# PLD/GAL
# 可编程逻辑器件原理和应用

应　钢
关　海　编著
伍能鹏

中国科学院希望高级电脑技术公司

PLD／GAL

# 可编程逻辑器件原理和应用

应 钢　　关 海　　伍能鹏　　编著

# 目　录

# 前　言

可编程逻辑器件早在七十年代就已面世，从那时起直到今天，随着技术的进步和需求的发展，已经历了数代的演变。这期间，技术越来越成熟，功能越来越强大，应用也越来越广泛。目前，在绝大多数稍复杂点的电子产品上都能见到可编程逻辑器件的身影。比较典型的有如：计算机主板及其各种功能卡，无线电话及其它各种先进的通迅设备，等等。笔者面前的计算机就有许多可编程逻辑器件。此外，联想汉卡、单片机开发装置（仿真器）上都有它的存在。

在国内，可编程逻辑器件的应用起步较晚。真正得到广泛应用还是近两年的事。尽管起步较晚，但来势很猛，越来越多的人意识到可编程逻辑器件的无可比拟的优点，许多单位都跃跃欲试，希望能把这一新兴器件用到自己的产品中去，以获得更大的效益。然而就笔者了解，目前，国内虽有广泛应用之势，却很少这方面的资料。

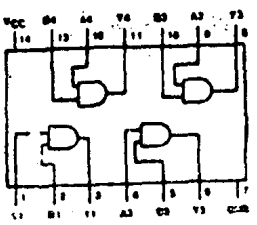
本书即是在这种形式下，结合现有的一些中外资料及应用中的体会，以尽可能通俗的笔法向大家介绍什么是可编程逻辑器件，可编程逻辑器件有多少种，它们都有什么特点，其工作原理是什么，以及如何为可编程逻辑器件编程。此外还要介绍可编程逻辑器件的一些典型应用实例。

本书第一、三、四、八章由关海编写，第二章由伍能鹏编写，其余各章均由应钢编写。

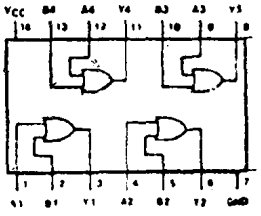
# 第一章　可编程逻辑器件介绍

## 1.1　引言

所谓逻辑器件，即可用来实现特定逻辑功能的电子器件。最基本的逻辑关系有“与”、“或”、“非”等。从这个意义上来说，已广为应用的门电路等都是逻辑器件。如74LS08（2输入四与门）实现的逻辑是“与”，74LS32（2输入四或门）实现的逻辑是“或”。如图1.1所示。当然，这两个器件实现的都是最简单的逻辑功能。还有很多实现很复杂逻辑功能的，典型的如各种高性能的微处理器等。



(a)74 LS 08



(b)74 LS 32

图1.1

这一类逻辑器件有一个共同点，就是其逻辑功能从生产厂家出来以后都是固定不变的。当然，你也可以提出要求，让生产厂家按你的逻辑功能要求特制，这就是所谓的订制器件。由于它是针对一些特殊应用专门制作的集成片，故硅片内的每一面积都会得到充分利用，因此在同一功能要求下，这种结构的芯片面积最小。但是除非你需要的器件数量很大，否则成本是很高的，而且设计生产的周期至少在国内是相当长的。

与定制（包括半定制）器件相对照，前面提到的74LS08、74LS23等在市场上到处都可买到的定型器件就称为标准单元器件。

凡是搞过硬件电路的技术人员都知道，用通用的标准逻辑器件组合出一块特定逻辑功能的板子是很麻烦的。先要进行逻辑电路设计，然后再进行印刷电路板设计，最后经焊接成为成品。且不说设计周期长，可靠性、可维修性差，只所占的空间就往往使某些方面的应用难以接受。用一块块简单逻辑器件搭成的逻辑功能板的集成度显然是比较低的。

当然，用分立元件亦能构成逻辑电路，这种电路灵活性大些，但不论从价格（相同功能情况下比较）、工时来看，还是从可靠性、可维修性来看都很差。本书中一般不涉及分立元件的问题。

为了摆脱前述几种逻辑器件在许多数字逻辑系统研制过程中，低水平费时费力的工作状况，为了获得最大的灵活性和最短的研制周期，就出现了本书要重点介绍的可编辑逻辑器件。

可编程逻辑器件简称为 PLD ( Programmable Logical Device )，是由编程来确定逻辑功能的器件的统称。属于一种门阵列结构，这类结构的主要吸引力在于它的柔性。应用时，由用户完成器件的逻辑功能构造。在大多数情况下硅片能够充分利用。使用这类器件能获得极大的设计灵活性，大大缩短研制周期，还能大大提高产品的集成度，使一些复杂的设计轻而易举的实现。无怪乎许多权威人士评价道：如今不懂 PLD 器件的工程师，就如同当年搞电子技术的不懂晶体管。

当然，若经常使用可编程逻辑器件，最好购买一套编程开发工具。这就必须要支出购买编程开发工具的费用，这个费用比购买器件本身贵得多。但要注意到，开发工具仅是一次性投资。这个费用折合到每个器件上就非常便宜了。况且以后许多器件编程设计时均可使用。

若不是经常性的使用可编程逻辑器件，也可以不购买开发工具，而委托有关部门代为编程。但至少应了解 PLD 器件的一般原理。

## 1.2 PLD 器件的符号约定

大多数典型的 PLD 器件都是由二级组合网络构成的。通常第一级是“与”阵列，第二级是“或”阵列。输入连接“与”阵列，在其中进行“与”逻辑组合，形成乘积项，然后乘积项转入“或”阵列，在“或”阵列中由不同的乘积项构成要求的逻辑函数输出。

逻辑编程的物理实现一般都是通过熔丝的熔断和连接完成的。

图1.2所示为一个 2 输入，1 输出的可编程逻辑器件的基本结构。

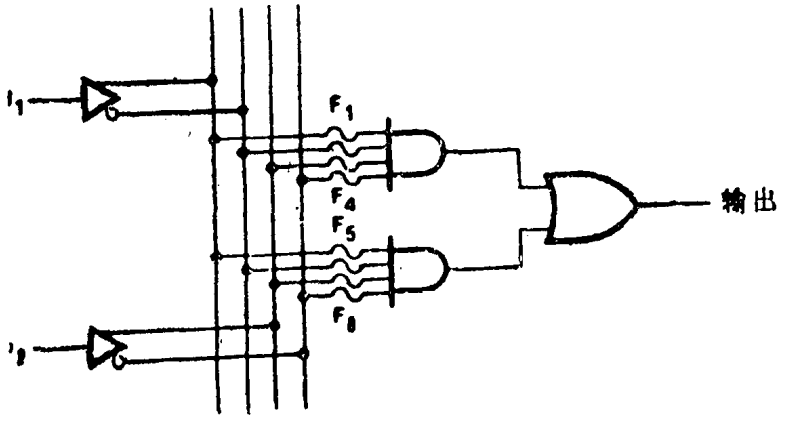



图1.2

图中有两个“与”项构成“与”阵列，一个“或”项构成“或”阵列，$F_1$— $F_8$为八个熔丝。对刚出厂尚未编程过的 PLD 器件，全部熔丝都是相连的。

熔丝熔断时　　F＝0

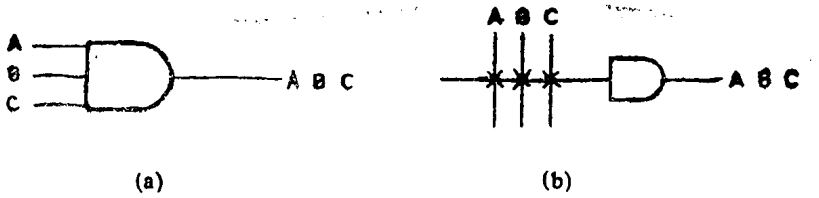
熔丝连接时　　F＝1

于是图 1.2 所实现的逻辑方程可表达为：

$$\text{输出}=(I_1+\bar{F}_1)(\bar{I}_1+\bar{F}_2)(I_2+\bar{F}_3)(\bar{I}_2+\bar{F}_4)$$
$$+(I_1+\bar{F}_5)(\bar{I}_1+\bar{F}_6)(I_2+\bar{F}_7)(\bar{I}_2+\bar{F}_8)$$

显然，对于稍大点的系统，用图 1.2 所用的表示方法就很不方便了。我们知道十数

个输入、十来个输出的 PLD 器件是很常用到的。为了使 PLD 的逻辑表示便于使用又易于理解，约定如下。即图 1.3 a 可用图 1.3 b 的方法表示。




(a) (b)

图 1.3

其中“×”表示用来完成逻辑“与”功能的一个未熔熔丝，即可编程连接；“⊹”表示固定连接；“+”表示擦除单元，不连接。

由于图 1.2 所示的约定能在芯片设计和逻辑图之间清楚的建立起一一对应的关系，也允许逻辑图和真值表结合成为紧密和易读的形式，因此已广泛用作 PLD 器件的简化表达形式。

图 1.2 的逻辑结构若用上述约定就可简化为图 1.4 的形式：

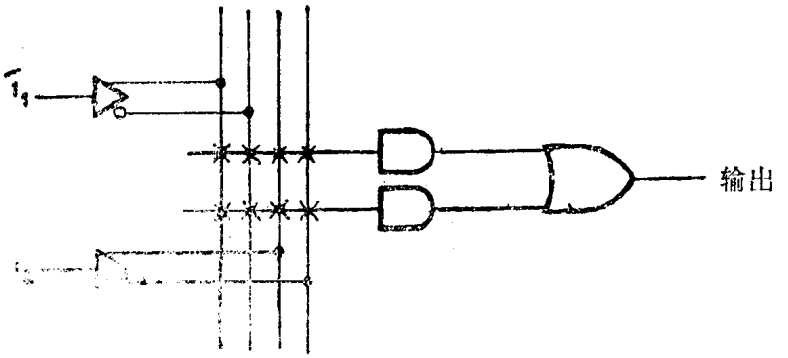



图 1.4

做为一个简单例子，考虑逻辑关系：

$$\text{输出} = I_1\bar{I}_2 + \bar{I}_1 I_2$$

的实现。

该逻辑关系的普通组合逻辑图见图 1.5，相应的 PLD 简化等效逻辑图见图 1.6。

## 1.3 PLD 器件的发展

有了 §1.2 节的约定，我们可以将 PLD 的发展用简化表达形式清楚地描述出来。

在 PLD 家族中，最初的成员可以说是 PROM 器件了。

PROM，在有些书中也写为 PROMS，即可编程只读存储器。这个名字很容易让人联想到存储器元件 RAM、ROM 等。PROM 的基本结构包括一个固定的“与”阵列，其输出加到一个可编程的“或”阵列上。它们普遍用来存储计算机程序和数据。在

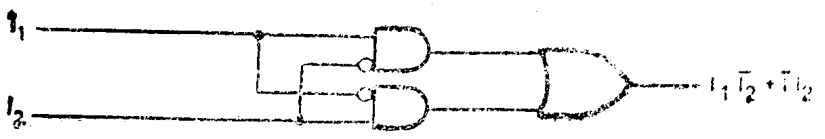


图 1.5

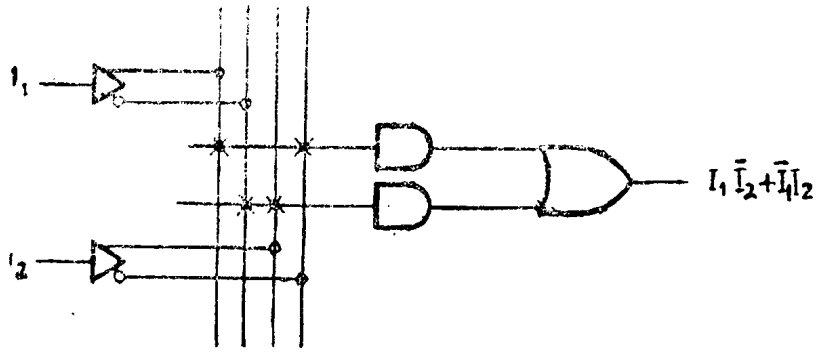


图 1.6

这类应用中，固定的输入是计算机存储器地址，输出是存储器单元的内容。PROM 价格低，易于编程，并具有各种容量和结构。图 1.7 是一个 16 字×4 位的 PROM 逻辑图。

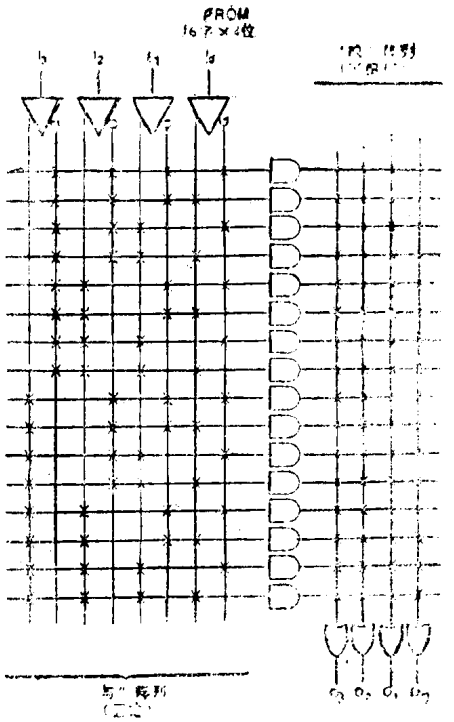


图 1.7 PROM结构示意图

此后出现了 PLA 器件，即可编程逻辑阵列。也称 FPLA（熔丝可编程逻辑阵列或现场可编程逻辑阵列）。这种器件是由可编程的“与”阵列和可编程的“或”阵列构成的。见图 1.8。由于设计者完全控制全部的输入和输出，使其在实现逻辑函数时有极大的灵活性，广泛应用于各种场合。然而这种结构对于实现简单逻辑函数显得尺寸过大，不太经济，且价规昂贵，不易掌握，其专用编程器的开支也较大。

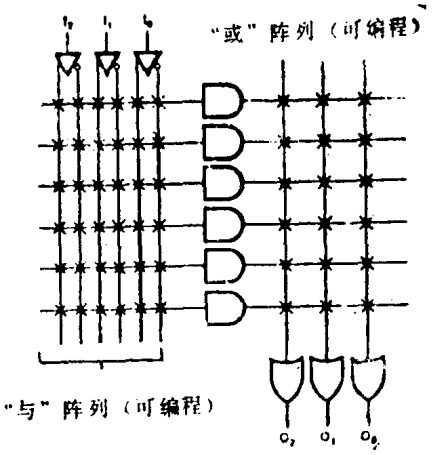



图 1.8　基本 FPLA 结构

PLS 器件，即可编程逻辑定序器件，也称 FPLS（熔丝可编程逻辑定序阵列），是一种与 PLA 器件在结构上类似的可编程基本逻辑时序器件。不同的是其输出一般都有寄存器反馈。

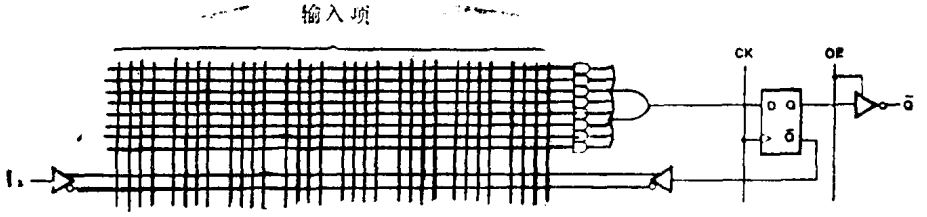



图 1.9　PLS 输出结构示意图

PAL 器件，即可编程阵列逻辑器件，则结合了 PLA 的灵活性和 PROM 的廉价和易于编程的特点。其基本逻辑结构包括有一个可编程的“与”阵列和一个固定的“或”阵列，见图1.10。“与”阵列的编程特性使输入项可以增多，而“或”阵列固定使器件简化。目前，PAL 器件是 PLD 中应用最广的。

尽管随着 IC 技术的发展，生产更大集成规模的 PLA，PAL 等传统可编程逻辑器件已成为可能，但有几个方面是不尽人意的。其一是传统的可编程逻辑器件的利用率随其规模的扩大而逐渐下降，统计表明在大多数设计中，PAL 器件的最小项只用三个或更少的“与”项，其它的“与”项都浪费掉了。其二是其寄存器数目和输入／输出引脚

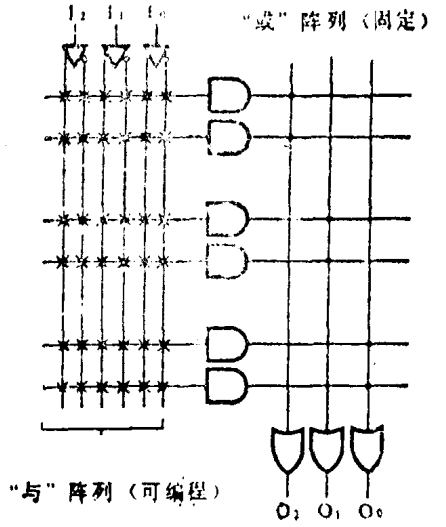



图 1.10　基本 PAL 器件的结构

有限，互连固定和设计灵活性仍不够好。

PGA 器件在这方面则有独到之处。PGA 的全称是可编程门阵列（Programable Gate Array）。也称 FPGA。细心的读者可能会发现这个名字与前面频频提到的"可编程逻辑"的说法不同，这里用的是"可编程门"。这正反映了 PGA 器件的特点。

通常 PGA 不划归 PLD 器件之列。

PGA 的芯片中有被互连网络包围的简单逻辑单元，芯片四周为输入／输出单元。其互连模式是可编程确定的。用户可以通过编程决定每个单元的功能及它们之间的互连关系。由于 PGA 器件集成度高、使用灵活、引脚数多（可多达 100 多条），因此，可望填补高性能微处理器和标准器件之间的空白。

有别于前述各类器件，美国晶格半导体公司（Lattice Semiconductor）在1983 年最新推出的可电擦写、可重复编程、可设置加密的一种新型器件成了 PLD 家族的后起之秀。这种新型器件取名为 GAL（Generic Array Logic），一般认为这是第二代的 PAL。目前，这也是最为理想的可多次编程的逻辑器件。

GAL 器件采用电擦除技术，无需紫外线照射就可随时进行修改。由于其内部有一特殊的结构控制字，因而使它芯片类型少，但功能齐全。目前普遍采用的芯片只有两种：GAL 16V 8（20 引脚）和 GAL 20V 8（24 引脚）。仅这两种 GAL 就能仿真所有的 PAL。在研制和开发新的电路系统时极为方便。

GAL 器件与PAL器件在结构上的区别是：PAL 器件把一个可编程的与阵连接到一个固定的或阵上输出。而 GAL 器件则是把一个可编程的与阵连接到输出逻辑宏单元（OLMC）上输出，通过对 OLMC 的编程，就可在符合各种逻辑电路要求方面，给设计者提供最大的灵活性，从而比 PAL 具有更多的功能。如图1.11。

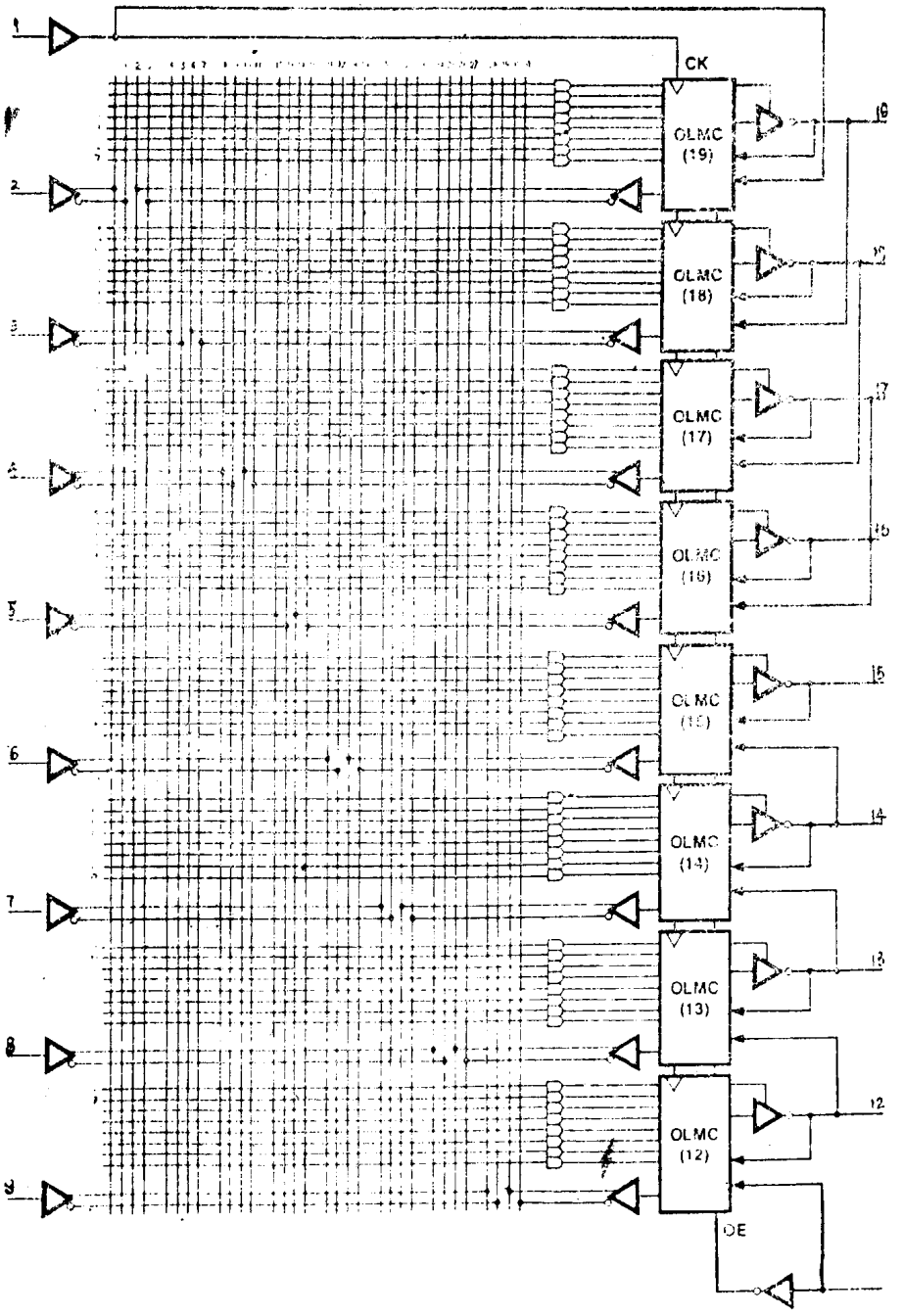



图 1.11　GAL 16V 8方框图

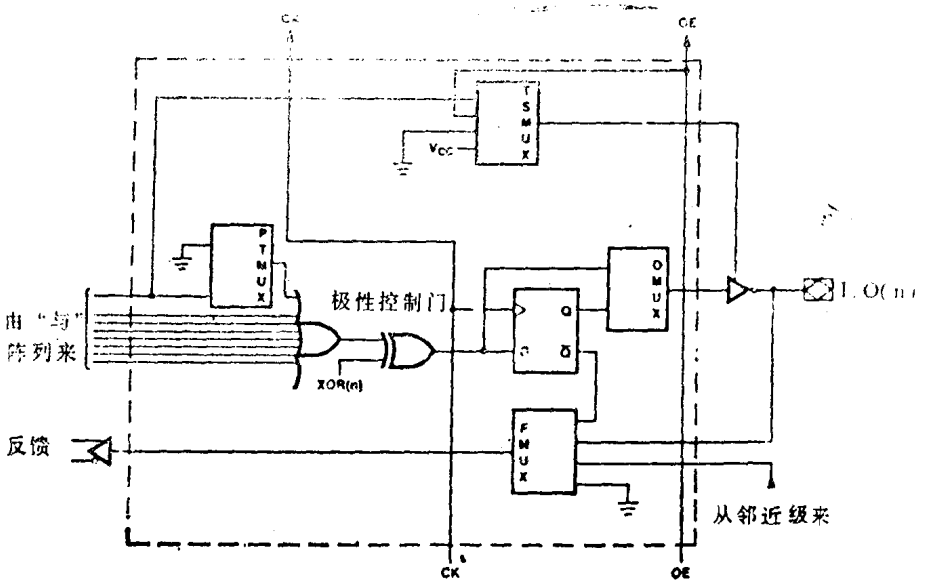



图 1.12　GAL 器件输出逻辑宏单元

# 第二章　数字逻辑系统设计基础

凡是可编程逻辑器件，无论是PROM、PLA还是PAL、GAL，在其设计和使用时，都离不开数字逻辑的基本原理和基本知识。而数字逻辑的设计过程，事实上是对逻辑代数的理解、应用过程。本章简要介绍有关数字逻辑系统的基础知识。

## 2.1　布尔代数

### 2.1.1　引论

"逻辑代数"更多的时候被称为"布尔代数"。这是因为逻辑代数理论首先来自于乔治·布尔（Geoge Boole）于1854年出版的《思维规律研究》（An Investigation of the laws of Thought）一书。

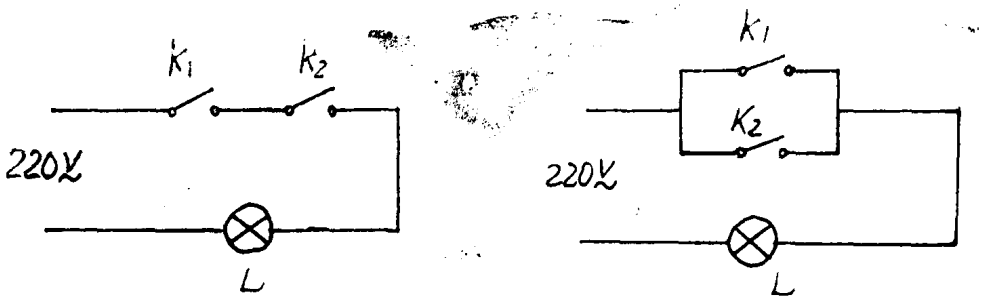
在这本书里，布尔主要研究了"真"和"假"，"是"和"非"等问题，并提出了用数学表示推理方法的符号逻辑。即用一位二进制变量所具有的两个可能的值："1"和"0"来表示逻辑意义上的"真"和"假"。

在我们日常生活中经常会碰到"真"和"假"，"是"和"非"等逻辑问题。例如，我们用两个开关控制一个灯泡，可以用两种方法。第一种，线路依次经过开关$K_1$、$K_2$以后再经过灯泡后形成回路，此时$K_1$与$K_2$是串联关系（见图2.1的（a））。如果改用第二种接法，线路分为两支，同时经过开关$K_1$和$K_2$后合并，再经过灯泡形成回路，此时$K_1$和$K_2$是并联关系（见图2.1的（b））。

在第一种情形中，当且仅当$K_1$与$K_2$两个开关都接通时电路才通畅（灯泡亮），只要$K_1$或$K_2$之中任意一个断开，灯泡就熄灭。

在第二种情形中，只要$K_1$和$K_2$之中的任意一个接通，灯泡就亮。仅当$K_1$和$K_2$都断开时灯泡才熄灭。

假如我们用"1"表示开关接通，用"0"表示开关断开，并且用"1"表示灯泡




(a)串联关系　　(b)并联关系

图2.1　一个例子

亮，用“0”表示灯泡不亮，就归纳出真值表2.1。

表2.1(a)$K_1$与$K_2$串联

| $K_1$ | $K_2$ | L |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

表2.1(b)$K_1$与$K_2$并联

| $K_1$ | $K_2$ | L |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

我们称第一种情形中两个开关在逻辑上构成“与”的关系，第二种情形中两个开关在逻辑上构成“或”的关系。将开关$K_1$和$K_2$看成逻辑变量，则灯泡L是$K_1$、$K_2$的函数，即逻辑变量的函数，称之为逻辑函数。逻辑函数由逻辑变量和逻辑运算构成。上面提到的“与”和“或”是最基本的逻辑运算。

### 2.1.2　基本逻辑运算

布尔代数中有三种最基本的逻辑运算，它们是“与”、“或”和“非”。这三种基本的逻辑运算可用来描述任何逻辑关系，构造任何逻辑函数。

当多个逻辑变量进行“与”运算时，只要其中有一个变量为假，则运算结果为假。

当多个逻辑变量进行“或”运算时，只要其中有一个变量为真，则运算结果为真。

“非”运算属于单目运算。当自变量为真时，结果为假，当自变量为假时，结果为真。非运算虽然简单，但在处理复杂数据的运算当中，尤其是在实际系统中起着非常重要的作用。

“与”运算的运算符为“AND”，或者“*”，或者“·”，或者省略。下面写法都是等价的：

A　AND　B

A　*　B

A·B

AB

“或”运算的运算符为“OR”，或者用“+”表示。下面的表达式是等价的：

A+B

A OR B

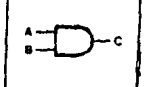
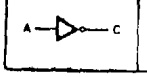
“非”运算的运算符为“NOT”，或者在自变量顶部写一横线，或者在自变量前写一斜线来表示。如

NOT A

和 $\overline{A}$ 或/A

是相互等价的。（以上关系，参见表2.2）

**表2.2 布尔代数的3种基本运算**

| 逻辑符号 | 真值表 | 逻辑表达式 | 功能 | 逻辑运算符 |
|---|---|---|---|---|
|  |  | C=A·B | 或 | “+” |
|  |  | C=A·B 或 C=A·B 或 C=AB | 与 | “·”或“ ” |
|  |  | C=A C=/A | 非 | “/”或“—” |

实际上，逻辑表达式往往是以上三种运算的混合运算。在混合运算表达式中，括号内的运算优先于括号外的运算，其次是“非”运算，再其次是“与”运算，再其次是或运算。

### 2.1.3 逻辑函数的表示方法

除了逻辑表达式以外，逻辑函数还有两个表示方法：真值表法和语言描述。

所谓真值表法，就是用穷举的方法，将自变量的全部可能的值与对应的逻辑函数值排列在一张表内，从而显示逻辑函数关系的方法。例如表2.3就是一张典型的真值表。

**表2.3 真值表**

| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

由真值表可以找到相应的逻辑表达式。方法是：从真值表中找出函数值为1的各变量的“与”（输入变量为1时写成原变量，输入变量为0时写成该变量的“非”），将它们进行“或”就得到逻辑表达式。

例如，表2.3中，找出C=1的自变量的“与”为：$A\overline{B}$和$\overline{A}B$，再进行“或”运算得到

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

逻辑函数可用语言描述来表达。如上例中C是A、B的“异或”。

## 2.2 布尔代数基本理论

本书简要介绍布尔代数的公理、定理。读者若有兴趣、可参阅有关书刊自己证明和分析它们。

### 2.2.1 公理

（1）公元的加与乘

（a） $x+0=x$

（b） $x*1=x$

（2）交换律

（a） $x+y=y+x$

（b） $x*y=y*x$

（3）分配律

（a） $x+(y*z)=(x+y)*(x+z)$

（b） $x*(y+z)=(x*y)+(x*z)$

（4）互补律

（a） $x+\bar{x}=1$

（b） $x*\bar{x}=0$

可以看出，以上公理是成对出现的。可以称其中一个是另一个的“对偶”。“对偶律”指出：对于一个公理、定理或等式，将所有项置换成它的对偶，则可产生一个在逻辑上等价的公理、定理或等式。对偶的形成规则是：

（a）0的对偶是1，1的对偶是0

（b）AND的对偶是OR，OR的对偶是AND

（C）其它对偶是其本身。

“对偶律”广泛应用于逻辑代数中，特别是在定理、命题证明时经常被采用。

### 2.2.2 定理

（1）等幂律

（a）$x*x=x$

（b）$x+x=x$

（2）0、1的特性

（a）$x*0=0$

（b）$x+1=1$

（c）$\bar{0}=1$

（d）$\bar{1}=0$

（3）吸收率

（a）$x*(x+y)=x$

（b）$x+(x*y)=x$

（4）第二吸收律

（a）$x*(\bar{x}+y)=x*y$

（b）$x+(\bar{x}*y)=x+y$

（5）结合律

（a）$x+(y+z)=(x+y)+z$

（b）$x*(y*z)=(x*y)*z$

（6）狄·莫根定理

（a）$\overline{(x+y)}=\bar{x}*\bar{y}$

（b）$\overline{(x*y)}=\bar{x}+\bar{y}$

（7）广义狄·莫根定理

（a）$\overline{x_1+x_2\cdots+x_n}=\overline{x_1}\,\overline{x_2}\cdots\overline{x_n}$

（b）$\overline{x_1x_2\cdots x}=\overline{x_1}+\overline{x_2}+\cdots+\overline{x_n}$

（8）还原律

$\overline{(\bar{x})}=x$

（9）包含律

（a）$xy_1+\bar{x}y_2+y_1y_2=xy_1+xy_2$

（b）$(x+y_1)(\bar{x}+y_2)(y_1+y_2)$

$=(x+y_1)(\bar{x}+y_2)$

（10）若$xy=y$且$x+y=y$、则$x=y$

### 2.2.3 布尔代数三个重要法则

（1）代入法则

若一等式成立，则将式中所有出现某变量的地方代之以一逻辑上相等的函数，则等式仍然成立。

（2）反演法则

已知F求F时，只需将F表达式中的与换成或、或换成与，0换成1、1换成0，原变量换成反变量，反变量换成原变量即可。

（3）对偶法则

若某等式成立，则其对偶一定成立。反之亦然。

一等式的对偶的对偶是该等式本身。

### 2.2.4 展开定理

（1）对于任一逻辑函数，可将其中任一逻辑变量提出，形成“积之和”的表达式：

$$F(x_1, x_2, \cdots, x_n)=x_1*F(1, x_2, \cdots, x_n)+\bar{x}_1*F(0, x_2, \cdots, x_n)$$

例如：

$$F=AB+BC+A\bar{C}$$
$$=A(1*B+BC+\bar{1}*C)+\bar{A}(0*B+B\bar{C}+\bar{0}*C)$$
$$=A(B+BC)+\bar{A}(B\bar{C}+C)$$
$$=AB+\bar{A}C$$

这实际上就是包含律（定理9）

（2）也可将任意逻辑表达式写成“和之积”的形式：

$$F(x_1, x_2, \cdots, x_n)=(x_1+F(0, x_2, \cdots, x_n))$$