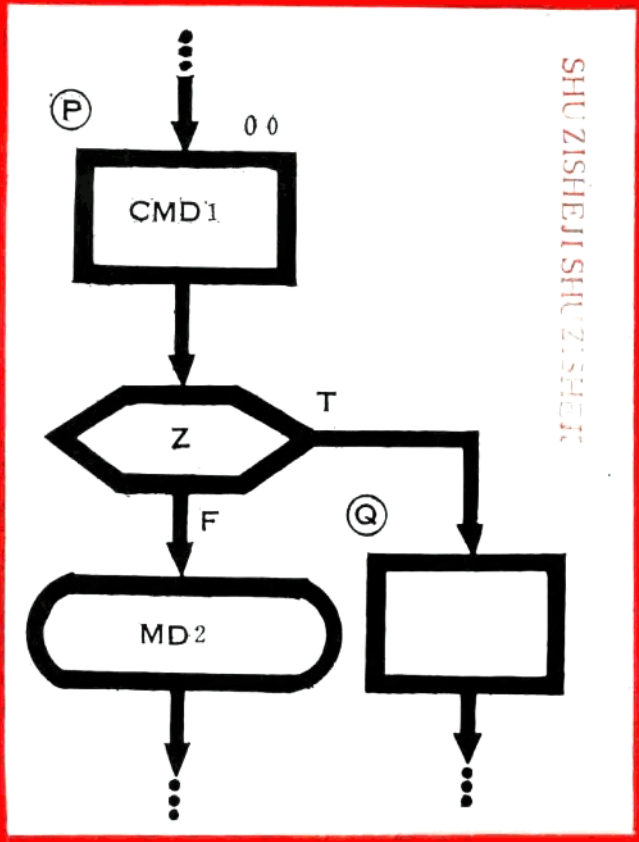
数字设计

数字设计

SHUZISHEJI SHUZISHEJI SHUZISHEJI



SHUZISHEJI SHUZISHEJI

安徽科学技术出版社

责任编辑：张晓红

数 字 设 计

何荣超　张为慧　编著

安徽科学技术出版社出版

（合肥市金寨路283号）

新华书店经销　　芜湖新华印刷厂印刷

开本：787×1092　1/16　印张：12.5　字数：316,000

1988年8月第1版　1988年8月第1次印刷

印数：00,001—1,500

ISBN 7-5337-0111-9/O·3　　定价：4.50元

序

何荣超和张为慧先生合作编著（以何荣超为主）的《数字设计》，是一本介绍数字系统设计技艺的、基础而实用的教学与工程用书。它是作者多年教学实践和理论研究的一个成果。

1979年本人回国讲学，在合肥工业大学与何先生开始合作，直至1982年我离开该校。其后，我们在学术上仍有交往，特别是对编写这本教材上许多问题的商讨没有间断。本人对何先生渴望在教学中反映国内外先进科技成果的精神甚为感动。1981年我们曾合作采用Winkel / Prosser所著的《The Art of Digital Design》（1980年出版）一书，代替传统的《数字逻辑》，为合肥工业大学计算机及其应用专业开课，获得学生肯定的评价。我曾提出合作编写一本更适合国内实际情况的教材，只是由于我离开合肥工大而未果，而何先生一直坚持这项工作。几年来他边教学，边写讲义，边实践，边修改，并先后发表过几篇研究论文，直至《数字设计》最后定稿，终于实现了我们的宿愿。

和传统的有关数字逻辑的许多教材相比，《数字设计》具有完全不同的风格，它是一本能适应现代数字系统工程设计要求的教材和工程技术人员的参考书。

随着微电子学的发展，由器件价格统治数字系统设计的观点和方法已经陈旧，以小规模集成电路为主体的传统数字逻辑中提供的那些设计技术和种种诀窍，已不能满足愈来愈复杂的数字系统设计的需要。和数字逻辑不同，《数字设计》的编写是以设计的系统性、清晰性和可靠性为指导思想，以中、大规模集成电路为基本模块，采用了自上而下的结构式的设计方法。

全书共分八章。前四章阐明设计的基础知识和必备的工具，它包括布尔代数、逻辑函数及其化简、用电路实现逻辑函数以及用来组成数字系统的各种中规模组合和时序集成电路基本模块等。和一般书籍不同，该书是从设计数字系统的角度讨论它们的，篇幅不大，而工程实用性很强。特别是其中介绍的填变量卡诺图及混合逻辑，它们是构成现代数字设计的强有力的工具。

后四章是这本书的主体。其中，第五章详尽地阐明了现代数字设计的基本观点和基本要求，并且提出了一套能适应这些要求的基本原则和设计方法，它是全书的核心。书中采用了七十年代提出并逐步获得实际应用的算法状态机表达控制算法，提高了设计的清晰性。当算法状态机和前四章介绍的设计工具结合后，使整个设计系统性大大加强。这一章着重介绍同步系统设计，因为同步系统是提高运行可靠性的主要手段，它是构成现代数字系统的主流。第六章以大量实例讨论设计过程中可能遇到的许多具体问题及相应的解决方法。这一章对帮助读者进行实际工程的数字系统设计是有意义的。第七章异步数字设计讨论的内容，主要是作者的研究成果。它不仅指出了异步设计的特殊性，而且提供了和同步设计风格一致的、系统的设计方法，从而使各种类型的数字系统有了统一的设计技术。最后一章，作者简要地介绍了微程序设计原理及其应用，它为读者进一步学习现代数字设计技术开拓了视野。

该书作者具有丰富的教学经验，全书内容充实、文字流畅、逻辑性强。《数字设计》一

书可作为高校计算机系和电气系有关专业的基本教材；同时，由于这本书对电工学及电子学等方面的预备知识要求甚低，因而十分适合数字系统爱好者作为自学之用。

赵鉴芳

1986.6.

注：本序撰写者原系美国哥伦比亚特区大学教授。

目　录

第一章　逻辑函数及其化简

数字设计的最终目的是把数字部件组织成能满足某种需要的可靠的数字系统。早期的数字设计方法受到器件价格的限制，从而把注意力集中在如何节省器件上。当前，由于数字集成电路的迅猛发展，价格低、功能强而且灵活性大的数字部件不断涌现，传统设计方法的复杂性成了设计的障碍。设计者希望寻求一种直截了当的、便于理解的正确可靠的设计方法，这种方法称作系统的设计方法，或工程设计方法。本书将主要介绍以中规模集成电路为基础的系统的数字设计方法。在讨论这个方法之前，必须掌握有关的设计工具。为此，我们将以四章篇幅介绍这些工具。

1.1 数字设计中的逻辑概念

数字系统中的数字部件都是由二值性的逻辑器件组成的。任何数字部件，乃至数字系统的任何输入或输出只有两种逻辑取值，即不是真值就是假值。当我们通过语言来表达一个复杂的思想时，总是离不开用“与”、“或”、“否”三个简单的字及其组合来表达这种思想中各种因素之间的关系。在数字设计中，我们也在同样的意义上用“与”、“或”、“非”来描述逻辑事件之间的逻辑关系。下面就来对它们作严格定义。

1.1.1 逻辑常数

逻辑常数指的是二值性逻辑的两个值，也就是真值（*Ture*）和假值（*False*）。因为在设计中将频繁地用到这个概念，所以有必要用缩写字来表示它们。这里，我们以 T 或 1 代表逻辑真值，而以 F 或 0 代表逻辑假值。请注意，这里的 1 和 0 绝不是十进制数的值，仅仅是真与假值的符号而已。1 和 T，0 和 F 在今后将交替使用。但在可能引起混乱的时候，例如在讨论用硬件实施设计时，倾向于使用 T 和 F 表示真值和假值。

1.1.2 逻辑变量

如果用文字符号代表一次逻辑事件，就把这个符号叫做逻辑变量。例如令

A = 按下按钮

则称 A 为逻辑变量，它仅仅是“按下按钮”的代号而已。变量 A 只有两种可能的取值：若已按下按钮，则 $A=T$，否则，$A=F$。

在一项实际的数字设计中，会遇到许多变量。因此给变量取名既要简短，又要能表达出

它所代表的逻辑事件的含义。用这个要求来衡量上述变量 A，它虽简短，但却没有直接表达出“按下按钮”这个事件的含义。为此，最好以事件名称的缩写来定义一个逻辑变量。对于上例，重新定义为：

PB = 按按钮 (*Press the button*)

或采用缩写的汉语拼音：

AAN = *An Anniu*

当在一个系统中出现许多相同性质的变量时，例如有三只操作按钮，则可以给它们编号，如 $PB1$，$PB2$，$PB3$ 等。本书逻辑变量名的定义总是字母符号在前，且所有的字母都用大写。

1.1.3 真值表

在数字设计中，我们感兴趣的是由若干个输入逻辑变量的逻辑组合产生的新的逻辑变量，这个新的逻辑变量叫做这些输入变量的逻辑函数。因为每一个输入变量都只具有 T 或 F 两种取值，它们不同取值的任何一种组合所对应的逻辑函数值，也必然只有真与假两种可能的值。只要能确定输入变量每一种可能组合时的函数值，就能得到一个确定的逻辑函数。若把输入变量与输出函数间的关系列成表格形式，这个表就叫真值表。例如，设 Z 是三个变量 A、B 和 C 的逻辑函数，其真值表如表 1-1 所示，其中表 1-1 (a) 用 0、1 表示，表 1-1 (b)

表1-1(a)

行序号	A C B	Z
0	0 0 0	0
1	0 0 1	1
2	0 1 0	1
3	0 1 1	0
4	1 0 0	1
5	1 0 1	0
6	1 1 0	0
7	1 1 1	0

表1-1(b)

行序号	A C B	Z
0	F F F	F
1	F F T	T
2	F T F	T
3	F T T	F
4	T F F	T
5	T F T	F
6	T T F	F
7	T T T	F

用 F、T 表示，这两种方法是等效的。由表可见，在选定了输入变量的排列顺序以后（例如表中的顺序是A、C、B），各变量值的各种可能的组合也是按一定顺序排列的，这个顺序就是二进制数的顺序(关于二进制数将在第二章讨论)。按这种标准形式作出的真值表称作标准真值表，也叫完全真值表，它包含各输入变量值所有各种可能的组合。表 1-1 是三个变量的标准真值表，共具有$2^3 = 8$ 行，从二进制数000到111。这些二进制数000到111和十进制数(0到7)是一一对应的，所以可以用十进制数编号来称呼标准真值表对应的行。例如表1-1的第 3 行，就指的是 011 这种变量组合。为了用更简洁的形式表示一张标准真值表所代表的函数，也可以把函数写成函数值的向量形式，如表1-1(a)可写成

$Z(A、C、B) = (0、1、1、0、1、0、0、0)$

1.1.4 逻辑运算符号的定义

现在来定义前述的与、或、非三种逻辑含义。

我们用在逻辑变量上加一横杠这种符号表示该变量的逻辑非。设A为逻辑变量，则

A 的逻辑非 $=\overline{A}$

$\overline{A}$ 读作“A 非”，或者“非 A”。其逻辑含义是对所指定逻辑事件的否定，即只有在事件不成立时，其结果才是真的。$\overline{A}$ 和A 之间有严格的逻辑定义，可用真值表描述成表1-2形式。

表1-2

A	$\overline{A}$
0	1
1	0

表1-3

A	B	$A\cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

我们用在两个逻辑变量间加一个“·”这样的符号表示两个变量的逻辑与。例如 A 与 B，写作 $A\cdot B$。其逻辑含义是，只有在 A 和 B 同时为真值时，$A\cdot B$ 才为真值，因为 A 和 B 分别是独立的变量，每一个都可以取值 1 或 0，且 A 和 B 任一组取值都对应一个确定的 $A\cdot B$ 的值，故逻辑与的定义可以用真值表表示成表 1-3 的形式。真值表也能用来定义多个变量的逻辑与，例如三个变量的逻辑与 $A\cdot B\cdot C$，其真值表如表 1-4 所示。总之，所谓几个变量的逻辑与，其含义指的是：只有在所有输入变量为真值时，它们的逻辑与才是真值。

表1-4

A	B	C	$A\cdot B\cdot C$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

表1-5

A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

这里要提醒注意的是，不要把 $A\cdot B$ 中的“·”符号随意略去而写成 AB，这样做会使人把 AB 误解为只是一个变量名。

逻辑或的符号是“+”。例如 A 或 B，写作 $A+B$。逻辑或的定义如表1-5真值表所示。其逻辑函数是，只要 A 或者 B 中的一个为真值，$A+B$ 就是真值。

请注意，这里的符号“+”指的是逻辑或运算，不是两个数的数值相加。在本书中，为了区分“逻辑或”和“数值加”这两种不同性质的运算，将一律采用“(+)”这样的符号代表“数值加”。例如：

$5=2(+)3$

对于多个变量的逻辑或，同样可以用真值表予以定义，如表 1-6 所示。由表可见，参与逻辑或的各输入变量中，即使只有一个为真值，其输出也为真值。

表1-6

AY	$PB1$	X	$AY+PB1+X$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

逻辑与、或、非的符号称作逻辑运算符。其中，非运算符仅仅是对单个变量而言的，而与、或运算符则一定要用在两个参与运算的变量中间，所以与、或运算符被认为是多输入运算符。

真值表在许多设计中是很有用的。但是我们希望找到更简单和有效的工具，以便使我们能象用普通代数运算那样处理更为复杂的逻辑关系。布尔代数就是一种逻辑运算符代数，又叫逻辑代数。下面就介绍一些逻辑代数的基本知识。

1.2 逻辑代数基础

在数字设计中，逻辑代数是很重要的。因为不仅它能对逻辑结构提供简洁的说明和化简的途径，而且它的逻辑运算符的逻辑功能能用实际器件予以实现。由于真值表及其有关的表格在数字设计中起着重要的作用，如何表达真值表和逻辑方程之间的关系，并通过化简方程最后用器件实现它，这些工作都要借助于逻辑代数。我们将围绕这些实际需要来介绍逻辑代数。

1.2.1 基本公式

首先要明确，在逻辑代数中变量只有真值（T 或 1）和假值（F 或 0），运算只能用与、或、非三种运算符。在复杂的逻辑表达式中，运算符的运算顺序是：最先是非运算，其次是与运算，最后是或运算。和普通代数一样，优先处理括弧内的运算。

上一节关于运算符定义的真值表中所体现的逻辑关系，是构成逻辑代数基本公式的依据。现在把三种基本运算关系归结如下：

与：$0\cdot 0=0$　　或：$0+0=0$　　非：$\overline{0}=1$

$0\cdot 1=0$　　$0+1=1$　　$\overline{1}=0$

$1\cdot 0=0$　　$1+0=1$

$1\cdot 1=1$　　$1+1=1$

由此得出下列推论：

$$A\cdot 0=0 \qquad A+1=1 \tag{1-1}$$

$$A\cdot 1=A \qquad A+0=A \tag{1-2}$$

$$A \cdot A = A \qquad A + A = A \tag{1-3}$$

$$A \cdot \overline{A} = 0 \qquad A + \overline{A} = 1 \tag{1-4}$$

$$\overline{\overline{A}} = A \tag{1-5}$$

从这些推论可以看出，与运算和或运算之间有对偶关系，只要把式中与、或符号对换，0和1也互换，就可以从一个等式得到另一个等式，这叫做对偶规则。利用这条规则能使逻辑公式扩展一倍。

下面介绍几条在逻辑设计中常常用到的基本公式（或基本定律）：

交换律 $A + B = B + A \qquad A \cdot B = B \cdot A$ (1-6)

结合律 $A + (B + C) = (A + B) + C$

$A \cdot (B \cdot C) = (A \cdot B) \cdot C$ (1-7)

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

$A \cdot (B + C) = A \cdot B + A \cdot C$ (1-8)

吸收律 $A + A \cdot B = A \qquad A \cdot (A + B) = A$ (1-9)

$A + \overline{A} \cdot B = A + B \qquad A \cdot (\overline{A} + B) = A \cdot B$ (1-10)

反演律 $\overline{A + B} = \overline{A} \cdot \overline{B} \qquad \overline{A \cdot B} = \overline{A} + \overline{B}$ (1-11)

以及附加公式

$$A \cdot B + \overline{A} \cdot B = B \qquad (A + B) \cdot (\overline{A} + B) = B \tag{1-12}$$

这些公式都可以用真值表证明。现以反演律$\overline{A \cdot B} = \overline{A} + \overline{B}$为例说明。分别对该式左、右两边作真值表如表1-7所示，表1-7(*a*)和表1-7(*b*)的输入变量相同，都是A和B，对于A、B每一种可能的组合(即真值表每一行)，两表分别对应的函数$\overline{A \cdot B}$和$\overline{A} + \overline{B}$都有相同的值，所以上式成立。

表1-7(*a*)

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

表1-7(*b*)

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

以上这些公式对于化简逻辑函数是十分有用的；其中反演律尤为重要，因为它能把与、或逻辑运算符互换。反演律可以扩展到多个变量，如

$$\overline{A + B + C} = \overline{A} \cdot \overline{B} \cdot \overline{C} \qquad \overline{A \cdot B \cdot C} = \overline{A} + \overline{B} + \overline{C}$$

关于化简函数的方法将在本章后面部分专门讨论，这里只举一个用逻辑代数公式化简函数的例子：

$$\begin{aligned}
\overline{A \cdot (B + C \cdot \overline{(B + \overline{A})}} &= \overline{A \cdot (B + C \cdot B + C \cdot \overline{A})} && \text{(分配律)} \\
&= \overline{A \cdot (B + C \cdot \overline{A})} && \text{(吸收律)} \\
&= \overline{A \cdot B + A \cdot (C \cdot \overline{A})} && \text{(分配律)} \\
&= \overline{A \cdot B + (A \cdot \overline{A}) \cdot C} && \text{(交换律)} \\
&= \overline{A \cdot B + 0 \cdot C} && \text{[由(1-4)式]} \\
&= \overline{A \cdot B} && \text{[由(1-1)式]} \\
&= \overline{A} + \overline{B} && \text{(反演律)}
\end{aligned}$$

1.2.2 从真值表获得逻辑方程

在数字设计中，真值表和逻辑方程是互相紧密配合的表达逻辑函数的手段。首先让我们来讨论如何从真值表获得一个函数的逻辑方程。从表 1－8 可见，只有在 A 为真值，B 为假值时，W 才为真值。一般表示成

表1-8

行序号	A	B	W
0	F	F	F
1	F	T	F
2	T	F	T
3	T	T	F

表1-9

行序号	A	B	Y
0	F	F	T
1	F	T	F
2	T	F	T
3	T	T	T

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

再看一个例子如表 1－9 所示。由真值表看出，每当 A 是假值和 B 是假值（第 0 行），或者每当 A 是真值 和B 是假值（第 2 行），或者每当 A 是真值和 B 是真值（第 3 行）时，Y 是真值，故而写作

$$Y = \overline{A} \cdot \overline{B} + A \cdot \overline{B} + A \cdot B \tag{1-13}$$

利用前述公式，上式可化简为

$$\begin{aligned} Y &= \overline{A} \cdot \overline{B} + A \cdot \overline{B} + A \cdot B \\ &= \overline{B} + A \cdot B \qquad [由（1-12）式] \\ &= \overline{B} + A \qquad [由（1-10）式] \end{aligned} \tag{1-14}$$

从另一角度看表 1－9，也可以说，只有在 A 是假值和 B 是真值（第 1 行）时，Y 才是假值，因此

$$\overline{Y} = \overline{A} \cdot B \tag{1-15}$$

很明显，从同一真值表能得到 Y 的几种不同的等效表达式，（为什么说它们是等效的?）这里有必要讨论一下函数的标准表达式。

式（1－13）是函数 Y 的与或表达式。式中由“·”运算符连接各变量的项称作与项，由“+”运算符把各与项连接起来的函数式称作与或表达式。一个变量，不管是以原变量的形式（如 A），还是以反变量的形式（如 $\overline{A}$）出现在一个与项中的次数，不得超过一次。例如 $\overline{A}$、$A \cdot \overline{B}$ 和 $\overline{A} \cdot B \cdot \overline{C}$ 都是与项，而 $A \cdot \overline{A}$ 和 $\overline{A} \cdot B \cdot B \cdot C$，虽然是正确的逻辑表达式，但却不能叫做与项。如果一个与项严格地包含真值表中所有的输入变量，这个与项称作最小项，它对应真值表中的一行。把真值表中使函数为真值（1 或 T）的各最小项用逻辑或符号连接起来，这样的与或式称作标准与或表达式，如上例式（1－13）。而式（1－14）是非标准与或表达式，因为它的与项不是最小项。如果从标准的真值表上，把函数值为假值（0 或 F）的各最小项用逻辑或符号连接起来，这样所得到的是函数非形式的标准与或表达式，如式（1-15）所示。

对于 n 个变量的函数，其标准真值表必包含 2^n 行，每一行是一个最小项。有时用 m_i 表示一个最小项，其中 i 是真值表某一行序号，对应于第 i 行，m_i 是真值。例如 $m_4 = A \cdot \overline{B} \cdot \overline{C}$，只有在变量 $A=1$，$B=0$，$C=0$ 时，$m_4=1$。它对应真值表第 4 行，因为二进制数 100

是十进制数的 4 。这里的最小项，指的是真值表上唯一为真值的一行。用最小项符号可以把函数的标准与或表达式写成使函数值为真值的各最小项之和，例如把前式（1-13）写成 $Y_{=}=m_0+m_2+m_3$；或者把函数的非写成使函数值为假值的各最小项之和，如把前式（1-15）写成 $\overline{Y}=m_1$。

标准或与表达式是从真值表获得函数表达式的另一种方法。和与或表达式相反，函数是由一组或项的与形式组成的，如 $(\overline{A}+B)\cdot(A+B)\cdot(\overline{B})$，其中，由“+”运算符把各个变量连接起来的项称作或项。一个变量，不管是以原变量形式，还是以反变量形式最多只允许在一个或项中出现一次。如果一个或项严格地包含真值表中所有的输入变量，即各变量都分别出现一次，这样的或项称作最大项，例如三变量函数，$A+\overline{B}+C$ 是最大项，而 $B+C$ 则不是。由“·”运算符连接一组最大项的函数表达式称作标准或与表达式，如 $W=(A+\overline{B}+\overline{C})\cdot(A+B+C)$，而 $W=(A+\overline{C})\cdot(\overline{A}+\overline{B}+\overline{C})$ 则不是。

怎样从真值表写出最大项呢？前面曾用最小项为真值来表达函数的真值或假值，是否可以设想以最小项为假值来表达函数呢？如前例

$$\overline{Y}=m_1 \quad 则 \quad Y=\overline{m}_1=M_1$$

前式 $\overline{Y}=m_1$ 的含义是，当 m_1 为真值时，Y 为假值，反过来说，只有在不是 m_1 时，Y 是真值。不是 m_1，即 $\overline{m}_1$。令 $M_1=\overline{m}_1$，所以 $Y=M_1$。M_1 指的是：只有在真值表的第一行（m_1），M_1 是假值，而在其余各行，M_1 都为真值。推而广之，M_i 的含义是，除了真值表第 i 行以外的其余各行，M_i 皆为真值，故 M_i 称作最大项。最大项与最小项的关系是

$$M_i=\overline{m}_i$$

设 $m_5=A\cdot B\cdot C$，则

$$M_5=\overline{m}_5=\overline{A\cdot B\cdot C}=\overline{A}+B+\overline{C}$$

为了从真值表得到函数的标准或与方程，必须把使函数为假值的各行表达成最大项，然后用与运算符把这些最大项连接起来。在前例真值表 1-9 中，只有第 1 行 Y 为假值，故

$$Y=A+\overline{B}=M_1$$

写某行最大项的方法是，先将该行每个输入变量值求反，再用或运算符逐个连接之。

要想从真值表得到函数非的标准或与方程，必须把使函数为真值的各行表达成最大项，然后再把这些最大项用与运算符连接起来。对前例则可得

$$\overline{Y}=(A+B)\cdot(\overline{A}+B)\cdot(\overline{A}+\overline{B})=M_0\cdot M_2\cdot M_3$$

很明显，两种用最大项表达的函数方程完全可以用最小项表达的方程经过逻辑代换导出。下面再以一个三变量函数的真值表（表1-10）为例，总结出下列标准逻辑方程，即

表1-10

行序号	J	K	L	W
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

函数 W 的标准与或方程

$$W=\bar{J}\cdot\bar{K}\cdot L+\bar{J}\cdot K\cdot\bar{L}\cdot+\bar{J}\cdot K\cdot L \tag{1-16}$$

$$\overline{W}=\bar{J}\cdot\bar{K}\cdot\bar{L}++J\cdot\bar{K}\cdot\bar{L}+J\cdot\bar{K}\cdot L+J\cdot K\cdot\bar{L}+J\cdot K\cdot L \tag{1-17}$$

函数W的标准或与方程

$$W=(J+K+L)\cdot(\bar{J}+K+L)\cdot(\bar{J}+K+\bar{L})\cdot(\bar{J}+\bar{K}+L)\cdot(\bar{J}+\bar{K}+\bar{L}) \tag{1-18}$$

$$W=(J+K+\bar{L})\cdot(J+\bar{K}+L)\cdot(J+\bar{K}+\bar{L}) \tag{1-19}$$

利用逻辑代数的基本公式，通过变换可以证明这四个式子是等效的。从写出这些式子的过程可以明显感觉到，与或方程比较符合人们的认识过程，所以在实际设计中，最常用的是与或式。

1.3 逻辑函数化简

数字设计的根本任务是用逻辑器件组装成所要求的数字系统。早先，由于器件集成度低，价格又高，所以在设计中强调尽可能减少器件的数量，以便降低总的价格。为此，人们提出了许多简化逻辑表达式的方法，并把大量的精力花在完善这些方法上。现在，数字逻辑器件不仅集成度高，功能性强，而且价格便宜，所以现代设计的重点是考虑集成电路块的合理组合，以及系统的清晰性。集成电路块是构成系统的单元电路，称作模块或结构块(*Building Block*)。这些模块本身的电路是比较复杂的，功能是比较强的。而把这些模块连接在一起的逻辑关系则是比较简单的，表达这种逻辑关系的逻辑方程不会是很复杂的，因此简化逻辑方程的任务也就不象以前那样重要了。当然，简化逻辑方程的工作也还是需要的。如果直接用逻辑代数基本公式变换逻辑方程，需要熟练的技巧，同时难以识别怎样才算最简，所以只有在简单而且直观的情况下，我们才采用这种方法。

与或式是逻辑方程最常见的形式，它不仅可以从真值表得到，而且在我们的设计过程中还可以通过其它步骤得到。简化与或式逻辑方程最常用的公式是

$$A\cdot B+\bar{A}\cdot B=(A+\bar{A})\cdot B=1\cdot B=B \tag{1-20}$$

它表明，对于方程中的两个与项，如果其中只有一个变量是以互非（也叫互补）的形式分别出现在两个与项中，而其余变量的形式完全相同，则可以消去那个互非的变量，把两项合并为一项。这个化简过程启发我们：如果把只有一个变量互非而其余变量完全相同的两个与项

表1-11(*a*)

行序号	J	L	K	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

表1-11(*b*)

J	L	K	Y
0	0	×	0
0	×	0	0
×	1	1	1
1	0	×	1
1	×	0	1
1	×	1	1

表1-11(*c*)

J	*L*	*K*	*Y*
0	0	×	0
0	×	0	0
×	1	1	1
1	0	×	1
1	×	×	1

表1-11(*d*)

J	*L*	*K*	*Y*
0	0	×	0
0	×	0	0
×	1	1	1
1	×	×	1

叫做逻辑相邻项，则函数的化简过程就是合并方程中逻辑相邻项的过程，每合并一次都能消去一个变量。考虑到最小项是一个标准的与项，函数化简可以通过合并包含在方程中的相邻最小项实现。这项工作能直接在真值表上进行，表1-11是一个例子。在表 1-11(*a*) 中，对于 $Y=0$ 而言，第 0 行分别和第 1 行、第 2 行为逻辑相邻项，这三行可以合并成两行，如表 1-11(*b*)中 $Y=0$ 的两行所示；对于 $Y=1$ 的几行来说，第 3 和 7 行，第 4 和 5 行，第 4 和 6 行，第 5 和 7 行可以分别合并成表1-11 (*b*) 中 $Y=1$ 的四行。根据上述方法，该函数可以进一步化简成表1-11(*c*)，最后化简成表1-11(*d*)。在化简过程中，以符号“×”代表某变量既可是 0 也可是 1，它意味着在真值表那一行的函数值和这个变量无关。由表1-11 (*d*) 可以得到函数的简化逻辑方程

$$Y=K\cdot L+J \qquad \overline{Y}=\overline{J}\cdot\overline{K}+\overline{J}\cdot\overline{L}$$

显然，用真值表化简函数也是相当麻烦的。原因在于，逻辑相邻项在真值表中不一定是相邻排列，而且每一次合并只能消去一个变量。下面介绍一种化简逻辑函数的图解方法，它既能避免上述的缺点，又易于掌握，是一个有效的化简工具。

1.3.1 卡诺图的构成

卡诺图是针对真值表法化简函数的缺点而提出来的一种图解方法。卡诺图仅仅是标准真值表的另一种作法而已：它把真值表中每一个最小项用一个小方格表示，并保证使逻辑上相邻的最小项在图中位置上也相邻。这样一来，合并相邻项的工作就极为直观，简单易行了。

图 1-1 是任意两变量函数的真值表和它的卡诺图。真值表中每一行在卡诺图中都有一个对应的小方格，也都是说，每一个输入变量值的组合在图中占具一个小方格。在两变量情况下，一个变量的值标在这些小方格的垂直边上（如图 1-1 中 $B=0$ 和 $B=1$），另一个变量的值写在水平边上（$A=0$ 和 $A=1$）。每一个小方格内填入的是函数值，某个格中的函数值 Y_i 是和真值表相应行(i)的函数值相对应的。例如图1-1卡诺图中左下方的小方格 1 是 $A=0$，$B=1$，故其函数值为 Y_1。

行序号	*A*	*B*	*Y*
0	0	0	Y_0
1	0	1	Y_1
2	1	0	Y_2
3	1	1	Y_3

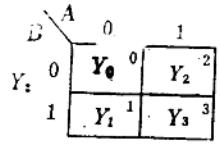


图1-1　两变量真值表和它的卡诺图

三变量和四变量函数也很容易用卡诺图表示。三变量图有 8 个小方格，分别对应于标准真值表的 8 行。图 1-2 是三变量卡诺图的两种标志方法，它们是等效的，而且都是常用的。图1-2(*a*)在垂直与水平边上清楚地表明了输入变量的值，而图1-2(*b*)只是标出了每个变量真值的位置。有人喜欢用图1-2(*a*)形式，也有人习惯图1-2(*b*)的形式，这取决于各人的选择，但应该对两种形式都熟悉。习惯上，把第三个变量（即真值表上最右边一个输入变量）放在垂直边，而以水平边表示第一和第二个变量。

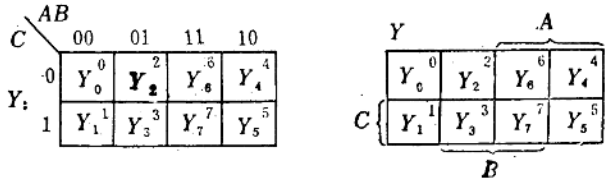


图1-2　三变量卡诺图的两种标志方式

仔细看一下图1-2(*a*)顶部标志的顺序(00，01，11，10)，可以发现，同一行两个相邻的小方格之间只有一个变量值不同，该行最左边和最右边的两个小方格（如第 0 和第 4 格，第 1 和第 5 格）也是这样。同一列的两个小方格也有相应的特性。这种特性叫做单位距离特性，它完全满足相邻项相邻排列的要求，也正是卡诺图能方便地简化逻辑函数的原因所在。

在三变量卡诺图的垂直边上加一个变量就能构成四变量卡诺图，它有 16 个小方格。图1-3是四变量卡诺图常见的两种标志方法。这个卡诺图中任何相邻的两个小方格之间，包括同一行最左边和最右边两格，同一列最上面和最下面两格，也都具有单位距离特性。

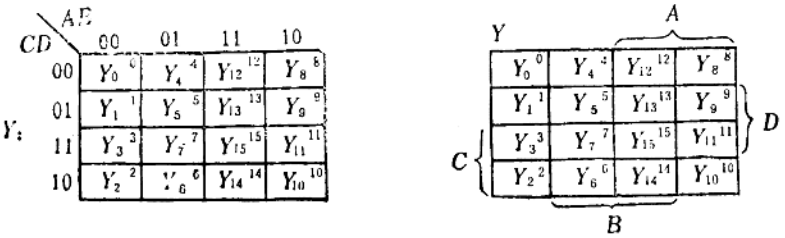


图1-3　四变量卡诺图的两种标志方法

多于四变量的卡诺图很少采用，因为它们的逻辑相邻项在图上的单位距离特性表现得不直观。

1.3.2 用卡诺图化简逻辑函数

用卡诺图化简逻辑函数大体分为三个步骤，即作图、化简和读图。

1. 作函数卡诺图

要用卡诺图化简函数，首先必须作出这个函数的卡诺图。如果这个待化简的函数是用标准真值表表示的，就先作出该函数的卡诺图；如果函数是以标准与或方程形式给出的，这也好办，因为式中一个与项和卡诺图一个小方格相对应。图1-4就是一个二变量函数用方程

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

和真值表以及卡诺图表示的例子。因为 Z 在 $\overline{A}\cdot\overline{B}$ 或是 $A\cdot\overline{B}$ 时为真值，故在卡诺图对应的小方格（0和2格）里填“1”，图中另外两个小方格（1和3格）应填“0”

$$Z=\overline{A}\cdot\overline{B}+A\cdot\overline{B}$$

A	B	Z
0	0	1
0	1	0
1	0	1
1	1	0

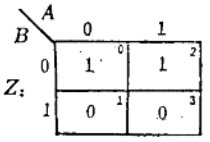


图1-4　逻辑函数的三种表示方法

现在着重讨论一般与或函数如何作卡诺图。设有三变量方程

$$V=A\cdot\overline{B}+B+A\cdot B\cdot C$$

式中第1项表示，当 $A=1$，$B=0$，不管 C 是1还是0，总是 $V=1$，所以应在卡诺图 A，B，$C=100$ 和 A，B，$C=101$ 两个格子（第4和第5格）里都填“1”；式中第2项 B 表明，只要 $B=1$，不管 A，C 是何值，总是 $V=1$，故应在卡诺图中凡是 $B=1$ 的格子里都填“1”，这样的格子有四格，它们是 A，B，$C=010$，011，110，111，即第2，3，6，7四格；式中第3项 $A\cdot B\cdot C$ 则只是在 A，B，$C=101$（第5格）时 $V=1$。由于第5格已在填 $A\cdot\overline{B}$ 项时填过“1”，故不必重复了（因为 $1+1=1$）。最后作出的函数 V 的卡诺图如图1-5所示。

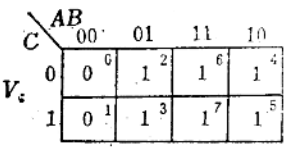


图1-5　$V=A\cdot\overline{B}+B+A\cdot\overline{B}\cdot C$ 的卡诺图

2．在卡诺图上化简函数

用卡诺图化简函数的原则和真值表法是一样的，即函数表达式中出现的两个逻辑相邻与项可以合并成一项，这就消去一个变量。在卡诺图中，逻辑相邻项就是位置相邻的两个格子，当它们所填的函数值相同时，就能够化简。以图1-4的 Z 函数卡诺图为例：图中，$B=0$ 的两个相邻格子里的函数值都是1。这意味着，只要 $B=0$，不管 A 是何值，总有 $Z=1$，所以 $Z=\overline{B}$。为了清楚地表明由哪些方格合并成一项，我们把要合并的那些格子画一个圈圈起来，如图1-6所示。在相邻为1的格子间画圈用的是卡诺图化简逻辑函数的基本方法。两变量卡诺图五种可能画圈的方式如图1-7所示，它们都遵守逻辑代数的基本公式。由这些图可以看出，两个格子组成的圈将消去一个变量，4个格子组成的圈会消去两个变量。以此推论，在多变量卡诺图中，8个格子组成的圈一定消除三个变量。

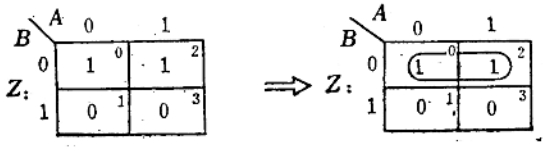


图1-6　圈相邻为1的格子