



**FPGA Design Based on Verilog**

# 基于Verilog的FPGA 设计基础

❖ 杜慧敏 李宥谋 赵全良 编著



西安电子科技大学出版社

<http://www.xduph.com>

TP312  
1878

国家自然科学基金资助

# 基于 Verilog 的 FPGA 设计基础

杜慧敏 李宥谋 赵全良 编著

西安电子科技大学出版社

2006

## 内 容 简 介

本书简要介绍了FPGA的编程技术,详细讨论了以Altera FPGA为代表的可编程器件的结构和特点、Altera Quartus II集成环境的使用以及目前工业界最常用的仿真工具Modelsim的使用。重点讲授了FPGA设计流程中的基本概念、所采用的步骤和应该遵循的原则,包括模块划分原则、可综合Verilog编码风格、验证程序的编写方法和静态时序分析等。另外,本书结合Altera公司的Nios II软核,简单介绍了基于SOPC的系统设计方法以及Altera SOPC Builder软件的使用方法。

本书可作为从事数字集成电路设计及相关工程技术人员的参考书,也可作为大专院校电子信息、自动控制等专业高年级本科生及研究生的教学用书。

### 图书在版编目(CIP)数据

基于Verilog的FPGA设计基础/杜慧敏等编著. —西安:西安电子科技大学出版社,2006.2

(国家自然科学基金资助)

ISBN 7-5606-1626-7

I. 基… II. 杜… III. ① 硬件描述语言—程序设计 ② 现场可编程门阵列—系统设计  
IV. ① TP312 ② TP332.1

中国版本图书馆CIP数据核字(2005)第155182号

责任编辑 王晓杰 云立实 臧延新

出版发行 西安电子科技大学出版社(西安市太白南路2号)

电 话 (029)88242885 88201467 邮 编 710071

<http://www.xduph.com>

E-mail: [xdupfb@pub.xaonline.com](mailto:xdupfb@pub.xaonline.com)

经 销 新华书店

印刷单位 西安文化彩印厂

版 次 2006年2月第1版 2006年2月第1次印刷

开 本 787毫米×1092毫米 1/16 印张 24.375

字 数 578千字

印 数 1~4000册

定 价 35.00元

ISBN 7-5606-1626-7/TP·0931

**XDUP 1918001-1**

\*\*\*如有印装问题可调换\*\*\*

本社图书封面为激光防伪覆膜,谨防盗版。

# 前 言

FPGA(现场可编程门阵列)是一种大规模可编程逻辑器件,自1984年第一片FPGA问世至今,FPGA已经历了20年的发展历史。在这20年的发展过程中,FPGA从最初的1200门,发展到现在的几百万门,器件的集成度不断提高。以Altera、Xilinx等为代表的FPGA厂家不断更新、优化产品架构和生产工艺,降低了FPGA的功耗和系统成本,推出了很多高性能、低价位的解决方案,将FPGA应用从传统的高端通信产品扩展到汽车和消费类电子产品中。

随着低成本FPGA的推广,越来越多的科技工作者开始应用FPGA解决实际问题,也有越来越多的高等学校开始在本科生中开设基于FPGA进行系统设计方面的课程,以满足社会对这些方面人才的需要。本书正是在这个背景下编写的,其目的是为工程技术人员和在校学生提供基于FPGA设计的基本方法。

在以FPGA为载体的设计过程中,必须遵循一定的设计流程、设计原则和方法。本书涵盖了基于FPGA设计全过程的主要内容,包括系统规范的制定、模块的设计划分、可综合代码的编写原则、仿真程序的编写和后端验证方法等,并给出了一些较为复杂的设计实例。除了设计方法之外,本书以Altera的FPGA为例,对FPGA内部结构做了深入的分析,介绍了世界著名的FPGA公司的产品及其特点。作为FPGA设计过程中必不可少的开发环境,本书介绍了Altera公司的Quartus II开发环境以及MIT的Modelsim仿真软件。另外,对于目前非常流行的SOPC设计也做了简要的介绍,可供嵌入式系统的设计人员参考。

本书的三位作者在基于FPGA设计方面有着丰富的经验,书中的部分内容也是作者多年工作的总结,希望能对读者有所帮助。

本书分为8章,其中第1章、第3章和第4章由杜慧敏执笔,第2章和第7章由李有谋执笔,第5章、第6章和第8章由赵全良执笔。另外,在本书的编写过程中,得到了韩俊刚教授、蒋林教授和谢庆胜硕士的大力支持,他们为本书提出了许多宝贵意见。邢立冬和张阿宁同志验证了本书中的设计实例。在此一并表示感谢。

本书可作为大专院校FPGA设计基础课程的教材。书中所有例子均是用Verilog语言描述的,因此,使用本教材的学生应先学习Verilog语言和数字电路设计两门课程。另外,本书也可以作为相关领域工程技术人员的参考资料。

由于编者的水平有限,书中难免存在缺点和错误,恳请各位专家和读者批评指正。作者的联系方式为:fv@xiyou.edu.cn, lym@xiyou.edu.cn, zql@xiyou.edu.cn。

编 者

2005年11月

# 目 录

|                              |    |
|------------------------------|----|
| 第 1 章 绪论.....                | 1  |
| 1.1 FPGA 概述.....             | 1  |
| 1.1.1 FPGA 发展的简要回顾.....      | 1  |
| 1.1.2 FPGA 与 ASIC.....       | 4  |
| 1.2 可编程逻辑器件的基本概念.....        | 5  |
| 1.3 简单可编程器件(SPLD)的结构.....    | 9  |
| 1.4 高密度可编程逻辑器件.....          | 12 |
| 1.4.1 复杂可编程逻辑器件 CPLD.....    | 12 |
| 1.4.2 现场可编程门阵列 FPGA.....     | 14 |
| 1.4.3 CPLD 和 FPGA 的区别.....   | 16 |
| 1.4.4 FPGA/CPLD 厂家简介.....    | 17 |
| 1.5 基于 FPGA 的设计流程与设计方法.....  | 18 |
| 1.5.1 基于 FPGA 的设计流程.....     | 18 |
| 1.5.2 自顶向下和自底向上的设计方法学.....   | 20 |
| 1.5.3 基于 IP 核的设计.....        | 20 |
| 1.6 EDA 技术简介.....            | 21 |
| 第 2 章 可编程逻辑器件.....           | 24 |
| 2.1 Altera 器件概述.....         | 24 |
| 2.1.1 FPGA 系列简介.....         | 24 |
| 2.1.2 EPLD 系列简介.....         | 25 |
| 2.1.3 结构化 ASIC 器件.....       | 26 |
| 2.1.4 FPGA 器件的配置芯片.....      | 26 |
| 2.2 Altera 的 EPLD 器件系列.....  | 27 |
| 2.2.1 EPLD 器件的特性.....        | 27 |
| 2.2.2 MAX9000 器件的结构.....     | 30 |
| 2.2.3 MAX II 器件的结构.....      | 34 |
| 2.3 Altera 的 FPGA 器件.....    | 36 |
| 2.3.1 简单 FPGA 器件.....        | 36 |
| 2.3.2 复杂 FPGA 器件.....        | 49 |
| 2.3.3 新型 FPGA 器件.....        | 52 |
| 2.4 Xilinx 公司产品简介.....       | 65 |
| 2.4.1 Xilinx CPLD 器件.....    | 65 |
| 2.4.2 Xilinx FPGA 器件的特性..... | 67 |

|                                   |            |
|-----------------------------------|------------|
| 2.4.3 Xilinx FPGA 器件的结构.....      | 71         |
| 2.5 Lattice 公司产品简介.....           | 75         |
| 2.5.1 Lattice CPLD 器件系列.....      | 75         |
| 2.5.2 Lattice FPGA 产品系列.....      | 76         |
| 2.5.3 FPSC 产品系列.....              | 77         |
| 2.5.4 低密度 PLD 产品系列.....           | 78         |
| 2.5.5 其他产品.....                   | 78         |
| 2.6 Actel 公司产品简介.....             | 78         |
| 2.6.1 Flash FPGA 器件.....          | 78         |
| 2.6.2 反熔丝 FPGA 器件.....            | 81         |
| 2.6.3 航空航天和军用器件.....              | 83         |
| <b>第 3 章 FPGA 设计入门.....</b>       | <b>84</b>  |
| 3.1 系统的抽象层次与高级硬件描述语言 Verilog..... | 84         |
| 3.2 用 Verilog 语言建立数字电路模型.....     | 87         |
| 3.2.1 代码的书写风格.....                | 87         |
| 3.2.2 可综合代码的编码风格.....             | 91         |
| 3.2.3 时序电路的设计.....                | 111        |
| 3.3 模块设计.....                     | 127        |
| 3.4 系统规范.....                     | 130        |
| 3.4.1 系统规范的内容.....                | 130        |
| 3.4.2 选择 FPGA.....                | 131        |
| <b>第 4 章 设计验证.....</b>            | <b>132</b> |
| 4.1 验证综述.....                     | 132        |
| 4.1.1 验证的概念.....                  | 132        |
| 4.1.2 验证和测试.....                  | 133        |
| 4.1.3 自顶向下和自底向上的验证方法.....         | 133        |
| 4.1.4 主要验证技术.....                 | 134        |
| 4.1.5 验证工具的介绍.....                | 136        |
| 4.1.6 验证计划和流程.....                | 138        |
| 4.2 功能验证.....                     | 140        |
| 4.2.1 验证程序(Testbench)的组成.....     | 140        |
| 4.2.2 实用构造 Testbench 技术.....      | 145        |
| 4.3 基于断言的验证.....                  | 165        |
| 4.4 时序验证.....                     | 168        |
| 4.4.1 静态时序分析概述.....               | 168        |
| 4.4.2 静态时序分析中的基本概念.....           | 171        |
| 4.4.3 假路径和多周期路径.....              | 176        |
| 4.4.4 时序验证中的系统任务.....             | 178        |

|                                  |            |
|----------------------------------|------------|
| <b>第 5 章 ModelSim 工具介绍 .....</b> | <b>179</b> |
| 5.1 ModelSim 概述 .....            | 179        |
| 5.1.1 基本仿真流程 .....               | 179        |
| 5.1.2 工程仿真流程 .....               | 180        |
| 5.1.3 多数据库仿真流程 .....             | 180        |
| 5.1.4 调试工具 .....                 | 181        |
| 5.2 ModelSim 工程 .....            | 181        |
| 5.2.1 创建一个新工程 .....              | 181        |
| 5.2.2 编译和加载设计 .....              | 183        |
| 5.2.3 利用文件夹组织工程 .....            | 183        |
| 5.2.4 在工程中进行仿真配置 .....           | 184        |
| 5.2.5 关于工程的其他基本操作 .....          | 186        |
| 5.2.6 Project 标签页及菜单简介 .....     | 186        |
| 5.2.7 指定文件属性和工程设置 .....          | 187        |
| 5.3 设计库 .....                    | 188        |
| 5.3.1 设计库简介 .....                | 188        |
| 5.3.2 使用设计库工作 .....              | 189        |
| 5.3.3 导入 FPGA 设计库 .....          | 192        |
| 5.4 Verilog 基本仿真 .....           | 192        |
| 5.4.1 创建工作的设计数据库 .....           | 192        |
| 5.4.2 编译设计 .....                 | 194        |
| 5.4.3 运行仿真 .....                 | 194        |
| 5.4.4 设置断点与源代码单步执行 .....         | 195        |
| 5.4.5 结束仿真 .....                 | 196        |
| 5.4.6 增量编译 .....                 | 196        |
| 5.5 在 Verilog 仿真中连接第三方资源库 .....  | 196        |
| 5.5.1 仿真连接资源库 .....              | 196        |
| 5.5.2 永久性映射资源库 .....             | 198        |
| 5.6 使用波形窗口 .....                 | 198        |
| 5.6.1 向波形窗口添加项目 .....            | 198        |
| 5.6.2 波形显示的图像缩放 .....            | 199        |
| 5.6.3 在波形窗口中使用游标 .....           | 200        |
| 5.6.4 存储波形窗口格式 .....             | 200        |
| 5.6.5 WLF 文件(Datasets) .....     | 200        |
| 5.7 使用数据流(dataflow)窗口进行调试 .....  | 202        |
| 5.7.1 编译并加载一个例子 .....            | 202        |
| 5.7.2 观察设计的连接性 .....             | 202        |
| 5.7.3 跟踪事件 .....                 | 204        |
| 5.7.4 追踪未知态 .....                | 205        |

|                                      |            |
|--------------------------------------|------------|
| 5.7.5 在 dataflow 窗口中显示层次结构 .....     | 206        |
| 5.8 存储器的查看与初始化 .....                 | 207        |
| 5.8.1 编译和装入设计举例 .....                | 207        |
| 5.8.2 查看存储器 .....                    | 208        |
| 5.8.3 保存存储器数据到一个文件 .....             | 210        |
| 5.8.4 初始化一个存储区 .....                 | 211        |
| 5.8.5 交互式调试命令 .....                  | 212        |
| 5.9 使用性能分析器仿真 .....                  | 214        |
| 5.9.1 性能分析器简介及本节的设计文件 .....          | 214        |
| 5.9.2 编译、加载例子的设计 .....               | 214        |
| 5.9.3 运行仿真 .....                     | 215        |
| 5.9.4 使用数据改进性能 .....                 | 216        |
| 5.9.5 过滤并保存数据 .....                  | 216        |
| 5.10 仿真代码覆盖情况 .....                  | 217        |
| 5.10.1 编译、加载例子的设计 .....              | 217        |
| 5.10.2 在主窗口中查看统计 .....               | 218        |
| 5.10.3 在源代码窗口中查看统计 .....             | 218        |
| 5.10.4 在信号窗口中查看状态翻转统计 .....          | 220        |
| 5.10.5 指定不进行覆盖率统计的行或文件 .....         | 220        |
| 5.10.6 创建代码覆盖率报告 .....               | 221        |
| 5.11 波形比较过程 .....                    | 221        |
| 5.11.1 波形比较器简介及本节的设计文件 .....         | 221        |
| 5.11.2 创建参考数据文件和测试数据文件 .....         | 222        |
| 5.11.3 比较仿真运行 .....                  | 222        |
| 5.11.4 查看比较数据 .....                  | 223        |
| 5.11.5 保存和重装比较数据 .....               | 225        |
| 5.12 ModelSim 自动运行 .....             | 225        |
| 5.12.1 创建简单的 DO 文件 .....             | 226        |
| 5.12.2 使用“启动 DO 文件”运行 ModelSim ..... | 227        |
| 5.12.3 命令行方式运行 ModelSim .....        | 227        |
| 5.12.4 与 ModelSim 一起使用 Tcl .....     | 228        |
| 5.13 使用 ModelSim 进行后仿真 .....         | 229        |
| <b>第 6 章 Quartus 集成环境 .....</b>      | <b>233</b> |
| 6.1 Quartus II 软件概述 .....            | 234        |
| 6.1.1 Quartus II 软件的安装 .....         | 234        |
| 6.1.2 Quartus II 软件工具授权 .....        | 236        |
| 6.2 Quartus II 设计流程简介 .....          | 237        |
| 6.3 设计输入 .....                       | 240        |
| 6.3.1 创建一个工程 .....                   | 240        |



|                                                     |     |
|-----------------------------------------------------|-----|
| 6.3.2 创建一个设计 .....                                  | 241 |
| 6.3.3 Quartus 使用举例 .....                            | 242 |
| 6.4 配置设计工程的编译约束 .....                               | 247 |
| 6.4.1 使用 Assignment Editor .....                    | 248 |
| 6.4.2 使用引脚规划器(Pin Planner) .....                    | 249 |
| 6.4.3 使用 Settings 对话框 .....                         | 250 |
| 6.5 综合设计 .....                                      | 250 |
| 6.5.1 使用 Quartus II Verilog HDL 及 VHDL 集成综合工具 ..... | 251 |
| 6.5.2 使用其他 EDA 综合工具 .....                           | 253 |
| 6.5.3 “Analysis & Synthesis” 的控制 .....              | 254 |
| 6.6 布局布线 .....                                      | 256 |
| 6.6.1 执行一个完整的增量编译 .....                             | 258 |
| 6.6.2 分析布局布线结果 .....                                | 258 |
| 6.6.3 布局布线的优化 .....                                 | 260 |
| 6.7 基于模块的设计 .....                                   | 265 |
| 6.7.1 Quartus II 基于模块化的设计流程 .....                   | 266 |
| 6.7.2 使用逻辑锁区域(LogicLock Regins) .....               | 266 |
| 6.7.3 在自顶向下增量编译流程中使用区域逻辑锁 .....                     | 268 |
| 6.7.4 在自底向上逻辑锁流程中保存中间综合结果 .....                     | 268 |
| 6.7.5 在 EDA 工具中集中使用逻辑锁 .....                        | 270 |
| 6.8 Quartus II 的时序分析(Timing Analysis) .....         | 271 |
| 6.8.1 在 Quartus II 软件中执行时序分析 .....                  | 271 |
| 6.8.2 进行前期的时序评估 .....                               | 274 |
| 6.8.3 查看时序分析结果 .....                                | 274 |
| 6.8.4 使用第三方 EDA 工具进行时序分析 .....                      | 276 |
| 6.9 时序逼近(Timing Closure) .....                      | 277 |
| 6.9.1 使用时序逼近底层图(Timing Closure Floorplan) .....     | 277 |
| 6.9.2 使用时序优化顾问 .....                                | 278 |
| 6.9.3 使用网表优化实现时序逼近 .....                            | 279 |
| 6.9.4 使用 LogicLock Regins 实现时序逼近 .....              | 280 |
| 6.9.5 使用增量编译达到时序逼近 .....                            | 280 |
| 6.10 功率分析(Power Analysis) .....                     | 280 |
| 6.11 对器件的编程与配置 .....                                | 283 |
| 6.12 调试 .....                                       | 286 |
| 6.12.1 使用 SignalTap II 逻辑分析仪 .....                  | 287 |
| 6.12.2 使用 SignalProbe 信号探针 .....                    | 289 |
| 6.12.3 使用 In-System Memory Content Editor .....     | 289 |
| 6.12.4 使用寄存器传输级查看器(RTL Viewer) .....                | 290 |
| 6.12.5 使用芯片编辑器 .....                                | 290 |

|                                          |            |
|------------------------------------------|------------|
| <b>第 7 章 FPGA 设计实例 .....</b>             | <b>291</b> |
| 7.1 74181ALU 运算器设计 .....                 | 291        |
| 7.1.1 74181ALU 的功能说明 .....               | 291        |
| 7.1.2 逻辑电路 .....                         | 292        |
| 7.1.3 Verilog 程序设计 .....                 | 292        |
| 7.1.4 ALU 运算器的功能验证 .....                 | 298        |
| 7.2 伪随机序列设计 .....                        | 300        |
| 7.2.1 m 序列 .....                         | 300        |
| 7.2.2 9 位的 LFSR 计数器 .....                | 302        |
| 7.2.3 数字序列的扰码 .....                      | 305        |
| 7.2.4 循环冗余校验 .....                       | 310        |
| 7.3 SDH 解帧器设计 .....                      | 321        |
| 7.4 8b/10b 编码设计 .....                    | 329        |
| 7.4.1 8b/10b 编码技术 .....                  | 330        |
| 7.4.2 8b/10b 编码器的设计 .....                | 333        |
| 7.4.3 程序代码简介 .....                       | 334        |
| 7.4.4 Testbench 程序设计 .....               | 339        |
| <b>第 8 章 Altera 系统级的 SOPC 开发 .....</b>   | <b>342</b> |
| 8.1 Altera IP 的使用 .....                  | 342        |
| 8.2 SOPC 开发流程概述 .....                    | 343        |
| 8.2.1 应用系统需求分析 .....                     | 344        |
| 8.2.2 使用 SOPC Builder 建立 SOPC 系统设计 ..... | 345        |
| 8.2.3 Nios II 监控软件的开发 .....              | 346        |
| 8.2.4 灵活运用 SOPC 开发流程 .....               | 348        |
| 8.3 Altera Nios CPU 简介 .....             | 348        |
| 8.4 Altera Nios 外设组件简介 .....             | 349        |
| 8.5 Altera 1C20 Demo 板介绍 .....           | 350        |
| 8.6 Altera 1C20 试验板上的 SOPC 系统开发实例 .....  | 353        |
| 8.6.1 开发实例功能介绍 .....                     | 353        |
| 8.6.2 开发设计步骤 .....                       | 354        |
| <b>附录 频率计系统的设计 .....</b>                 | <b>370</b> |
| <b>参考文献 .....</b>                        | <b>380</b> |

# 第 1 章

## 绪 论

现场可编程门阵列 FPGA(Field Programmable Gate Array)是 20 世纪 80 年代中期由美国 Xilinx 公司首先推出来的。随着半导体加工工艺的不断发展, FPGA 在结构、速度、工艺、集成度和性能等方面都有了极大的改进和提高, 与之相对应的设计方法学和自动化设计工具也得到迅速的发展。其中, 设计工具的自动化为 FPGA 的应用和发展起到了推波助澜的作用。本章将简要回顾一下 FPGA 器件和自动化设计工具的发展过程, 并简要介绍基于 FPGA 设计的流程。

### 1.1 FPGA 概述

#### 1.1.1 FPGA 发展的简要回顾

简单地讲, 现场可编程门阵列 FPGA 是一种可以编程的数字集成电路 IC(Integrated Circuit), 它包含了可配置的逻辑块以及逻辑块之间的互连线。所谓的现场可编程是指设计人员可以通过在工作现场(如实验室、调试现场等)完成对这些逻辑块和连线的配置, 以实现或改变复杂的电子系统的功能。

为了对 FPGA 有一个较全面的认识, 我们简要回顾一下与 FPGA 发展相关的技术。如图 1.1 所示, 图中的白色区域表示已开始研究该技术的时间, 但是由于种种原因没有得到很好的推广。

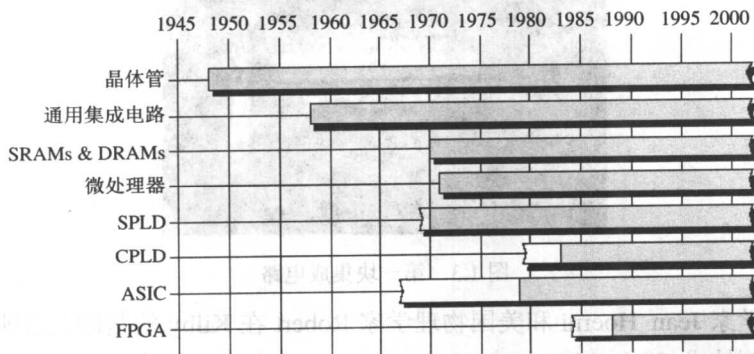


图 1.1 FPGA 技术发展过程

## 1) 晶体管

1947 年 12 月 23 日, 美国物理学家 William Shockley, Walter Brattain 和 John Bardeen 在贝尔实验室首先制造出第一只点接触式锗晶体管, 如图 1.2 所示。他们三人也因对半导体物理做出的杰出贡献在 1956 年获得诺贝尔物理学奖。1950 年制造出的双极型晶体管, 也称 BJT(Bipolar Junction Transistor)。

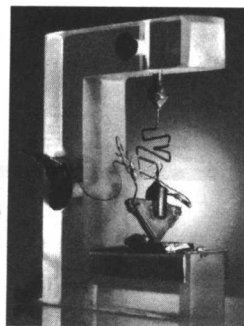


图 1.2 第一只晶体管

20 世纪 50 年代后期可以用硅片生产晶体管。由于硅比锗要便宜的多, 因此 BJT 工艺的晶体管得到了广泛的应用。根据连接关系的不同, BJT 工艺的晶体管可以构建出 TTL (Transistor-Transistor Logic) 类型的数字逻辑门, 也可以构造 ECL(Emitter-Coupled Logic) 类型的数字逻辑门。ECL 门的速度比 TTL 门的快, 但是功耗比 TTL 门大。

1962 年, 美国科学家 Steven Hofstein 和 Fredric Heiman 在普林斯顿的 RCA 研究试验室发明了一种被称为金属氧化物场效应管 MOSFET(Metal-Oxide Semiconductor Field-Effect Transistors) 工艺的元件, 简称 FET。虽然当时的 FET 比 BJT 的速度慢, 但是它们具有廉价、体积小和功耗小等特点。有两种类型的 FET, 一种称为 NMOS, 另一种称为 PMOS。由 PMOS 和 NMOS 可以构造出 CMOS 逻辑门(Complementary Metal-Oxide Semiconductor), 这种方式实现的逻辑门速度上比 TTL 要稍稍慢一些, 但是两者的工艺是一致的。CMOS 工艺的逻辑门的最大优点就是它的静态功耗非常小。CMOS 工艺是现代超大规模集成电路的主流工艺。

## 2) 通用集成电路(IC)

第一个晶体管是以分立元件的形式出现的, 它可以用小的金属外壳独立封装。在这个基础上, 人们开始想到在一个半导体上生产出一个完整的电路。1952 年 5 月, 英国雷达专家 G.W.A.Dummer 首先提出集成电路的概念。时隔 6 年之后, 1958 年 9 月, 美国德州仪器 TI(Texas Instruments) Jack Kilby 成功地生产出第一个由五个元件构成的简单晶体振荡器集成电路, 如图 1.3 所示。2000 年, Jack Kilby 因发明集成电路而和其他物理学家一起获得了物理学的诺贝尔奖, 他是诺贝尔历史上少有的获此殊荣的工程师。



图 1.3 第一块集成电路

瑞士物理学家 Jean Hoerni 和美国物理学家 Robert 在 Kilby 的基础上发明了现代 IC 生产中用到的光刻技术(Optical Lithographic Techniques), 奠定了现代 IC 生产的基础。20 世纪 60 年代中期, TI 开始为军用和商用生产 5400 系列和 7400 系列的 TTL IC。这些 IC 完成非

常简单的逻辑门。例如，2 输入 NAND 门 7400，2 输入 NOR 7402 等。有时，把这些逻辑也称为粘和逻辑(英文称为 Jelly-bean Logic)。

### 3) SRAM, DRAM 和微处理器

20 世纪 60 年代后期到 70 年代前期，数字 IC 得到了飞速的发展。1970 年 Intel 公司宣布 1024 bit 的 DRAM 诞生了，当时著名的仙童(Fairchild)半导体公司生产出第一个 256 bit 的 SRAM，它是 FPGA 的基础。1971 年，Intel 公司生产出第一个包含 2300 个晶体管，每秒钟完成 60 000 个操作的微处理器。

### 4) SPLD 和 CPLD

第一个可编程的 IC 是在 1970 年以 PROM 的形式出现的，通常被称为可编程逻辑元件 PLD(Programmable Logic Devices)。刚开始的 PLD 结构比较简单，只能完成简单的功能，把这种 PLD 称为简单的 PLD，即 SPLD。到 20 世纪 70 年代末以后，出现了规模更大，集成度更高的 PLD，可以完成非常复杂的功能，我们把这类 PLD 称为复杂 PLD，即 CPLD。1984 年，Altera 公司推出了结合 CMOS 和 EPROM 工艺的 CPLD，这种 CPLD 可以实现较复杂的功能，功耗相对又比较小，将 CPLD 元件的性能提高了一大步。

### 5) ASIC

在 FPGA 的发展史上，专用集成电路 ASIC 的实现方法对 FPGA/CPLD 的发展有着重要的影响。按实现方法划分，ASIC 可以被分为全定制 ASIC 和半定制 ASIC。半定制 ASIC 中又分为门阵 ASIC、标准单元 ASIC 和结构化 ASIC。

在数字 IC 发展的早期，除了存储器之外，主要有两大类 IC，一类是由 TI 和 Fairchild 公司生产的功能相对简单的数字 IC，它们是以现货方式提供给用户。另外一类就是根据用户特殊需求设计生产的全定制的 IC，称为专用集成电路 ASIC(Application Specific Integrated Circuit)。在全定制的 ASIC 中，设计工程师完成用于芯片加工的每个掩膜层的控制，ASIC 厂商并不在硅片上预先放置任何元件，也不提供任何预定义的逻辑门和功能，而是借助于特殊的工具，由工程师手工完成每个晶体管的尺寸设计，然后根据这些晶体管创建高层的功能。例如，如果工程师需要一个速度快的逻辑门，那么他可以改变构造该门的晶体管的尺寸。全定制的设计非常复杂，同时也非常耗时，但是生产的芯片在面积、功耗、速度等方面具有优势。

### 6) Micromatrix 和 Micromosaic

20 世纪 60 年代中期，Fairchild 半导体公司引入了被称为 Micromatrix 的元件，该元件中包含了 100 个左右的未连接的裸晶体管。为了使这种元件能完成特定的功能，设计工程师通过手工的方式绘制这些晶体管之间的连线。虽然这种方式费时，且易出错，但是却允许工程师在一定的时间内，用较高的代价形成一个定制的元件。

到了 20 世纪 60 年代的后期，Fairchild 半导体公司生产了 Micromosaic 元件。它包含了几百个没有连接的晶体管，这些晶体管可以在以后连接并实现 150 个左右的与/或非门。Micromosaic 的主要特性就是设计工程师可以通过布尔表达式说明元件要完成的功能，然后通过计算机程序决定哪些晶体管进行互联，同时计算机程序也构造用于生产元件的光掩模。这在 IC 历史上具有重要的意义，它是现代 ASIC 中门阵的先驱。

### 7) 门阵

门阵的概念最早由 IBM 和富士通等公司在 20 世纪 60 年代后期提出的，然而直到 20

世纪 70 年代中期, 基于 CMOS 的门阵工艺成熟后该项技术才得以推广。门阵的基本思想是预先在硅片上生产一些没有连接关系的称为基本单元的晶体管和电阻, 上面几层作为晶体管互连的金属层用全定制掩模的方式确定。门阵实现方式的缺点是大多数设计可能不能完全使用门阵的内部资源, 造成一些浪费。另外, 门阵的布局和布线优化程度也不高。

#### 8) 标准单元

为了克服门阵的缺点, 在 20 世纪 80 年代初, 推出了基于标准单元的 ASIC。标准单元有许多地方与门阵非常相似, ASIC 厂家定义了设计工程师用到的单元库, 同时也提供了软宏和硬宏的库。这些库包括处理器、通信接口以及 ROM 和 RAM 等, 设计人员可以根据需要决定是否购买和重用这些被称为知识产权 IP(Intelligence Property)的标准单元。与门阵不同的是, 标准单元并不预先在芯片上生产元件, 而是通过软件工具决定网表中逻辑门之间的互连, 并产生用于生产的每层定制的光掩膜。

#### 9) FPGA

在 20 世纪 80 年代, 在数字 IC 和可编程逻辑元件之间存在着一个空挡, 即 SPLD 和 CPLD 具有易修改和设计周期短等特点, 但由于结构和资源等方面的限制, 不能实现非常复杂的功能。另外一方面是 ASIC, 它可以实现非常复杂的功能, 但开发周期长并且不可修改。为了填补这个空挡, 1984 年, Xilinx 公司开始在市场上推出一种现场可编程门阵列器件, 即 FPGA。FPGA 结构类似于门阵列 ASIC, 但它是可编程的。很多时候, FPGA 被称为是一种可编程的 ASIC。

#### 10) 结构化 ASIC

结构化 ASIC 是在 20 世纪 90 年代初提出的概念, 但是这一概念直到 10 年之后才得到 ASIC 厂商的重视。2003 年, Altera 首先推出了 Hard Copy Structured ASIC。结构化 ASIC 类似于门阵列, 底层单元都已经做好了, 掩膜也是现成的。用户只需要对几层的金属连线和通孔进行编程, 大大节约了掩膜层的成本。结构化 ASIC 主要适合中规模产量需要, 单芯片成本低于 FPGA, 但高于标准单元 ASIC。结构化 ASIC 的主要特点是和其本身 FPGA 100%兼容(包括功能、管脚和时序等), 用户验证过的 FPGA 原型设计, 可直接由厂家在几周内转换成 ASIC。由于结构化 ASIC 成本低, 同时生产周期比较短, 因此, 一些 ASIC 厂家现在正在积极推广结构化 ASIC 器件。

### 1.1.2 FPGA 与 ASIC

IC 的种类非常多, 从完成简单逻辑功能的 IC 到完成复杂系统功能的系统芯片应有尽有。我们感兴趣的两类芯片是可编程逻辑器件 PLD 和专用集成电路 ASIC, 其中可编程逻辑器件按其规模划分为低密度可编程逻辑器件和高密度可编程逻辑器件, FPGA 是高密度可编程逻辑器件。

与通用 IC 不同的是, 这两类芯片都可以根据用户的需要实现特殊功能。其中, ASIC 是为用户定制的芯片, 需要经过 ASIC 厂家生产, 它可以完成非常复杂的系统功能, 芯片的规模也可以非常大。与通用集成电路相比, ASIC 在构成电子系统时具有以下几个方面的优越性:

(1) 缩小系统的体积, 减轻系统的重量, 降低系统的功耗和提高系统的性能。

(2) 提高可靠性。用 ASIC 芯片进行系统集成后, 外部连线减少, 因而可靠性明显提高。

(3) 可增强保密性。电子产品中的 ASIC 芯片对用户来说相当于一个“黑匣子”，难以仿造。

(4) 在大批量应用时，可显著降低成本。

而 PLD 也可以根据用户的需要完成特殊的功能，其中低密度可编程逻辑器件只能完成简单的逻辑功能，而高密度逻辑可编程器件(如 CPLD 和 FPGA)则可以实现非常复杂的系统功能。

与 ASIC 不同的是，PLD 是在市面上可以购买的，其实现功能可以在现场进行修改，而 ASIC 一旦生产就不能修改了。

FPGA 的主要用途有两个方面：

(1) 作为 ASIC 设计的快速原型系统。生产 ASIC 的费用非常昂贵，这其中包含了两方面的费用，一是设计 ASIC 的工具费用，另外就是 ASIC 中不可回归的工程费用，即通常所言的 NRE(Nonrecurring Engineering)费用。正如前面所述，一旦 ASIC 产生，就不能修改，设计中的任何微小的错误，都可能导致 ASIC 的失败，如果修改后重新投片，需要向 ASIC 厂家再支付一笔 NRE。因此，许多 ASIC 设计人员在流片之前，先用 FPGA 系统验证 ASIC 设计。与流片费用相比，购买 FPGA 的价格要低得多。另外，如果购买了某个厂家的 FPGA，FPGA 的供应商会提供相应的开发系统。从经济的角度讲，FPGA 的开发费用要小得多。但是，如果 ASIC 用量非常大，NRE 费用平摊到每个芯片上时，ASIC 单片价格就比购买 FPGA 的价格要低，因此，在大批量使用时，还是考虑用 ASIC 而不是 FPGA。

(2) 验证新算法的物理实现。很多应用场合，设计人员提出一些新的算法，为了验证算法硬件的可实现性和算法的正确性，通常也用 FPGA 作为实现的一种载体。

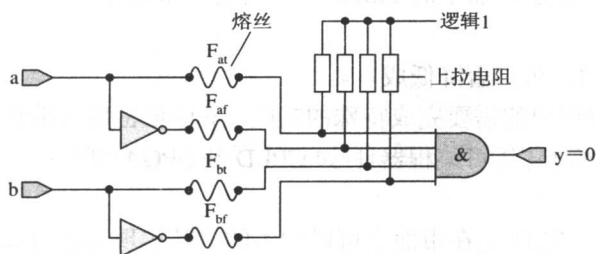
随着半导体工艺的进步，FPGA 厂家也在生产一些比较廉价的 FPGA，因此在使用数量不多的时候，也可以考虑用 FPGA 而不用 ASIC。此外，由于电子产品更新换代的速度加快，许多产品为了快速占领市场，也在大量使用 FPGA。

## 1.2 可编程逻辑器件的基本概念

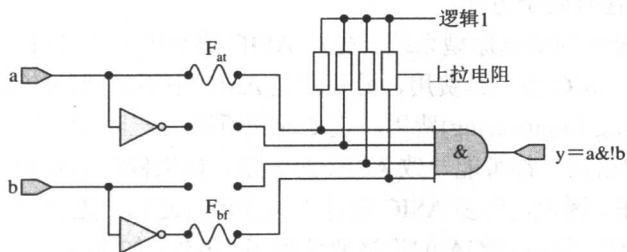
FPGA 区别于 ASIC 的一个重要特点就是 FPGA 是现场可编程的。本小节简单介绍可编程器件的一些相关技术。

### 1. 熔丝链技术

第一个在可编程逻辑器件中用到的技术是熔丝链技术。出厂时元件之间是通过熔丝连接的，如图 1.4(a)所示。图中所示的熔丝类似日常生活中所用的保险丝，当电流非常大时，熔丝便熔断。当熔丝熔断后，与熔丝相连接的输入被断开，由于上拉电阻的作用，输入端的逻辑值为逻辑高电平。设计工程师可以根据设计需要通过施加规定宽度和幅度的脉冲电流，熔断熔丝，完成相应的功能。例如当图 1.4(b)中  $F_{bt}$  和  $F_{af}$  熔断后，实现的逻辑功能是： $y=a \&!b$ 。熔丝一旦熔断，便不可再恢复，因此，编程是一次性的。虽然熔丝链技术不是 FPGA 的编程技术，但它是 FPGA 编程技术发展的起点。



(a)



(b)

图 1.4 熔丝电路

(a) 通过熔丝连接的电路; (b) 熔丝熔断后的电路

## 2. 反熔丝技术

一个普通的熔丝一般是构成一个连接，直到有一个过量的电流通过熔丝并烧断为止。反熔丝则正好相反，它在施加电压之前是断开的，而在施加了电压后形成导体。图 1.5 说明了反熔丝电路的形成。

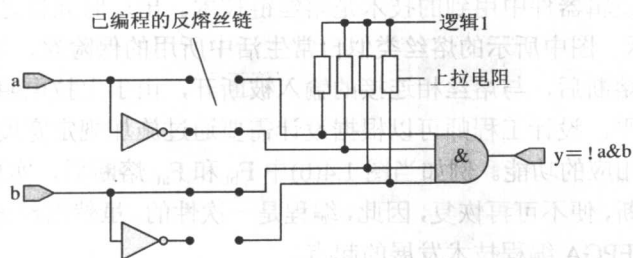
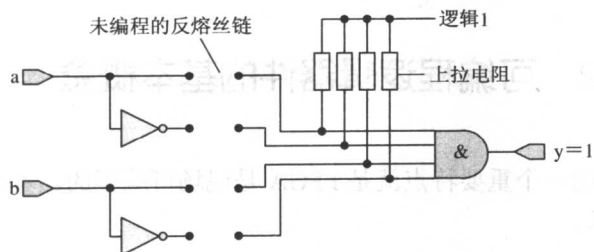


图 1.5 反熔丝电路



反熔丝技术的基本原理是在绝缘体分开的两个导体之间跨接一个反熔丝链，当在这个反熔丝链上施加一个大电压时，反熔丝链熔化，从而在两个导体之间产生电流。反熔丝隔开了 FPGA 上的互连线，而编程器烧断反熔丝后形成了永久的连接，这个过程不可逆转。因此，基于反熔丝的 FPGA 是不可再编程的。

反熔丝的 FPGA 器件的主要优点是：速度快、功耗低、非易失、抗辐射性好和保密性好。因此，它被应用到航天和军事系统中。它的缺点是不可重复设计，要求有编程器及专用的插座，封装都是 BGA；另外，由于反熔丝，所以它需要一定的编程电流。

### 3. 可编程只读存储器 PROM(Programmable Read Only Memory)

ROM 数据的存储是靠生产厂家特定的掩膜将数据写入存储器，如果用户需修改已存储的数据，则需要由生产厂家重新生产一套新的掩膜，经过加工测试然后再交回用户，这个周期比较长，对用户来讲，非常不方便。PROM 的出现解决了这一问题。PROM 允许通过专用的编程器将数据“烧录”到存储器中，这个过程叫做“编程”，烧录后的数据同样能保持断电后不丢失。

一次性 PROM 单元是由熔丝和二极管或三极管构成的，如图 1.6 所示。当大电流通过熔丝时，熔丝断开，从而切断原来的连接。PROM 产品在出厂时，所有存储单元均被加工成同一状态“0”(或“1”)。用户对 PROM 编程是逐字逐位进行的，根据需要写入的信息，按字线和位线选择某个存储单元，通过脉冲电流将该三极管的熔丝熔断，使该存储单元的状态被改变成与原状态相反的状态。熔丝一旦熔断，便不可再恢复，因此，编程是一次性的，PROM 的这种特性影响了它的使用，目前这种器件很少使用。

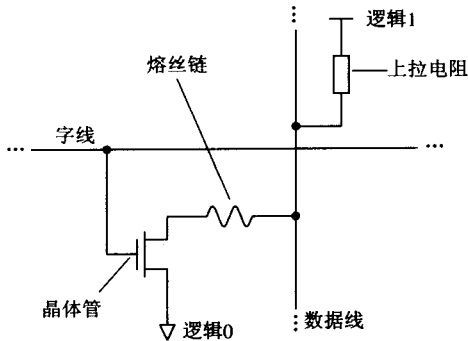


图 1.6 PROM 结构

### 4. EPROM 技术

由于 PROM 是一次性的，一旦编程完毕，其内容便不能再改变，如果数据烧录错误，那么 PROM 只能报废。Intel 公司在 1971 年首次开发了可擦除可编程的只读存储器 EPROM(Erasable PROM)，这种器件由于允许用户利用编码器对器件反复编程、擦除，因而得到了广泛的应用。这种器件通过施加高压信号进行编程，将器件置于紫外线下就可以擦除其内容。

EPROM 型的晶体管与标准的 MOS 管具有相同的结构，而与 MOS 不同的是，它增加了另外一个被称为浮栅的多晶硅，有时把这个晶体管称为浮栅雪崩注入 MOS 管，即 FAMOS 管。FAMOS 管的栅极全部被二氧化硅绝缘层包着，没有引出线，呈悬浮状，所以称作浮栅。图 1.7 给出了这两种晶体管的结构。