普通高等教育“十一五”国家级规划教材

（高职高专教育）

电子设计自动化

杨　静　主　编

杨　静　游周密　沈明山　编

高等教育出版社

Higher Education Press

普通高等教育“十一五”国家级规划教材
（高职高专教育）

电子设计自动化

杨　静　　主　编
杨　静　游周密　沈明山　　编

高等教育出版社

内容简介

本书以详细的实例介绍了 VHDL 硬件描述语言和可编程逻辑器件、Nios Ⅱ嵌入式系统开发流程，可以使读者较快地了解可编程逻辑器件和 Nios Ⅱ嵌入式系统先进的设计方法、开发流程和开发手段。

全书共七章，主要包括 CPLD/FPGA 的基本知识、CPLD/FPGA 基本开发流程、VHDL 语法与使用实例、状态机设计方法、基于 Quartus Ⅱ软件的多种逻辑设计流程、Nios Ⅱ嵌入式系统软/硬件设计流程和数字系统设计实例。

本书以实践为基础，图文并茂，开发流程完整详尽，可作为高职高专院校应用电子技术、电子信息工程技术、通信、电气自动化等专业学生的学习或实训教材，亦可作为电子设计竞赛 VHDL 硬件描述语言与 CPLD/FPGA 开发赛前辅导参考资料，也可供电子类在职研究开发人员和技术人员参考。

图书在版编目(CIP)数据

电子设计自动化/杨静主编. —北京：高等教育出版社，2009.6

ISBN 978-7-04-026672-6

Ⅰ. 电… Ⅱ. 杨… Ⅲ. ①电子电路-电路设计：计算机辅助设计-高等学校-教材 ②硬件描述语言，VHDL-程序设计 Ⅳ. TN702 TP312

中国版本图书馆 CIP 数据核字(2009)第 073919 号

策划编辑	刘 洋	**责任编辑**	李葛平	**封面设计**	张志奇	**责任绘图**	吴文信
版式设计	王艳红	**责任校对**	俞声佳	**责任印制**	尤 静		

出版发行	高等教育出版社	**购书热线**	010-58581118
社 址	北京市西城区德外大街 4 号	**咨询电话**	400-810-0598
邮政编码	100120	**网 址**	http://www.hep.edu.cn
总 机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经 销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印 刷	人民教育出版社印刷厂	**畅想教育**	http://www.widedu.com
开 本	787×1092 1/16	**版 次**	2009 年 6 月第 1 版
印 张	16	**印 次**	2009 年 6 月第 1 次印刷
字 数	380 000	**定 价**	20.80 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

物料号 26672-00

前　言

随着大规模集成电路和电子计算机技术的发展，电子产品设计方法发生了根本性的变革，以电子计算机辅助分析和设计为基础的电子设计自动化（Electronic Design Auntomation，EDA）技术已广泛用于集成电路与数字系统的设计中。电子设计自动化技术已成为现代电子系统设计的关键技术，是新一代电子设计工程师以及从事电子技术开发和研究人员的必备技能。

随着电子产品的集成化和复杂程度的提高，采用先进的电子器件和电子电路设计方法，可以大幅度缩短产品设计周期，并使设计产品小型化、低功耗、高速度、高性能，提高产品的竞争能力。

了解新的设计技术应当尽快掌握它的设计流程。本教材在编写时以实践为基础，以详细的实例让初学者了解基于 Quartus Ⅱ 和 Nios Ⅱ IDE 的可编程逻辑器件和片上系统（System On a Programmable Chip，SOPC）开发的基本流程，目的是为了让初学者尽快了解可编程逻辑器件先进的设计方法、基本开发流程和常用设计开发手段。

本教材共七章。第 1 章为 CPLD/FPGA 的基本知识，初学者应先了解这部分内容。第 2 章以两个实例让初学者了解 CPLD/FPGA 基于硬件描述语言和原理图的基本开发流程，并熟悉 Quartus Ⅱ 软件的基本使用。第 3 章较详细地介绍了 VHDL 语法和使用实例，可用于 CPLD/FPGA 小规模数字系统设计开发或底层模块设计。第 4 章通过实例较详细地介绍了状态机设计方法，使初学者了解较大规模数字系统的设计方法。第 5 章通过实例详细介绍了 Quartus Ⅱ 软件的 LPM 参数化宏模块、存储器、嵌入式锁相环、嵌入式逻辑分析仪 SignalTap Ⅱ 等多种逻辑设计流程以及层次化设计流程，它主要适用于较大规模 FPGA 设计开发。第 6 章通过一个实例详细介绍了 Nios Ⅱ 嵌入式系统软、硬件设计流程，它主要适用于在大容量 FPGA 中嵌入微处理器数字系统的设计开发。第 7 章通过 3 个数字系统设计实例，介绍了基于可编程逻辑器件进行数字系统设计的完整流程。

本教材由杨静、游周密编写，沈明山负责部分实例的编写和验证。杨静负责全书的统稿。

本教材初稿由王毓银教授审阅和修改，提出了许多宝贵意见。在此表示衷心的感谢。

由于电子设计自动化技术发展迅速，作者水平有限，本教材一定还存在不少缺点和不足之处，殷切期望读者指正。

编　者

2009 年 5 月

目　录

第 1 章　可编程逻辑器件 …… 1
1.1　可编程逻辑器件(PLD)基本结构 …… 1
1.1.1　可编程逻辑器件概述 …… 1
1.1.2　可编程逻辑器件基本结构 …… 1
1.1.3　可编程逻辑器件编程技术 …… 6
1.2　PAL 和 GAL 器件 …… 8
1.2.1　PAL 器件的基本结构 …… 8
1.2.2　GAL 器件的基本结构 …… 8
1.3　CPLD 基本结构 …… 9
1.3.1　Xilinx 公司 XC7300 系列器件结构 …… 10
1.3.2　Altera 公司 MAX7000 系列器件结构 …… 11
1.3.3　Altera 公司 FLEX10K 系列器件结构 …… 12
1.4　FPGA 基本结构 …… 13
本章小结 …… 16
习题 …… 17
第 2 章　Quartus Ⅱ软件基本设计流程 …… 18
2.1　可编程逻辑器件设计流程 …… 18
2.2　硬件描述语言逻辑设计流程 …… 20
2.2.1　建立新工程 …… 20
2.2.2　建立硬件描述语言逻辑设计源文件 …… 21
2.2.3　编译器选项设置 …… 22
2.2.4　器件引脚配置 …… 24
2.2.5　设计编译 …… 25
2.2.6　查看编译后结果 …… 27
2.2.7　逻辑功能仿真分析 …… 30
2.2.8　时序分析 …… 33
2.2.9　器件编程 …… 34
2.2.10　硬件描述语言逻辑设计流程实训题目 …… 36
2.3　原理图逻辑设计流程 …… 37
2.3.1　建立新工程 …… 37
2.3.2　创建新原理图文件 …… 38
2.3.3　放置元器件符号 …… 38
2.3.4　定义输入和输出引线 …… 40
2.3.5　逻辑符号之间的连接 …… 40
2.3.6　建立默认逻辑符号 …… 42
2.3.7　原理图逻辑设计流程实训题目 … 42
本章小结 …… 43
第 3 章　VHDL 硬件描述语言初阶 …… 44
3.1　VHDL 基本结构 …… 44
3.1.1　VHDL 最小结构 …… 44
3.1.2　VHDL 基本结构 …… 45
3.1.3　VHDL 基本语句 …… 46
3.2　VHDL 语言数据类型及运算操作符 …… 49
3.2.1　VHDL 数据对象 …… 49
3.2.2　VHDL 基本数据类型 …… 52
3.2.3　VHDL 预定义属性 …… 64
3.2.4　VHDL 逻辑运算符 …… 67
3.2.5　VHDL 关系运算符 …… 68
3.2.6　VHDL 算术运算符 …… 69
3.2.7　VHDL 并置运算符 …… 70
3.2.8　VHDL 符号运算符 …… 70
3.2.9　VHDL 省略赋值操作符 …… 71
3.2.10　VHDL 并列符 …… 71
3.3　VHDL 顺序处理语句 …… 72
3.3.1　信号代入语句 …… 72
3.3.2　变量赋值语句 …… 74
3.3.3　case 语句 …… 75
3.3.4　if 语句 …… 77
3.3.5　loop 语句 …… 78
3.3.6　next 语句 …… 80
3.3.7　exit 语句 …… 81

3.3.8 wait 语句 …… 82
3.3.9 null 语句 …… 83
3.4 VHDL 并行处理语句 …… 83
3.4.1 进程(process)语句 …… 83
3.4.2 并发信号代入语句 …… 85
3.4.3 条件信号代入语句 …… 85
3.4.4 选择信号代入语句 …… 86
3.4.5 块(block)语句 …… 87
3.4.6 元件声明/元件例化(component)语句 …… 88
3.4.7 生成(generate)语句 …… 91
3.5 VHDL 库和程序包 …… 93
3.5.1 VHDL 库 …… 93
3.5.2 VHDL 程序包 …… 93
3.6 VHDL 子程序 …… 95
3.6.1 函数语句 …… 95
3.6.2 过程语句 …… 96
3.7 组合逻辑电路设计示例 …… 98
3.7.1 逻辑门电路设计示例 …… 98
3.7.2 编码器设计示例 …… 99
3.7.3 译码器设计示例 …… 101
3.7.4 数据选择器设计示例 …… 103
3.7.5 运算器设计示例 …… 104
3.7.6 奇偶校验电路设计示例 …… 105
3.8 时序逻辑电路设计示例 …… 106
3.8.1 基本触发器示例 …… 106
3.8.2 寄存器示例 …… 107
3.8.3 计数器示例 …… 108
3.8.4 序列信号发生器示例 …… 110
本章小结 …… 111
习题 …… 112
第 4 章 硬件描述语言逻辑设计进阶 …… 118
4.1 状态机设计 …… 118
4.1.1 Moore 型状态机设计方法 …… 118
4.1.2 Mealy 型状态机设计方法 …… 120
4.1.3 MDS 图设计方法 …… 122
4.1.4 ASM 图设计方法 …… 125
4.2 硬件描述语言层次化设计 …… 128
4.2.1 “自上而下”层次化设计概述 …… 128
4.2.2 VHDL 层次化设计方法 …… 129
本章小结 …… 131
习题 …… 132
第 5 章 Quartus Ⅱ软件混合设计流程 …… 133
5.1 LPM 参数化宏模块逻辑设计流程 …… 133
5.1.1 建立新工程 …… 133
5.1.2 创建加法器宏模块符号 …… 134
5.1.3 创建减法器宏模块符号 …… 136
5.1.4 创建乘法器宏模块符号 …… 136
5.1.5 创建除法器宏模块符号 …… 137
5.1.6 创建数据选择器宏模块符号 …… 137
5.1.7 顶层逻辑设计 …… 138
5.1.8 LPM 参数化宏模块逻辑设计流程实训题目 …… 138
5.2 层次化设计流程 …… 140
5.2.1 建立新工程 …… 141
5.2.2 创建 4 位加法器模块符号 …… 141
5.2.3 创建 4 位减法器模块符号 …… 142
5.2.4 创建 2 选 1 数据选择器模块符号 …… 142
5.2.5 创建七段显示译码器模块符号 …… 144
5.2.6 顶层逻辑设计 …… 144
5.2.7 层次逻辑设计浏览 …… 147
5.2.8 层次化设计流程实训题目 …… 147
5.3 存储器逻辑设计流程 …… 148
5.3.1 建立新工程 …… 148
5.3.2 创建存储器初始化文件 …… 148
5.3.3 创建存储器模块符号 …… 149
5.3.4 创建正弦波信号发生器原理图 …… 150
5.3.5 正弦波信号发生器逻辑功能仿真 …… 151
5.3.6 使用嵌入式逻辑分析仪 SignalTap Ⅱ …… 152
5.3.7 使用在系统嵌入式寄存器数据编辑器 …… 156
5.3.8 移出嵌入式逻辑分析仪 SignalTap Ⅱ …… 160
5.3.9 存储器逻辑设计流程实训

题目 …… 160
5.4 嵌入式锁相环 PLL 模块设计流程 … 161
5.4.1 建立新工程 …… 161
5.4.2 创建锁相环 PLL 模块 …… 161
5.4.3 锁相环分频输出时序仿真 …… 164
5.4.4 嵌入式锁相环 PLL 模块设计流程实训题目 …… 165
本章小结 …… 165
第 6 章 Nios Ⅱ嵌入式系统软硬件设计流程 …… 166
6.1 典型 Nios Ⅱ嵌入式系统开发流程 … 166
6.2 生成可调试的 Nios Ⅱ系统 …… 167
6.3 生成 Nios Ⅱ系统顶层原理图 …… 173
6.4 Nios Ⅱ系统下载 …… 175
6.5 Nios Ⅱ系统软件编写 …… 177
6.6 Nios Ⅱ系统软件调试 …… 183
6.7 Nios Ⅱ嵌入式系统开发流程实训题目 …… 186
本章小结 …… 187
第 7 章 数字系统设计实训 …… 188
7.1 数字系统设计实训 1 …… 188
7.1.1 数字钟系统总体设计 …… 188
7.1.2 数字钟系统底层模块设计流程 …… 191
7.1.3 数字钟系统中层模块设计流程 …… 204
7.1.4 数字钟系统顶层模块设计流程 …… 212
7.1.5 数字钟系统顶层模块层次结构与设计下载验证 …… 214
7.1.6 数字钟实训题目 …… 216
7.2 数字系统设计实训 2 …… 217
7.2.1 可调低频正弦波信号发生器总体设计 …… 217
7.2.2 可调低频正弦波信号发生器底层模块设计流程 …… 220
7.2.3 可调低频正弦波信号发生器顶层模块设计流程 …… 227
7.2.4 可调低频正弦波信号发生器设计验证 …… 228
7.2.5 可调低频正弦波信号发生器设计实训题目 …… 231
7.3 数字系统设计实训 3 …… 231
7.3.1 Nios Ⅱ嵌入式软核基本硬件环境的建立 …… 231
7.3.2 Nios Ⅱ嵌入式系统软件开发 …… 240
本章小结 …… 243
参考书目 …… 245

第1章　可编程逻辑器件

1.1　可编程逻辑器件(PLD)基本结构

1.1.1　可编程逻辑器件概述

用于传统数字系统设计的基本器件主要为标准逻辑器件，如 TTL74 系列和 CMOS4000 系列等。标准逻辑器件的主要缺点是逻辑规模小、功耗大、可靠性低。设计一个数字系统往往要用多片标准器件，因此数字系统布局布线复杂，占用的印制电路板面积较大。

20 世纪 70 年代，世界各半导体厂家竞相开发了专用集成电路(Application Specific Integrated Circuit，ASIC)产品。ASIC 可分为全定制、半定制和可编程逻辑器件(Programmable Logic Device，PLD)三大类。其中，半定制、全定制 ASIC 产品的开发需要半导体厂家参与，设计周期长，开发费用高。而 PLD 的设计开发不需要半导体厂家的参与，适用于一般设计者使用，是集成电路中发展最快的器件之一。PLD 器件与标准逻辑器件相比，其主要特点是：

(1) 逻辑规模大。PLD 器件已进入大规模和超大规模集成电路时代。一片 PLD 器件的规模可达几十万甚至上百万逻辑门。用一片 PLD 器件就可实现一个数字系统，使电子产品体积小、功耗低、可靠性高。

(2) 硬件的软设计。采用 PLD 器件设计数字系统的主要工作是利用计算机及 PLD 开发软件进行逻辑设计、功能仿真，可大大降低系统设计成本。此外，还可利用优化元件库或专用模块库进行设计，提高设计效率，缩短设计周期。

(3) 在采用 PLD 器件设计逻辑电路时，设计者需要利用 PLD 器件开发软件和硬件。PLD 器件开发软件根据设计要求，可自动进行逻辑电路设计输入、编译、逻辑划分、优化和模拟，得到一个满足设计要求的 PLD 编程数据。逻辑功能模拟通过后，还需将 PLD 编程数据下载到 PLD 器件中，使 PLD 器件具有设计所要求的逻辑功能。

1.1.2　可编程逻辑器件基本结构

新买来的可编程逻辑器件不含有任何逻辑信息，需由用户往里存储逻辑信息，这一过程称为器件编程。那么，其逻辑信息是如何在可编程器件中保存的呢？下面进行详细介绍。

1. 组合逻辑和时序逻辑的表示

数字逻辑电路一般分为两种类型，一种称为组合逻辑电路，另一种称为时序逻辑电路。典型

的组合逻辑电路是1位半加法器，设输入加数和被加数为 A、B，和输出及溢出指示为 F、C。1位半加法器输入－输出关系真值表如表1-1-1所示，其逻辑表达式为

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

表1-1-1　1位半加法器输入-输出关系真值表

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

典型的时序电路是同步十进制计数器，设输入为计数时钟 CP，输出为 Q_4、Q_3、Q_2、Q_1。十进制计数器输入－输出关系如表1-1-2所示，其状态转移方程为

表1-1-2　同步十进制计数器状态转移表

Q_4^N	Q_3^N	Q_2^N	Q_1^N	Q_4^{N+1}	Q_3^{N+1}	Q_2^{N+1}	Q_1^{N+1}
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0

$$\begin{cases} Q_1^{N+1}=\overline{Q}_1^N\cdot CP\uparrow \\ Q_2^{N+1}=[\overline{Q}_4^N\,\overline{Q}_2^N Q_1^N+Q_2^N\,\overline{Q}_1^N]\cdot CP\uparrow \\ Q_3^{N+1}=[\overline{Q}_3^N Q_2^N Q_1^N+Q_3^N\,\overline{Q}_1^N+Q_3^N\,\overline{Q}_2^N]\cdot CP\uparrow \\ Q_4^{N+1}=[\overline{Q}_4^N Q_3^N Q_2^N Q_1^N+Q_4^N\,\overline{Q}_1^N]\cdot CP\uparrow \\ Z=Q_4^N Q_1^N \end{cases}$$

无论组合逻辑电路还是时序逻辑电路，它们都可以用表格或与－或逻辑方程表示。可编程逻辑器件的内部有两种基本器件结构：(1) 与－或阵列结构，对应逻辑方程描述；(2) 查找表结构，对应真值表或状态转移表逻辑描述。

2. 与－或阵列结构

基于**与**－**或**阵列的可编程逻辑器件基本结构如图 1-1-1 所示。在这类可编程逻辑器件中，实际上已内置了多个**与**逻辑门和多个**或**逻辑门，它们一般都是按照一定规律排列在器件中，用于完成**与**逻辑运算和**或**逻辑运算。此外，为了满足输入为反变量和时序输出及输出反馈等要求，还内置了多个输入变量变换电路和输出类型控制电路。

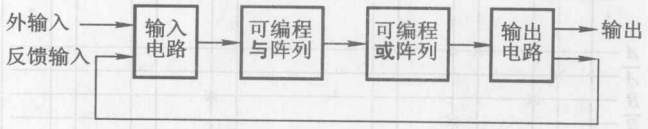

图 1-1-1　与－或阵列结构

3. 与－或结构可编程逻辑器件基本组成

由图 1-1-1 可知，**与**－**或**阵列结构可编程逻辑器件主要由四部分构成，即输入电路、可编程**与**阵列、可编程**或**阵列和输出电路。

（1）输入电路

输入电路的主要作用是将外输入信号或反馈输入信号转换成其相应的原变量和反变量，其实现电路及简化符号如图 1-1-2 所示。

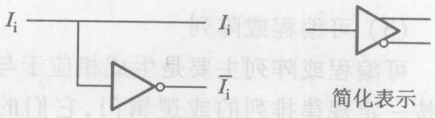

图 1-1-2　输入电路

（2）可编程**与**阵列

可编程**与**阵列主要是生成相相应于**与**－**或**逻辑表达式中的**与**项，即乘积项。在可编程逻辑器件中，有多个按一定规律排列的**与**逻辑门，它们的输入来自输入电路的输出，其示意图如图 1-1-3所示。

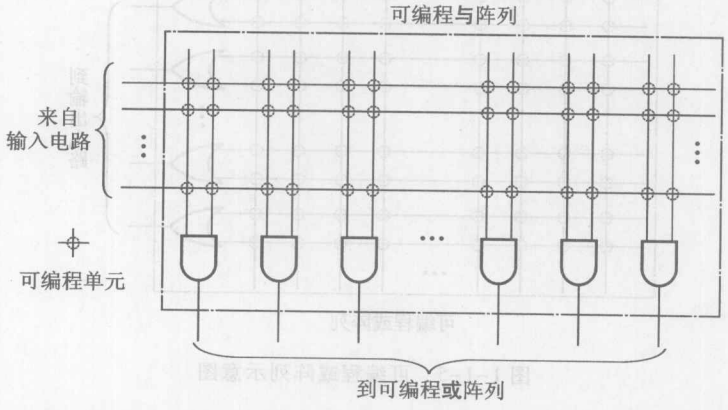

图 1-1-3　可编程**与**阵列示意图

在图 1-1-3 中，两条导线交叉部分的圆圈表示可编程，称为编程单元。通常，如果两条导线相连接，一般用“×”符号表示此编程单元编程连接。如果两条导线本身就固定连接，用“•”符号表示为此编程单元固定连接（不可编程）。无任何标记则表示此编程单元编程为不连接。用“×”、“•”等符号表示的阵列图也称为逻辑映象图。

例如，1 位半加法器输入－输出关系可用逻辑表达式表示为

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

则可编程**与**阵列逻辑映象图如图 1-1-4 所示。

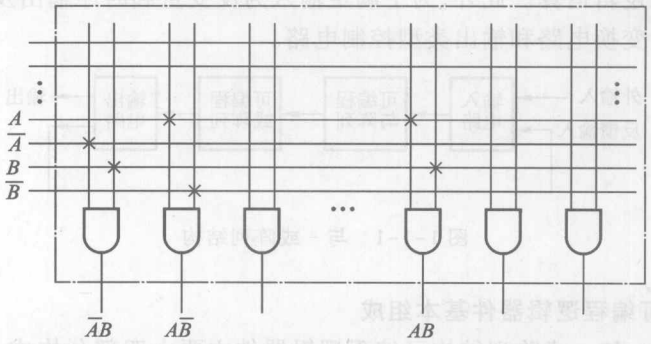

图 1-1-4　一位半加法器可编程**与**阵列逻辑映象图

(3) 可编程**或**阵列

可编程**或**阵列主要是生成相应于**与**－**或**逻辑表达式中的**或**项。在可编程逻辑器件中，有多个按一定规律排列的**或**逻辑门，它们的输入来自可编程**与**阵列的输出，其示意图如图 1-1-5 所示。

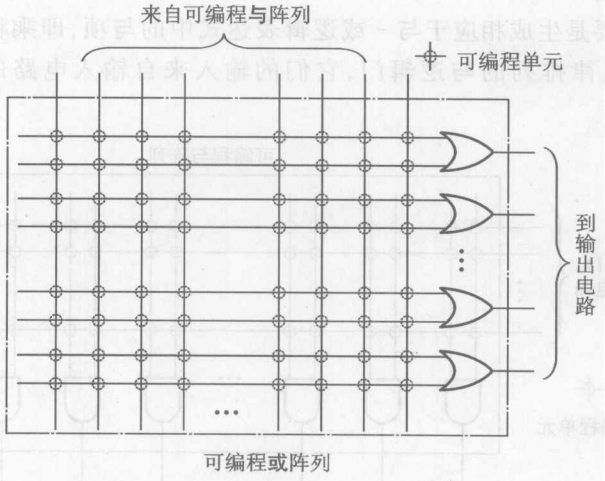

图 1-1-5　可编程**或**阵列示意图

同可编程逻辑**与**阵列一样，在图 1-1-5 两条导线交叉部分的圆圈表示可编程。例如，用可编程逻辑器件实现 1 位半加法器，则可编程**或**阵列逻辑映象图如图 1-1-6 所示。

(4) 输出电路

输出电路主要是完成直接或寄存输出及输出信号的反馈。在可编程逻辑器件中，组合逻辑电路可直接输出，而对时序逻辑电路，则必须经寄存器输出或反馈输出。输出电路有多个按一定

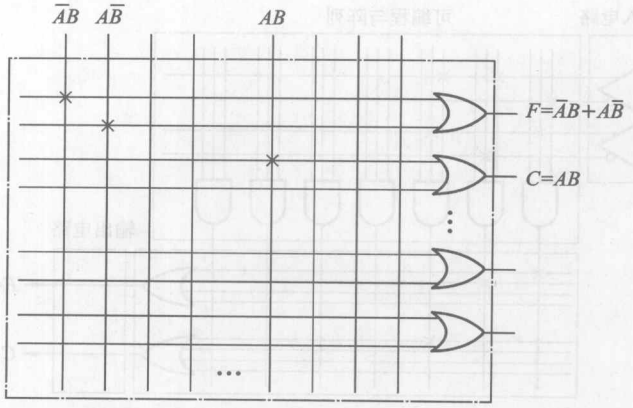

图 1-1-6　1 位半加法器可编程或阵列逻辑映象图

规律排列的寄存器，它们的输入来自可编程或阵列的输出，其带异步控制输入的输出电路如图 1-1-7所示。

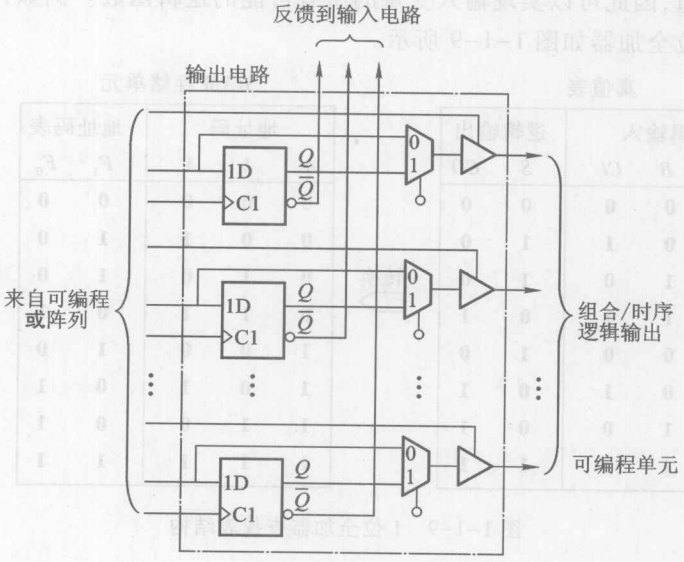

图 1-1-7　输出电路

综上所述，如用**与－或**结构可编程逻辑器件实现 1 位半加法器，则可编程逻辑器件逻辑映象图如图 1-1-8 所示。

4. 查找表结构

查找表（Look Up Table，LUT）结构与**与－或**阵列结构的主要区别是在实现逻辑运算上。**与－或**阵列结构是用可编程**与**阵列和**或**阵列来实现逻辑运算，而在查找表结构可编程逻辑器件中，用存储逻辑的存储单元来实现逻辑运算。

查找表实际上是一个根据逻辑真值表或状态转移表设计的 RAM 逻辑函数发生器。在查找

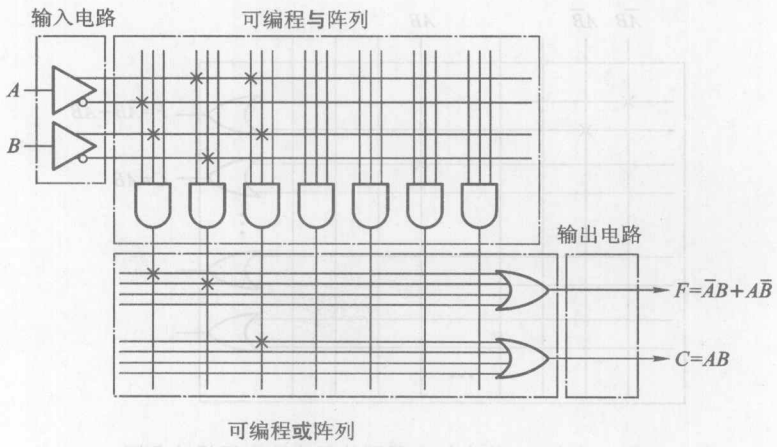

图 1-1-8　1 位半加法器可编程逻辑器件逻辑映象图

表结构中，RAM 存储器预先加载要实现的逻辑函数真值表，输入变量作为地址用来从 RAM 存储器中选择输出逻辑值，因此可以实现输入变量的所有可能的逻辑函数。例如，用查找表结构可编程逻辑器件实现 1 位全加器如图 1-1-9 所示。

真值表

逻辑输入			逻辑输出	
A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

转换

RAM 存储单元

地址码			地址码表	
A_2	A_1	A_0	F_1	F_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

查找输出

图 1-1-9　1 位全加器查找表结构

图 1-1-9 中，输入的信号组合首先转换为 RAM 存储单元的地址码，根据地址码查找相对应的存储单元，然后将相对应的存储单元存储的逻辑值送到输出电路。例如，当前输入引脚 $A(A_2)$、$B(A_1)$、$CI(A_0)$ 为 **110**，则内部逻辑控制电路将输出指针指向存储单元的倒数第二行，然后将该地址行对应的输出 $F_1(S)=\mathbf{0}$ 和 $F_0(CO)=\mathbf{1}$ 送到输出引脚，实现 1 位加法器的运算。

1.1.3　可编程逻辑器件编程技术

在采用 PLD 设计逻辑电路时，设计者需要利用 PLD 开发软件和硬件。PLD 开发软件根据设

计要求，可自动进行逻辑电路设计输入、编译、逻辑划分、优化和模拟，得到一个满足设计要求的PLD编程数据。逻辑功能模拟通过后，还需将PLD编程数据下载至PLD中，使PLD具有设计所要求的逻辑功能。

可编程逻辑器件从编程技术上一般分为两类，一类是一次性编程，另一类是可多次编程。一次性可编程器件在编程后不能重复编程和修改，因此不适于数字系统的研制、开发和实验阶段使用。可多次编程器件大多采用场效应管作开关元件，控制存储器存储编程信息，它们可采用EPROM、EEPROM、Flash或SRAM等工艺制造。

1. 熔丝编程和反熔丝编程技术

熔丝技术在早期的可编程逻辑器件中得到应用，使用熔丝技术的可编程逻辑器件属于一次性可编程器件，编程后不能再重复编程和修改。

在可编程逻辑器件内部互连节点上设有相应的熔丝，当可编程逻辑器件开发软件形成编程数据文件后，设计者可利用编程器将该编程数据写入PLD中。在编程操作前，器件所有连接节点的熔丝未断开。编程操作时，对不需要连接的节点加远大于正常工作电流的编程电流，使该节点熔丝断开，而对需要连接的节点（映象图用×符号表示），熔丝保留。编程操作后，可编程逻辑器件内部的熔丝图就等效于要完成的逻辑电路，即具有设计所要求的逻辑功能。

在可编程逻辑器件内部，熔丝编程烧断后，该连接节点将永久开路，因此这类器件只能一次编程，不能重复修改。为了保证熔丝熔化时产生的金属物不影响器件其他部分，熔丝还需要留出较大的保护空间，因此熔丝占用的芯片面积较大。为了克服熔丝编程的缺点，可利用反熔丝编程技术。反熔丝编程是通过击穿介质达到连通线路的目的。

2. 浮栅编程技术

浮栅编程技术主要包括紫外线擦除、电编程的EPROM以及电擦除、电编程的EEPROM和快闪存储器（Flash）。这几种结构的可编程逻辑器件都是采用悬浮栅存储电荷的方法来保存编程数据，因此在断电时，存储的数据不会丢失，编程数据可长期保存或多次修改。

3. 在系统编程技术

采用在系统编程ISP（In System Programmable）技术，由于这类器件内含有产生编程电压的电源泵及编程控制电路，因而不需要外配编程器，可直接对印制电路板上的在系统可编程逻辑器件进行编程。

4. JTAG编程技术

联合测试行动小组JTAG（Joint Test Action Group）在20世纪80年代中期制定了边界扫描测试BST（Boundary Scan Test）技术，1990年被修改后成为IEEE的一个标准（IEEE1149.1—1990），即JTAG标准。边界扫描测试技术最初是为了实现高密度电路板级和芯片级测试，后又利用对支持JTAG接口的可编程逻辑器件进行在线编程。

5. 在线可重配置技术

这类器件利用静态随机存储器SRAM存储信息，不需要在编程器上编程，可直接在印制电路板上对器件编程。通常编程信息存于外附加的EPROM、EEPROM或系统的硬盘上，在系统工作之前，先将存于器件外的编程信息输入到器件内的SRAM，在系统上电时，这些编程数据立即写入到可编程逻辑器件中，从而实现对可编程逻辑器件的动态配置。

1.2 PAL 和 GAL 器件

根据可编程**与**阵列和**或**阵列的结构及存储单元密度，可编程逻辑器件分为高密度可编程逻辑器件和低密度可编程逻辑器件。低密度可编程逻辑器件根据历史发展，又可分为 PAL 和 GAL 两种类型。

1.2.1 PAL 器件的基本结构

可编程阵列逻辑 PAL(Programmable Array Logic)器件是 20 世纪 70 年代末期出现的一种低密度、一次性可编程逻辑器件。它是第一个具有典型实用意义的可编程逻辑器件。

在 20 世纪 70 年代，由于集成电路制造工艺及计算机处理数据速度的限制，与前述图 1-1-1 所示的 PLD 结构相比，PAL 器件采用的是可编程的**与**阵列、固定的**或**阵列和多种输出反馈单元结构，如图 1-2-1 所示。由于它逻辑规模小，适用范围小，目前已很少使用。

图 1-2-1 PAL 器件基本结构

1.2.2 GAL 器件的基本结构

通用阵列逻辑 GAL(Generic Array Logic)器件是继 PAL 器件之后，在 20 世纪 80 年代中期推出的一种低密度可编程逻辑器件。它在结构上采用了“**与**-输出逻辑宏单元 OLMC(Output Logic Macro Cell)”结构形式。在工艺上吸收了 EEPROM(Electrically Erasable PROM)的浮栅技术，从而使 GAL 器件具有可擦除、可重新编程、数据可长期保存和可重新组合结构的特点。因此，GAL 器件比 PAL 器件功能更加全面，结构更加灵活，它可取代大部分中小规模的数字集成电路和 PAL 器件，增加了数字系统设计的灵活性。

GAL 器件基本结构如图 1-2-2 所示。在电路结构上，器件继承了 PAL 器件**与**阵列可编程和**或**阵列固定的基本结构，但在输出电路中全部采用了可编程的输出逻辑宏单元 OLMC。

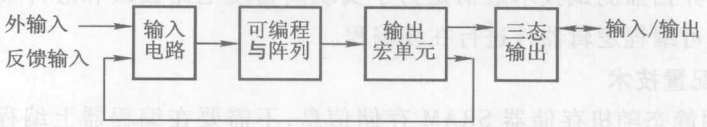

图 1-2-2 GAL 器件基本结构

GAL 器件型号不同，可编程的输出宏单元也不尽相同。典型的 GAL16V8 可编程输出宏单

元 OLMC 的内部结构如图 1-2-3 所示。每个 OLMC 包含或阵列中的一个或门、一个可编程异或门、一个 D 触发器和四个可编程多路开关。

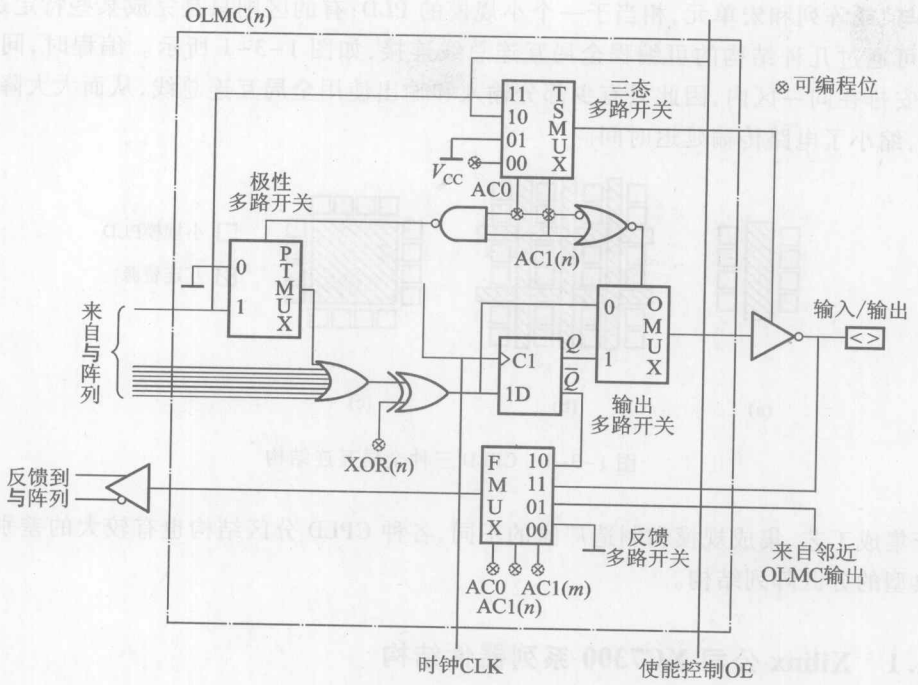

图 1-2-3　OLMC 内部结构

采用 GAL 器件，可以使系统设计方便灵活，系统体积缩小，可靠性和保密性提高，还可以提高系统速度并降低功耗。但 GAL 和 PAL 一样，都属于简单可编程逻辑器件，它们的共同缺点是逻辑阵列规模小，每个器件仅相当于几十个等效逻辑门，不适用于较复杂的逻辑电路的设计，并且也不能完全杜绝编程数据的非法抄袭。

GAL 器件的这些不足之处，在复杂可编程器件 CPLD 和 FPGA 中得到了较好的解决。

1.3　CPLD 基本结构

随着集成工艺的发展，PLD 的集成规模越来越大，当前 PLD 的集成规模已从低密度的 PAL 和 GAL 器件发展到万门以上的复杂可编程逻辑器件 CPLD(Complex Programmable Logic Device)系列。与 GAL 器件一样，CPLD 也是可进行多次编程改写、多次擦除的逻辑器件。CPLD 采用 CMOS EPROM、EEPROM、闪速存储器和 SRAM 等编程技术，从而构成了高密度、高速度和低功耗的逻辑可编程器件。

随着 PLD 集成规模的增大，器件的 I/O 引脚数目和 D 触发器数目大大增加，如果仍采用一个与阵列，则输入或反馈到与门的输入端数和与阵列的规模必然急剧增大。但实际设计时，每个与门所需的输入端数常常不是很多，因而器件的与门利用率较低。此外，当阵列大到一定程度

时，将使电路传输延迟增加，工作频率降低。为了克服上述缺点，集成规模较大的 CPLD 大多采用各种分区的阵列结构，即将整个器件分成若干个区。有的区包含有若干的 I/O 端、输入端及规模较小的**与**、**或**阵列和宏单元，相当于一个小规模的 PLD；有的区则只是完成某些特定逻辑功能。各区之间可通过几种结构的可编程全局互连总线连接，如图 1-3-1 所示。编程时，同一模块的电路一般安排在同一区内，因此只有少部分输入和输出使用全局互连总线，从而大大降低了逻辑阵列规模，缩小了电路传输延迟时间。

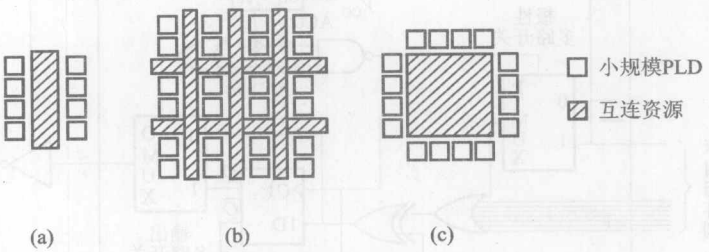

图 1-3-1　CPLD 三种全局互连结构

由于集成工艺、集成规模和制造厂家的不同，各种 CPLD 分区结构也有较大的差别。下面介绍几种典型的分区阵列结构。

1.3.1　Xilinx 公司 XC7300 系列器件结构

Xilinx 公司 XC7300 系列器件采用了 0.8μm CMOS EPROM 工艺，其结构由多个可编程功能模块组成，如图 1-3-2 所示。在这个结构中，包含了两种不同类型的功能模块，即快速功能模块 FFB(Fast Function Blocks)和高集成度功能模块 FB(Function Blocks)。这两个模块以及 I/O 模块通过一个通用互连矩阵 UIM(Universal Interconnect Matrix)连接。

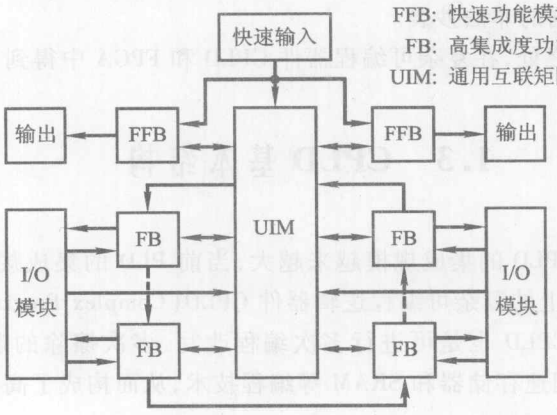

图 1-3-2　XC7300 系列结构

快速功能模块 FFB 的输入可以从 UIM、快速专用输入引脚信号中选择，它提供了引脚到引