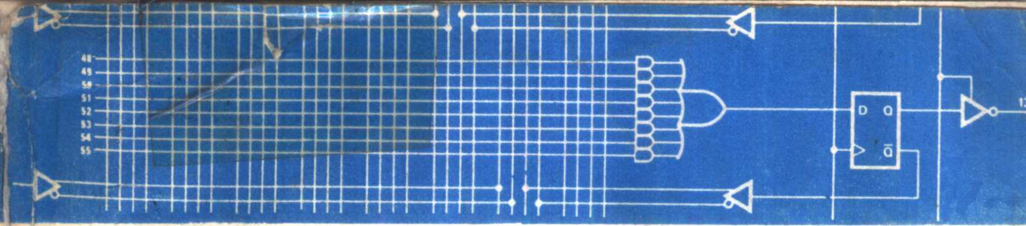
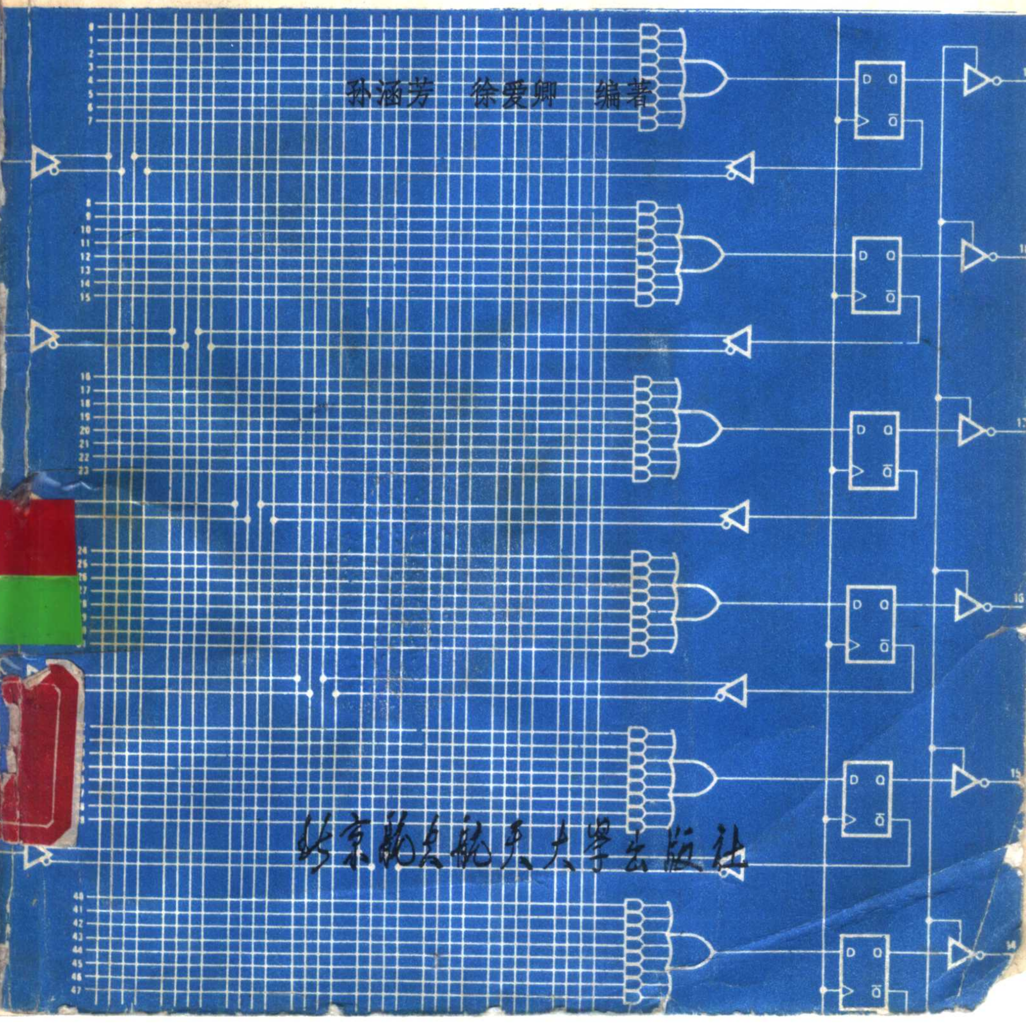


可编程逻辑器件

PAL和GAL



孙涵芳　徐爱卿　编著

北京航空航天大学出版社

可编程逻辑器件

PAL和GAL

孙涵芳
徐爱卿 编著

北京航空航天大学出版社

简　　介

PAL和GAL是已经获得广泛应用的两种可编程逻辑器件（PLD），它们属于专用集成电路（ASIC）的一个重要分支。PLD可灵活地编程以实现各种逻辑功能，PAL和GAL都有加密功能，GAL器件还可以反复编程，且编程快捷。PLD是构成数字系统的理想器件，可在很大程度上取代传统的中小规模集成的标准器件。

本书着重叙述PAL和GAL器件的逻辑功能、性能特点和开发工具，并以丰富的实例说明器件的应用。本书还以适当的篇幅介绍了数字逻辑设计的基础知识。

本书可供从事数字系统开发工作的广大科技工作者阅读，也可作为大中专院校有关数字电路和逻辑设计课程的教学参考书。

可编程逻辑器件

PAL 和 GAL

KEBIANCHENG LUOJI QIJIAN

•

孙涵芳、徐爱卿　编著

责任编辑　杨昌竹

•

北京航空航天大学出版社出版

北京密云华都印刷厂印刷

新华书店总店科技发行所发行　各地新华书店经售

•

开本：850×1168　1/32　印张：8　字数：215千字

1990年2月第一版　1990年2月第一次印刷

印数：5000册　定价：4.05元

ISBN 7-81012-160-X/TP·026

前　言

专用集成电路ASIC (Application-Specific Integrated Circuit) 是采用大规模集成工艺制造的数字逻辑器件，是继中小规模标准数字逻辑器件（如74系列的TTL器件）之后发展起来的新技术。作为ASIC的一个分支，可编程逻辑器件PLD (Programmable Logic Device) 的发展，已有十多年的历史。从70年代初期推出第一种PLD器件——可编程只读存储器PROM (Programmable Read Only Memory) 至80年代中推出通用阵列逻辑GAL (Generic Array Logic)，PLD已有了长足的发展。由于可编程，使用方便灵活，这类器件特别适用于产品开发阶段和小批量生产的系统中。它们已经在国外获得广泛的应用，在国内也引起科技工作者的注目，近年来已经在不同领域中获得应用。

我国有关部门已经把推广应用ASIC技术作为近期内技术突破口之一。我们编写这本书，旨在推广应用这项新技术，使它能尽快地成为从事数字系统设计的广大科技工作者的有力工具。

由于水平所限，书中难免有错误和欠妥之处，敬请读者批评指正。

编著者

1989.12于东方科技（北京）有限公司

北京信息工程学院

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

绪　论

在数字系统中，大量使用数字逻辑器件。我们可把这些器件分为3大类：

- 中小规模集成的标准器件，如74和54系列的TTL器件，74CH和CD4000系列的CMOS器件等等；
- 大规模和超大规模集成的微处理器，这类器件是靠软件来实现逻辑功能的；
- 专用集成电路ASIC（Application-Specific IC）。

第一类标准器件可说是目前世界上用得最广泛的集成器件。芯片本身价格低廉，性能好，但其集成度低，功能有限，灵活性比较差。构成系统时，有大量的芯片间的连线，要采用各种不同性能和功能的芯片。最终导致系统可靠性差，费用高，功耗多，体积大。

微处理器是靠软件配置的逻辑器件，它具有其它类型器件所难以匹敌的灵活性，用户几乎可以随心所欲地靠它来实现各种不同的逻辑功能。但是，在某些场合下，速度低是这类器件的致命弱点；此外，在微处理器应用系统中，经常还要求众多的外围接口芯片。作者曾为之大力推广的单片机多少缓解了后一问题。

ASIC器件可以弥补前两类芯片存在的缺陷。ASIC本身又可划分为3类：标准单元（Standard Cell），门阵列（Gate Array）和可编程逻辑器件 PLD（Programmable Logic Device）。

标准单元是一些预先配置好、经过测试、具有一定功能和性能的逻辑块。制造商把自己生产的标准单元纳入单元数据库，这些数据库中甚至可以包括一些微处理器。用户可以从标准单元数据库中选取合用的逻辑块，以构成自己所需要的专用功能芯片。

但用这类标准单元构成专用芯片时，要经过一定的设计过程，通常要花费数周或数月的时间，生产过程还得花数周时间；另外，用这种途径获得所需的芯片，还使客户承担额外的风险，因为在调试阶段，逻辑设计常常还要修改，这将使系统开发过程更长，代价更高。

门阵列是预先制造好的硅阵列，一般包括几种最基本的逻辑，如与非门、触发器和各种缓冲器等。在这类“半成品”的芯片中，留有一定的连线区。设计者可以选取上述最基本的逻辑单元，确定连线方式，以构成所需要的功能芯片。生产厂家根据所设计的连线方式，对门阵列的连线区进行最后的布线，就可以获得所需的芯片。

上述两种ASIC电路都要由用户向芯片生产厂家定做，适于批量较大的产品中使用。近年来Xinlinx公司推出一种可编程门阵列PGA产品，可由终端用户编程。对于小批生产的系统，或在系统开发过程中，则采用PLD器件更为适宜。PLD是用户可配置的逻辑器件。在小批量情况下，PLD的成本较低，它更为灵活，并且有较高的性能，可靠性高，可以利用简单而有效的开发工具，承担的风险较小，设计周期短。

PROM、EPROM 和E^2PROM 都是可编程逻辑器件，它们一般只用于储存CPU要执行的指令和表格常数，是数字计算机中的一个部分。本书中叙述的PLD与普通的可编程只读存储器在功能上略有区别，它们可以单独使用，实现一定的逻辑功能，但两者的基本单元的结构是一样的。

与可编程ROM器件相似，PLD也可以分为一次性写入PLD，紫外线擦除、电可改写的PLD和电擦除、电可改写的PLD。它们分别采用双极型熔丝式工艺、UVCMOS 工艺和 E^2CMOS 工艺。

双极型熔丝式工艺PLD是最早推出的PLD器件，第二章中叙述的PAL就是采用这种工艺的 PLD 器件。它具有较高的速

度，但功耗较大，对系统电源和冷却方面的要求较高，限制了芯片本身的功能密度。它的另一个重大缺点是采用熔丝式的一次性写入工艺，因而无法在编程前对它进行完善的测试，制造商只能对它采用较复杂的仿真（功能）和相关（开关性能）测试法。由于只能一次性编程，修改设计后，原有的器件就不能再利用了。

UVCMOS工艺虽然能弥补双极型工艺的一些缺陷，但它本身也有很多不足之处。由于CMOS工艺，功耗较低，具有可擦除可改写的能力，但器件速度较低，擦除过程较长。相对于双极型工艺，UVCMOS的可测试性提高了。但这种工艺的擦除时间很长（一般为20分，改进后的紫外线擦除可以把擦除时间缩短到10分钟以内），且要用紫外光源。为了实现较完善的测试，须对芯片进行反复的擦除改写，这将使生产商付出高昂代价。作为折衷，生产商只对芯片进行一次性擦除，故只能进行部分的实际测试，也不能达到100%的测试可靠性。此外，为了紫外线擦除，要采用代价较高的窗口式封装。有的生产商为了降低成本，采用了无窗口式的封装，这样一来也就失去了反复使用的优点。

E^2CMOS是一种比较先进的工艺，本书重点介绍的GAL器件就是采用这种工艺的。GAL是英文字母 Generic Array Logic 的字首缩写词，是LATTICE公司的专用注册商标名。正如它的名称叫法，它具有较通用的结构，为用户提供了最大的设计和使用上的灵活性。

与其它的PLD器件相比，GAL具有功能上和输出结构上的最大通用性和灵活性。采用20脚和24脚两种GAL器件，在功能和性能上几乎可以取代整个74LS系列、54LS系列、74HC系列和CD4000系列的器件（只有少量的例外）。借助于简单有力的开发工具，可以大大简化设计和开发过程。无论对制造商还是对用户，由于品种单一，可以显著地降低管理费用。GAL具有双极型器件的高速性能，而功耗却只有它的1/2或1/4。E^2CMOS工艺使它可以反复擦除、改写，且擦除时间仅为10毫秒，不需要采

用紫外光源，不需要窗口式的封装，在封装前后都可以对芯片进行反复的、完善的、实际的测试，因此制造商可以提供100%测试可靠性的保证。

总之，GAL 是一种结构灵活、性能优越、功能可靠的可编程逻辑器件。美国还有其它生产PLD的公司如Altera 公司，该公司正在研制一种叫做MAX的多重矩阵PLD器件。据称，“一个MAX 器件有可能替代50个或更多的TTL 封装件”。作者有志于向我国广大科技工作者大力推广这种先进技术。我们深信，对于大多数从事应用技术的读者而言，若能把单片微型计算机技术与GAL技术结合起来，一定会使你开发的应用系统在性能价格比上更上一层楼。

本书第一章简略介绍逻辑设计的基本知识及PLD电路的习惯表示法，对于熟悉逻辑设计的读者可以只阅读其中最后一节。第二章概括叙述PAL 的内部结构和工作原理，PAL 和GAL有很多相似之处，且前者电路结构更简单些，因此从PAL 入手来阐述PLD的工作原理，更容易为一般读者所接受。第三章叙述GAL 的内部结构和工作原理，是本书的重点之一。PAL和GAL的开发都要依赖于简易而有力的开发工具，两者的开发工具有极大的相似性，第四章将详细介绍这些工具。最后一章——第五章将给出众多的应用实例，通过这些实例，可以帮助读者进一步掌握PAL 和GAL 的原理及应用。

第一章　数字逻辑设计的基本知识

可编程逻辑器件PAL和GAL在数字系统（包括计算机系统）中有着美好的应用前景，80年代中期问世的一些电子产品中已采用了这类器件，它们正在逐渐取代一些标准的中小规模逻辑器件。借助于强有力的开发工具，人们可以很少甚至根本就不理会这些器件的内部结构而学会使用它。但是开发应用这类器件都不可缺少逻辑设计的知识。

本书旨在向读者推荐和介绍可编程逻辑器件PAL和GAL，无意于系统地叙述逻辑设计的知识，但作为应用PLD器件的必备知识，本章将对逻辑电路的基本概念和设计方法作概括性的叙述。本章的最后一节将介绍PLD电路的表示法。

1.1　一些基本概念

1854年英国数学家和逻辑学家乔治·布尔(George Boole)提出了逻辑代数，也称布尔代数，这种代数系统中的变量和函数都只能取0（假）和1（真）两种值。在逻辑代数中，最基本的运算只有3种："与"（AND）、"或"（OR)和"非"(NOT)。逻辑与也叫逻辑乘，其运算符常用"·"表示，其运算结果称为"积"；逻辑或也叫逻辑加，其运算符常用"+"表示，运算结果称为"和"；它们都是多元运算。逻辑非是一元运算，运算符为"—"。这3种基本运算分别对应于数字逻辑电路中的3种基本门电路：与门、或门和非门。如图1-1所示。它们的逻辑关系可用真值表来表示，见表1-1。

一个函数，若其自变量和函数值都只能取0或1两种值，这样

此为试读，需要完整PDF请访问：www.ertongbook.com

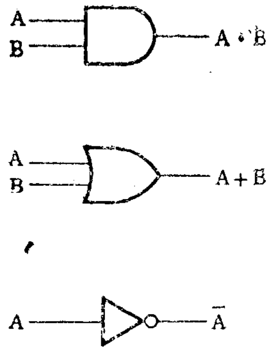


图1-1　3种基本逻辑运算

表1-1　3种基本逻辑运算的真值表

A	B	P = A · B	P = A + B	P = $\overline{A}$	P = $\overline{B}$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

的函数就称为逻辑函数。一个逻辑函数可用多种形式来表示，逻辑代数式和真值表是两种最常用的表示法。图 1-1 中的式子和表 1-1 就是最基本的逻辑函数的两种表示法。逻辑代数式在形式上和普通代数式相同，都是由一些代表变量的标识符和运算符（包括括号）组成。但逻辑代数中没有除法和减法。逻辑代数的运算优先顺序和普通代数一样。因此，

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

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

$$\overline{A + B} \neq \overline{A} + \overline{B}$$ （非运算相当于普通代数中取负）

等等。真值表是将逻辑变量的各种可能取值和相应函数值排列组成的表格。一个n 元变量的逻辑函数真值表，应有2^n种可能的取值。

除了3种最基本的逻辑运算之外，常用到的逻辑运算还有：与非(NAND)，或非(NOR)，异或(XOR)和异或非（XNOR）。在二元运算中，可分别表示为$\overline{A\cdot B}$、$\overline{A+B}$、$A\oplus B=A\cdot\overline{B}+\overline{A}\cdot B$和$\overline{A\oplus B}=A\cdot B+\overline{A}\cdot\overline{B}$。它们的真值表见表1-2。与这些运算相对应的门电路的逻辑图示于图1-2。

表1-2　一些常用逻辑函数的真值表

A	B	$\overline{A\cdot B}$	$\overline{A+B}$	$A\oplus B$	$\overline{A\oplus B}$
0	0	1	1	0	1
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1

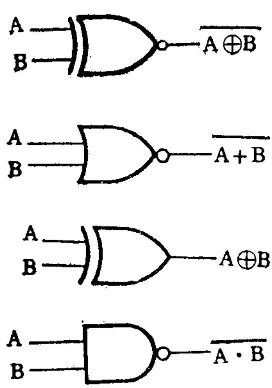


图1-2　一些常用的门电路

1.2　逻辑代数的基本定律和运算规则

很多参考文献都详细地叙述和证明了逻辑代数的基本定律，我们仅将这些定律综合起来，列于表1-3中。该表是按对偶的规律排列的，表的左部与右部相互对偶。表中作有“*”记号的公式是逻辑代数的特有规律，未标有“*”者是一般代数系统普遍适用的规律。互补律，也称反演律，它就是著名的德·摩尔根

(De Morgan) 定理，在逻辑简化或转化中十分有用。

表1-3　逻辑代数的基本定义和定

基本定义 逻辑非 $\bar{0}=1^*$ 逻辑加 $0+0=0$ $0+1=1$ $1+0=1$ $1+1=1^*$	**基本定义** 逻辑非 $\bar{1}=0^*$ 逻辑乘 $1\cdot 1=1$ $1\cdot 0=0$ $0\cdot 1=0$ $0\cdot 0=0$
反转律 $\bar{\bar{A}}=A$	
单变量式 $A+0=A$　$A+1=1^*$ $A+A=A^*$　$A+\bar{A}=1^*$	**单变量式** $A\cdot 1=A$　$A\cdot A=A^*$ $A\cdot 0=0$　$A\cdot\bar{A}=0^*$
多变量式 代数律 交换律 $A+B=B+A$ 结合律 $(A+B)+C$ $=A+(B+C)$ 分配律 $A+B\cdot C$ $=(A+B)(A+C)^*$ 吸收律 $A+A\cdot B=A^*$ $A+\bar{A}\cdot B=A+B^*$ $A\cdot B+A\cdot\bar{B}=A^*$ $AB+\bar{A}C+BCD$ $=AB+\bar{A}C^*$ 互补律 $\overline{A+B}=\bar{A}\cdot\bar{B}^*$ 交叉互换律 $AB+\bar{A}C=(A+C)(\bar{A}+B)^*$	**多变量式** 代数律 交换律 $A\cdot B=B\cdot A$ 结合律 $(A\cdot B)\cdot C$ $=A\cdot(B\cdot C)$ 分配律 $A\cdot(B+C)$ $=AB+AC$ 吸收律 $A\cdot(A+B)=A^*$ $A\cdot(\bar{A}+B)=A\cdot B^*$ $(A+B)(A+\bar{B})=A^*$ $(A+B)(\bar{A}+C)(B+C+D)$ $=(A+B)(\bar{A}+C)^*$ 互补律 $\overline{A\cdot B}=\bar{A}+\bar{B}^*$ 交叉互换律 $(A+B)(\bar{A}+C)=AC+\bar{A}B^*$

逻辑符"+"与"·"互换
逻辑变量"0"与"1"互换
相互对偶

注：表中标有 * 者表示逻辑代数的特有规律。

在逻辑代数的公式推导中，有3条十分有用的规则，可以帮助我们由一些基本公式得到更有用的公式，这3条规则是：

1．代入规则

在任何一个逻辑等式中，将等式两边所有出现某一变量A的地方，都代之以同一函数P，则等式仍然成立，这个规则叫做代入规则。这是大家十分熟悉的规则，不再细论。

2．反演规则

对任何一个逻辑函数表达式P，若

1）所有“·”换成“+”，所有“+”换成“·”；

2）所有“0换成“1”，所有“1”换成“0”；

3）所有原变量换成反变量，反变量换成原变量

则所得的表达式就是P的反函数$\overline{P}$。

举一个简单的例子：在逻辑图中，我们常用两个等价电路来表示同一个逻辑门——或门，见图1-3。事实上，图左部的电路对应于公式$P=A+B$，根据反演规则，可以得到另一个公式$\overline{P}=\overline{A}\cdot\overline{B}$，这正是图左部的电路所实现的逻辑关系。

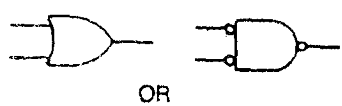


图1-3　或门的两种表示法

运用反演规则时，要特别注意运算符的优先级问题。例如，对$P=\overline{A}\cdot\overline{B}+C\cdot D$，应先对$\overline{A}\cdot\overline{B}$和$C\cdot D$运用反演规则，然后再对“+”反演，因此，它的反演公式是$\overline{P}=(A+B)\cdot(\overline{C}+\overline{D})$，而不是$\overline{P}=\overline{A}+B\cdot\overline{C}+D$。

3．对偶规则

对于任何一个逻辑函数表达式P，如果把其中“+”换成“·”，“·”换成“+”；“1”换成“0”，“0”换成“1”，那么，可得到一个新的表达式P'，称P'是P的对偶式。显然，P也是P'的对偶式。若已知$P_1=P_2$，则有$P_1'=P_2'$，反之亦然，这就是对偶规则。

根据对偶规则，表1-3右半部分的公式完全可以由左半部分的

公式推得，这样，我们就把需要熟记的公式减少了一半。在运用对偶规则时，同样需要注意运算符的优先级问题。

1.3 逻辑表达式的代数化简法

逻辑表达式的化简具有重大的现实意义，逻辑表达式越简单，电路实现时所用的元件就越少，电路就越简单，费用就会降低。因此一名优秀的逻辑设计人员应熟练地掌握逻辑化简方法。

逻辑化简有多种方法，本节叙述代数化简法。

化简后的逻辑表达式通常采用5种最终的形式：与-或表达式、或-与表达式、与非-与非表达式、或非-或非表达式和与-或-非表达式。这5种表达方法是可以互相转换的。现举一个2:1多路开关为例，设B和C为2个输入端，A为控制端，Q为输出端，电路要求当A = 1时，Q = B；A = 0时，Q = C。显然，可用图1-4所示的逻辑图实现之。其对应的逻辑方程为：

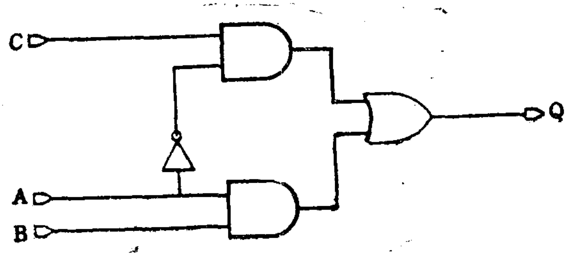


图1-4　2:1多路开关逻辑图

$$Q = AB + \bar{A}C \qquad \text{与-或表达式}$$

公式右部表达式是2个逻辑乘积项之和，因此可称为积-和表达式，或称为与-或表达式。运用交叉互换律和德·摩尔根定理（互补律），可得其它表达形式：

$$
\begin{aligned}
Q &= (\bar{A} + B)(A + C) && \text{或-与表达式} \\
&= \overline{\overline{AB}\cdot\overline{\bar{A}C}} && \text{与非-与非表达式} \\
&= \overline{\overline{\bar{A} + B} + \overline{A + C}} && \text{或非-或非表达式}
\end{aligned}
$$

$$=\overline{A}\overline{B}+\overline{A}\overline{C} \qquad \text{与-或-非表达式}$$

或-与表达式是逻辑和项之积，故也称为和-积式。由于本书介绍的PAL和GAL器件的结构适用于实现与-或式的逻辑关系，故下面将只介绍如何把复杂的逻辑表达式化简为最简的与-或表达式。

大家熟悉的PROM、EPROM和E²PROM存储器结构适于实现或-与表达式。顺便提一句，上面列出的式子中，与非-与非表达式和或非-或非表达式中都只用到一种运算符，我们可以证明，任何复杂的逻辑表达式都可以化为与非-与非式或或非-或非式。在数学上，把可用于表达任何逻辑函数的逻辑运算符的集合，称为逻辑运算的功能完全集。因此“与非”或“或非”运算本身就构成了一个功能完全集。这个概念对指导现实生产具有重要意义，因为一家生产工厂只需要制造与功能完全集相应的门电路，就可以通过这些门电路的相互联系实现任何逻辑功能。因此，一家工厂可专门生产与非门电路，而利用电路的互联，再生产出千变万化的逻辑器件。

逻辑表达式的代数化简法就是用逻辑代数中的基本定律和常用公式进行化简。这种化简方法极大地依赖于人们的经验和对公式运用的熟练程度。除代数律中的几个基本公式和著名的德·摩尔根定理之外，与-或表达式化简过程中常用到的基本公式还有（见表1-3）：

$$A+A\cdot B=A \tag{1}$$

$$A+\overline{A}\cdot B=A+B \tag{2}$$

$$A\cdot B+A\cdot\overline{B}=A \tag{3}$$

$$A\cdot B+\overline{A}\cdot C+B\cdot C=A\cdot B+\overline{A}\cdot C \tag{4}$$

或

$$A\cdot B+\overline{A}\cdot C+B\cdot C\cdot D=A\cdot B+\overline{A}\cdot C \tag{4'}$$

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

根据上述几个公式，可以归纳出5种常用的化简方法：

1. 吸收法　即利用式（1）吸收多余的乘积项。例如：

此为试读，需要完整PDF请访问：www.ertongbook.com