

第 2 章 逻辑代数基础

逻辑代数是布尔代数应用于数字系统与数字电路的产物，是数字系统与数字电路分析和设计的数学理论基础和工具。无论何种形式的数字系统，都是由一些基本的逻辑电路所组成的。为了解决数字系统分析和设计中的各种具体问题，必须掌握逻辑代数这一重要数学工具。本章将从实用的角度介绍逻辑代数的基本概念、基本定理和规则、逻辑代数的表示形式以及逻辑函数的化简。

2.1 逻辑代数的基本概念

英国数学家 George Boole 于 1854 年提出了将人的逻辑思维规律和推理过程归结为一种数学运算的代数系统，即布尔代数 (Boolean Algebra)。1938 年贝尔实验室研究员 Claude E. Shannon 将布尔代数的一些基本前提和定理应用于继电器电路的分析与描述上，提出了开关代数的概念。随着数字电子技术的发展，机械触点开关逐步被无触点电子开关所取代，现在已经很少使用“开关代数”这个概念了，为了与数字系统逻辑设计相适应，人们现在更多地习惯采用“逻辑代数”这个概念，逻辑代数是布尔代数的特例，它是采用二值逻辑运算的基本数学工具，主要研究数字电路输入和输出之间的因果关系，即输入和输出之间的逻辑关系。因此，逻辑代数是布尔代数应用于数字系统与数字电路的产物，是数字系统与数字电路分析和设计的数学理论基础工具，被广泛地使用于计算机、通信、自动化等领域研究数字电路。本章将着重介绍逻辑代数的基本概念、基本定律和规则、逻辑函数的表示及逻辑函数的化简。

2.1.1 逻辑代数的定义

逻辑代数是代数形式来研究逻辑问题的数学工具。其变量可以用字符 A, B, C, D 等表示，并被称为逻辑变量。逻辑变量只有两种取值，一般用“0”和“1”来表示，而这里的“0”和“1”仅仅是一种符号的代表，没有数量的含义，也没有大小和正负的含义，只是用来表示研究问题的两种可能性，如电平的高与低、电流的有与无、命题的真与假、事物的是与非、开关的通与断。

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“与”、“或”、“非”三种基本运算构成，记为 $L = \{K, +, \cdot, -, 0, 1\}$ 。这个代数系统满足下列公理。

公理 1 交换律

对于任意的逻辑变量 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

对于任意的逻辑变量 A，有：

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

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

公理5 互补律

对于任意的逻辑变量 A，存在唯一的 \bar{A} ，使得：

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

2.1.2 逻辑代数的基本运算

在数字系统与数字电路中，开关的通与断、电平的高与低、信号的有与无等两种稳定的物理状态都可以用“0”和“1”这两个逻辑值来表示，但是对于一个复杂数字系统，仅用逻辑变量的取值来反映单个开关元件的两种状态是不够的，还必须反映这个复杂数字系统中各开关元件之间的联系，为了描述这些联系，逻辑代数中定义了“与”、“或”、“非”三种基本运算。

一、与运算

在逻辑问题中，某个事件受若干个条件影响，若所有的条件都成立，则该事件才能成立，这样的逻辑关系被称为与逻辑。

例如，图 2.1 的电路中，灯 F 由开关 A 和开关 B 串联控制，开关 A 和开关 B 的状态组合有四种，这四种不同的状态组合与灯 F 的亮灭之间的关系如表 2-1 所示，由表 2-1 可见，只有当开关 A 和开关 B 同时闭合时，灯 F 才亮，否则，灯 F 灭。因此灯 F 与开关 A 和开关 B 之间的关系就形成了与逻辑关系。

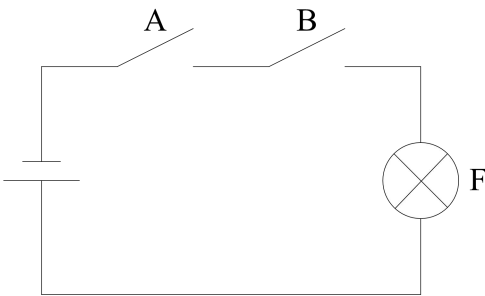


图 2.1 串联开关电路

表 2-1 串联开关电路功能表

开关 A 的状态	开关 B 的状态	灯 F 的状态
断开	断开	灭
断开	闭合	灭
闭合	断开	灭
闭合	闭合	亮

在逻辑代数中，与逻辑关系用与运算来描述，与运算又称逻辑乘，其运算符号为“ \cdot ”或“ \wedge ”。则图 2.1 的电路可以用与运算表示为：

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

与运算符“ \cdot ”也可以省略，即 $F = A \cdot B = AB$ 。

为了用逻辑代数来分析图 2.1 的电路，假设“0”表示开关断开，“1”表示开关闭合，“0”表示灯灭，“1”表示灯亮，这样电路状态关系表 2-1 可变换成表 2-2 所示的真值表。所谓真值表是指把逻辑变量的所有可能的取值组成及其对应的结果构成一个二维表格，这张表被称为真值表。因此与运算可以用表 2-2 描述。

表 2-2 与运算真值表

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

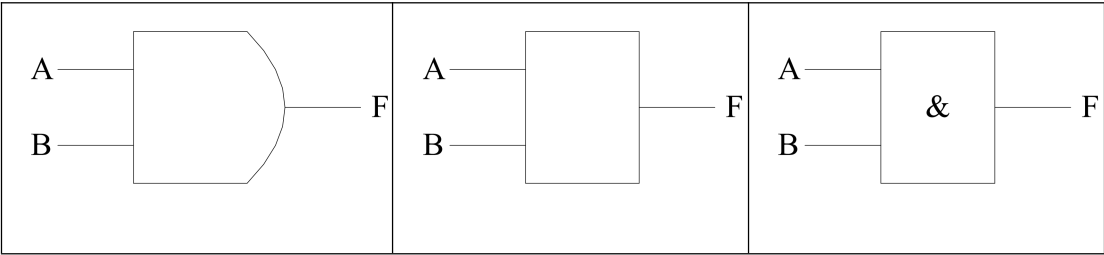
由表 2-2 可得出与运算的运算法则为：

$$\begin{aligned} 0 \cdot 0 &= 0 & 0 \cdot 1 &= 0 \\ 1 \cdot 0 &= 0 & 1 \cdot 1 &= 1 \end{aligned}$$

在数字系统与电路中，实现与运算的逻辑电路称为“与门”，其逻辑符号为如表 2-3 所示。本书采用国家标准符号。

表 2-3 与门逻辑符号

国际常用符号	曾用符号	国家标准符号
--------	------	--------



二、或运算

在逻辑问题中，某个事件受若干个条件影响，只要其中一个或一个以上条件成立，则该事件便可成立，这样的逻辑关系被称为或逻辑。

例如，图 2.2 的电路中，灯 F 由开关 A 和开关 B 并联控制，开关 A 和开关 B 的状态组合有四种，这四种不同的状态组合与灯 F 的亮灭之间的关系如表 2-4 所示，由表 2-4 可见，只要开关 A 和开关 B 中有一个闭合或两个均闭合时，灯 F 便亮，否则，灯 F 灭。因此灯 F 与开关 A 和开关 B 之间的关系就形成了或逻辑关系。

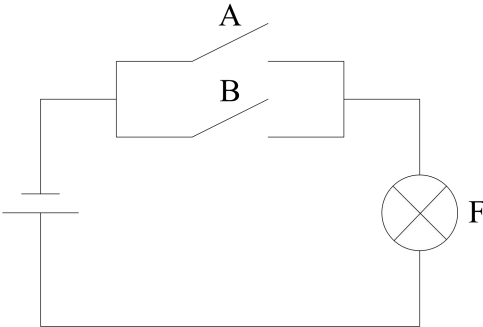


图 2.2 并联开关电路

表 2-4 并联开关电路功能表

开关 A 的状态	开关 B 的状态	灯 F 的状态
断开	断开	灭
断开	闭合	亮
闭合	断开	亮
闭合	闭合	亮

在逻辑代数中，或逻辑关系用或运算来描述，或运算又称逻辑加，其运算符号为“+”或“ \vee ”。则图 2.2 的电路可以用与运算表示为：

$F = A + B$ 或者 $F = A \vee B$

为了用逻辑代数来分析图 2.2 的电路，假设“0”表示开关断开，“1”表示开关闭合，“0”表示灯灭，“1”表示灯亮，这样电路状态关系表 2-4 可变换成表 2-5 所示的真值表。因此或运算可以用表 2-5 描述。

表 2-5 或运算真值表

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

由表 2-5 可得出或运算的运算法则为：

$0 + 0 = 0$

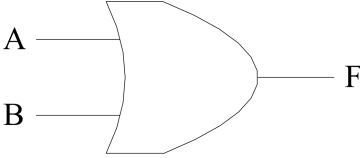
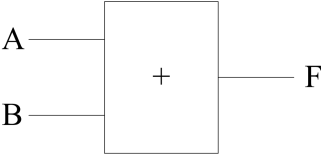
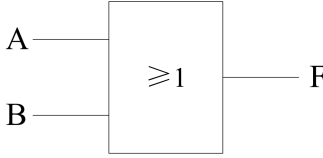
$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 1$

在数字系统与电路中，实现或运算的逻辑电路称为“或门”，其逻辑符号为如表 2-6 所示。本书采用国家标准符号。

表 2-6 或门逻辑符号

国际常用符号	曾用符号	国家标准符号
		

三、非运算

在逻辑问题中，某个事件的成立取决于条件的否定，即事件与事件的成立条件之间构成矛盾，这样的逻辑关系被称为非逻辑。

例如，图 2.3 的电路中，灯 F 和开关 A 并联，开关 A 两种不同的状态与灯 F 的亮灭之间的关系如表 2-7 所示，由表 2-7 可见，开关 A 闭合，则灯 F 灭，否则，灯 F 亮。因此灯 F 与开关 A 之间的关系就形成了非逻辑关系。

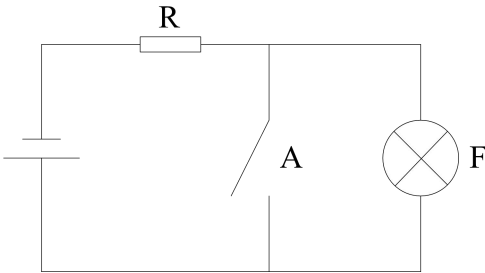


图 2.3 开关与灯并联电路

表 2-7 开关与灯并联电路功能表

开关 A 的状态	灯 F 的状态
断开	亮

闭合	灭
----	---

在逻辑代数中，非逻辑关系用非运算来描述，非运算又称逻辑反，其运算符号为“ $-$ ”。则图 2.3 的电路可以用与运算表示为：

$$F = \overline{A}$$

为了用逻辑代数来分析图 2.3 的电路，假设“0”表示开关断开，“1”表示开关闭合，“0”表示灯灭，“1”表示灯亮，这样电路状态关系表 2-7 可变换成表 2-8 所示的真值表。因此非运算可以用表 2-8 描述。

表 2-8 非运算真值表

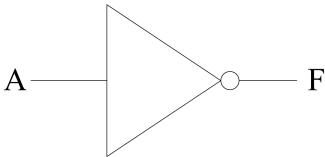
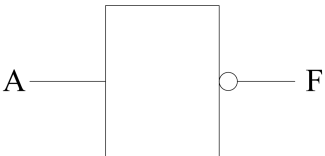
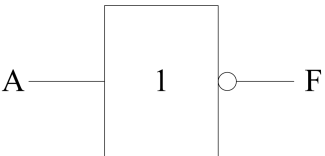
A	F
0	1
1	0

由表 2-8 可得出或运算的运算法则为：

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

在数字系统与电路中，实现非运算的逻辑电路称为“非门”，其逻辑符号为如表 2-9 所示。本书采用国家标准符号。

表 2-9 非门逻辑符号

国际常用符号	曾用符号	国家标准符号
		

2.1.3 逻辑代数的复合运算

在实际电路中，利用与门、或门和非门之间的不同组合可构成复合门电路，完成复合逻辑运算。常见的复合门电路有与非门、或非门、与或非门、异或门和同或门。

一、与非逻辑

与逻辑和非逻辑的复合逻辑称为与非逻辑，它可以看成与逻辑后面加了一个非逻辑，实现与非逻辑的电路称为与非门。其表达式如下：

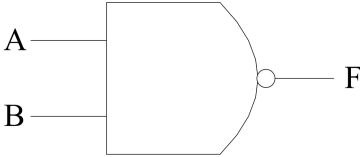
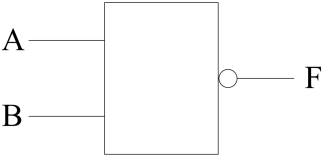
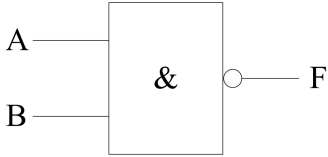
$$F = \overline{A \cdot B} = \overline{AB}$$

与非运算的真值表见表 2-10，与非门的逻辑符号见表 2-11。

表 2-10 与非运算真值表

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

表 2-11 与非门逻辑符号

国际常用符号	曾用符号	国家标准符号
		

二、或非逻辑

或逻辑和非逻辑的复合逻辑称为或非逻辑，它可以看成或逻辑后面加了一个非逻辑，实现或非逻辑的电路称为或非门。其表达式如下：

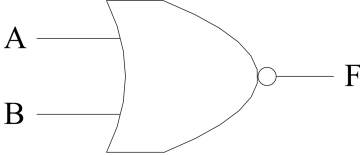
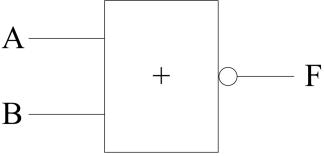
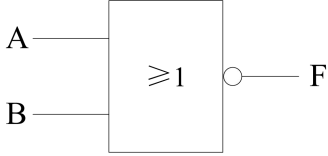
$$F = \overline{A + B}$$

或非运算的真值表见表 2-12，或非门的逻辑符号见表 2-13。

表 2-12 或非运算真值表

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

表 2-13 或非门逻辑符号

国际常用符号	曾用符号	国家标准符号
		

三、与或非逻辑

与或非逻辑是三种基本逻辑（与、或、非）的组合，也可以看成是与逻辑和或非逻辑的组合。其表达式如下：

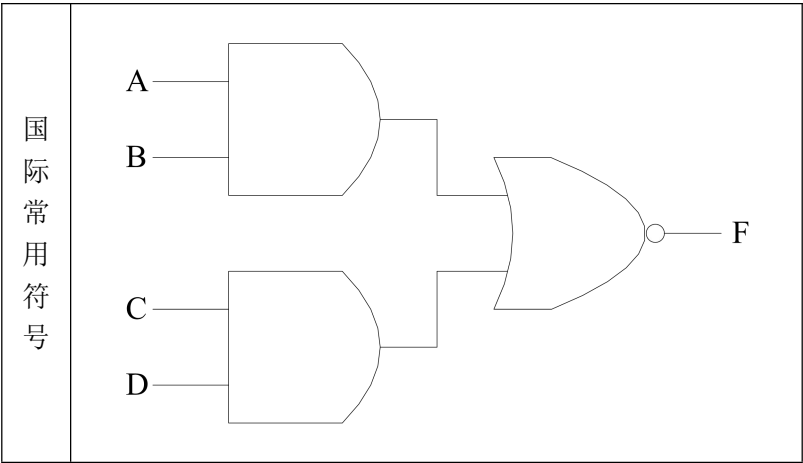
$$F = \overline{AB+CD}$$

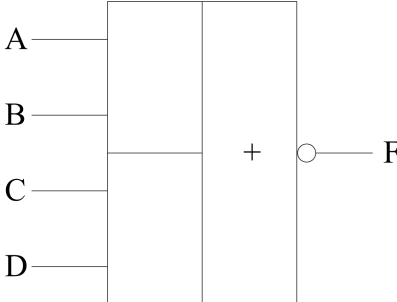
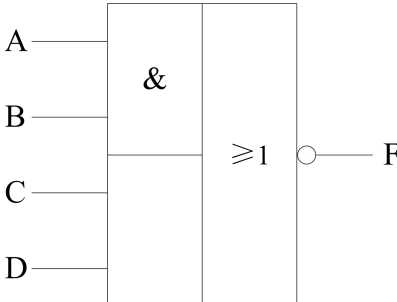
与或非运算的真值表见表 2-14，与或非门的逻辑符号见表 2-15。

表 2-14 与或非运算真值表

A	B	C	D	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

表 2-15 与或非门逻辑符号



曾用符号	
国家标准符号	

四、异或逻辑

异或逻辑是指当两个输入逻辑变量取值不同时输出为 1，相同时输出为 0。实现异或逻辑的电路称为异或门。其表达式如下：

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

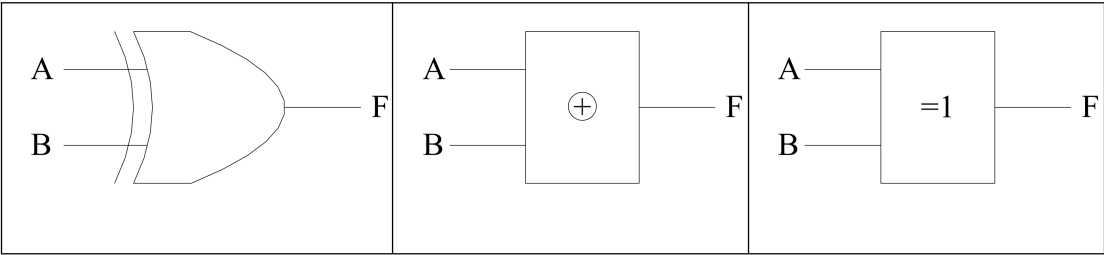
异或运算的真值表见表 2-16，异或门的逻辑符号见表 2-17。

表 2-16 异或运算真值表

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

表 2-17 异或门逻辑符号

国际常用符号	曾用符号	国家标准符号
--------	------	--------



五、同或逻辑

同或逻辑是指当两个输入逻辑变量取值相同时输出为 1，不同时输出为 0，偶数个变量的同或逻辑和异或逻辑互为反运算。实现同或逻辑的电路称为同或门。其表达式如下：

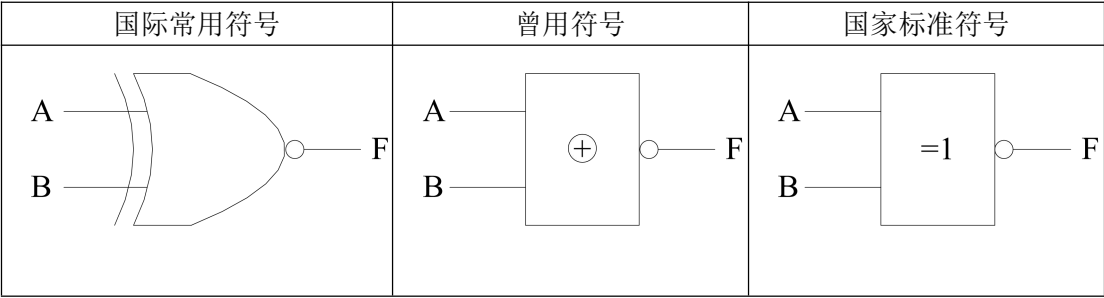
$$F = A \odot B = AB + \overline{A}\overline{B}$$

同或运算的真值表见表 2-18，同或门的逻辑符号见表 2-19。

表 2-18 同或运算真值表

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

表 2-19 同或门逻辑符号



2.1.4 逻辑函数的表示法及逻辑函数间的相等

一、逻辑函数的定义

逻辑函数中函数的定义与普通代数中函数的定义极为相似，同时逻辑函数具有其自身的特点：

- (1) 逻辑变量和逻辑函数的取值只有 0 和 1 两种可能；
- (2) 逻辑函数和逻辑变量之间的关系是由“与”、“或”、“非”三种基本运算决定的。

从数字电路的角度看，逻辑函数可以定义为：

设某一逻辑电路的输入变量为 A_1, A_2, \dots, A_n ，输出变量为 F ，当 A_1, A_2, \dots, A_n

的取值确定后，F 的值就被唯一确定下来，则称 F 是 A_1, A_2, \dots, A_n 的逻辑函数，记为：

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

二、逻辑函数的相等

与普通代数一样，逻辑函数也存在相等的问题。

如果有两个都是 n 个变量的逻辑函数：

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

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

若对于这 n 个逻辑变量的 2^n 种组合中的任意一组取值， F_1 和 F_2 都相等，则称函数 F_1 和 F_2 相等，记为： $F_1 = F_2$ 。

判断两个逻辑函数是否相等，通常有两种方法。一种方法是列出逻辑变量的所有可能的取值组合，并按逻辑运算计算出各种取值组合下两个函数的对应值，然后进行比较判断两个逻辑函数是否相等。另一种方法是用逻辑代数的公理、定理、公式和规则等进行数学证明。

三、逻辑函数的表示法

逻辑函数的表示方法有很多种，例如逻辑表达式、逻辑电路图、真值表、卡诺图、波形图、硬件描述语言等。本节重点讨论逻辑表达式、逻辑电路图、真值表和波形图，卡诺图和硬件描述语言将在后面的章节进行介绍。

1、逻辑表达式

逻辑表达式是逻辑变量和“与”、“或”、“非”三种运算符所构成的式子。将逻辑函数输入和输出之间的关系写成逻辑表达式就得到了逻辑函数的逻辑表达式。例如，逻辑函数

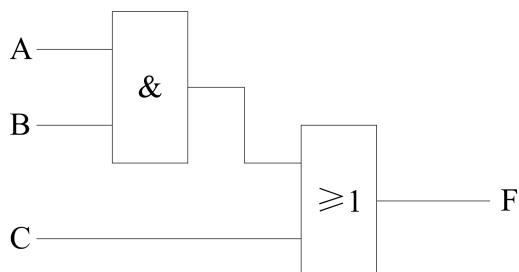
$$F = f(A, B, C) = AB + C$$

是一个由 A、B、C 三个变量进行逻辑运算所构成的逻辑表达式。

2、逻辑电路图

将逻辑函数的逻辑表达式中各变量之间的与、或、非等逻辑关系用逻辑门电路的图形符号表示出来，就得到了逻辑函数的逻辑电路图。

逻辑函数 $F = f(A, B, C) = AB + C$ 的逻辑电路图如图 2.4 所示。

图 2.4 逻辑函数 $F = f(A, B, C) = AB + C$ 的逻辑电路图

3、真值表

前面曾经提到过真值表，即将 n 个输入变量所有 2^n 个取值组合下对应的输出值计算出来，列成表格，就得到了逻辑函数的真值表。

逻辑函数 $F = f(A, B, C) = AB + C$ 的真值表如表 2-20 所示。

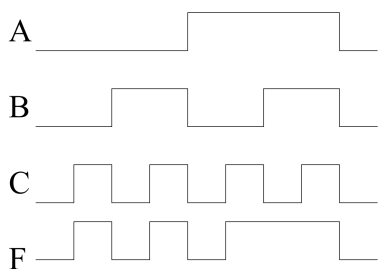
表 2-20 逻辑函数 $F = f(A, B, C) = AB + C$ 的真值表

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

4、波形图

用逻辑电平的高、低变化来动态地表示逻辑函数的输入变量与输出变量之间的关系，按时间顺序依次排列出来，就得到了逻辑函数的波形图，也称为时序图。由于波形图是一种动态图形语言，非常直观，在数字系统分析和测试中可以利用计算机仿真工具和逻辑分析仪等来分析逻辑函数。

逻辑函数 $F = f(A, B, C) = AB + C$ 的波形图如图 2.5 所示。

图 2.5 逻辑函数 $F = f(A, B, C) = AB + C$ 的波形图

2.2 逻辑代数的基本定律、规则和常用公式

根据逻辑代数中的与、或、非三种基本运算，可以推导出逻辑代数运算的一些基本定律、规则和常用公式。它们为逻辑函数的化简提供了理论依据，也为分析和设计逻辑电路提供了重要工具。

2.2.1 基本定律

根据 2.1.1 节的公理，可以推导出下列常用的定律。

定律 1 重叠律

$$A + A = A \quad AA = A$$

证明：

$$A + A = A \cdot 1 + A \cdot 1 \quad (\text{0-1 律})$$

$$= A \cdot (1 + 1) \quad (\text{分配律})$$

$$= A \cdot 1 \quad (\text{或运算法则})$$

$$= A \quad (\text{0-1 律})$$

该定律说明一个变量多次自与、自或的结果仍为其自身，即逻辑代数中不存在倍乘和方幂运算。

定律 2 吸收律

$$A + AB = A \quad A(A + B) = A$$

证明：

$$A + AB = A \cdot 1 + A \cdot B \quad (\text{分配律})$$

$$= A \cdot (1 + B) \quad (\text{互补律})$$

$$= A \cdot 1 \quad (\text{0-1 律})$$

$$= A \quad (\text{0-1 律})$$

该定律说明逻辑表达式中某一项包含了式中另一项，则该项可以去掉。

定律3 消去律

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

证明:

$$A + \overline{A}B = (A + \overline{A})(A + B) \quad (\text{分配律})$$

$$= 1 \cdot (A + B) \quad (\text{互补律})$$

$$= A + B \quad (\text{0-1律})$$

定律4 并项律

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

证明:

$$AB + A\overline{B} = A(B + \overline{B}) \quad (\text{分配律})$$

$$= A \cdot 1 \quad (\text{互补律})$$

$$= A \quad (\text{0-1律})$$

定律5 复原律

$$\overline{\overline{A}} = A$$

证明: 令 $\overline{\overline{A}} = X$, 因而存在唯一的 X, 使得:

$$X\overline{A} = 0, \quad X + \overline{A} = 1 \quad (\text{互补律})$$

但是 $A\overline{A} = 0, \quad A + \overline{A} = 1 \quad (\text{互补律})$

这样, X 和 A 都满足互补律, 因此根据互补律的唯一性, 可得 $A=X$, 即 $\overline{\overline{A}} = A$ 。

该定律说明了“否定的否定等于肯定”这一规律。

定律6 冗余律

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

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

证明:

$$AB + \overline{A}C + BC = AB + \overline{A}C + BC(A + \overline{A}) \quad (\text{互补律})$$

$$= AB + \overline{AC} + BCA + BC\overline{A} \quad (\text{分配律})$$

$$= AB + \overline{AC} + ABC + \overline{ABC} \quad (\text{交换律})$$

$$= AB(1 + C) + \overline{AC}(1 + B) \quad (\text{分配律})$$

$$= AB + \overline{AC} \quad (0-1 \text{ 律})$$

该定律说明当逻辑表达式中的某个变量（如 A）分别以原变量和反变量的形式出现在两项中时，该两项中其他变量（如 B、C）组成的第三项（如 BC）是多余的，可从式中去掉。

冗余律的推广：

$$AB + \overline{AC} + BCX_1X_2 \cdots X_n = AB + \overline{AC}$$

$$(A + B)(\overline{A} + C)(B + C + X_1 + X_2 + \cdots + X_n) = (A + B)(\overline{A} + C)$$

冗余律的推广说明若第三项中除了前两项的剩余部分以外，还含有其他部分，它仍然是多余的。

定律 7 摩根律

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

证明：

$$(\overline{AB}) + (A + B) = (\overline{AB} + A) + B \quad (\text{结合律})$$

$$= (\overline{B} + A) + B \quad (\text{消去律})$$

$$= A + (\overline{B} + B) \quad (\text{交换律、结合律})$$

$$= A + 1 \quad (\text{互补律})$$

$$= 1 \quad (0-1 \text{ 律})$$

而且

$$(\overline{AB})(A + B) = \overline{AB}A + \overline{AB}B \quad (\text{分配律})$$

$$= 0 + 0 \quad (\text{互补律})$$

$$= 0 \quad (0-1 \text{ 律})$$

这样 \overline{AB} 和 $A + B$ 都能满足互补律，根据互补律的唯一性，可得 $\overline{A + B} = \overline{AB}$ 。

摩根律的推广（n 变量摩根律）：

$$\overline{X_1 + X_2 + \cdots + X_n} = \overline{X_1} \overline{X_2} \cdots \overline{X_n}$$

$$\overline{X_1 X_2 \cdots X_n} = \overline{X_1} + \overline{X_2} + \cdots + \overline{X_n}$$

以上定律的证明还可以通过真值表进行,读者可以自行尝试。

2.2.2 重要规则

逻辑代数有4条重要规则:代入规则、反演规则、对偶规则和展开规则。这些规则在逻辑运算中十分有用,可以将原有的定律和公式加以扩充和扩展。

一、代入规则

代入规则是指在任何含有某变量(如A)的逻辑等式中,如果将等式中所有出现该变量的地方都以同一个逻辑函数(如F=B+C),则等式仍然成立。

【例 2-1】 已知等式 $A(B+C)=AB+AC$, 将 $F=D+E$ 代替等式中的变量B后,试证明新等式仍然成立。

将 $F=D+E$ 代替等式中的变量B后,有:

等式左边 $= A[(D+E)+C] = A(D+E+C) = AD+AE+AC$

等式右边 $= A(D+E)+AC = AD+AE+AC$

所以代入以后得到的新等式仍然成立。

代入规则在推导公式中有重要意义。利用这条规则可以将逻辑代数基本定律中的变量用任意逻辑函数代替,从而推导出更多的公式。

例如,利用代入规则可以推导出n变量的摩根律,即:

$$\overline{X_1 + X_2 + \cdots + X_n} = \overline{X_1} \overline{X_2} \cdots \overline{X_n}$$

$$\overline{X_1 X_2 \cdots X_n} = \overline{X_1} + \overline{X_2} + \cdots + \overline{X_n}$$

证明: 由于 $\overline{X_1 + X_2} = \overline{X_1} \overline{X_2}$, 将 $F = X_2 + X_3$ 代替等式中的变量 X_2 后, 根据代入规则新等式仍然成立, 可得:

$$\overline{X_1 + X_2 + X_3} = \overline{X_1} \overline{X_2} \overline{X_3}$$

在将 $F = X_3 + X_4$ 代替等式中变量 X_3 后, 根据代入规则新等式仍然成立, 可得:

$$\overline{X_1 + X_2 + X_3 + X_4} = \overline{X_1} \overline{X_2} \overline{X_3} \overline{X_4}$$

依次类推, 则可得n变量的摩根律: $\overline{X_1 + X_2 + \cdots + X_n} = \overline{X_1} \overline{X_2} \cdots \overline{X_n}$ 。

二、反演规则

由原函数求反函数的过程称为反演。对于任何一个逻辑函数F, 在保持函数运算顺序

不变的情况下，如果将函数表达式中所有的“ \cdot ”变为“ $+$ ”、“ $+$ ”变为“ \cdot ”、“ 0 ”变为“ 1 ”、“ 1 ”变为“ 0 ”、原变量变为反变量、反变量变为原变量，就得到了逻辑函数 F 的反函数 \overline{F} ，即若逻辑函数 $F = f(X_1, X_2, \dots, X_n, 0, 1, +, \cdot)$ ，则 $\overline{F} = f(\overline{X_1}, \overline{X_2}, \dots, \overline{X_n}, 1, 0, \cdot, +)$ ，这就是反演规则。

使用反演规则时要注意以下三点：

- (1) 在应用反演规则时需保持原函数表达式运算顺序不变；
- (2) 在非运算符下有两个以上的变量时，非符号应保持不变；
- (3) 反演规则实际上是摩根律的推广，用反演规则求得的反函数和用摩根律求得的反函数是一致的。

【例 2-2】 已知逻辑函数 $F = AB + C\overline{D}$ ，求其反函数。

解：根据反演规则可求得其反函数为： $\overline{F} = (\overline{A} + \overline{B})(\overline{C} + D)$ 。

【例 2-3】 已知 $F = AB + \overline{ABC} + \overline{BD}$ ，求其反函数：

解：根据反演规则可求得其反函数为： $\overline{F} = (\overline{A} + \overline{B})(\overline{A + B + C})(B + D)$ 。

三、对偶规则

对于任何一个逻辑函数 F ，在保持函数运算顺序不变的情况下，如果将函数表达式中所有的“ \cdot ”变为“ $+$ ”、“ $+$ ”变为“ \cdot ”、“ 0 ”变为“ 1 ”、“ 1 ”变为“ 0 ”，就得到了逻辑函数 F 的对偶函数 F' ，即若逻辑函数 $F = f(X_1, X_2, \dots, X_n, 0, 1, +, \cdot)$ ，则 $F' = f(X_1, X_2, \dots, X_n, 1, 0, \cdot, +)$ ，这就是对偶规则。

【例 2-4】 已知逻辑函数 $F = AB + C\overline{D}$ ，求其对偶函数。

解：根据对偶规则可求得其对偶函数为： $F' = (A + \overline{B})(C + \overline{D})$ 。

【例 2-5】 已知 $F = AB + \overline{ABC} + \overline{BD}$ ，求其对偶函数：

解：根据对偶规则可求得其对偶函数为： $F' = (A + B)(\overline{A + B + C})(\overline{B} + \overline{D})$ 。

对偶函数和原函数具有如下特点：

- (1) 原函数与其对偶函数互为对偶函数，或者说原函数的对偶函数的对偶函数是原函数本身；
- (2) 若两个逻辑函数相等，则它们的对偶函数也相等，反之亦然。
可以利用对偶规则的特点来证明两个函数相等。

【例 2-6】 利用对偶规则证明等式 $A + BC = (A + B)(A + C)$ 。

证明：令 $F_1 = A + BC$ ， $F_2 = (A + B)(A + C)$ ，则两函数的对偶函数为：

$$F_1' = A(B + C) = AB + AC \qquad F_2' = AB + AC$$

由 $F_1' = F_2'$ ，可得 $F_1 = F_2$ ，因此等式成立。

4、展开规则

对于任何一个逻辑函数 $F = f(X_1, X_2, \dots, X_n)$ 可以将其中任意一个变量（例如 X_1 ）分离出来，并展开成：

$$\begin{aligned} F &= f(X_1, X_2, \dots, X_n) \\ &= \overline{X_1}f(0, X_2, \dots, X_n) + X_1f(1, X_2, \dots, X_n) \\ &= [X_1 + f(0, X_2, \dots, X_n)][\overline{X_1} + f(1, X_2, \dots, X_n)] \end{aligned}$$

这就是展开规则。展开规则的正确性验证可以令 $X_1=0$ 或 $X_1=1$ 分别代入便得证。

【例 2-7】 化简函数 $F = A[AB + \overline{A}C + (A + D)(\overline{A} + E)]$

解：根据展开规则有：

$$\begin{aligned} F &= A[AB + \overline{A}C + (A + D)(\overline{A} + E)] \\ &= \overline{A}\{0[0B + 1C + (0 + D)(1 + E)]\} + A\{1[1B + 0C + (1 + D)(0 + E)]\} \\ &= 0 + A(B + E) \\ &= A(B + E) \end{aligned}$$

2.3 逻辑函数表达式的形式与变换

任意一个逻辑函数对应一个唯一的真值表，但是其逻辑表达式却不是唯一的。本节将从理论分析的角度介绍逻辑函数表达式的基本形式、标准形式及其相互转换，作为后面逻辑函数化简的基础。

2.3.1 逻辑函数表达式的基本形式

对于同一个逻辑函数表达式可以有多种形式，例如与或式、或与式、与非式、或非式、与或非式，其中与或表达式（积之和）和或与表达式（和之积）是逻辑函数表达式的两种基本形式。

1、与或表达式

一个逻辑函数表达式中包含若干个“与项”，每个“与项”中可有一个或多个以原变量或反变量形式出现的字母，所有这些“与项”的“或”就构成了该逻辑函数的与或表达式。而“与”运算对应的是“逻辑乘”运算，“与项”就对应了“乘积项”，“或”运算对应的是“逻辑加”运算，因此与或表达式又被称为“积之和”。

例如，逻辑函数 $F = AB + B\bar{C}$ 由 2 个“与项” AB 和 $B\bar{C}$ 组成，这 2 个“与项”又通过“或”运算形成了该逻辑函数表达式。

2、或与表达式

一个逻辑函数表达式中包含若干个“或项”，每个“或项”中可有一个或多个以原变量或反变量形式出现的字母，所有这些“或项”的“与”就构成了该逻辑函数的或与表达式。而“或”运算对应的是“逻辑加”运算，“或项”就对应了“和项”，“与”运算对应的是“逻辑乘”运算，因此或与表达式又被称为“和之积”。

例如，逻辑函数 $F = (A + B)(B + \bar{C})$ 由 2 个“或项” $A + B$ 和 $B + \bar{C}$ 组成，这 2 个“或项”又通过“与”运算形成了该逻辑函数表达式。

逻辑函数还可以表示成其他形式，例如 $F = (AB + C)(BD + \bar{C})$ 既不是与或表达式，也不是或与表达式，但是所有的逻辑函数都可以转换成与或表达式或者或与表达式。

2.3.2 逻辑函数表达式的标准形式

通过前面的介绍可以看出，一个逻辑函数的真值表是唯一的，但是它的逻辑表达式不唯一，那么逻辑函数是否存在一个唯一的表达形式呢？答案是肯定的，这就是逻辑函数表达式的标准形式。逻辑函数表达式有两种标准形式：标准的与或表达式（最小项之和）和标准的或与表达式（最大项之积）。首先，先介绍最小项和最大项的概念和性质。

1、最小项

对于一个具有 n 个变量的函数的与项，它包含全部 n 个变量，其中每个变量都以原变量或者反变量的形式出现且仅出现一次，这样的与项称为最小项。任何一个函数都可以用最小项之和的形式来表示，这种函数表达式称为标准的与或表达式（最小项之和）。

例如：一个三变量的逻辑函数 $F(A, B, C) = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC$ ，其变量按 A 、 B 、 C 顺序排列，由 3 个最小项组成，这个函数表达式就是标准的与或表达式。

由最小项的定义可知， n 个变量的函数最多可以组成 2^n 个最小项。三个变量最多可

以组成 $2^3=8$ 个最小项： $\overline{A}\overline{B}\overline{C}$ 、 $\overline{A}\overline{B}C$ 、 $\overline{A}B\overline{C}$ 、 $\overline{A}BC$ 、 $A\overline{B}\overline{C}$ 、 $A\overline{B}C$ 、 $AB\overline{C}$ 和 ABC ，

其他不同的变量组合，例如 AB 、 $\overline{A}(B+C)$ 等都不满足最小项的条件，所以均不是最小项。

为了描述和书写方便，通常 m_i 表示最小项。按照最小项中的原变量记为 1，反变量记为 0，且当变量顺序确定后，1 和 0 按顺序排列成一个二进制数，而于这个二进制数相对应的十进制数就是最小项的下标 i ，表 2-21 列出了三变量函数的全部的最小项。

表 2-21 三变量函数中的最小项和最大项

变量的各组取值 A B C	对应的最小项及其编号		对应的最大项及其编号	
	最小项	编号	最大项	编号
0 0 0	$\overline{A}\overline{B}\overline{C}$	m_0	$A+B+C$	M_0
0 0 1	$\overline{A}\overline{B}C$	m_1	$A+B+\overline{C}$	M_1
0 1 0	$\overline{A}B\overline{C}$	m_2	$A+\overline{B}+C$	M_2
0 1 1	$\overline{A}BC$	m_3	$A+\overline{B}+\overline{C}$	M_3
1 0 0	$A\overline{B}\overline{C}$	m_4	$\overline{A}+B+C$	M_4
1 0 1	$A\overline{B}C$	m_5	$\overline{A}+B+\overline{C}$	M_5
1 1 0	$AB\overline{C}$	m_6	$\overline{A}+\overline{B}+C$	M_6
1 1 1	ABC	m_7	$\overline{A}+\overline{B}+\overline{C}$	M_7

因此，逻辑函数 $F(A,B,C) = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + ABC = m_2 + m_6 + m_7$ ，若借用数学中常用的符号“ Σ ”表示累计的逻辑加运算，该函数也可以简写成如下形式：

$F(A,B,C) = \Sigma m(2,6,7)$ ，其中符号“ Σ ”表示各项的或运算，后面括号内的数字表示函数的各最小项。等式左边括号内的字母列出所有的变量和它的排列顺序。变量的顺序是很重要的，一旦确定后，就不能任意改变，否则会造成表达式错误。

由表 2-21 可以看出最小项有以下性质：

- 1) 对于任意一个最小项 m_i ，只有一组变量的取值才能使其值为 1。

例如，最小项 $m_4 = \overline{A}BC$ ，只有当 $ABC=100$ 时， m_4 的值才为 1，而对于 ABC 的其他取值， m_4 均为 0。

2) 任意两个不同的最小项之积恒为 0，即 $m_i \cdot m_j \equiv 0, i \neq j$ 。

例如，对于三变量 A、B、C 的两个最小项 m_0 和 m_4 ，则 $m_0 \cdot m_4 = (\overline{A}\overline{B}\overline{C}) \cdot (\overline{A}BC) = 0$ 。

3) n 个变量的全部最小项之和为 1，即 $\sum_{i=0}^{2^n-1} m_i = 1$ 。

例如，对于三变量 A、B、C，其所有的最小项之和为：

$$\begin{aligned} \sum_{i=0}^{2^3-1} m_i &= m_0 + m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7 \\ &= \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + A\overline{B}C + AB\overline{C} + ABC \\ &= \overline{A}\overline{B} + \overline{A}B + A\overline{B} + AB \\ &= \overline{A} + A = 1 \end{aligned}$$

4) n 个变量的任何一个最小项有 n 个相邻最小项。

所谓相邻最小项是指两个最小项中仅有一个变量不同，且该变量为同一变量的原变量和反变量。因此两个相邻最小项相加以后一定能合并成一项，并消去这一对以原变量和反变量形式出现的因子。例如，三变量 A、B、C 组成的最小项 m_0 和 m_1 为相邻最小项，

$$m_0 + m_1 = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C = \overline{A}\overline{B}(\overline{C} + C) = \overline{A}\overline{B}。$$

2、最大项

对于一个具有 n 个变量的函数的或项，它包含全部 n 个变量，其中每个变量都以原变量或者反变量的形式出现且仅出现一次，这样的或项称为最大项。任何一个函数都可以用最大项之积的形式来表示，这种函数表达式称为标准的或与表达式（最大项之积）。

例如：一个三变量的逻辑函数 $F(A, B, C) = (A + B + C)(A + B + \overline{C})(\overline{A} + B + \overline{C})$ ，其变量按 A、B、C 顺序排列，由 3 个最大项组成，这个函数表达式就是标准的或与表达式。

由最大项的定义可知，n 个变量的函数最多可以组成 2^n 个最大项。三个变量最多可以组成 $2^3=8$ 个最大项： $A + B + C$ 、 $A + B + \overline{C}$ 、 $A + \overline{B} + C$ 、 $A + \overline{B} + \overline{C}$ 、 $\overline{A} + B + C$ 、

$\overline{A} + B + \overline{C}$ 、 $\overline{A} + \overline{B} + C$ 和 $\overline{A} + \overline{B} + \overline{C}$ ，其他不同的变量组合，例如 $A + B$ 、 $\overline{A} + BC$ 等都不满足最大项的条件，所以均不是最大项。

为了描述和书写方便，通常 M_i 表示最大项。按照最大项中的原变量记为 0，反变量记为 1，且当变量顺序确定后，1 和 0 按顺序排列成一个二进制数，而于这个二进制数相对应的十进制数就是最大项的下标 i ，表 2-21 列出了三变量函数的全部的最大项。

因此，逻辑函数 $F(A, B, C) = (A + B + C)(A + B + \overline{C})(\overline{A} + B + \overline{C}) = M_0 M_1 M_5$ ，若借用数学中常用的符号“ \prod ”表示累计的逻辑加运算，该函数也可以简写成如下形式： $F(A, B, C) = \prod M(0, 1, 5)$ ，其中符号“ \prod ”表示各项的与运算，后面括号内的数字表示函数的各最大项。等式左边括号内的字母列出所有的变量和它的排列顺序。变量的顺序是很重要的，一旦确定后，就不能任意改变，否则会造成表达式错误。

由表 2-21 可以看出最大项有以下性质：

1) 对于任意一个最小项 M_i ，只有一组变量的取值才能使其值为 0。

例如，最大项 $M_3 = A + \overline{B} + \overline{C}$ ，只有当 $ABC=011$ 时， M_3 的值才为 0，而对于 ABC 的其他取值， M_3 均为 1。

2) 任意两个不同的最大项之和恒为 1，即 $M_i + M_j \equiv 1, i \neq j$ 。

例如，对于三变量 A、B、C 的两个最大项 M_0 和 M_3 ，则

$$M_0 + M_3 = (A + B + C) + (A + \overline{B} + \overline{C}) = 1。$$

3) n 个变量的全部最大项之积为 0，即 $\prod_{i=0}^{2^n-1} M_i = 0$ 。

例如，对于两变量 A、B，其所有的最大项之和为：

$$\prod_{i=0}^{2^2-1} M_i = M_0 M_1 M_2 M_3 = (A + B)(A + \overline{B})(\overline{A} + B)(\overline{A} + \overline{B}) = A \cdot \overline{A} = 0$$

4) n 个变量的任何一个最大项有 n 个相邻最大项。

3、最大项和最小项之间的关系

在同一逻辑问题中，下标相同的最小项和最大项之间存在互补关系，即有：

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

例如, 对于三变量 A、B、C 的最小项 $m_0 = \overline{ABC}$, 则 $\overline{m_0} = \overline{\overline{ABC}} = A + B + C = M_0$ 。

2.3.3 逻辑函数表达式的转换

虽然逻辑函数表达式的形式是多种多样的, 但是各种表达式形式是可以转换的, 任何一个逻辑函数不管是什么形式, 都可以将其转换成为标准的与或表达式及标准的或与表达式的形式。求一个函数表达式标准形式有两种方法: 代数转换法和真值表转换法。

一、代数转换法

代数转换法是利用逻辑代数的公理、定律和规则对逻辑函数表达式的形式进行转换得到逻辑函数的标准形式。

用代数转换法求一个逻辑函数的标准的与或表达式一般分两步:

第一步, 将逻辑函数表达式转换成一般的与或表达式的形式;

第二步, 反复使用形如 $A = A(B + \overline{B})$, 将表达式中所有的非最小项的与项扩展成最小项。

【例 2-8】 求逻辑函数 $F(A, B, C) = \overline{(\overline{AB} + \overline{BC}) \cdot \overline{AB}}$ 的标准的与或表达式形式。

解: 第一步, 将逻辑函数表达式转换为一般的与或表达式, 即:

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

第二步, 把所有的与项扩展成最小项, 若某与项中缺少函数变量 B, 则用 $(B + \overline{B})$ 与上这一项, 再用分配律将其拆成两项。即:

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

与之类似, 用代数转换法求一个逻辑函数的标准的或与表达式也分两步:

第一步, 将逻辑函数表达式转换成一般的或与表达式的形式;

第二步,反复使用形如 $A = (A+B)(A+\bar{B})$, 将表达式中所有的非最大项的或项扩展成最大项。

【例 2-9】 求逻辑函数 $F(A,B,C) = \overline{AB} + \overline{AC} + \overline{BC}$ 的标准的或与表达式形式。

解: 第一步, 将逻辑函数表达式转换为一般的或与表达式, 即:

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

第二步, 把所有的或项扩展成最大项。即:

$$\begin{aligned}
 F(A,B,C) &= (\bar{A} + \bar{B})(A + \bar{B} + \bar{C})(\bar{A} + \bar{B} + C) \\
 &= (\bar{A} + \bar{B} + C)(\bar{A} + \bar{B} + \bar{C})(A + \bar{B} + \bar{C})(\bar{A} + \bar{B} + C) \\
 &= (\bar{A} + \bar{B} + C)(\bar{A} + \bar{B} + \bar{C})(A + \bar{B} + \bar{C}) \\
 &= M_6 + M_7 + M_3 \\
 &= \prod M(3,6,7)
 \end{aligned}$$

二、真值表转换法

逻辑函数如果用真值表表示, 那么真值表的每一行变量组合就对应了一个最小项。如果对应该行的函数值为 1, 则函数的标准与或表达式中应该包含对应该行的最小项; 如果对应该行的函数值为 0, 则函数的标准与或表达式中不包含对应该行的最小项。因此, 求一个逻辑函数的标准与或表达式时, 可以列出该函数的真值表, 然后根据真值表写出逻辑函数的标准与或表达式。

【例 2-10】 求逻辑函数 $F(A,B,C) = \overline{AB} + \overline{BC}$ 的标准的与或表达式形式。

解: 首先, 列出逻辑函数 F 的真值表如表 2-22 所示。

表 2-22 逻辑函数 $F(A,B,C) = \overline{AB} + \overline{BC}$ 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

由表 2-22 的真值表可知，逻辑函数 F 的函数值为 1 的行有第 3、5、6、7 行，所对应的最小项为 m_2 、 m_4 、 m_5 、 m_6 ，则逻辑函数的标准与或表达式为：

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

类似地，逻辑函数真值表的每一行变量组合也对应了一个最大项。如果对应该行的函数值为 0，则函数的标准或与表达式中应该包含对应该行的最大项；如果对应该行的函数值为 1，则函数的标准或与表达式中不包含对应该行的最大项。因此，求一个逻辑函数的标准或与表达式时，可以列出该函数的真值表，然后根据真值表写出逻辑函数的标准或与表达式。

例 2-11：求逻辑函数 $F(A,B,C) = \overline{AC} + \overline{ABC}$ 的标准的或与表达式形式。

解：首先，列出逻辑函数 F 的真值表如表 2-23 所示。

表 2-23 逻辑函数 $F(A,B,C) = \overline{AC} + \overline{ABC}$ 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

由表 2-23 的真值表可知，逻辑函数 F 的函数值为 0 的行有第 1、3、6、7、8 行，所对应的最小项为 M_0 、 M_2 、 M_5 、 M_6 、 M_7 ，则逻辑函数的标准或与表达式为：

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

2.4 逻辑函数的化简

逻辑函数表达式有各种不同的表示形式，即使同一类型的表达式也可能有繁有简。对于某一个逻辑函数来说，尽管函数表达式的形式不同，但它们所描述的逻辑功能却是相同的。在数字系统中，逻辑函数的表达式和逻辑电路是一一对应的，表达式越简单，用逻辑电路去实现也就越简单。通常，从逻辑问题直接归纳出的逻辑函数表达式不一定是最简单的，因此为了降低系统成本、减小复杂度、提高可靠性，必须对逻辑函数进行化简。

逻辑函数表达式是什么形式才能认为是最简呢？通过衡量逻辑函数最简表达式的标准是表达式中的项数最少，每项中包含的变量最少。这样用逻辑电路去实现是，用的逻辑门的数量最少，每个逻辑门的输入端也最少。

逻辑函数化简的方法有很多种，最常用的方法是代数化简法和卡诺图法。

2.4.1 代数化简法

代数化简法是利用逻辑代数的公理、定律和规则对逻辑函数表达式进行化简。一个逻辑函数可以有多种表达形式，而最基本的就是与或表达式。如果有了最简与或表达式，通过逻辑函数的基本定律和规则进行变换，就可以得到其他形式的最简表达式。

一、与或表达式的化简

最简的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，每个与项中的变量的个数最少。

化简与或表达式的常用方法有：

1、并项法

利用逻辑代数的并项律 $AB + A\bar{B} = A$ ，将两个与项合并成一个与项，合并后消去一个

变量，例如： $ABC + AB\bar{C} = AB$ ， $ABC + \overline{ABC} = A$ 。

2、吸收法

利用逻辑代数的吸收律 $A + AB = A$ ，消去多余的项，例如： $B + ABC = B$ ，

$$\overline{AB} + \overline{AB}CD(E + F) = \overline{AB}。$$

3、消去法

利用逻辑代数的消去律 $A + \overline{A}B = A + B$ ，消去多余的变量，例如：

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

4、配项法

利用公理 0-1 律的 $A \cdot 1 = A$ 和公理互补律的 $A + \bar{A} = 1$ ，先从函数表达式中选择某些与项，并配上所缺少的一个合适的变量，然后再利用并项法、吸收法和消去法等方法进行化简。例如：

$$\begin{aligned}
 \overline{AB} + \overline{BC} + \overline{BC} + \overline{AB} &= \overline{AB} + \overline{BC} + \overline{BC}(A + \bar{A}) + \overline{AB}(C + \bar{C}) \\
 &= \overline{AB} + \overline{BC} + \overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC} \\
 &= (\overline{AB} + \overline{ABC}) + (\overline{BC} + \overline{ABC}) + (\overline{ABC} + \overline{ABC}) \\
 &= \overline{AB} + \overline{BC} + \overline{AC}
 \end{aligned}$$

5、冗余法

利用逻辑代数的冗余律 $AB + \bar{A}C + BC = AB + \bar{A}C$ ，消去多余的变量，例如：

$$\begin{aligned}
 \overline{ABCD} + \overline{AE} + BE + C\overline{DE} &= \overline{ABCD} + E(\bar{A} + B) + C\overline{DE} \\
 &= \overline{ABCD} + E\overline{AB} + C\overline{DE} \\
 &= \overline{ABCD} + E\overline{AB}
 \end{aligned}$$

$$\begin{aligned}
 AD + \bar{A}C + \bar{B}C + BC\bar{D} &= AD + C(\bar{A} + \bar{B}) + BC\bar{D} \\
 &= AD + C\overline{AB} + BC\bar{D} \\
 &= C\overline{AB} + AD + BC\bar{D} + ABC \\
 &= (C\overline{AB} + ABC) + AD + BC\bar{D} \\
 &= C + AD
 \end{aligned}$$

上面介绍的例子比较简单，而实际中遇到的逻辑函数往往比较复杂，化简时应灵活地使用逻辑代数的公理、定律和规则，综合运用各种方法。下面给出几个相对复杂一些的例子。

【例 2-11】 化简逻辑函数 $F = \overline{AC} + ABC + A\overline{CD} + CD$ 。

解： $F = \overline{AC} + ABC + A\overline{CD} + CD = A(\bar{C} + BC) + C(\overline{AD} + D)$

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

$$= A + AB + CD = A + CD$$

【例 2-12】 化简逻辑函数

$$F = AC + \overline{BC} + \overline{BD} + \overline{CD} + A(B + \overline{C}) + \overline{ABCD} + \overline{ABDE}。$$

$$\text{解: } F = AC + \overline{BC} + \overline{BD} + \overline{CD} + A(B + \overline{C}) + \overline{ABCD} + \overline{ABDE}$$

$$= AC + \overline{BC} + \overline{BD} + (\overline{CD} + \overline{ABCD}) + \overline{\overline{ABC}} + \overline{ABDE}$$

$$= AC + (\overline{BC} + \overline{\overline{ABC}}) + \overline{BD} + \overline{CD} + \overline{ABDE}$$

$$= AC + \overline{BC} + A + \overline{BD} + \overline{CD} + \overline{ABDE}$$

$$= A + \overline{BC} + \overline{BD} + \overline{CD}$$

$$= A + \overline{BC} + \overline{BD}$$

【例 2-13】 化简逻辑函数 $F = A(B + \overline{C}) + \overline{A}(\overline{B} + C) + BCDE + \overline{BC}(D + E)F$ 。

$$\text{解: } F = A(B + \overline{C}) + \overline{A}(\overline{B} + C) + BCDE + \overline{BC}(D + E)F$$

$$= AB + \overline{AC} + \overline{AB} + \overline{AC} + BCDE + \overline{BC}(D + E)F$$

$$= (AB + \overline{AC} + BCDE) + [\overline{AC} + \overline{AB} + \overline{BC}(D + E)F]$$

$$= AB + \overline{AC} + \overline{AC} + \overline{AB}$$

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

$$= AB + \overline{AC} + \overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC}$$

$$= (AB + \overline{ABC}) + (\overline{AC} + \overline{ABC}) + (\overline{ABC} + \overline{ABC})$$

$$= AB + \overline{AC} + \overline{BC}$$

二、或与表达式的化简

同与表达式的化简类似，最简的或与表达式也应满足两个条件：

- (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、定律中的或与形式，并综合运用前

面介绍的与或表达式化简是提出的各种方法进行化简。但是如果对于公理、定律中的或与形式不太熟悉，也可以利用对偶规则对逻辑函数两次求对偶的方法来化简，即：首先对逻辑函数的或与表达式求对偶，得到其对偶函数的与或表达式，再按与或表达式化简的方法求出对偶函数的最简与或表达式，最后对最简的对偶函数再求对偶，即可得到逻辑函数的最简的或与表达式。

【例 2-14】 化简逻辑函数 $F = (A + B)(A + \bar{B})(B + C)(A + C + D)$ 。

解：逻辑函数 F 的对偶函数为：

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

再对 F' 求对偶，可得： $F = A(B + C)$ 。

2.4.2 卡诺图化简法

卡诺图是美国工程师 Karnaugh 于 20 世纪 50 年代提出的。卡诺图是逻辑函数真值表的一种图形表示。

一、卡诺图的结构

卡诺图由 2^n 个小方格构成的正方形或长方形的图形，其中 n 表示变量的个数。每个小方格对应一个最小项，并在按照在逻辑上相邻的最小项在几何上也相邻的原则进行排列。两个最小项的逻辑相邻是指这两个最小项只有一个变量互为反变量，其余变量都完全相同，因此要实现逻辑相邻的最小项在几何上也相邻，就需要卡诺图上的变量按照格雷码的顺序排列。

图 2.6 为二变量的卡诺图，它由 $2^2=4$ 个方格组成。每一列和每一行上的 0 和 1 分别代表变量 A 和 B 的值。

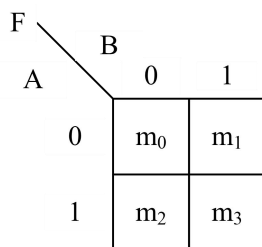


图 2.6 二变量的卡诺图

类似，可以画出三变量、四变量和五变量的卡诺图，如图 2.7、图 2.8、图 2.9 所示。

		BC			
		00	01	11	10
F	A				
		0	1	1	0
	0	m ₀	m ₁	m ₃	m ₂
	1	m ₄	m ₅	m ₇	m ₆

图 2.7 三变量的卡诺图

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

图 2.8 四变量的卡诺图

		DE			
		00	01	11	10
F	BC				
		00	01	11	10
	00	m ₀	m ₁	m ₃	m ₂
	01	m ₄	m ₅	m ₇	m ₆
	11	m ₁₂	m ₁₃	m ₁₅	m ₁₄
	10	m ₈	m ₉	m ₁₁	m ₁₀

A=0

		DE			
		00	01	11	10
F	BC				
		00	01	11	10
	00	m ₁₆	m ₁₇	m ₁₉	m ₁₈
	01	m ₂₀	m ₂₁	m ₂₃	m ₂₂
	11	m ₂₈	m ₂₉	m ₃₁	m ₃₀
	10	m ₂₄	m ₂₅	m ₂₇	m ₂₆

A=1

图 2.9 五变量的卡诺图

二、卡诺图的表示

卡诺图实际上是由真值表变换而来的，真值表有多少行，卡诺图就有多少个小方格，而卡诺图上的每个小方格就代表着真值表上的一行，也代表着一个最小项或最大项。将逻辑函数用卡诺图表示，需分以下四种情况。

1、最小项表达式

因为构成逻辑函数的每一个最小项，其逻辑取值都是使函数值为 1 的最小项，所以

填写卡诺图时,在构成函数的每个最小项相应的小方格中填上 1,而其他方格填上 0 即可。也就是说,任何一个逻辑函数都等于它的卡诺图中填 1 的那些最小项之和。

【例 2-15】 画出逻辑函数 $F(A, B, C, D) = \sum m(1, 3, 6, 7)$ 的卡诺图。

解: 先画一个四变量的卡诺图, 在对应最小项 m_1 、 m_3 、 m_6 和 m_7 的位置填入 1, 其余小方格中填入 0, 得到逻辑函数 $F(A, B, C, D) = \sum m(1, 3, 6, 7)$ 的卡诺图如图 2.10 所示。

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

图 2.10 逻辑函数 $F(A, B, C, D) = \sum m(1, 3, 6, 7)$ 的卡诺图

2、最大项表达式

因为相同编号的最小项和最大项之间存在互补关系, 所以使逻辑函数值为 0 的那些最小项的编号与构成函数的最大项表达式中的那些最大项编号相同, 按这些最大项的编号在卡诺图的相应小方格中填入 0, 其余方格上填入 1 即可。

【例 2-16】 画出逻辑函数 $F(A, B, C, D) = \prod M(3, 4, 8, 9, 11, 15)$ 的卡诺图。

解: 先画一个四变量的卡诺图, 在对应最大项 M_3 、 M_4 、 M_8 、 M_9 、 M_{11} 和 M_{15} 的位置填入 0, 其余小方格中填入 1, 得到逻辑函数 $F(A, B, C, D) = \prod M(3, 4, 8, 9, 11, 15)$ 的卡诺图如图 2.11 所示。

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

图 2.11 逻辑函数 $F(A, B, C, D) = \prod M(3, 4, 8, 9, 11, 15)$ 的卡诺图

3、任意与或表达式

对于任意的与或表达式，可以先将其转换为标准的与或表达式，再按最小项表达式的方法填入卡诺图。另外，也可以不经转换直接填写。任意的与或表达式填入卡诺图的方法是：首先分别将每个与项的原变量用 1 表示，反变量用 0 表示，在卡诺图上找出交叉的小方格并填入 1，没有交叉点的小方格填入 0。

【例 2-17】 画出逻辑函数 $F(A, B, C, D) = AB + BC + CD$ 的卡诺图。

解：先画一个四变量的卡诺图，与项 AB 用 11 表示，对应的最小项为 m_{12} 、 m_{13} 、 m_{14} 和 m_{15} ，在卡诺图对应小方格中填入 1。与项 BC 用 11 表示，对应的最小项为 m_6 、 m_7 、 m_{14} 和 m_{15} ，在卡诺图对应小方格中填入 1。与项 CD 用 11 表示，对应的最小项为 m_3 、 m_7 、 m_{11} 和 m_{15} ，在卡诺图对应小方格中填入 1。

所以，逻辑函数 $F(A, B, C, D) = AB + BC + CD$ 的最小项包含 m_3 、 m_6 、 m_7 、 m_{11} 、 m_{12} 、 m_{13} 、 m_{14} 和 m_{15} ，其卡诺图如图 2.12 所示。

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

图 2.12 逻辑函数 $F(A, B, C, D) = AB + BC + CD$ 的卡诺图

4、任意或与表达式

与任意的与或表达式类似，对于任意的或与表达式只要当任意一项的或项为 0 时，函数的取值就为 0。要使或项为 0，只需将组成该或项的原变量用 0、反变量用 1 代入即可。故任意或与表达式对应的卡诺图的填入方法是：首先将每个或项的原变量用 0、反变量用 1 代入，在卡诺图上找出交叉的小方格并填入 0，然后在其余小方格上填入 1 即可。

【例 2-18】 画出逻辑函数 $F(A, B, C, D) = (A + C)(\bar{B} + \bar{D})(C + D)$ 的卡诺图。

解：先画一个四变量的卡诺图，或项 $A+C$ 对应的最大项为 M_0 、 M_1 、 M_4 和 M_5 ，在卡诺图对应小方格中填入 0。或项 $\bar{B} + \bar{D}$ 对应的最大项为 M_5 、 M_7 、 M_{13} 和 M_{15} ，在卡诺图对应小方格中填入 0。或项 $C+D$ 对应的最大项为 M_0 、 M_4 、 M_8 和 M_{12} ，在卡诺图对应

小方格中填入 0。

所以，逻辑函数 $F(A, B, C, D) = (A + C)(\overline{B} + \overline{D})(C + D)$ 的最大项包含 M_0 、 M_1 、 M_4 、 M_5 、 M_7 、 M_8 、 M_{12} 、 M_{13} 和 M_{15} ，其卡诺图如图 2.13 所示。

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

图 2.13 逻辑函数 $F(A, B, C, D) = (A + C)(\overline{B} + \overline{D})(C + D)$ 的卡诺图

三、卡诺图的性质

卡诺图的特点是任意两个逻辑相邻的最小项（或最大项）在几何上也相邻。在卡诺图中，相邻项有以下三种形式：

1、几何相邻

即几何位置上相邻的最小项，如四变量卡诺图中与 m_0 的相邻最小项 m_1 和 m_4 ，这些最小项对应的小方格与 m_0 对应的小方格分别相连。如图 2.14 所示。

F AB \ CD		CD			
		00	01	11	10
AB	00	m_0	m_1		
	01	m_4			
	11				
	10				

图 2.14 几何相邻

2、相对相邻

同一行的两端以及同一列的两端为相对相邻，如四变量卡诺图中 m_0 相对相邻的最小项 m_2 和 m_8 ， m_0 和 m_2 处于同一行的两端， m_0 和 m_8 处于同一列的两端。如图 2.15 所示。

		CD			
		00	01	11	10
F	AB				
	00	m_0			m_2
	01				
	11				
	10	m_8			

图 2.15 相对相邻

3、重叠相邻

五变量卡诺图中的 m_3 与 m_1 、 m_2 、 m_7 为几何相邻，与 m_{11} 相对相邻，与 m_{19} 则是重叠相邻。对这种情形，可将卡诺图左右两边的矩形重叠，凡上下重叠的最小项即为重叠相邻。只有 5 个及其以上变量的卡诺图中可能存在重叠相邻。如图 2.16 所示。

F		DE			
		00	01	11	10
BC	00			m_3	
	01				
	11				
	10				

A=0

F		DE			
		00	01	11	10
BC	00			m_{19}	
	01				
	11				
	10				

A=1

图 2.16 重叠相邻

四、用卡诺图化简逻辑函数

用卡诺图进行逻辑函数化简的一般步骤：

1、函数化为基本形式之一（与或表达式、或与表达式）；

2、画出函数对应的卡诺图，在相应的格子里填入 0 和 1；

3、找出可以合并的相邻项。每 2^m 个相邻项可以合并为一个卡诺图。先画大的卡诺圈，后画小的卡诺圈，每次画圈时尽量圈未被圈过的格子，以提高画圈的效率，至少要圈一个以前没有圈过的格子，以避免重复画圈。如果是求最简与或表达式，则圈为 1 的格子；求最简或与表达式，则圈为 0 的格子；

4、检查第（3）步画圈的情况，确保每个需要圈的格子至少被圈一次，不要遗漏；

5、根据卡诺图写最简形式。

【 例 2-19 】 用 卡 诺 图 化 简 逻 辑 函 数

$F(A, B, C, D) = ABC + AB\bar{C}D + \bar{A}BCD + \bar{B}CD$ ，求出其最简的与或表达式。

解：逻辑函数 $F(A, B, C, D) = ABC + AB\bar{C}D + \bar{A}BCD + \bar{B}CD$ 对应的卡诺图如图 2.17 (a) 所示。根据卡诺图化简的方法，圈卡诺圈，如图 2.17 (b) 所示。

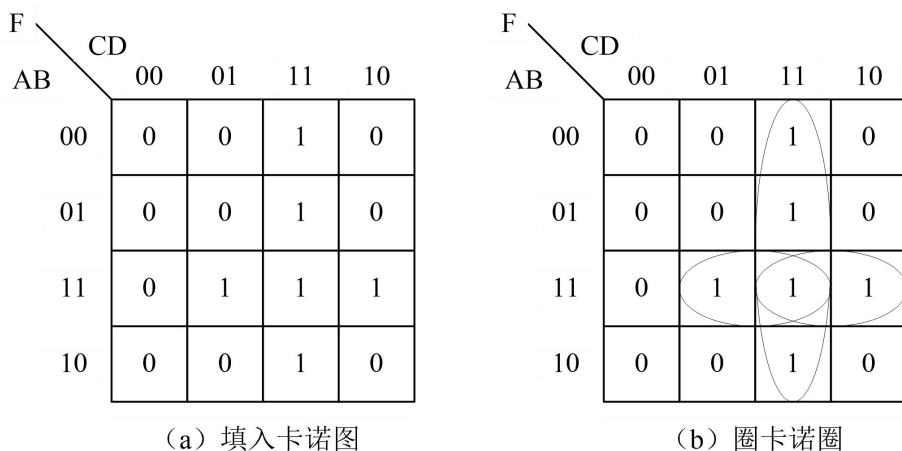


图 2.17 例 2-19 的卡诺图

因此逻辑函数的最简与或表达式为 $F(A, B, C, D) = ABC + ABD + CD$ 。

【例 2-20】 用卡诺图化简逻辑函数 $F(A, B, C, D) = \sum m(3, 4, 5, 6, 7, 9, 11, 13, 15)$ ，求出其最简的与或表达式。

解：逻辑函数 $F(A, B, C, D) = \sum m(3, 4, 5, 6, 7, 9, 11, 13, 15)$ 对应的卡诺图如图 2.18 (a) 所示。根据卡诺图化简的方法，圈卡诺圈，如图 2.18 (b) 所示。

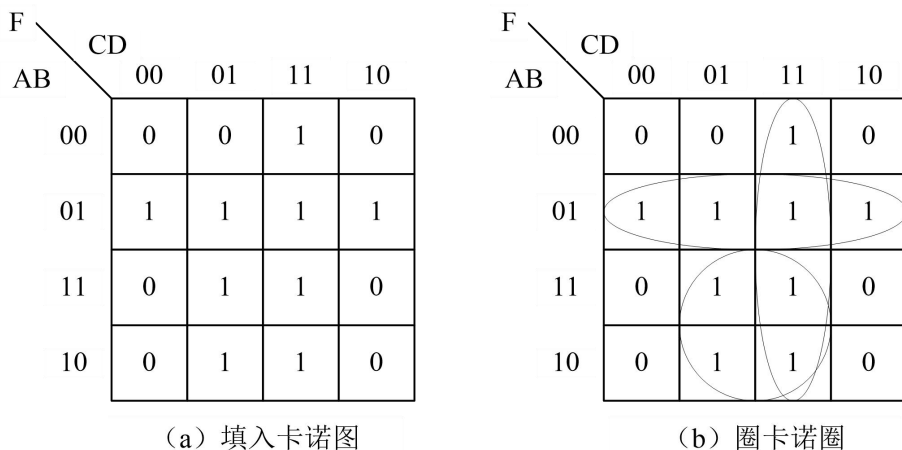


图 2.18 例 2-20 的卡诺图

因此逻辑函数的最简与或表达式为 $F(A, B, C, D) = \overline{A}B + AD + CD$ 。

【例 2-21】 用卡诺图化简逻辑函数 $F(A, B, C, D) = \prod M(3, 4, 6, 7, 11, 12, 13, 14, 15)$ ，求出其最简的与或表达式。

解：逻辑函数 $F(A, B, C, D) = \prod M(3, 4, 6, 7, 11, 12, 13, 14, 15)$ 对应的卡诺图如图 2.19 (a) 所示。根据卡诺图化简的方法，圈卡诺圈，如图 2.19 (b) 所示。

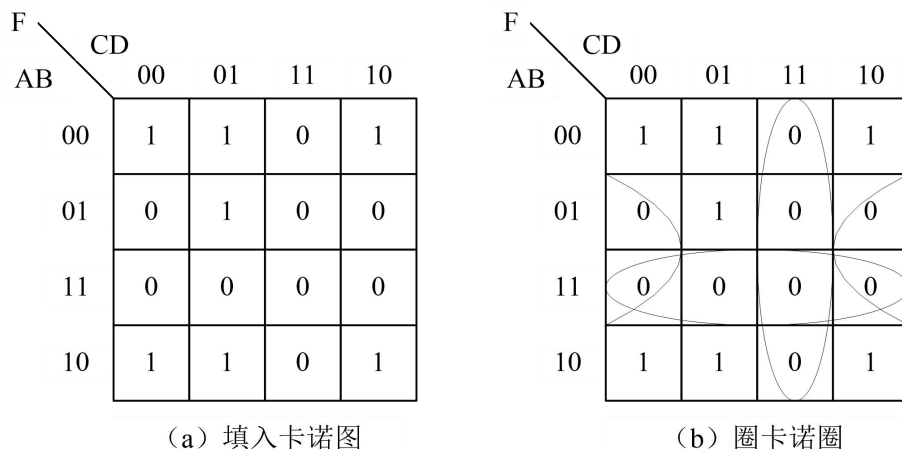


图 2.19 例 2-21 的卡诺图

因此逻辑函数的最简与或表达式为 $F(A, B, C, D) = (\overline{A} + \overline{B})(\overline{B} + D)(\overline{C} + \overline{D})$ 。

2.4.3 包含无关项的逻辑函数的化简

对于一个逻辑函数来说，如果针对逻辑变量的每一组取值，逻辑函数都有一个确定的值相对应，则这类逻辑函数称为完全描述逻辑函数。但是，在某些实际问题中，其输出并不是与 2^n 种输入组合都有关，而是仅与其中的一部分输入组合有关，而与另一部分的输入组合无关。例如，一个电路输入为 8421BCD 码，则其输入变量中的 16 种组合中 1010~1111 不会出现。当函数输出与某些输入组合无关时，这些输入组合就称无关项，又称任意项或约束项。这里的“无关”有两个含义：(1) 这些输入组合在正常操作中不会出现；(2) 即使这些输入组合可能出现，但输出实质上与它们无关。换句话说，当输入出现这些组合时，其所对应的输出值可以为 0，也可以为 1。

与无关项相关的函数就称为包含无关项的逻辑函数，或称为具有约束条件的逻辑函数。若以 d_i 表示无关项，则约束条件（或称约束方程）表示为 $\sum d_i = 0$ 。所以，无关最小项可以随意加到函数表达式中或不加到函数表达式中，而并不影响函数的实际逻辑功能。根据这一特点，化简含有无关项的逻辑函数时，只要使得表达式最简，无关项可以

取 0，也可以取 1。

【例 2-22】 用卡诺图化简逻辑函数 $F(A, B, C, D) = \sum m(0, 1, 5, 7) + \sum d(4, 6)$ ，求出其最简的与或表达式。

解：逻辑函数 $F(A, B, C, D) = \sum m(0, 1, 5, 7) + \sum d(4, 6)$ 对应的卡诺图如图 2.20 (a) 所示。根据卡诺图化简的方法，圈卡诺圈，如图 2.20 (b) 所示。

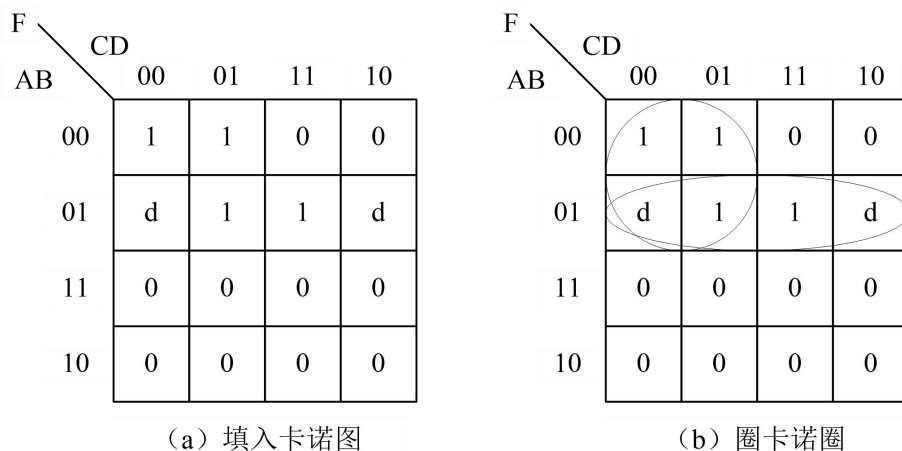


图 2.20 例 2-22 的卡诺图

因此逻辑函数的最简与或表达式为 $F(A, B, C, D) = \overline{A}B + \overline{A}C$ 。

2.4.4 多输出逻辑函数的化简

对于单个逻辑函数的化简问题，已经进行了系统讨论。但在实际问题中，存在着根据一组相同输入变量产生多个输出函数的情况。对于一个具有相同输入变量的多输出逻辑电路，如果只是孤立地将单个输出函数一一化简，然后直接拼接在一起，通常并不能保证整个电路最简，因为各输出函数之间往往存在可共享的部分，这就要求在化简时把多个输出函数当作一个整体考虑，以整体最简为目标。以与或表达式为例来介绍多输出函数的化简。

衡量多输出函数最简的标准是：

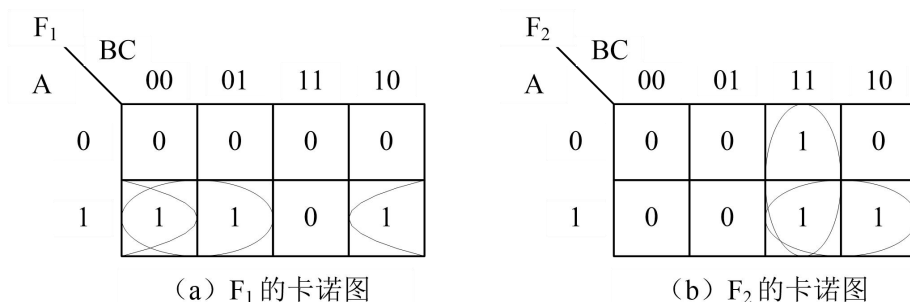
- (1) 所有逻辑表达式中包含的不同与项总数最少；
- (2) 在满足上述条件的前提下，各不同与项中所含的变量总数最少。

多输出函数化简的关键是充分利用各函数之间可供共享的部分（公共项）。例如，某逻辑电路有两个输出函数：

$$F_1(A, B, C) = \overline{A}B + \overline{A}C$$

$$F_2(A, B, C) = AB + BC$$

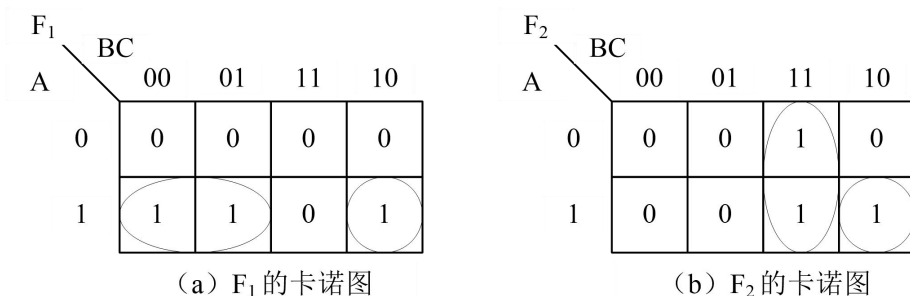
其对应的卡诺图如图 2.21 所示。从卡诺图可以看出，就单个函数而言， F_1 和 F_2 均已达到最简。此时，两个函数表达式共含 4 个不同的与项，4 个不同的与项所包含的变量总数为 8 个。

图 2.21 函数 F_1 和 F_2 的卡诺图

假如按图 2.22 所示的卡诺图化简上述函数，则可得到函数表达式为：

$$F_1(A, B, C) = \overline{A}\overline{B} + ABC$$

$$F_2(A, B, C) = ABC + BC$$

图 2.22 修改后的函数 F_1 和 F_2 卡诺图

这样处理以后，尽管从单个函数来看，上述两个表达式均未达到最简。但从整体来说，由于恰当地利用了两个函数的共享部分，使两个函数表达式中不同与项总数由原来的 4 个减少为 3 个，各不同与项中包含变量总数由 8 个减少为 7 个，从而使整体得到了进一步简化。

用卡诺图化简多输出函数一般分为两步进行。首先按单个函数的化简方法用卡诺图对各函数逐个进行化简。然后，在卡诺图上比较两个以上函数的相同 1 方格部分，看是否能够通过改变卡诺图的画法找出公共项。在进行后一步时要注意：第一，卡诺圈的变动必须在两个或多个卡诺图的相同 1 方格部分进行，只有这样，对应的项才能供两个或多个函数共享；第二，卡诺圈的变动必须以使整体得到进一步简化为原则。

【例 2-23】 用卡诺图化简多输出函数

$$F_1(A, B, C, D) = \sum m(2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$

$$F_2(A, B, C, D) = \sum m(2, 3, 5, 6, 7, 10, 11, 14, 15)$$

$$F_3(A, B, C, D) = \sum m(6, 7, 8, 9, 13, 14, 15)$$

解：画出三个函数的卡诺图如图 2.23 所示。

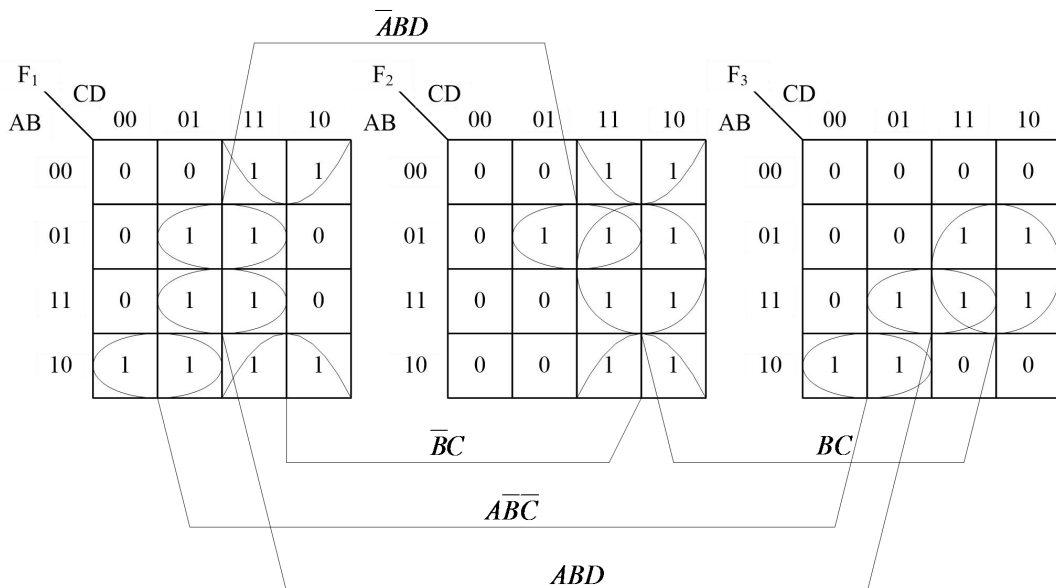


图 2.23 例 2-4-13 的卡诺图

考虑到各函数共享问题，可按图 2.23 所示的卡诺圈的画法，使函数从整体上得到进一步简化，化简结果为：

$$F_1(A, B, C, D) = \overline{A}BD + ABD + \overline{A}\overline{B}\overline{C} + \overline{B}\overline{C}$$

$$F_2(A, B, C, D) = \overline{A}BD + \overline{B}\overline{C} + BC$$

$$F_3(A, B, C, D) = ABD + \overline{A}\overline{B}\overline{C} + BC$$

2.5 本章小结

本章首先学习了逻辑代数的基本概念，认识了逻辑代数的定义、基本运算、复合运算和逻辑函数的表示，其次学习了逻辑代数的基本定律、规则和常用公式，介绍了逻辑函数表达式的基本形式和标准形式，最后学习了逻辑函数常用的化简方法：代数化简法和卡诺图化简法，并且介绍了逻辑函数化简在实际工程中的一些典型问题。

具体关键知识点梳理如下：

- 1、逻辑代数定义了三种基本运算：与、或、非，由与、或、非之间不同组成可构成很多复合运算。

- 2、逻辑代数有很多基本定律、规则和常用公式，利用它们可以证明逻辑等式、变换逻辑函数表达式的形式和化简逻辑函数。
- 3、逻辑函数的反演规则可以用来求解逻辑函数的反函数，逻辑函数的对偶规则可以用来求解逻辑函数的对偶函数。
- 4、逻辑函数表达式有两种标准的表达形式：最小项之和（标准与或表达式）和最大项之积（标准或与表达式），最小项和最大项的性质都是设计和分析数字电路的基础。
- 5、本教材介绍了两种最常用的逻辑函数化简方法：代数化简法和卡诺图化简法，代数化简法是利用逻辑代数的各种定律、规则和公式以数学推导的形式对逻辑函数表达式进行化简；卡诺图化简法是利用画卡诺图、圈卡诺圈和合并相邻项的图形方法对逻辑函数表达式进行化简。逻辑函数表达式在后续的数字电路设计和分析中对应的是实际电路，因此逻辑函数表达式的形式的复杂程度直接关系到电路的复杂程度，因此逻辑函数化简是优化数字电路设计的有效手段之一。

2.6 习题

原书习题