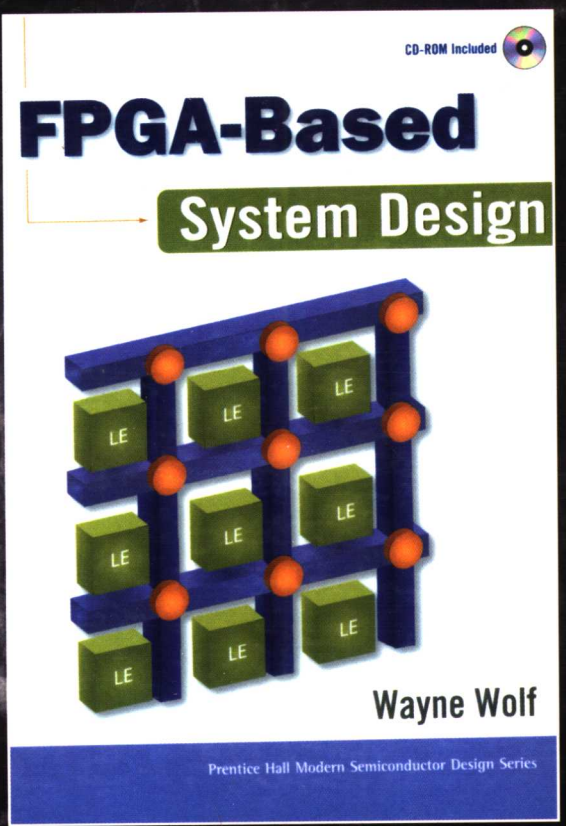
计 算 机 科 学 丛 书

基于FPGA的系统设计

(美) Wayne Wolf 普林斯顿大学 著 闫敬文 等译



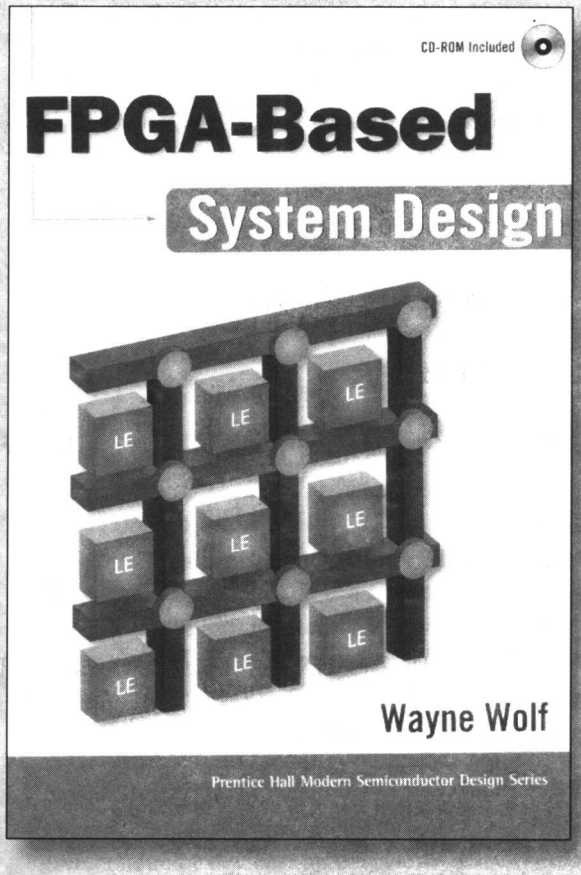
FPGA-Based System Design

机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

基于FPGA的系统设计

(美) Wayne Wolf 普林斯顿大学 著　　闫敬文 等译



FPGA-Based System Design

机械工业出版社
China Machine Press

本书以 VLSI 设计方法系统讲解 FPGA（现场可编程门阵列）系统设计方面的基本原理，帮助解决应用 FPGA 进行大系统设计时所遇到的问题。主要内容包括：VLSI 的要点（如 VLSI 制造工艺、电路设计、连线、组合逻辑、时序机以及系统结构方面的内容）、VLSI 的现代设计方法和 FPGA 最有价值的特性等。本书提供了丰富的实例，读者可结合实例和附赠光盘中的 FPGA 设计工具 XSE 来学习和实现基于 FPGA 的系统设计。

本书注重基础，强调应用，提供了大量的图表、例题和习题，可作为高等院校的电子、电气、计算机、信息、控制类专业高年级本科生和研究生教材，也可供相关工程技术人员参考。

Simplified Chinese edition copyright © 2006 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *FPGA-Based System Design* (ISBN 0-13-142461-0) by Wayne Wolf, Copyright © 2004.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice Hall Professional Technical Reference.

本书封面贴有 Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

版权所有，侵权必究。

本书法律顾问　北京市展达律师事务所

本书版权登记号：图字：01-2004-4412

图书在版编目（CIP）数据

基于 FPGA 的系统设计/（美）沃尔夫（Wolf, W.）著；闫敬文等译. -北京：机械工业出版社，2006.5

（计算机科学丛书）

书名原文：FPGA-Based System Design

ISBN 7-111-18707-5

Ⅰ.基… Ⅱ.①沃… ②闫… Ⅲ.可编程序逻辑器件-系统设计 Ⅳ.TP332.1

中国版本图书馆 CIP 数据核字（2006）第 022863 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：许　萍　姜淑欣

北京诚信伟业印刷有限公司印刷 • 新华书店北京发行所发行

2006 年 5 月第 1 版第 1 次印刷

184mm×260mm • 17.75 印张

定价：36.00 元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

FPGA 以其功能强大、开发过程投资少、周期短、可反复修改、保密性能好和开发工具智能化等特点成为当今可编程逻辑器件设计领域首选的器件之一。随着 FPGA 设计技术的发展，FPGA 在硬件设计中的作用越来越明显。目前全国有数百万的硬件工程师在设计中运用着各种型号的 FPGA。FPGA 设计技术已经成为当今硬件工程师和 IC 工程师的必备技能，越来越多的大学生和硬件工程师急需学习和掌握这门技术。

设计人员只有在真正理解 VLSI 的基础上，才能更好地利用 FPGA 进行大系统设计。目前，结合 VLSI 系统优点等知识系统介绍现代 VLSI 设计的书籍非常少，而本书就是以 VLSI 设计方法系统地讲解 FPGA 系统设计方面的基本原理，举例介绍了如何应对用 FPGA 进行大系统设计时所遇到的问题。因此，我们翻译了本书，旨在推广基于 FPGA 的系统设计和应用方面的技术，把国外优秀的 FPGA 设计思想介绍给国内的读者。

本书以 VLSI 设计方法系统讲解 FPGA 系统设计方面的基本原理，解决如何应用 FPGA 进行大系统设计所遇到的问题：首先介绍了 VLSI 的制造工艺、电路设计、连线、组合逻辑、时序机以及系统结构等方面的内容，其次介绍了 VLSI 现代设计方法和 FPGA 最有价值的特性。本书实例丰富、浅显易懂，有助于读者理解全书的精髓。

特别需要指出的是，本书以“系统设计”思想进行 FPGA 设计，而不是以面向具体应用进行 FPGA 设计，也就是说，不针对某一种规模某一具体 FPGA 芯片进行设计。这种定位有利于帮助读者建立 FPGA 系统设计的概念，帮助读者利用 FPGA 最佳性能设计出大系统。建议读者在阅读本书时，结合基础知识学习完整设计实例，包括在 FPGA 系统上验证和运行书中实例系统的源代码。

在翻译过程中，我们遇到了很多问题。如不同作者在表达同一个意思时使用的术语不同，如第 3 章有三个词：fabrics（可译为：结构、结构体系、体系结构、构造）、architecture（可译为：结构）和 structure（可译为：结构）。按照中文意思如何区分这三个词的具体含义，是翻译这类术语时的难题之一。最后经过认真地比较其他教材中的术语，再结合汉语中的建筑结构的思想进行了处理。“fabrics”译为“层构”，就是指制造集成电路时制造的不同层；“architecture”译为“结构体”或“整体结构”，是指 FPGA 本身的整体结构；“structure”译为“结构”，指局部的一般结构的意思。这样这三个词就区分开了。类似这样的问题还有很多。我们处理的方式是结合汉语习惯并结合目前我们所掌握的电子技术知识来进行选择。

但为了进一步学习，只学习理论知识、进行基本实践还是不够的，还需要进行深入的研究和大量的实际设计，只有这样才能培养出真正的 FPGA 设计经验。为此我们今后还要组织队伍加强这方面的研究工作，将教学中的实际例子和研究生在研究过程中的项目设计放到网页上（http://isip.xmu.edu.cn.），供读者免费下载、学习和测试。

除译者本人外，参加本书的翻译人员有赖耀林、林媛、刘晓玲、陈嘉臻和余见等。但因水平有限，译文中必然存在错误和不足之处。在阅读过程中，若发现不当之处或是错误译文，欢迎批评指正！本人的 E-mail 地址：yjwen@xmu.edu.cn，xdyjwen@tom.com（永久）。

闫敬文

前　言

本书是一次尝试。在完成《*Modern VLSI Design*》第三版后不久，我开始意识到越来越多过去建立在传统硅片上的数字设计现在可以在现场可编程门阵列（field programmable gate array，FPGA）上实现。传统的 VLSI 系统设计在一段时间内不会很快消失，但越来越多的设计者将致力于 FPGA 的设计，而且他们中的很多人从来没有设计过定制的芯片。

然而，基于 FPGA 的大系统设计者只有在真正理解 VLSI 的基础上，才能更好地利用 FPGA。事实上，很多系统设计者把 FPGA 简单地当作一个黑匣子处理，结果失去了很多适合于 FPGA 内优化设计的机会。FPGA 的结构主要由 VLSI 约束所决定：例如逻辑单元结构、可编程的互连结构、互连网络、配置和输出引脚等。理解 VLSI 器件特性如何影响 FPGA 层构设计，可以帮助设计者更好地理解如何利用 FPGA 的最佳特性以及如何降低它的局限性所造成的影响。

例如，在 FPGA 互连网络中，多数现代 FPGA 结构给设计者提供了几种不同类型的连线：局部的、通用的和全局的。为什么会存在这些不同类型的连线？因为随着连线长度的增加，连线驱动变得很难，这就需要额外的电路来驱动长连线，从而使 FPGA 更昂贵。了解这些不同类型的连线是如何工作的，能帮助设计者确定一个特殊的逻辑连线是否需要一种较昂贵类型的连线。

今天的 FPGA 的确是很神奇。高端 FPGA 能拥有几百万个门。芯片上有几片 FPGA 和一片或多片 CPU，就能提供一个完整的嵌入式计算平台。不论系统是用 FPGA 还是用传统的硅片来设计，这些大系统的很多设计技术都是相同的。特别是当我们想要更好地利用 VLSI 结构中的硅特性时，更是如此。

考虑到 VLSI 系统中的这些优点，我决定以《*Modern VLSI Design*》为出发点，写一本基于 FPGA 的系统设计的书。《*Modern VLSI Design*》的读者将从本书看到以下熟悉的内容：VLSI 特性、电路、组合和时序逻辑设计，以及系统结构的知识。当然，本书也增加了一部分新内容，一些与 FPGA 有关的内容，另外还有一些是对数字系统设计的新展望。

编写本书的一个主要目的是为那些对 VLSI 感兴趣的设计者和只想简单使用 FPGA 来设计大型系统的人提供教材。本书第 2 章主要是对 VLSI 的回顾：包括它的构造、电路、互连特性等。本书的其他部分，分别介绍 VLSI 设计细节，对 VLSI 不感兴趣的读者可以跳过这些内容；而那些把 FPGA 当作 VLSI 器件并想深入了解 FPGA 设计的读者可以在闲暇时间学习这部分内容。

第 3 章是对 FPGA 的基本的可编程结构的一个概述。因为 FPGA 制造商提供的产品资料一直在变，因此本章并不是一本数据手册。它的目的是介绍 FPGA 的一些基本概念，比较在可编程逻辑上解决基本问题的不同方法。至于如何利用 FPGA 结构将是本书剩余部分要讨论的内容。

第 4 章和第 5 章分别详细介绍了组合和时序逻辑设计。为了符合大多数设计的主要目标（如尺寸、速度和功耗），这两章分别描述了数字逻辑的规范和优化方法，另外还介绍了

试读结束，需要全本PDF请购买 www.ertongbook.com

Verilog 和 VHDL 两种语言。但是本书并不是任何一种语言的权威参考书。硬件描述语言是当今数字系统设计的一种折中选择。对这些硬件描述语言及其基本原理的理解是成功设计数字系统的基本要求。为了理解如何最佳利用逻辑和物理综合，我们同时也需要了解用于最佳逻辑和时序机设计的工具。

第 6 章介绍了大型数字系统的结构。介绍了存储-转移设计，把它作为一种数字设计结构的方法论。在该章中，将一个简单的 DSP 系统当作一个设计例子。但并不是把该 DSP 系统当成 CPU 设计的一个经典范例，而是想通过这个简单的例子，看到很多不同的设计问题。

第 7 章通过研究一个建立在 FPGA 上的大规模系统来总结回顾本书中所学的知识，FPGA 平台包含 CPU 和 FPGA 层构，它允许设计者在单个芯片上综合硬件和软件来解决设计难题。多 FPGA 系统能够实现超大型的设计；单个多 FPGA 系统能通过编程实现许多不同的逻辑设计。

ASIC（专用集成电路）设计的未来发展会怎样？我并不认为它会消失。因为人们仍然需要高密度和高性能的芯片，而这些也只有传统的硅片才能提供。但是我还是认为 FPGA 将会成为实现数字系统的一种主要方式。

Xilinx 公司已经允许我们制作包含 Xilinx 学习设计工具（XSE）的光盘。本书中的例子都在这些工具上调试过，读者能跟着这些例子熟悉该工具的使用，同时也可以使用该工具来设计自己的例子。这些工具能使我们更容易地理解概念。非常感激 Xilinx 公司为本书提供的帮助。

在下面的网页上能找到一些和本书相关的材料：

http://www.ee.princeton.edu/~wolf/fpga-book

该网页包含各个章节的概述，提供了额外的资源、一些例子实验和勘误表。我希望读者喜欢这本书，需要时可以给我发邮件。我的邮箱地址是 wolf@princeton.edu。

我非常感谢 2003 年春季 ELE 462 班的同学，他们在 VLSI 课程上非常有耐心地完成我的试验。我也非常感谢 Jiang Xu 和 Li Shang，他们是我那个学期的助教。他们为 FPGA 设计改进了基础结构，并且帮我调试了 DSP 设计。Mike Butts 和 Mohammed Khalid 在分割算法上给了我许多有用的建议。Steven Brown、Jonathan Rose、Zvonko Vranesic 和 William Yu 将他们论文中的图形直接提供给本书，并且为读者提供了原始资料的数据。他们使得我的工作简化了很多。我非常感激 Andre De Hon、Carl Ebeling、Yankin Tanurhan 和 Steve Trimberger，他们审阅评论了我整个草稿，同时在如何改进这本书上给了我很多精彩的建议。我非常感激 Ivo Bolsens、Anna Acevedo 和 Jeff Weintraub，他们使我从总体上掌握了 Xilinx 知识，特别是允许我在这本书中包含 Xilinx ISE 磁盘。当然我得感谢我的编辑 Bernard Goodwin，感谢他为了本书不倦地努力。在本书中出现的问题，无论大小，当然全都由我负责。

Wayne Wolf
Princeton，New Jersey

目　录

译者序

前言

第 1 章　基于 FPGA 的系统 …… 1

1.1　概述 …… 1

1.2　基本概念 …… 1

1.2.1　布尔代数 …… 1

1.2.2　原理图与逻辑符号 …… 3

1.3　数字设计和 FPGA …… 4

1.3.1　FPGA 的作用 …… 4

1.3.2　FPGA 的类型 …… 5

1.3.3　FPGA 与定制 VLSI 的比较 …… 6

1.4　基于 FPGA 的系统设计 …… 7

1.4.1　目标和方法 …… 7

1.4.2　分级设计 …… 8

1.4.3　设计抽象 …… 9

1.4.4　方法学 …… 11

1.5　小结 …… 12

1.6　习题 …… 12

第 2 章　VLSI 技术 …… 13

2.1　概述 …… 13

2.2　制造工艺 …… 13

2.3　半导体特性 …… 15

2.4　CMOS 逻辑门 …… 20

2.4.1　静态互补门 …… 20

2.4.2　门延迟 …… 22

2.4.3　功耗 …… 27

2.4.4　驱动大负载 …… 28

2.4.5　低功耗门 …… 29

2.4.6　开关逻辑 …… 32

2.5　连线 …… 34

2.5.1　连线的结构 …… 35

2.5.2　连线的寄生现象 …… 35

2.5.3　金属线模型 …… 38

2.5.4　通过 *RC* 传输线的延迟 …… 38

2.5.5　*RC* 传输线中的缓冲器插入 …… 40

2.5.6　*RC* 线间的串扰 …… 41

2.6　寄存器和随机存储器 …… 43

2.6.1　寄存器的结构 …… 43

2.6.2　随机存储器 …… 44

2.7　封装与焊盘 …… 49

2.7.1　封装 …… 49

2.7.2　焊盘 …… 51

2.8　小结 …… 52

2.9　习题 …… 53

第 3 章　FPGA 层构 …… 55

3.1　概述 …… 55

3.2　FPGA 的体系结构 …… 55

3.3　基于 SRAM 的 FPGA …… 57

3.3.1　概述 …… 57

3.3.2　逻辑器件 …… 58

3.3.3　互连网络 …… 61

3.3.4　配置 …… 64

3.4　永久性编程的 FPGA …… 66

3.4.1　反熔丝 …… 66

3.4.2　Flash 配置 …… 67

3.4.3　逻辑模块 …… 67

3.4.4　互连网络 …… 70

3.4.5　编程 …… 71

3.5　芯片的 I/O …… 71

3.6　FPGA 层构的电路设计 …… 73

3.6.1　逻辑器件 …… 73

3.6.2　互连 …… 77

3.7　FPGA 的层构体系 …… 80

3.7.1　逻辑器件参数 …… 81

3.7.2　互连结构 …… 82

3.7.3　引脚输出 …… 83

3.8　小结 …… 84

3.9 习题 …… 84
第4章 组合逻辑 …… 86
4.1 概述 …… 86
4.2 逻辑设计过程 …… 86
4.3 硬件描述语言 …… 104
4.3.1 用 HDL(硬件描述语言)进行建模 …… 104
4.3.2 Verilog …… 108
4.3.3 VHDL …… 110
4.4 组合网络延迟 …… 113
4.4.1 延迟描述 …… 113
4.4.2 门延迟和连线延迟 …… 114
4.4.3 扇出 …… 115
4.4.4 路径延迟 …… 116
4.4.5 延迟和物理设计 …… 118
4.5 功率和能量优化 …… 121
4.6 算术逻辑 …… 122
4.6.1 数字描述 …… 122
4.6.2 组合移位器 …… 122
4.6.3 加法器 …… 123
4.6.4 ALU …… 129
4.6.5 乘法器 …… 131
4.7 FPGA 的逻辑实现 …… 136
4.7.1 句法引导翻译 …… 136
4.7.2 用宏来实现逻辑 …… 137
4.7.3 逻辑综合 …… 137
4.7.4 工艺无关逻辑优化 …… 138
4.7.5 工艺相关逻辑优化 …… 142
4.7.6 FPGA 的逻辑综合 …… 143
4.8 FPGA 的物理设计 …… 143
4.8.1 布局 …… 144
4.8.2 布线 …… 147
4.9 再次考察逻辑设计过程 …… 149
4.10 小结 …… 161
4.11 习题 …… 161
第5章 时序机 …… 164
5.1 概述 …… 164
5.2 时序机设计过程 …… 164
5.3 时序设计格式 …… 165
5.3.1 状态转换和寄存器传输级模型 …… 165
5.3.2 有限状态机理论 …… 169
5.3.3 状态赋值 …… 171
5.3.4 Verilog 建模风格 …… 174
5.4 时序法则 …… 179
5.4.1 触发器和锁存器 …… 179
5.4.2 时序规则 …… 180
5.5 性能分析 …… 184
5.5.1 基于触发器系统的性能 …… 185
5.5.2 基于锁存器系统的性能 …… 187
5.5.3 时钟偏差 …… 189
5.5.4 调整 …… 194
5.6 功率优化 …… 194
5.7 小结 …… 195
5.8 习题 …… 195
第6章 整体结构 …… 197
6.1 概述 …… 197
6.2 行为级设计 …… 197
6.2.1 数据路径控制器结构 …… 197
6.2.2 时间调度和分配 …… 198
6.2.3 功率 …… 213
6.2.4 流水线技术 …… 214
6.3 设计方法学 …… 220
6.3.1 设计过程 …… 220
6.3.2 设计标准 …… 221
6.3.3 设计验证 …… 223
6.4 设计举例 …… 224
6.5 小结 …… 229
6.6 习题 …… 229
第7章 大规模系统 …… 232
7.1 概述 …… 232
7.2 总线 …… 232
7.2.1 协议和规范 …… 232
7.2.2 总线的逻辑设计 …… 235
7.2.3 微处理器和系统总线 …… 239
7.3 FPGA 平台 …… 242
7.3.1 FPGA 平台的整体结构 …… 242
7.3.2 串行 I/O …… 246

7.3.3 存储器 …… 247
7.3.4 CPU 和嵌入式乘法器 …… 248
7.4 多 FPGA 系统 …… 250
7.4.1 多 FPGA 系统中的约束 …… 251
7.4.2 多 FPGA 之间的连线 …… 251
7.4.3 多 FPGA 分割 …… 253
7.5 新型结构 …… 254
7.5.1 用 FPGA 搭建的机器 …… 254
7.5.2 可替代的 FPGA 层构 …… 255
7.6 小结 …… 256
7.7 习题 …… 256
附录 A 术语表 …… 257
附录 B 硬件描述语言 …… 263
参考文献 …… 270

第 1 章　基于 FPGA 的系统

1.1　概述

本章是为本书的其他部分打基础的。1.2 节介绍一些布尔代数和原理图的基本概念。1.3 节介绍 FPGA 并说明其重要性。1.4 节介绍怎样用 FPGA 设计复杂的数字系统。

1.2　基本概念

本节将介绍逻辑设计中的一些基本概念。在这部分，将介绍一些术语，这些术语将贯穿本书的其他部分。

1.2.1　布尔代数

我们用布尔代数来表示数字电路的逻辑函数。香农［Sha38］指出开关网络（比如电灯开关）可以用布尔函数建模。虽然今天的逻辑门电路通常不是用开关来建立的，但是仍然认为布尔代数是其基本表达形式。我们用布尔代数描述**组合逻辑**函数，用布尔函数描述输入组合；不使用包含存在（$\exists x f(x)$）量词或全称（$\forall x g(x)$）量词的函数。

在逻辑表达中使用一些很标准的符号：如果 a 和 b 是变量，那么 a'（或 $\bar{a}$）是变量 a 的反，$a \cdot b$（或 ab）是变量 a、b 的**与**（AND），而 $a+b$ 是变量 a、b 的**或**（OR）。另外，**与非**（NAND）函数 $(ab)'$ 用符号 |○ 表示，或非（NOR）函数 $(a+b)'$ 用 a NOR b 表示，而异或（a XOR $b=ab'+a'b$）函数使用符号⊕（学过代数的读者应该知道 XOR 和 AND 形成一个环）。图 1-1 总结了常见逻辑函数的名称和符号。

名称	符号
NOT	‘、~
AND	、∧、&
NAND	|
OR	+、∨
NOR	NOR
XOR	⊕
XNOR	XNOR

图 1-1　逻辑表达式中的函数符号

我们用字母 a 和 a' 分别表示变量的原码与反码形式。为了理解逻辑表达式和逻辑门之间的关系，可以考虑这个问题的最简单模型，然后转换结果。

现在复习一些用于变换表达式的基本代数规则。有些规则和算术规则类似，而有些规则是布尔代数特有的：

- **幂等律**　$a \cdot a=a$，$a+a=a$。
- **反演律**　$a+a'=1$，$a \cdot a'=0$。
- **同一律**　$a+0=a$，$a \cdot 1=a$。
- **交换律**　$a+b=b+a$，$a \cdot b=b \cdot a$。
- **零元律**　$a \cdot 0=0$，$a+1=1$。
- **对合律**　$(a')'=a$。

○ 是一个点和经过点的虚线。C 语言中把它当作 OR 使用。

- **吸收律** $a+ab=a$。
- **结合律** $a+(b+c)=(a+b)+c$，$a\cdot(b\cdot c)=(a\cdot b)\cdot c$。
- **分配律** $a\cdot(b+c)=ab+ac$，$a+bc=(a+b)(a+c)$。
- **德·摩根律** $(a+b)'=a'\cdot b'$，$(a\cdot b)'=a'+b'$。

虽然有时布尔代数可能显得很抽象，但是它提供的一些数学结果却直接关系到逻辑电路的物理特性。布尔代数中的**完备性**和**无冗余性**问题对逻辑设计来说非常重要。

如果使用一组逻辑函数集可以产生所有可能的布尔表达式，那么就说这个逻辑函数集是**完备**的——也就是说，任意由+、·和′组合而成的所有可能函数，都存在一个由将要测试的函数写成的等价式。通常通过归纳测试一组函数是否可以产生所有逻辑公式来检测这组函数是否完备。由下面最基本的公式，很容易证明与非（NAND）函数是完备的：

- 1：$a\mid(a\mid a)=a\mid a'=1$。
- 0：$\{a\mid(a\mid a)\}\mid\{a\mid(a\mid a)\}=1\mid 1=0$。
- a'：$a\mid a=a'$。
- ab：$(a\mid b)\mid(a\mid b)=ab$。
- $a+b$：$(a\mid a)\mid(b\mid b)=a'\mid b'=a+b$。

由这些基本公式可以产生所有可能的逻辑公式。所以{|}函数集可以产生任意的逻辑函数。同样地，任意一个公式也可以单独地用或非（NOR）式表达。

然而，与函数AND和或函数OR的组合并不是完备的。这很容易证明：因为无法从AND和OR组合中直接产生1或0。如果NOT也加入这个函数集，则可以产生所有的公式，如$a+a'=1$等。事实上，{′,·}和{′,+}都是完备集。

用来实现逻辑函数的任何电路技术都必须能够实现一个完整的函数集。静态电路、补偿电路通常实现NAND或NOR函数，但有些电路系列并不能实现一个完整的函数集。不完备的逻辑函数集给逻辑设计者增加了额外负担，因为设计者必须确保用正确的形式指定了逻辑函数。

在不改变真值表的情况下，没有可以从表达式中删除的字母，那么这个逻辑表达式就是**无冗余**的。比如，$ab+ab'$是冗余的，因为它可以简化为a。无冗余式及其关联的逻辑网络有一些重要特性：无冗余式比其逻辑等价的冗余式要简单；对于特定的制造缺陷，保证逻辑网络具有可测性。然而，无冗余式并不是万能的。无冗余式不像**最小项**一个最小项表达式可以有很多无冗余的表达形式，有些简单，有些复杂，所以找到一种无冗余的表达式并不能保证该设计是最简单的。无冗余式经常引入附加延迟，如果没有进行逻辑网络的冗余，这可能很难消除。在设计门电路网络前，简化逻辑表达式对面积和延迟都很重要。某些明显的简化可以通过手工完成；用CAD工具可以实现复杂的表达式的较难简化。

我们常谈论一个函数的**正逻辑**（on-set）和**负逻辑**（off-set）。正逻辑是指可以令函数为真的输入值集；负逻辑是指可以令函数为假的输入值集。一般都用正逻辑表示函数，但是使用负逻辑也是可以的（只要清楚使用的是哪一种方式）。

在组合逻辑中通常利用**无关项**。有两种无关项：**输入无关项**和**输出无关项**。尽管它们的名字类似，但它们的作用完全不同。

输入无关项是一种速记符号。观察图1-2的函数，在这个函数中，输入“$a=0$，$b=0$”和“$a=0$，$b=1$”有相同的结果。当写真值表时，可以划掉这两行，用一个带有无关项符号

试读结束，需要全本PDF请购买 www.ertongbook.com

"-"的单个函数表示。可以把 $a'b'+a'b$ 简化成 a'，把 b 作为输入无关项（这个完整的逻辑函数是 $f=a'+ab$，可以通过画卡诺图来验证）。

输出无关项是一种非完备的描述逻辑函数的方法，是不被使用的输入组合。观察图 1-3 的函数。在这里，输入组合 $a=0$，$b=1$ 的函数值是用无关项符号"-"表示（在输入无关项中也是使用同样的符号）。这意味着对这个输入组合，并不关心输出值为 0 或 1。在门电路中实现这个函数时，可以选择我们最容易实现的值。在图中，可以看到，将函数输出值选为 1 可以简化表达式。

a	b	f
0	0	1
0	1	1
1	0	0
1	1	1

完全指定

a	b	f
0	-	1
1	0	0
1	1	1

有无关项

图 1-2 输入无关项的例子

a	b	f
0	0	1
0	1	-
1	0	0
1	1	1

图 1-3 输出无关项的例子

通常把布尔表达式用**因式**形式表示。比如表达式 $f=a'+ab'$ 可以写成一个单一函数。如果定义一个新的函数 $g=ab'$，可以把 f 改写为 $f=a'+g$。这个变换本身并没有什么意义，但如果 ab' 项也用于其他地方，那么 g 函数就变成了一个通用代入式。与之相对的因式求逆：在 f 中把 g 反代回去，f 可重写为 $f=a'+ab'$。

1.2.2 原理图与逻辑符号

在使用电路元器件符号前，先简要地复习一下它们。图 1-4 是一些电路元器件的符号：n 型和 p 型晶体管，电容、电阻和电源结点（正电源 V_{DD} 和负电源 V_{SS}）。

图 1-5 是一些逻辑门符号：与非门（NAND）、或非门（NOR）等。图 1-6 是两种常用的寄存器传输器件：多路选择器（mux）和算术逻辑单元（ALU）。

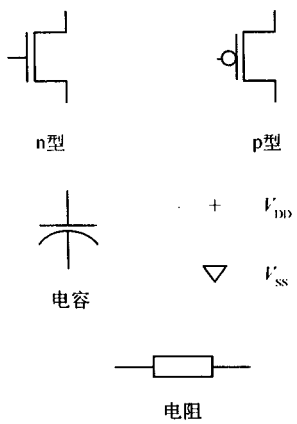


图 1-4 电路元器件的图表符号

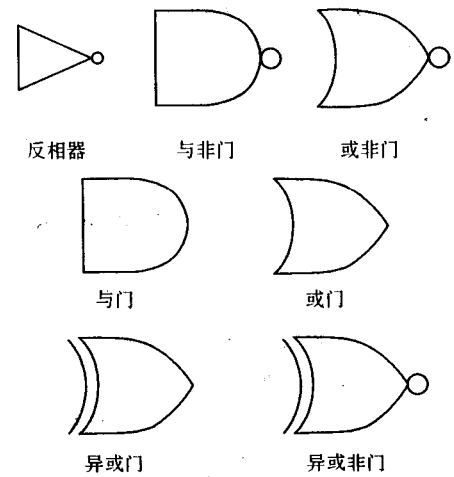


图 1-5 逻辑门的图表符号

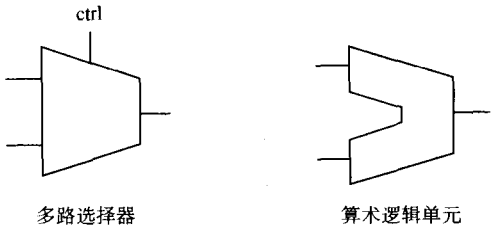


图 1-6 寄存器传输器件的图表符号

1.3 数字设计和 FPGA

1.3.1 FPGA 的作用

现场可编程门阵列（Field-programmable gate array，FPGA）填补了数字系统设计的空白，是对微处理器的补充。尽管微处理器能用于许多场合，但是它们依靠软件才能实现其功能，因此比起定制芯片，它们一般运行速度比较慢而且功耗大。同样地，FPGA 也不是定制芯片，因此，它们无法像那些为某一应用而设计的定制芯片那样擅长完成特定功能。FPGA 一般也比定制逻辑芯片的运行速度慢而且功耗大，同时相对较贵；所以人们认为定制芯片更便宜。

然而，由于它们是标准器件，因而能够弥补定制芯片的一些不足。

- 从完成设计到取得一个可工作的芯片之间不用等待，可以把程序写入 FPGA 并立即进行测试。
- FPGA 是一种出色的制作样机工具。当在最终设计中用到 FPGA 时，可以更简单、更容易地完成从样机到产品的飞跃。
- 同种类型的 FPGA 可以用于不同类型的设计中，以降低库存费用。

从 FPGA 研制成功到现在的这 20 年内，FPGA 应用领域发展迅速。从 20 世纪 70 年代以来，**可编程逻辑器件**（Programmable logic device，PLD）已占领了市场。这些器件使用两级逻辑结构实现可编程逻辑。第一级逻辑结构是“与门”阵列，一般是一次性写入的；而第二级逻辑结构是“或门”阵列，是可编程的。PLD 器件一般是通过反熔丝（antifuse）方式（也就是通过高压形成连接）进行编程的。

它们大多数时候用作**胶合逻辑**（glue logic）——即将系统的主要元器件连接在一起的逻辑。通常用于样机设计，因为它们是可编程的，并且可以在几分钟内嵌入电路板中。但是通常不用它们来做最后的产品。可编程逻辑器件在使用它的系统中通常并不是主要器件。随着数字系统越来越复杂，更高密度的可编程逻辑需求越来越多，PLD 器件的两级逻辑结构的局限性也越来越明显。

两级逻辑结构对相对较小的逻辑功能是非常有用的，但随着集成度的提高，两级逻辑结构的效率降低。FPGA 通过使用任意深度的多级结构提供可编程逻辑，使用可编程的逻辑单元和可编程的互连结构来建立多级逻辑功能。

一般认为是 Ross Freeman 研制了 FPGA。他的 FPGA [Fre89] 包括可编程逻辑器件和一个可编程的互连结构，通过 SRAM 而不是反熔丝方式编程。这样可以按照标准 VLSI 加

工流程生产 FPGA，节省资金并提供更多的加工选择。同时也能对电路中的 FPGA 进行重新编程；在 FLASH 闪存没有广泛使用前，这是一个特别吸引人的特性。

Xilinx 和 Altera 公司早期都销售基于 SRAM 的 FPGA。Actel 公司则研制了另一种反熔丝结构的 FPGA。这种结构无法进行现场重编程，在无需重新配置的情况下这是一种优点。Actel 公司的 FPGA 在连线通路上使用多取向的逻辑结构组织。

多年以来，FPGA 主要是胶合逻辑和样机设计的工具。今天，它们被用于各种各样的数字系统：

- 高速电信设备的组成部分；
- 家庭个人视频录像机（PVR）的视频加速器。

FPGA 已经成为数字系统实现的主流器件。

1.3.2 FPGA 的类型

迄今为止，我们一直没有对 FPGA 进行定义。一个好的定义一方面可以区分 FPGA 和较小的可编程器件（如 PLD），另一方面又可以区分 FPGA 和定制芯片。以下定义了 FPGA 的一些特征：

- **它们是标准器件。**它们不是为了某一个特定功能设计的，使用者可以用它们进行特定目的的编程。
- **它们实现多级逻辑。**FPGA 内部的逻辑模块可以连接成任意深度的网络。而 PLD 只使用两级的与非/或非函数来实现所有的逻辑。

由于 FPGA 实现多级逻辑，因此，它同时需要可编程逻辑模块和可编程互连结构。PLD 使用固定的连线，只改变加在连线上的逻辑函数。相反，FPGA 需要对逻辑模块进行编程并将它们连接到一起，以实现逻辑功能。逻辑和互联的结合被称为**层构**（fabric），因为它拥有规则的结构，可以先通过设计工具进行高效优化，再把指定的逻辑映射到 FPGA 中。

FPGA 的一个重要特征是它可被编程。如图 1-7 所示，FPGA 编程与微处理器编程有很大的区别。微处理器是一个可存储程序的计算机。计算机系统包含 CPU 和存储指令、数据的独立存贮器。FPGA 编程（也称为**特色**）交织在 FPGA 的逻辑结构中。FPGA 不取指令——FPGA 的编程过程直接实现了逻辑功能和互联。

在 FPGA 编程中采用了大量技术。一些

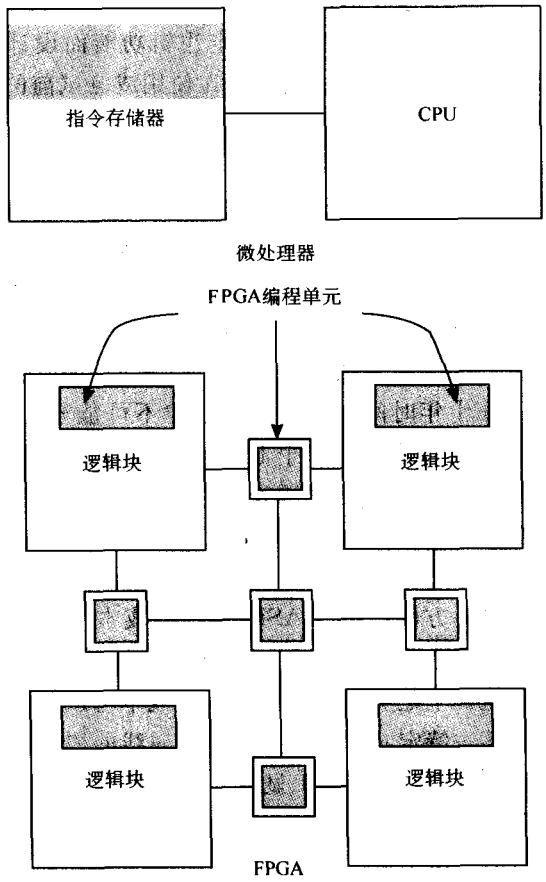


图 1-7 CPU 与 FPGA

FPGA是一次性编程；另一些则是可再编程。可再编程的FPGA器件也称为**可重配置**(reconfigurable)器件。可重配置的FPGA在样机制作中很受欢迎，因为它们在每次编程后还能再用。可重配置的系统在系统运行期间可以不停地编程。这样允许一块硬件实现几个不同的功能。当然，这些不同的功能无法同时实现，但当系统运行在不同的模式中时，可重配置能力是十分有用的。比如，径向计算机的显示操作有水平（景物）和垂直（人物）模式。当用户旋转显示，水银开关使完成显示功能的FPGA重新编程以显示新的模式。

一般，FPGA使用细粒度结构逻辑。在传统FPGA中用组合逻辑器件实现少数逻辑门电路和寄存器的功能。随着芯片的变大，出现了粗粒度结构的FPGA。这些芯片内的单独逻辑器件可以实现一个多位的ALU和寄存器。对某些类型的函数，粗粒度结构的FPGA能更有效地利用芯片的面积。

更新类型的FPGA包含FPGA层构以外的东西。**平台FPGA**包括几种不同类型的结构，所以大型系统的每个部分都可以用最适合的结构来高效地实现。典型的平台FPGA包含一个CPU，所以某些功能可以用软件方式运行。它还包括专用总线逻辑，这样，系统内可轻易地包含像PCI之类的总线接口。

1.3.3 FPGA与定制VLSI的比较

可以替换FPGA的主要器件是**专用集成电路**ASIC（application-specific IC）。与FPGA不同，ASIC是为了实现特定逻辑功能而设计的。ASIC的设计包括制作IC掩模版。ASIC必须在一条生产线上制成，在使用或测试前的生产过程就需要数月时间。ASIC和全定制设计明显不同：全定制设计有一个全定制版图，而ASIC对逻辑门使用预设计版图。现在，除微处理器外，大的数字芯片几乎都不包含大量的定制版图。

如前所述，由于ASIC是为了特定目的设计的，所以它具有某些明显的优点：一般比等同的FPGA运行得快，消耗的功耗低；在大量生产时，它们也很便宜。然而，有两种趋势使得许多系统设计者在定制逻辑的设计中选择FPGA而不是ASIC。

一方面，摩尔定律指出了集成电路产量的实质性增长规律。摩尔定律指出一块芯片上可生产的晶体管数目每18个月将翻一番。如图1-8所示，半导体工业已经将这个速度保持了几十年时间。即使摩尔定律不可能永远继续下去，但已经能够大量地生产具有几千万晶体管的芯片。单个芯片上可以集成这么多的晶体管，去掉晶体管，简化芯片上的设计逻辑任务就变得很诱人而且是越来越有必要了。对一个给定的功能，FPGA比ASIC使用更多的晶体管，可是FPGA可以在几天内设计完成，而普通的ASIC需要一年左右的设计周期。

另一方面，生产ASIC的掩模成本很高。一条IC生产线的基本技术性能指标是它可以生产的最小晶体管宽度。随着线宽的减小，生产成本也不断提高，一套现代化半导体生产设备价值几十亿美元。然而，生产成本可以分摊到生产线上生产的所有元器件上。相比之下，决定某一特定芯片的晶体管和金属线的掩模仅仅只能用于该芯片。掩模的成本必须完全加入芯片成本中。如表1-1显示，掩模成本正以指数级增长。随着VLSI线宽的减小，掩模成本将开始超过制造芯片的设计人员的成本。随着掩模成本的急剧上涨，标准器件变得更加具有吸引力。由于FPGA是适用于不同功能的可编程标准器件，因此可望FPGA在高密度芯片的IC市场上占据更大的份额。

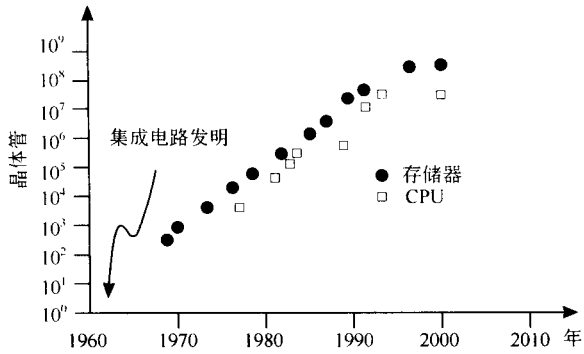


图 1-8 摩尔定律

表 1-1 掩模成本是晶体管线宽的函数

掩模成本	工艺
0.09μm	$1 000 000
0.18μm	$250 000
0.25μm	$120 000
0.35μm	$60 000

1.4 基于 FPGA 的系统设计

1.4.1 目标和方法

实现逻辑功能只是 FPGA 或任何数字系统设计必须达到的一个目标。为了使设计获得成功，还必须满足以下属性：

- **性能** 逻辑器件必须在要求的速度下工作。性能可以用以下几种方法来衡量，比如吞吐量和等待时间。时钟频率经常作为性能的一个衡量因素。
- **功率/能量** 芯片通常在一定的能量或功率预算下工作。在电池供电的系统中，能量消耗明显是很关键的。即使系统是用于电力网的，热耗散也会浪费金钱，因此必须加以控制。
- **设计时间** 不可能无限期地设计系统。由于 FPGA 是标准器件，因此它在设计时间上有一些优势。它们可以作为样机，可以快速编程，而且它们可以用作最终设计的一部分。
- **设计成本** 设计时间是设计成本的一个重要组成部分，但是还需要考虑其他的因素，比如对支持工具的投入等。FPGA 的开发工具一般比定制 VLSI 开发工具便宜。
- **生产成本** 生产成本是多次复制系统的成本。在通常情况下，FPGA 的编程费用比 ASIC 要高。然而，由于它们是标准器件，这有助于降低它们的生产成本。

为了解决以下问题，以致于设计变得特别艰难：

- **多级抽象** FPGA 设计要求通过多级细节来细化设计思想。从芯片必须做什么的设计规范开始，设计者必须创建一个可完成指定功能的结构，然后把结构扩展到逻辑设计。
- **多样且互相有矛盾的成本** 成本是用美元计算的，比如为完成软件某一特定部分而必须预留一些其他部分的花费。成本也可能用最终产品的 FPGA 性能或功耗来衡量。
- **较短的设计时间** 电子产品市场变化速度惊人。更快地研制出芯片意味着降低成本和增加收入；较慢推出产品则可能意味着根本不赚钱。

设计项目可能是伴随着大量多样的信息产生的。其中一些是早期设计的修正版；另外一些则是所发布标准的实现。在这些情况下，功能是规定好的，而且要实现的某些方面也可能是清楚的。其他项目并没有很好的基础，而且开始设计时，可能只有一些关于芯片任务的基

本概念。**需求**（requirement）这个术语是要求系统做什么的书面描述，例如“ADSL 基带调制解调器的功能”。**规范**（specification）这个术语是对功能的更正式的描述，比如一个 ADSL 基带功能的仿真器程序。性能、功耗和成本这样一些属性被称为**非功能需求**（non-functional requirement）。

1.4.2 分级设计

分级设计（Hierarchical design）是一种处理复杂数字系统设计的标准方法。被广泛应用于编程中：流程不是大量的原始语句的列表，而是较简化的程序。每个流程把任务分割成更小的操作，每一步都是简单的能直接编写的流程。这个技术一般称为**分治法**（divide-and-conquer）——程序的复杂度通过把它递归分割成可处理的片断加以克服。

芯片设计者采用把芯片分割为分级器件组的分治方法。如图 1-9 所示，全加器包含**主体**和许多**引脚**——该全加器有 5 个引脚：a、b、cin、cout 和 sum。如果把这个全加器作为一种**类型**的定义，我们可以设计很多这种类型的**实体**。重复使用元器件通常是非常有用的，比如可以用 n 个全加器组成 n-位加法器。每个元器件实体都有一个专有**名称**。由于同种类型的所有元器件具有相同的引脚，通过同时给定元器件实体名和引脚名可以指定在特定元器件上的引脚；通常，实体名和引脚名之间用逗号隔开。如果有两个全加器 add1 和 add2，可以把 add1. sum 和 add2. sum 作为不同的**引出端**（引出端是一元器件引脚对）。

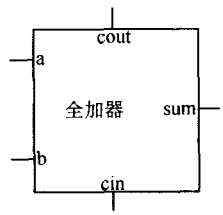


图 1-9 器件引脚

图 1-10 显示一个稍微复杂一点的器件，它包含了两个器件。图 1-10 上标明了每个器件的实体名和类型名，比如 large(bx) 中，large 是实体名，而 bx 是类型名。图 1-10 上也标明了每个器件的引脚名和引脚连接的网络名。因此，可以通过两种等价的方式列出电路的电气连接：**网表**和**器件列表**。网表提供与每个网络连接的引出端。以下是图 1-10 的顶层器件的网表：

```
net2: large.p1, b1.a;
net1: large.p2, b1.c;
cxnet: b1.b, sand.i1;
o1net: large.p3, sand.o1;
o2net: sand.o2, large.p4.
```

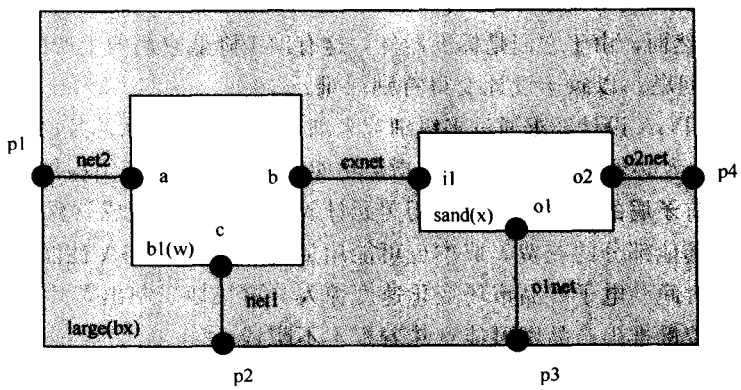


图 1-10 分级逻辑设计

试读结束，需要全本PDF请购买 www.ertongbook.com