑函数都和逻辑变量一样, 只有0和1两种可能的取值。

代入规则的意义：

利用代入规则可以将逻辑代数公理、定理中的变量用任意函数代替，从而推导出更多的等式。这些等式可直接当作公式使用，无需另加证明。

例如，用逻辑函数 $F = f(A_1A_2 \dots A_n)$ 代替公理 $\bar{A} + A = 1$ 中的 A ，可得到逻辑等式

$$\bar{f}(A_1A_2 \dots A_n) + f(A_1A_2 \dots A_n) = 1$$

注意：使用代入规则时，必须将等式中所有出现同一变量的地方均以同一函数代替，否则代入后的等式将不成立。



逻辑代数基础

二、反演规则

若将逻辑函数表达式F中所有的“·”变成“+”，“+”变成“·”，“0”变成“1”，“1”变成“0”，原变量变成反变量，反变量变成原变量，并保持原函数中的运算顺序不变，则所得到的新的函数 \bar{F} 为原函数F的反函数。

即：“·” \leftrightarrow “+”，“0” \leftrightarrow “1”，原变量 \leftrightarrow 反变量

例如，已知函数 $F = \bar{A} \cdot B + C \cdot \bar{D}$ 根据反演规则可得到

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

练习题：求 $F = \bar{A} + \bar{B}(C + D\bar{E})$ 的反演



逻辑代数基础

注意：使用反演规则时，应保持原函数式中运算符号的优先顺序不变！

例如，已知函数 $F = \overline{A} + \overline{B} \cdot (C + D\overline{E})$ ，根据反演规则得到的反函数应该是

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

而不应该是

$$\overline{F} = A \cdot B + \overline{C} \cdot \overline{D} + E \quad \times! \text{ 错误}$$

三、对偶规则

如果将逻辑函数表达式 F 中所有的“ \cdot ”变成“ $+$ ”，“ $+$ ”变成“ \cdot ”，“0”变成“1”，“1”变成“0”，并保持原函数中的运算顺序不变，则所得到的新的逻辑表达式称为函数 F 的对偶式，并记作 F' 。例如，

$$F = A \cdot B + \overline{B} \cdot (C + 0) \quad ; \quad F' = (A + B) \cdot (\overline{B} + C \cdot 1)$$

注意:求逻辑表达式的对偶式时，同样要保持原函数的运算顺序不变。

若两个逻辑函数表达式F和G相等，则其对偶式F' 和G' 也相等。这一规则称为对偶规则。

根据对偶规则，当已证明某两个逻辑表达式相等时，即可知道它们的对偶式也相等。

例如，已知 $AB + \overline{AC} + BC = AB + \overline{AC}$ ，根据对偶规则对等式两端的表达式取对偶式，即可得到等式

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

显然，利用对偶规则可以使定理、公式的证明减少一半。



逻辑代数基础

2. 2. 2 复合逻辑

实际应用中广泛采用“与非”门、“或非”门、“与或非”门、“异或”门等门电路。这些门电路输出和输入之间的逻辑关系可由3种基本运算构成的复合运算来描述，故通常将这种逻辑关系称为复合逻辑，相应的逻辑门则称为复合门。

一、与非逻辑

与非逻辑是由与、非两种逻辑复合形成的，可用逻辑函数表示为

逻辑功能：只要变量A、B、C、…中有一个为0，则函数 $F = \overline{A \cdot B \cdot C \dots}$

F为1；仅当变量A、B、C、…全部为1时，函数F为0。

实现与非逻辑的门电路称为“与非”门。

使用与非门可以实现与、或、非三种基本运算：

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

或： $F = \overline{\overline{A \cdot 1 \cdot B \cdot 1}} = \overline{\overline{A} \cdot \overline{B}} = A + B$

非： $F = \overline{A \cdot 1} = \overline{A}$

只要有了与非门便可组成实现各种逻辑功能的电路，通常称与非门为通用门。



逻辑代数基础

二、或非逻辑

或非逻辑是由或、非两种逻辑复合形成的，可表示为

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

逻辑功能：只要变量A、B、C…中有一个为1，则函数F为0；仅当变量A、B、C…全部为0时，函数F为1。实现或非逻辑的门电路称为“或非”门。

同样，或非逻辑也可以实现与、或、非3种基本逻辑。以两变量或非逻辑为例：

与： $F = \overline{\overline{A+0} + \overline{B+0}} = \overline{\overline{A} + \overline{B}} = A \cdot B$

或： $F = \overline{\overline{A+B} + \overline{0}} = \overline{\overline{A+B}} = A + B$

非： $F = \overline{A+0} = \overline{A}$

或非门同样可实现各种逻辑功能，是一种**通用门**。

三、与或非逻辑

与或非逻辑是由3种基本逻辑复合形成的，逻辑函数表达式的形式为

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

逻辑功能：仅当每一个“与项”均为0时，才能使F为1，否则F为0。实现与或非功能的门电路称为**“与或非”门**。

显然，可以仅用与或非门去组成实现各种功能的逻辑电路。但实际应用中这样做一般会很不经济，所以，与或非门主要用来实现与或非形式的函数。必要时可将逻辑函数表达式的形式变换成与或非的形式。



逻辑代数基础

四、异或逻辑及同或逻辑

1. 异或逻辑

异或逻辑是一种**两变量逻辑关系**，可用逻辑函数表示为

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

逻辑功能：变量A、B取值相同，F为0；变量A、B取值相异，F为1。

实现异或运算的逻辑门称为“**异或门**”。

根据异或逻辑的定义可知：

$$A \oplus 0 = A \quad A \oplus 1 = \overline{A}$$

$$A \oplus A = 0 \quad A \oplus \overline{A} = 1$$

当多个变量进行异或运算时，可用两两运算的结果再运算，也可两两依次运算。 **练习题：**求10110110110, 1000100101异或

逻辑代数基础



例如， $F = A \oplus B \oplus C \oplus D$

$$= (A \oplus B) \oplus (C \oplus D) \quad (\text{两两运算的结果再运算})$$
$$= [(A \oplus B) \oplus C] \oplus D \quad (\text{两两依次运算})$$

注意：在进行异或运算的多个变量中，若有奇数个变量的值为1，则运算结果为1；若有偶数个变量的值为1，则运算结果为0。

同或逻辑也是一种两变量逻辑关系，其逻辑函数表达式为

$$F = A \odot B = \overline{A} \cdot \overline{B} + AB$$

式中，“ \odot ”为同或运算的运算符。

功能逻辑：变量A、B取值相同，F为1；变量A、B取值相异，F为0。

实现同或运算的逻辑门称为“同或门”。

同或逻辑与异或逻辑的关系既互为相反，又互为对偶。即有：

$$\begin{aligned}\overline{A \oplus B} &= \overline{\overline{AB} + \overline{A}\overline{B}} = (\overline{A} + B)(A + \overline{B}) \\ &= AB + \overline{A} \cdot \overline{B} = A \odot B \\ (A \oplus B)' &= (\overline{AB} + \overline{A}\overline{B})' = (A + \overline{B})(\overline{A} + B) \\ &= \overline{A} \cdot \overline{B} + AB = A \odot B\end{aligned}$$

注意：当多个变量进行同或运算时，若有奇数个变量的值为0，则运算结果为0；反之，若有偶数个变量的值为0，则运算结果为1。

由于同或实际上等同于异或之非，所以实际应用中通常用异或门加非门实现同或运算。



逻辑代数基础

2.3 逻辑函数表达式的形式与变换

任何一个逻辑函数，其表达式的形式都不是唯一的。下面介绍逻辑函数表达式的基本形式、标准形式及其相互转换。

2.3.1 逻辑函数表达式的两种基本形式

两种基本形式：指“与-或”表达式和“或-与”表达式。

一、“与-或”表达式

“与-或”表达式：是指由若干“与项”进行“或”运算构成的表达式。例如，

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

“与项”有时又被称为“积项”，相应地“与-或”表达式又称为“积之和”表达式。

二、“或-与”表达式

“或-与” 表达式：是指由若干“或项”进行“与”运算构成的表达式。

例如，

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

“或项”有时又被称为“和项”，相应地“或-与”表达式又称为“和之积”表达式。



逻辑代数基础

逻辑函数表达式可以被表示成任意的混合形式。例如，

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

注意：逻辑函数的基本形式都不是唯一的。例如

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

2.3.2 逻辑函数表达式的标准形式

为了在逻辑问题的研究中使逻辑功能和唯一的逻辑表达式对应，引入了逻辑函数表达式的**标准形式**。逻辑函数表达式的标准形式是建立在**最小项**和**最大项**概念的基础之上的。



逻辑代数基础

一、最小项和最大项

1. 最小项

(1) 定义：如果一个具有n个变量的函数的“与项”包含全部n个变量，每个变量都以原变量或反变量形式出现一次，且仅出现一次，则该“与项”被称为最小项。有时又将最小项称为标准“与项”。

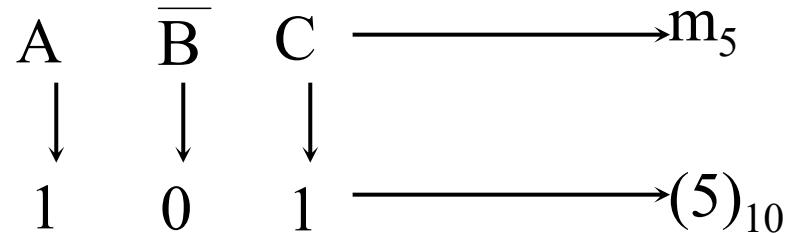
(2) 最小项的数目：n个变量可以构成 2^n 个最小项。

例如，3个变量A、B、C可以构成 $\bar{A} \cdot \bar{B} \cdot \bar{C}$ 、 $\bar{A} \cdot \bar{B} \cdot C$ 、...、 $A \cdot B \cdot C$ 共8个最小项。

(3) 简写：用 m_i 表示最小项。

下标*i*的取值规则是：按照变量顺序将最小项中的原变量用1表示，反变量用0表示，由此得到一个二进制数，与该二进制数对应的十进制数即下标*i*的值。

例如，3变量A、B、C构成的最小项 $\bar{A}\bar{B}C$ 可用 m_5 表示。因为



(4) 性质—— 最小项具有如下四条性质。

性质1：任意一个最小项，其相应变量有且仅有一种取值使这个最小项的值为1。并且，最小项不同，使其值为1的变量取值不同。

在由n个变量构成的任意“与项”中，最小项是使其值为1的变量取值组合数最少的一种“与项”，这也就是最小项名字的由来。



逻辑代数基础

性质2：相同变量构成的两个不同最小项相“与”为0。

因为任何一种变量取值都不可能使两个不同最小项同时为1，故相“与”为0。即

$$m_i \cdot m_j = 0$$

性质3：n个变量的全部最小项相“或”为1。

通常借用数学中的累加符号“ Σ ”，将其记为

$$\sum_{i=0}^{2^n - 1} m_i = 1$$

性质4：n个变量构成的最小项有n个相邻最小项。

相邻最小项：是指除一个变量互为相反外，其余部分均相同的最小项。例如，三变量最小项ABC和 $\bar{A}BC$ 相邻。

2. 最大项

定义：如果一个具有n个变量函数的“或项”包含全部n个变量，每个变量都以原变量或反变量形式出现一次，且仅出现一次，则该“或项”被称为最大项。有时又将最大项称为标准“或项”。

数目：n个变量可以构成 2^n 个最大项。

例如，3个变量A、B、C可构成 $A+B+C$ 、 $A + B + \bar{C}$ 、 \cdots 、 $\bar{A} + \bar{B} + \bar{C}$ 共8个最大项。



逻辑代数基础

简写：用 M_i 表示最大项。

下标*i*的取值规则是：将最大项中的原变量用0表示，反变量用1表示，由此得到一个二进制数，与该二进制数对应的十进制数即下标 *i* 的值。例如，3变量A、B、C构成的最大项 $\overline{A} + B + \overline{C}$ 可用 M_5 表示。因为

$$\begin{array}{c} \overline{A} + B + \overline{C} \longrightarrow M_5 \\ \downarrow \quad \downarrow \quad \downarrow \\ 1 \quad 0 \quad 1 \longrightarrow (5)_{10} \end{array}$$

性质：最大项具有如下四条性质：

性质1 任意一个最大项，其相应变量有且仅有一种取值使这个最大项的值为0。并且，最大项不同，使其值为0的变量取值不同。

在n个变量构成的任意“或项”中，最大项是使其值为1的变量取值组合数最多的一种“或项”，因而将其称为最大项。

性质2 相同变量构成的两个不同最大项相“或”为1。因为任何一种变量取值都不可能使两个不同最大项同时为0，故相“或”为1。即 $M_i + M_j = 1$

性质3 n 个变量的全部最大项相“与”为0。通常借用数学中的累乘符号“ Π ”将其记为：

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

性质4 n 个变量构成的最大项有 n 个相邻最大项。相邻最大项是指除一个变量互为相反外，其余变量均相同的大项。

两变量最小项、最大项的真值表如下。

2变量最小项、最大项真值表

变 量	最 小 项				最 大 项			
	$\bar{A} \cdot \bar{B}$	$\bar{A}B$	$A\bar{B}$	AB	$A + B$	$A + \bar{B}$	$\bar{A} + B$	$\bar{A} + \bar{B}$
	m_0	m_1	m_2	m_3	M_0	M_1	M_2	M_3
0 0	1	0	0	0	0	1	1	1
0 1	0	1	0	0	1	0	1	1
1 0	0	0	1	0	1	1	0	1
1 1	0	0	0	1	1	1	1	0

真值表反映了最小项、最大项的有关性质。

3. 最小项和最大项的关系

在同一问题中，下标相同的最小项和最大项互为反函数。或者说，相同变量构成的最小项 m_i 和最大项 M_i 之间存在互补关系。即

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

例如，由3变量A、B、C构成的最小项 m_3 和最大项 M_3 之间有

$$\overline{m_3} = \overline{\overline{A}BC} = A + \overline{B} + \overline{C} = M_3$$

$$\overline{M_3} = \overline{A + \overline{B} + \overline{C}} = \overline{A}BC = m_3$$



逻辑代数基础

二、逻辑函数表达式的标准形式

逻辑函数表达式的标准形式有**标准“与-或”表达式**和**标准“或-与”表达式**两种类型。

1. 标准“与 - 或”表达式

由若干最小项相“或”构成的逻辑表达式称为**标准“与-或”表达式**，也叫做**最小项表达式**。

例如，如下所示为一个3变量函数的标准“与-或”表达式

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

该函数表达式又可简写为

$$\begin{aligned} F(A, B, C) &= m_1 + m_2 + m_4 + m_7 \\ &= \sum m(1,2,4,7) \end{aligned}$$



逻辑代数基础

2. 标准“或-与”表达式

由若干最大项相“与”构成的逻辑表达式称为标准“或-与”表达式，也叫做最大项表达式。

例如， $A + B + \bar{C}$ 、 $\bar{A} + B + \bar{C}$ 、 $\bar{A} + \bar{B} + \bar{C}$ 为3变量构成的3个最大项，对这3个最大项进行“与”运算，即可得到一个3变量函数的标准“或-与”表达式

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

该表达式又可简写为

$$F(A, B, C) = M_1 M_5 M_7 = \prod M(1, 5, 7)$$



逻辑代数基础

2.3.3 逻辑函数表达式的转换

将一个任意逻辑函数表达式转换成标准表达式有两种常用方法。

一、代数转换法

所谓代数转换法，就是利用逻辑代数的公理、定理和规则进行逻辑变换，将函数表达式从一种形式变换为另一种形式。

1. 求标准“与-或”式

一般步骤如下：

第一步：将函数表达式转换成一般“与-或”表达式。

第二步：反复使用 $X = X(Y + \bar{Y})$ 将表达式中所有非最小项的“与项”扩展成最小项。



逻辑代数基础

例 将逻辑函数表达式 $F(A, B, C) = \overline{(A\bar{B} + B\bar{C}) \cdot \bar{AB}}$ 转换成标准“与-或”表达式。

解 第一步：将函数表达式转换成“与-或”表达式。即

$$\begin{aligned} F(A,B,C) &= \overline{(A\bar{B} + B\bar{C}) \cdot \bar{AB}} \\ &= \overline{\bar{A}\bar{B} + B\bar{C}} + AB \\ &= (\bar{A} + B)(\bar{B} + C) + AB \\ &= \bar{A} \cdot \bar{B} + \bar{A}C + BC + AB \end{aligned}$$

第二步：把“与-或”式中非最小项的“与项”扩展成最小项。

$$\begin{aligned} F(A,B,C) &= \bar{A} \cdot \bar{B}(\bar{C} + C) + \bar{A}C(\bar{B} + B) + (\bar{A} + A)BC + AB(\bar{C} + C) \\ &= \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C + \bar{A}BC + ABC + ABC + \bar{ABC} + ABC \\ &= \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C + \bar{A}BC + ABC + \bar{ABC} + ABC \end{aligned}$$

所得标准“与-或”式的简写形式为

$$\begin{aligned} F(A,B,C) &= m_0 + m_1 + m_3 + m_6 + m_7 \\ &= \sum m(0,1,3,6,7) \end{aligned}$$

当给出函数表达式已经是“与-或”表达式时，可直接进行第二步。

随堂练习：求下列逻辑表达式的标准表达式

1. $F(A, B, C) = AB + \bar{A}C$

2. $F(A, B, C, D) = \overline{AB + AD + \bar{B}C}$

2. 求一个函数的标准“或-与”式

一般步骤：

第一步：将函数表达式转换成一般“或-与”表达式。

第二步：反复利用定理 $A = (A + B)(A + \bar{B})$ 把表达式中所有非最大项的“或项”扩展成最大项。



逻辑代数基础

例 将逻辑函数表达式 $F(A, B, C) = \overline{(AB + \overline{A}C)} + \overline{B}\overline{C}$ 变换成标准“或-与”表达式。

解 第一步：将函数表达式变换成“或-与”表达式。即

$$\begin{aligned} F(A, B, C) &= \overline{(AB + \overline{A}C)} + \overline{B}\overline{C} \\ &= \overline{AB} \cdot \overline{\overline{A}C} + \overline{B}\overline{C} \\ &= (\overline{A} + \overline{B})(A + \overline{C}) + \overline{B}\overline{C} \\ &= [(\overline{A} + \overline{B})(A + \overline{C}) + \overline{B}] [(\overline{A} + \overline{B})(A + \overline{C}) + C] \\ &= (\overline{A} + \overline{B} + \overline{B})(A + \overline{C} + \overline{B})(\overline{A} + \overline{B} + C)(A + \overline{C} + C) \\ &= (\overline{A} + \overline{B})(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + C) \quad = 1 \end{aligned}$$



逻辑代数基础

第二步：将所得“或-与”表达中的非最大项扩展成最大项。即

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

该标准“或-与”表达式的简写形式为

$$F(A, B, C) = M_3 \cdot M_6 \cdot M_7 = \prod M(3,6,7)$$

当给出函数已经是“或-与”表达式时，可直接进行第二步。

另一种解法：求该表达式对偶的标准“与-或”表达式后，再对偶

随堂练习：利用对偶的方式求取上式的标准或与表达式

二、真值表转换法

1. 求标准“与-或”式

逻辑函数的最小项表达式与真值表具有一一对应的关系。

假定函数F的真值表中有k组变量取值使F的值为1，其他变量取值下F的值为0，那么，函数F的最小项表达式由这k组变量取值对应的k个最小项相或组成。

具体：真值表上使函数值为1的变量取值组合对应的最小项相“或”，即可构成一个函数的标准“与-或”式。



逻辑代数基础

例 将函数表达式 $F(A, B, C) = A\bar{B} + B\bar{C}$ 变换成标准“与-或”表达式。

解：首先，列出F的真值表如下表所示，然后，根据真值表可直接写出F的最小项表达式

$$F(A, B, C) = \sum m(2, 4, 5, 6)$$

函数 $F(A, B, C) = A\bar{B} + B\bar{C}$ 的真值表

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$$F(A, B, C) = \sum m(2, 4, 5, 6)$$



逻辑代数基础

2. 求一个函数的标准“或-与”式

逻辑函数的最大项表达式与真值表之间同样具有一一对应的关系。

假定在函数F的真值表中有p组变量取值使F的值为0，其他变量取值下F的值为1，那么，函数F的最大项表达式由这p组变量取值对应的p个最大项“相与”组成。

具体：真值表上使函数值为0的变量取值组合对应的最大项相“与”即可构成一个函数的标准“或-与”式。



逻辑代数基础

例 将函数表达式 $F(A, B, C) = \overline{AC} + A\overline{B}\overline{C}$ 表示成最大项表达式的形式。

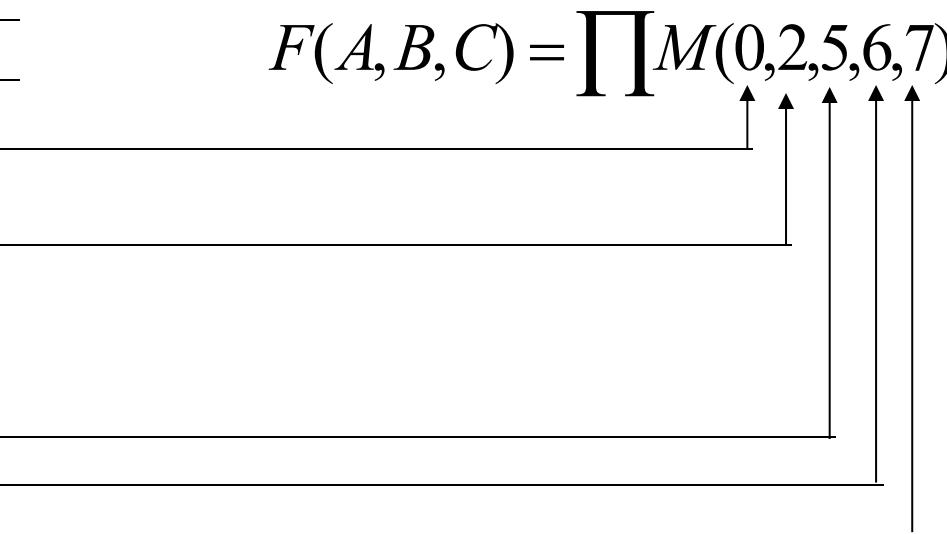
解：首先，列出F的真值表如下表所示。然后，根据真值表直接写出F的最大项表达式

$$F(A, B, C) = \prod M(0, 2, 5, 6, 7)$$

函数 $F(A, B, C) = \overline{AC} + A\overline{B} \cdot \overline{C}$ 的真值表

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$$F(A, B, C) = \prod M(0, 2, 5, 6, 7)$$





由于函数的真值表与函数的两种标准表达式之间存在一一对应的关系，而任何一个逻辑函数的真值表是唯一的，可见，任何一个逻辑函数的两种标准形式也是唯一的。

逻辑函数表达式的唯一性给我们分析和研究逻辑问题带来了很大的方便。

2.3 逻辑函数化简

实现某一逻辑功能的逻辑电路的复杂性与描述该功能的逻辑表达式的复杂性直接相关。

为了降低系统成本、减小复杂度、提高可靠性，必须对逻辑函数进行化简。

由于“与-或”表达式和“或-与”表达式可以很方便地转换成任何其他所要求的形式。因此，从这两种基本形式出发讨论函数化简问题，并将重点放在“与-或”表达式的化简上。

逻辑函数化简有3种常用方法。即：**代数化简法**、**卡诺图化简法**和**列表化简法**。

2.4.1 代数化简法

代数化简法就是运用逻辑代数的公理、定理和规则对逻辑函数进行化简的方法。

一、“与-或”表达式的化简

最简“与-或”表达式应满足两个条件：

1. 表达式中的“与”项个数最少；
2. 在满足上述条件的前提下，每个“与”项中的变量个数最少。

满足上述两个条件可以使相应逻辑电路中所需门的数量以及门的输入端个数均为最少，从而使电路最经济。

几种常用方法如下：

1. 并项法

利用定理7中的 $A\bar{B} + AB = A$ ，将两个“与”项合并成一个“与”项，合并后消去一个变量。例如，

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

2. 吸收法

利用定理3中 $A + AB = A$ ，吸收多余的项。例如，

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



逻辑代数基础

3 . 消去法

利用定理4中, $A + \overline{AB} = A + B$ 消去多余变量。例如,

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

4 . 配项法

利用公理4和公理5中的 $A \cdot 1 = A$ 及 $A + \overline{A} = 1$, 先从函数式中适当选择某些“与”项, 并配上其所缺的一个合适的变量, 然后再利用并项、吸收和消去等方法进行化简。例如,

$$\begin{aligned} & A\overline{B} + B\overline{C} + \overline{B}C + \overline{A}B \\ &= A\overline{B} + B\overline{C} + (A + \overline{A}) \cdot \overline{B}C + \overline{A}B \cdot (C + \overline{C}) \\ &= \underbrace{A\overline{B} + B\overline{C} + A\overline{B}C}_{\text{配项}} + \underbrace{\overline{B}C + \overline{A}B}_{\text{吸收}} + \overline{A}B\overline{C} \\ &= A\overline{B} + B\overline{C} + \overline{A}C \end{aligned}$$



逻辑代数基础

实际应用中遇到的逻辑函数往往比较复杂，化简时应灵活使用所学的公理、定理及规则，综合运用各种方法。

例 化简 $F = BC + D + \overline{D} \cdot (\overline{B} + \overline{C}) \cdot (AD + B\overline{C})$

解

$$\begin{aligned} F &= BC + D + \overline{D} \cdot (\overline{B} + \overline{C}) \cdot (AD + B\overline{C}) \\ &= BC + D + (\overline{B} + \overline{C}) \cdot (AD + B\overline{C}) \\ &= BC + D + \overline{BC} \cdot (AD + B\overline{C}) \\ &= BC + D + AD + B\overline{C} \\ &= B + D \end{aligned}$$

例 化简 $F = \overline{\overline{A}}(\overline{B} + \overline{C})(A + \overline{B} + C)\overline{\overline{\overline{ABC}}}$

解
$$\begin{aligned} F &= \overline{\overline{A}}(\overline{B} + \overline{C}) \cdot (A + \overline{B} + C)\overline{\overline{\overline{ABC}}} \\ &= (A + \overline{\overline{B}} + \overline{\overline{C}})(A + \overline{B} + C)(A + B + C) \\ &= (A + \overline{B}C) \cdot (A + C) \\ &= A + \overline{B}CC \\ &= A + \overline{B}C \end{aligned}$$

随堂练习

1. $F = \overline{AB} + \overline{A}CD + \overline{B}CD$
2. $F = ABC + A\overline{B}\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$
3. $F = \overline{A}B + A\overline{B} + \overline{A}\overline{B}C + ABC$
4. $F = \overline{A}B + AC + \overline{B}\overline{C} + A\overline{B} + \overline{A}\overline{C} + BC$
5. $F = ACE + \overline{A}BE + \overline{B}\overline{C}\overline{D} + BE\overline{C} + DE\overline{C} + \overline{A}E$

二、“或-与”表达式的化简

最简“或-与”表达式应满足两个条件：

1. 表达式中的“或”项个数最少；
2. 在满足上述条件的前提下，每个“或”项中的变量个数最少。

用代数化简法化简“或-与”表达式可直接运用公理、定理中的“或-与”形式，并综合运用前面介绍“与-或”表达式化简时提出的各种方法进行化简。



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

解
$$\begin{aligned} F &= (A + B) \cdot (\overline{B} + C) \cdot (A + C + \overline{D}) \cdot (A + C) \\ &= (A + B) \cdot (\overline{B} + C) \cdot (A + C) \\ &= (A + B)(\overline{B} + C) \end{aligned}$$

此外，可以采用两次对偶法。具体如下：

第一步：对“或-与”表达式表示的函数F求对偶，得到“与-或”表达式 F' ；

第二步：求出 F' 的最简“与-或”表达式；

第三步：对 F' 再次求对偶，即可得到F的最简“或-与”表达式。



逻辑代数基础

例 化简 $F = (A + \bar{B}) \cdot (\bar{A} + B) \cdot (B + C) \cdot (\bar{A} + C)$

解 第一步：求F的对偶式F'；

$$F' = A\bar{B} + \bar{A}B + BC + \bar{A}C$$

第二步：化简F'；

$$\begin{aligned} F' &= A\bar{B} + \bar{A}B + BC + \bar{A}C \\ &= A\bar{B} + \bar{A}B + (B + \bar{A})C \\ &= A\bar{B} + \bar{A}B + A\bar{B}C \\ &= A\bar{B} + A\bar{B} + C \end{aligned}$$

第三步：对F'求对偶，得到F的最简“或-与”表达式。

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

归纳：

代数化简法的优点是：不受变量数目的约束；当对公理、定理和规则十分熟练时，化简比较方便。

缺点是：没有一定的规律和步骤，技巧性很强，而且在很多情况下难以判断化简结果是否最简。



逻辑代数基础

2.4.1 代卡诺图化简法

卡诺图化简法具有**简单、直观、容易掌握等优点**，在逻辑设计中得到广泛应用。

一、卡诺图的构成

卡诺图是一种平面方格图，每个小方格代表一个最小项，故又称为**最小项方格图**。

结构特点：

- (1) n 个变量的卡诺图由 2^n 个小方格构成；
- (2) 几何图形上处在**相邻、相对、相重**位置的小方格所代表的最小项为相邻最小项。

逻辑代数基础



2变量、3变量、4变量卡诺图如图(a)、(b)、(c)所示。

		A	0	1
		B	m ₀	m ₂
0	0	m ₁	m ₃	
	1			

(a)

		A	B	00	01	11	10	
		C	m ₀	m ₂	m ₆	m ₄		
0	0	m ₁	m ₃	m ₇	m ₅			
	1							

(b)

		A	B	00	01	11	10	
		CD	m ₀	m ₄	m ₁₂	m ₈		
00	01	m ₁	m ₅	m ₁₃	m ₉			
	11	m ₃	m ₇	m ₁₅	m ₁₁			
10	00	m ₂	m ₆	m ₁₄	m ₁₀			

(c)

逻辑代数基础



从各卡诺图可以看出，在n个变量的卡诺图中，能从图形上直观、方便地找到每个最小项的n个相邻最小项。

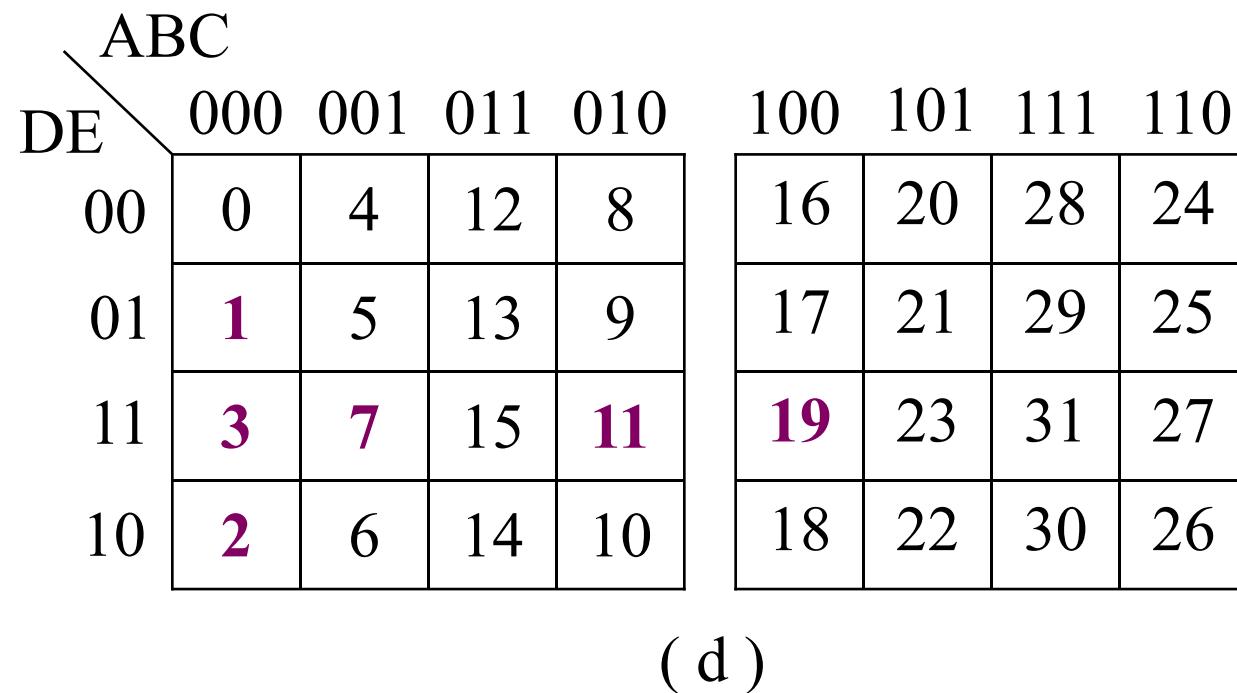
例如，四变量卡诺图中，如 m_5 的4个相邻最小项分别是和 m_5 相连的 m_1 , m_4 , m_7 m_{13} 。

m_2 的4个相邻最小项除了与之几何相邻的 m_3 和 m_6 之外，另外两个是处在“相对”位置的 m_0 （同一列的两端）和 m_{10} （同一行的两端）。这种相邻称为相对相邻。

AB	00	01	11	10
CD \	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

此外，处在“相重”位置的最小项相邻，如五变量卡诺图中的 m_3 ，除了几何相邻的 m_1 ， m_2 ， m_7 和相对相邻的 m_{11} 外，还与 m_{19} 相邻。这种相邻称为**重叠相邻**。

5变量卡诺图如图 (d) 所示。



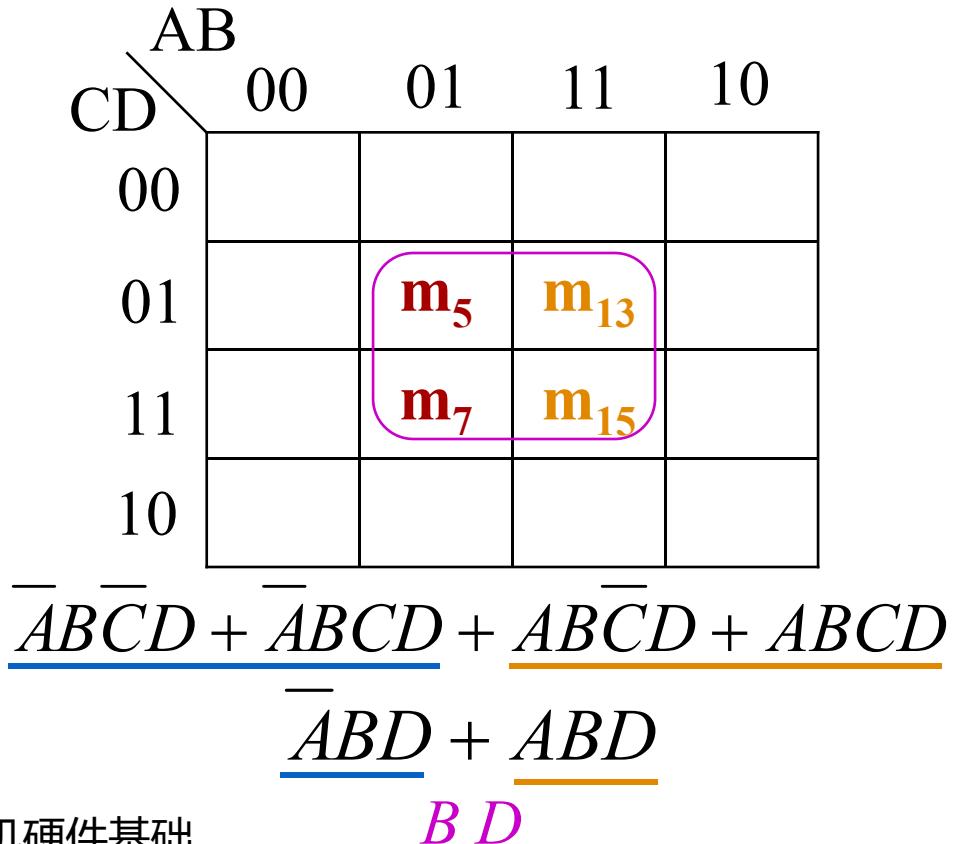
5变量卡诺图



逻辑代数基础

二、卡诺图的性质

性质：可以从图形上直观地找出相邻最小项合并。合并的理论依据是并项定理： $\bar{A}\bar{B} + A\bar{B} = A$



用卡诺图化简逻辑函数的基本原理：把卡诺图上表征相邻最小项的相邻小方格“圈”在一起进行合并，达到用一个简单“与”项代替若干最小项的目的。

通常把用来包围那些能由一个简单“与”项代替的若干最小项的“圈”称为卡诺圈。

三、逻辑函数在卡诺图上的表示

1. 给定逻辑函数为标准“与-或”表达式

当逻辑函数为标准“与-或”表达式时，只需在卡诺图上找出和表达式中最小项对应的小方格填上1，其余小方格填上0，即可得到该函数的卡诺图。

例如，3变量函数 $F(A, B, C) = \sum m(1, 2, 3, 7)$ 的卡诺图如下图所示。

		AB	00	01	11	10
		C	0	1	0	0
A	B	0	0	1	0	0
		1	1	1	1	0

$F(A, B, C) = \sum m(1, 2, 3, 7)$ 的卡诺图



逻辑代数基础

2. 逻辑函数为一般“与-或”表达式

当逻辑函数为一般“与-或”表达式时，可根据“与”的公共性和“或”的叠加性作出相应卡诺图。

例如，4变量函数 $F(A, B, C, D) = AB + CD + \overline{A}\overline{B}C$ 的卡诺图如图所示。

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

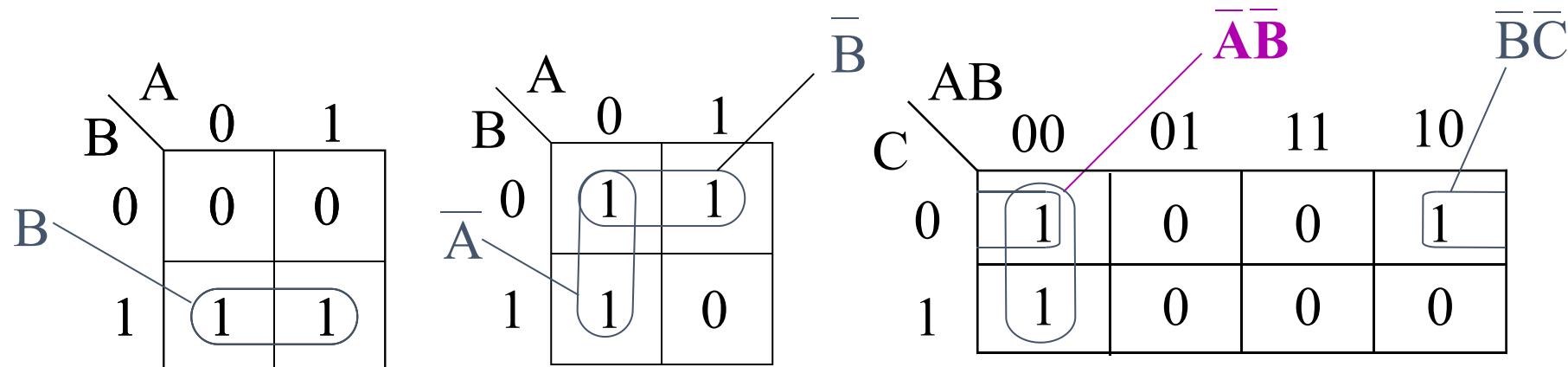
为了叙述的方便，通常将卡诺图上填1的小方格称为**1方格**，填0的小方格称为**0方格**。**0方格有时用空格表示**。

四、卡诺图上最小项的合并规律

当一个函数用卡诺图表示后，究竟哪些最小项可以合并呢？下面以2、3、4变量卡诺图为例予以说明。

1. 两个小方格相邻，或处于某行(列)两端时，所代表的最小项可以合并，合并后可消去一个变量。

例如，下图给出了2、3变量卡诺图上两个相邻最小项合并的典型情况的。



两个相邻最小项合并的情况

2. 四个小方格组成一个大方格、或组成一行（列）、或处于相邻两行（列）的两端、或处于四角时，所代表的最小项可以合并，合并后可消去两个变量。

例如，下图给出了3变量卡诺图上四个相邻最小项合并的典型情况的。

		AB	C			
		00	01	11		
		0	1	0	0	1
		1	1	0	0	1

		AB	C			
		00	01	11		
		0	0	1	1	0
		1	0	1	1	0

逻辑代数基础



4变量卡诺图上四个相邻最小项合并的典型情况：

		AB	$\bar{B}\bar{D}$		BD	
		CD	00	01	11	10
AB	CD	00	1	0	0	1
		01	0	1	1	0
AB	CD	11	0	1	1	0
		10	1	0	0	1

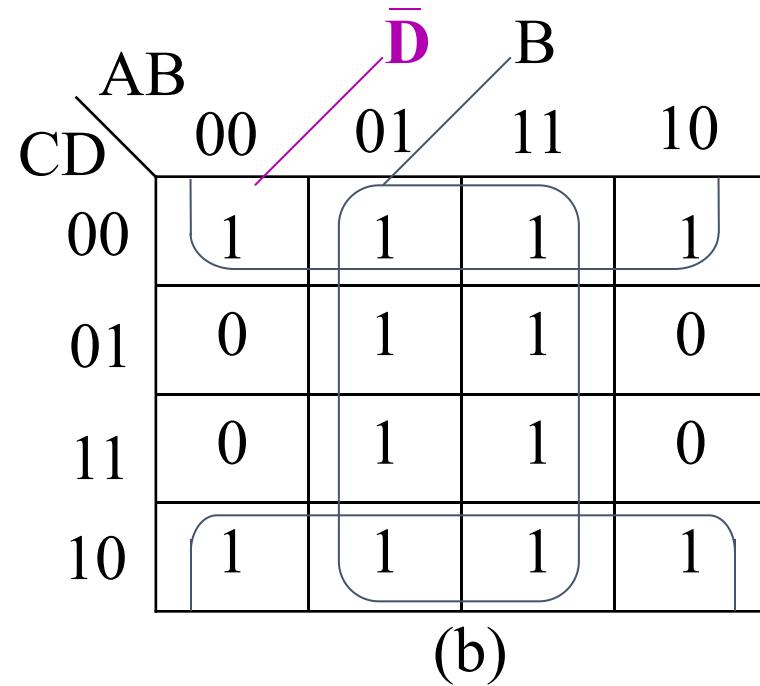
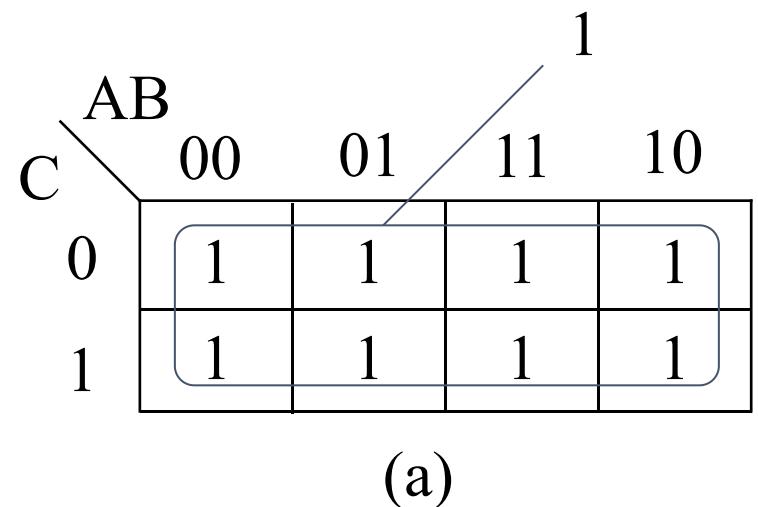
		AB	$\bar{B}D$		$B\bar{D}$	
		CD	00	01	11	10
AB	CD	00	0	1	1	0
		01	1	0	0	1
AB	CD	11	1	0	0	1
		10	0	1	1	0

		AB	$\bar{A}B$		CD	
		CD	00	01	11	10
AB	CD	00	0	1	0	0
		01	0	1	0	0
AB	CD	11	1	1	1	1
		10	0	1	0	0

四个相邻最小项合并的几种情况

3. 八个小方格组成一个大方格、或组成相邻的两行(列)、或处于两个边行(列)时，所代表的最小项可以合并，合并后可消去三个变量。

例如，下图给出了3、4变量卡诺图上八个相邻最小项合并的典型情况的。



8个相邻最小项合并的两种情况

n个变量卡诺图中最小项的合并规律归纳如下：

- (1) 卡诺圈中小方格的个数必须为 2^m 个，m为小于或等于n的整数。
- (2) 卡诺圈中的 2^m 个小方格有一定的排列规律，具体地说，它们含有m个不同变量，(n-m)个相同变量。
- (3) 卡诺圈中的 2^m 个小方格对应的最小项可用(n-m)个变量的“与”项表示，该“与”项由这些最小项中的相同变量构成。
- (4) 当m = n时，卡诺圈包围了整个卡诺图，可用1表示，即n个变量的全部最小项之和为1。



逻辑代数基础

五、卡诺图化简逻辑函数的步骤

1. 几个定义

蕴涵项:在函数的“与-或”表达式中，每个“与”项被称为该函数的蕴涵项(Implicant)。

在函数卡诺图中，任何一个1方格所对应的最小项或者卡诺圈中的 2^m 个1方格所对应的“与”项都是函数的蕴涵项。

质蕴涵项:若函数的一个蕴涵项不是该函数中其他蕴涵项的子集，则此蕴涵项称为质蕴涵项(Prime Implicant),简称为质项。

在函数卡诺图中，按照最小项合并规律，如果某个卡诺圈不可能被其他更大的卡诺圈包含，那么，该卡诺圈所对应的“与”项为质蕴涵项。

必要质蕴涵项：若函数的一个质蕴涵项包含有不被函数的其他任何质蕴涵项所包含的最小项，则此质蕴涵项被称为必要质蕴涵项(Essential Prime Implicant)，简称为必要质项。

在函数卡诺图中，若某个卡诺圈包含了不可能被任何其他卡诺圈包含的1方格，那么，该卡诺圈所对应的“与”项为必要质蕴涵项。

2. 求逻辑函数最简“与-或”表达式的一般步骤

第一步：作出函数的卡诺图；



第二步：在卡诺图上圈出函数的全部质蕴涵项；



第三步：从全部质蕴涵项中找出所有必要质蕴涵项；



第四步：求函数的最简质蕴涵项集。

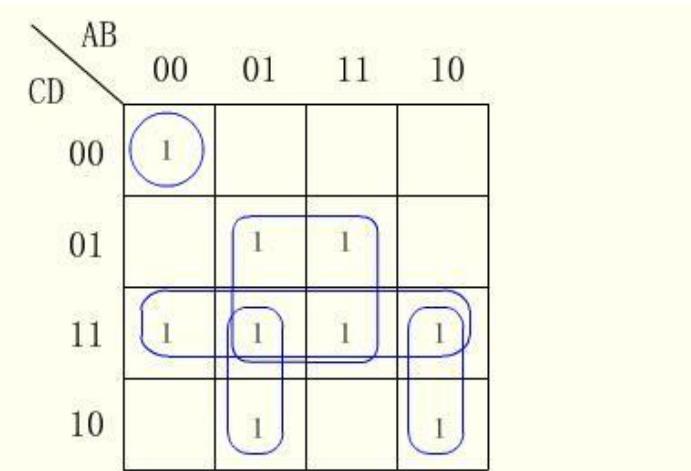
当函数的所有必要质蕴涵项尚不能覆盖卡诺图上的所有1方格时，则从剩余质蕴涵项中找出最简的所需质蕴涵项，使它和必要质蕴涵项一起构成函数的最小覆盖。

逻辑代数基础



例 用卡诺图化简逻辑函数 $F(A, B, C, D) = \sum m(0, 3, 5, 6, 7, 10, 11, 13, 15)$

解 用卡诺图化简给定函数的过程如下图所示。



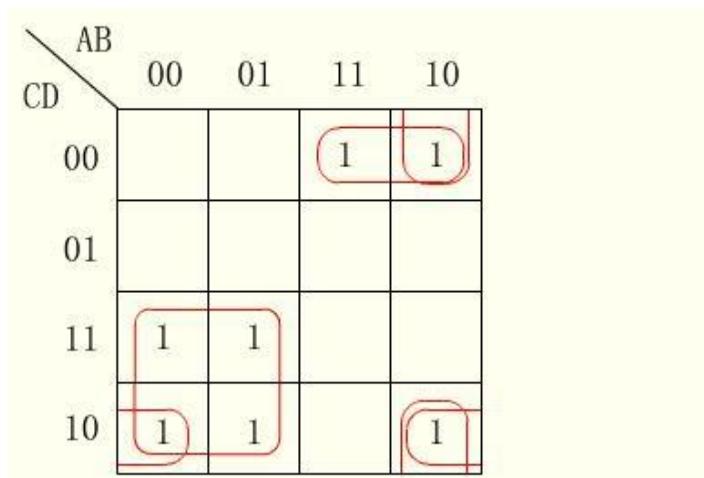
该题中，5个必要质蕴涵项已将函数的全部最小项覆盖，故将各卡诺圈对应的与项相或即可得到函数F的最简“与-或”表达式为

$$F(A, B, C, D) = \overline{ABCD} + \overline{ABC} + A\overline{B}C + BD + CD$$

例 用卡诺图化简逻辑函数

$$F(A, B, C, D) = \sum m(2, 3, 6, 7, 8, 10, 12)$$

解 用卡诺图化简给定函数的过程如下图所示。



即 $F(A, B, C, D) = \overline{A}C + \overline{A}\overline{C}D' + A\overline{B}D'$
或者 $F(A, B, C, D) = \overline{A}C + \overline{A}\overline{C}D' + \overline{B}CD'$

这里，两个“与-或”式的复杂程度相同。由此可见，一个函数的最简“与-或”表达式不一定是唯一的。

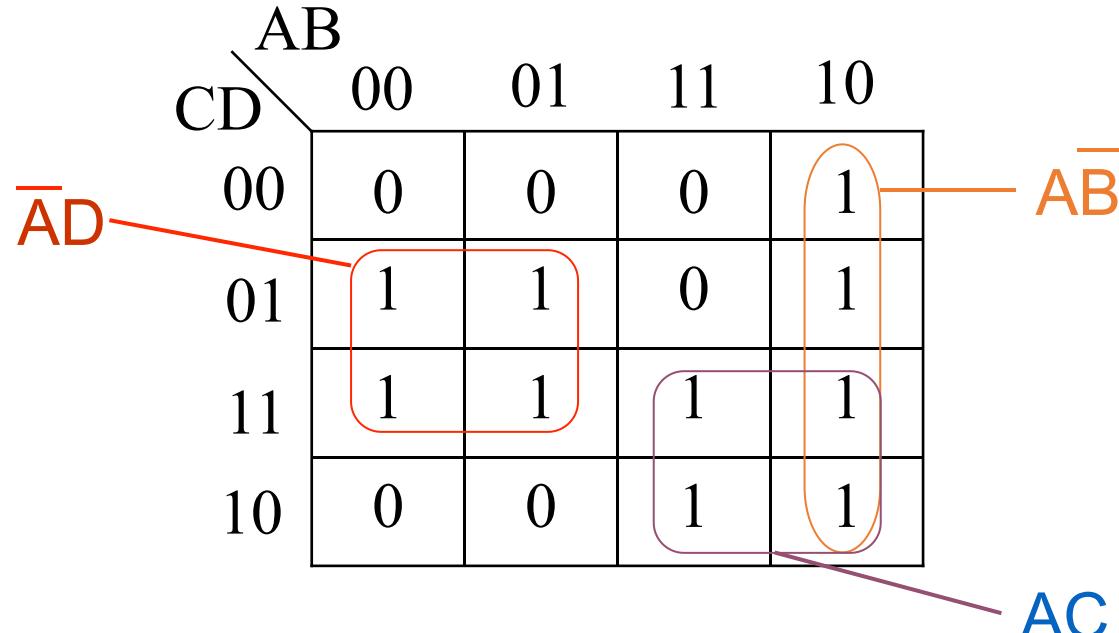


逻辑代数基础

例 用卡诺图化简逻辑函数

$$F(A, B, C, D) = \sum m(1, 3, 5, 7, 8, 9, 10, 11, 14, 15)$$

解 用卡诺图化简给定函数，如下图所示。



函数F的最简“与-或”表达式为

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

用卡诺图化简总的原则是：

- (1) 在覆盖函数中所有最小项的前提下，卡诺圈的个数 应达到最少；
- (2) 在满足合并规律的前提下卡诺圈应达到最大；
- (3) 根据合并的需要，每个最小项可以被多个卡诺圈包围。

练习题 5-(2), 6

3. 求逻辑函数最简“或-与”表达式的一般步骤

●当给定逻辑函数为“与-或”表达式或标准“与-或”表达式时，通常采用“两次取反法”。具体如下：

(1) 作出F的卡诺图，求出反函数 \bar{F} 的最简“与-或”表达式(合并卡诺图上的0方格)；

(2) 对 \bar{F} 的最简“与-或”表达式取反，得到函数F的最简“或-与”表达式。

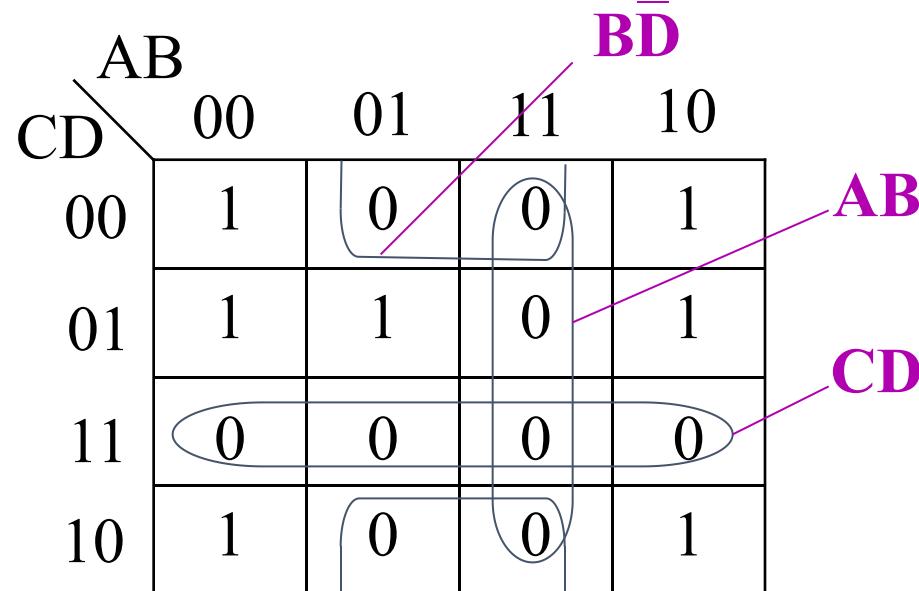
逻辑代数基础



例 用卡诺图求逻辑函数 $F(A, B, C, D) = \sum m(0,1,2,5,8,9,10)$ 的最简“或-与”表达式。

解 用卡诺图化简给定函数的过程如右图所示，得到：

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



再对反函数 \overline{F} 的最简“与-或”表达式 $\overline{F} = AB + CD + B\bar{D}$ 两边取反，即可求得函数的最简“或-与”表达式

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



● 当给定逻辑函数为“或-与”表达式或标准“或-与”表达式时，通常采用“两次对偶法”。具体如下：

(1) 作出 F 对偶式 F' 的卡诺图，并求出 F' 的最简“与-或”表达式；

(2) 对 F' 的最简“与-或”表达式取对偶，得到函数 F 的最简“或-与”表达式。

**补充：例题讲解2-10，
练习题 5-(4)**



卡诺图化简逻辑函数具有方便、直观、容易掌握等优点。但受到变量个数的约束，当变量个数大于6时，画图以及对图形的识别都变得相当复杂。

为了克服卡诺图的不足，引入了另一种化简方法——列表化简法。

(详见教材有关内容)