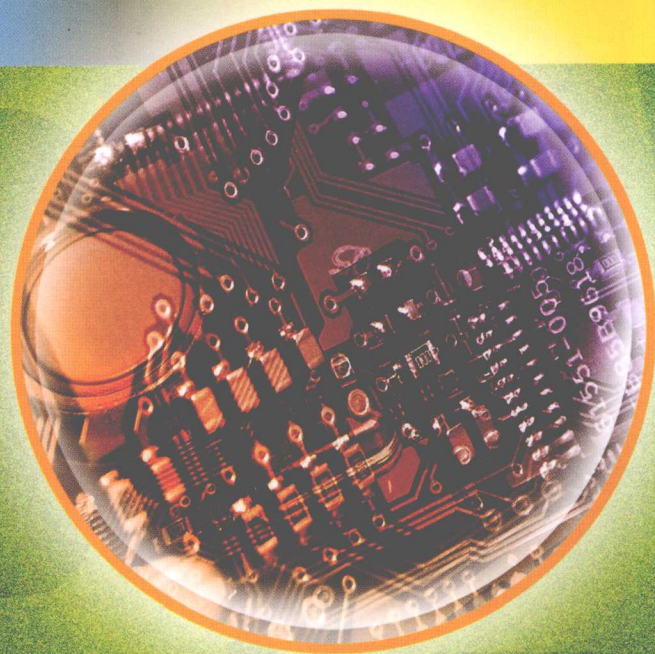


EDA工具应用丛书

# VHDL 开发精解 与实例剖析

云创工作室 詹仙宁 田耘 编著



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
<http://www.phei.com.cn>

<http://www.phei.com.cn>



EDA 工具应用丛书

# VHDL 开发精解与实例剖析

云创工作室 詹仙宁 田耘 编著

電子工業出版社.

Publishing House of Electronics Industry

北京 • BEIJING

电话: (010) 88538888  
 地址: 北京市海淀区中关村大街1号 邮编: 100080  
 电子邮箱: zls@phei.com.cn 网站: www.phei.com.cn

## 内 容 简 介

本书语言简洁,层次清晰,主要包括 VHDL 核心知识介绍和程序设计实践两部分。在介绍基础知识之后更多侧重于对 VHDL 可综合设计中的难点进行详细剖析,包括 VHDL 高级层次化设计、面向仿真的 VHDL 程序代码设计、基于 TEXT IO 的测试平台的应用,对实践中使用的物理器件平台(Xilinx Spartan 3E FPGA)和程序开发软件(Xilinx ISE 10.1)的使用进行了详细介绍。最后以数据加密标准 DES 为例,从算法原理介绍到系统方案设计,到程序代码设计,到系统仿真与综合,详细介绍了 VHDL 语言用于数字系统设计的流程、方法和技巧。

本书内容丰富,讲解清楚,适合作为高等院校微电子、电子、通信等专业的教材或教学参考书,同时也可以作为广大硬件电路设计工程师的工具书或培训教材。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

## 图书在版编目(CIP)数据

VHDL 开发精解与实例剖析/詹仙宁编著. —北京:电子工业出版社,2009.9

(EDA 工具应用丛书)

ISBN 978-7-121-09623-5

I. V… II. 詹… III. 硬件描述语言, VHDL—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2009)第 176149 号

责任编辑:高买花 曲 昕

印 刷:北京京师印务有限公司

装 订:

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本:787×1 092 1/16 印张:23.25 字数:595 千字

印 次:2009 年 9 月第 1 次印刷

印 数:4 000 册 定价:43.00 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010) 88258888。

# 前言

目前,EDA 技术已经成为现代电子设计领域的基本手段,涵盖印制电路板(PCB)设计、FPGA 芯片开发、芯片设计以及系统验证等诸多领域。硬件描述语言(HDL)是 EDA 技术中的一个重要组成部分,可应用于 FPGA 开发和芯片设计等领域。VHDL 语言是两个主流 HDL 语言之一,它以强大的系统描述能力、规范的程序设计结构、灵活的语句表达风格和多层次的仿真测试手段,受到了业界的普遍认同和广泛接受,在全球范围具有广泛的用户群。特别是在我国,90%以上的高校都以 VHDL 语言教学为主。本书通过 Xilinx 公司的 FPGA 平台来介绍 VHDL 在 FPGA/CPLD 上的程序开发相关知识与方法。

## 本书结构

本书语言简洁,层次清晰,主要内容包括 VHDL 语言基础知识介绍和程序设计实践两部分。第 1 章至第 9 章为 VHDL 基础知识,主要包括以下内容:

- VHDL 的语言背景知识,内容包括可编程逻辑器件(PLD)、电子设计自动化(EDA)、硬件描述语言(HDL)等基本概念,以及 VHDL 的发展历史等(第 1 章);
- VHDL 程序的结构(第 2 章);
- VHDL 基本语法要素,包括词法单元、数据对象、数据类型、运算操作符、属性等(第 3 章);
- VHDL 描述语句,主要包括并行描述语句和顺序描述语句(第 4 章);
- 有限状态机的 VHDL 设计(第 5 章);
- VHDL 程序设计难点解析,总结和分析可综合 VHDL 程序设计中的某些重点问题(第 6 章);
- 高级层次化 VHDL 程序设计,主要介绍参数化 VHDL 模块的设计方法(第 7 章);
- VHDL 程序的综合,介绍 VHDL 程序的综合流程以及某些 VHDL 语法成分的综合结果(第 8 章);
- 面向仿真的 VHDL 程序设计(第 9 章)。

其中,第 7 章至第 9 章是 VHDL 基础知识的高级部分,可作为进一步学习提高之用。第 10 章至第 13 章为 VHDL 程序设计实践,主要内容包括:

- Xilinx 公司 FPGA 产品简介,介绍典型的 VHDL 程序实现所使用的物理载体,本书所有内容均基于 Xilinx 公司 FPGA 平台进行介绍(第 10 章);
- Xilinx ISE Design Suite 10.1 使用指南,介绍 Xilinx 最新的 FPGA 开发工具 ISE 10.1 的使用(第 11 章);
- 基本功能模块的 VHDL 实现,详细叙述了常见的数字功能模块的 VHDL 实现,这些模块包括:分频器、同步整形电路、按键扫描电路、数码管接口电路、VGA 接口电路、PS/2 接口电路、SPI 接口电路、LCD 接口电路以及伪随机序列产生器等,同时介绍了 Xilinx FPGA 常用的底层硬核模块的调用(第 12 章);
- DES 算法的 VHDL 实现,以一个较大的数字电路为例介绍复杂数字系统的 VHDL 设计方法(第 13 章)。

## 本书特色

- 介绍 VHDL 语法知识与强化数字电路概念的有机结合。目前,常见的 VHDL 语法书籍往往忽视了数字电路设计的基本概念,而关于数字电路设计的书籍又对 VHDL 介绍得很简单。与上述大多数同类书籍不同的是,本书在 VHDL 基础知识介绍部分注重知识的实际应用,通过大量的程序实例加以深入说明;而在程序设计实践部分又注重程序实例所包含的基本语法知识的介绍,让读者在实践中总结语法的应用,实现了语法学习过程与程序设计实践过程的有机结合。
- 可综合 VHDL 程序设计与面向仿真 VHDL 程序设计的有机结合。不注重 VHDL 语言的可综合特性和可仿真特性的区别是大多数同类书籍的又一缺点。本书自始至终强调两者具有重大区别,并在第 9 章单独介绍用于仿真的 VHDL 程序设计,因而为读者确立可综合 VHDL 程序设计的基本原则提供了良好的教学环境。
- 语言表达力求具有亲和力,通俗易懂。本书中有一大部分内容是作者自身学习过程的经验总结,对这一部分的介绍从初学者的角度出发,力求语言表达的亲和力与通俗性。
- 具有很强的实用性。本书介绍了 Xilinx 最新的 EDA 工具软件——Xilinx ISE Design Suite 10.1 的使用,具有很强的实用性。
- 例题难度循序渐进,有利初学者学习。

本书由云创工作室、詹仙宁和田耘共同编写,詹仙宁负责各章节的主要编写工作,田耘完成部分内容的编写,并对全书进行统编。

本书所有例子都经过作者的精心挑选,每个例子都具有一定的代表性,并在难度上循序渐进,有利于初学者的学习。而且素材丰富,讲解清楚,适合作为高等院校微电子、电子、通信等专业的教材或教学参考书,同时也可以作为广大硬件电路设计工程师的工具书或培训教材。

## 致 谢

感谢云峻岭、吴晶以及北京邮电大学信息与通信工程学院创新实验室所有的老师和同学们,他们为本书提出了许多宝贵的意见和建议。

此外,本书在写作过程中参考了众多国内外同类书籍,吸收了很多新的知识,在这里一并表示感谢。由于作者水平有限,书中难免有错误和表述不准确的地方,希望广大读者批评指正。

电子邮箱:zhanxn@gmail.com。

编著者

2009.7.25

# 目 录

|                           |    |
|---------------------------|----|
| 第 1 章 概 述 .....           | 1  |
| 1.1 可编程逻辑器件 .....         | 1  |
| 1.1.1 可编程逻辑器件概述 .....     | 1  |
| 1.1.2 可编程逻辑器件发展历史 .....   | 1  |
| 1.1.3 可编程逻辑器件分类 .....     | 2  |
| 1.2 EDA 技术与硬件描述语言 .....   | 3  |
| 1.2.1 EDA 技术简介 .....      | 4  |
| 1.2.2 硬件描述语言简介 .....      | 5  |
| 1.3 VHDL 语言简介 .....       | 5  |
| 1.3.1 VHDL 语言概述 .....     | 5  |
| 1.3.2 使用 VHDL 的优势 .....   | 6  |
| 1.3.3 VHDL 语言程序开发流程 ..... | 7  |
| 1.3.4 给初学者的建议 .....       | 8  |
| 1.4 本章总结 .....            | 8  |
| 第 2 章 VHDL 语言程序结构 .....   | 9  |
| 2.1 VHDL 程序结构概述 .....     | 9  |
| 2.2 VHDL 程序基本组成 .....     | 12 |
| 2.2.1 实体声明 .....          | 12 |
| 2.2.2 结构体 .....           | 14 |
| 2.2.3 库和程序包调用 .....       | 16 |
| 2.3 本章总结 .....            | 18 |
| 第 3 章 VHDL 语言基本要素 .....   | 19 |
| 3.1 词法单元 .....            | 19 |
| 3.1.1 关键字 .....           | 19 |
| 3.1.2 标志符 .....           | 19 |
| 3.1.3 数字 .....            | 20 |
| 3.1.4 字符和字符串 .....        | 20 |
| 3.1.5 位串 .....            | 21 |
| 3.1.6 注释 .....            | 21 |
| 3.2 数据对象 .....            | 21 |
| 3.2.1 信号 .....            | 22 |
| 3.2.2 变量 .....            | 23 |
| 3.2.3 常量 .....            | 25 |
| 3.2.4 文件 .....            | 25 |
| 3.2.5 数据对象小结 .....        | 25 |
| 3.3 数据类型 .....            | 25 |
| 3.3.1 VHDL 预定义数据类型 .....  | 26 |
| 3.3.2 用户自定义数据类型 .....     | 33 |
| 3.3.3 数据类型转换 .....        | 36 |

|                          |    |
|--------------------------|----|
| 3.3.4 数据类型小结 .....       | 36 |
| 3.4 运算操作符 .....          | 37 |
| 3.4.1 赋值运算符 .....        | 37 |
| 3.4.2 逻辑运算符 .....        | 37 |
| 3.4.3 算术运算符 .....        | 39 |
| 3.4.4 关系运算符 .....        | 40 |
| 3.4.5 移位运算符 .....        | 42 |
| 3.4.6 并置运算符 .....        | 43 |
| 3.4.7 运算操作符的优先级 .....    | 43 |
| 3.5 属性 .....             | 44 |
| 3.5.1 数组的属性 .....        | 44 |
| 3.5.2 信号的属性 .....        | 44 |
| 3.6 本章总结 .....           | 45 |
| 第4章 VHDL 语言描述语句 .....    | 46 |
| 4.1 VHDL 语句概述 .....      | 46 |
| 4.2 基本的 VHDL 并行语句 .....  | 47 |
| 4.2.1 并行信号赋值语句 .....     | 47 |
| 4.2.2 进程 .....           | 53 |
| 4.2.3 元件例化语句 .....       | 58 |
| 4.2.4 生成语句 .....         | 63 |
| 4.2.5 块语句 .....          | 65 |
| 4.2.6 子程序和子程序调用 .....    | 65 |
| 4.3 基本的 VHDL 顺序语句 .....  | 74 |
| 4.3.1 顺序赋值语句 .....       | 75 |
| 4.3.2 IF 语句 .....        | 75 |
| 4.3.3 CASE 语句 .....      | 78 |
| 4.3.4 LOOP 语句 .....      | 80 |
| 4.3.5 NULL 语句 .....      | 85 |
| 4.3.6 WAIT 语句 .....      | 85 |
| 4.4 本章总结 .....           | 86 |
| 第5章 有限状态机的 VHDL 设计 ..... | 87 |
| 5.1 有限状态机概述 .....        | 87 |
| 5.1.1 有限状态机的概念和分类 .....  | 87 |
| 5.1.2 有限状态机的状态转移图 .....  | 88 |
| 5.1.3 有限状态机设计流程 .....    | 89 |
| 5.2 有限状态机设计重点解析 .....    | 90 |
| 5.2.1 状态编码方式 .....       | 90 |
| 5.2.2 状态机的容错处理 .....     | 91 |
| 5.2.3 有限状态机的性能指标 .....   | 91 |
| 5.2.4 有限状态机设计思想 .....    | 91 |
| 5.3 有限状态机的 VHDL 描述 ..... | 92 |
| 5.3.1 “三进程”描述 .....      | 92 |
| 5.3.2 “双进程”描述 .....      | 95 |

|       |                     |     |
|-------|---------------------|-----|
| 5.3.3 | “单进程”描述             | 97  |
| 5.3.4 | 状态机的 VHDL 描述总结      | 99  |
| 5.4   | 有限状态机 VHDL 设计实例     | 102 |
| 5.4.1 | 交通灯控制器              | 102 |
| 5.4.2 | 乒乓游戏机               | 106 |
| 5.5   | 本章总结                | 110 |
| 第 6 章 | VHDL 程序设计难点解析       | 111 |
| 6.1   | 面向硬件的设计思维           | 111 |
| 6.1.1 | 硬件电路设计不是编写计算机指令     | 111 |
| 6.1.2 | 模块化与层次化是硬件电路设计的基本方法 | 112 |
| 6.1.3 | 硬件电路可以从不同抽象层次进行描述   | 113 |
| 6.1.4 | 结构体的描述方式            | 113 |
| 6.2   | 组合电路与时序电路           | 115 |
| 6.2.1 | 概述                  | 115 |
| 6.2.2 | 组合电路设计              | 116 |
| 6.2.3 | 时序电路设计              | 117 |
| 6.2.4 | 同步时序电路与异步时序电路       | 118 |
| 6.3   | 程序中的信号与变量           | 118 |
| 6.3.1 | 概述                  | 119 |
| 6.3.2 | 信号赋值与变量赋值           | 119 |
| 6.3.3 | 信号的应用——进程间通信        | 123 |
| 6.3.4 | 变量赋值的应用——算法描述       | 125 |
| 6.4   | 进程中的多边沿触发问题         | 126 |
| 6.4.1 | 概述                  | 127 |
| 6.4.2 | 不可综合的多沿触发           | 127 |
| 6.4.3 | 多沿触发协同工作            | 128 |
| 6.5   | 基本算术运算实现及其难点        | 130 |
| 6.5.1 | 数字在硬件电路中的表示         | 130 |
| 6.5.2 | 数据溢出及其处理            | 131 |
| 6.5.3 | 数据截断                | 138 |
| 6.6   | 锁存器                 | 139 |
| 6.6.1 | 锁存器概述               | 139 |
| 6.6.2 | 锁存器的产生及处理策略         | 139 |
| 6.6.3 | 锁存器的应用              | 141 |
| 6.7   | 本章总结                | 141 |
| 第 7 章 | VHDL 高级层次化设计        | 143 |
| 7.1   | 层次化设计概述             | 143 |
| 7.1.1 | 层次化设计的优点            | 143 |
| 7.1.2 | VHDL 层次化设计基础        | 144 |
| 7.2   | 基于 VHDL 的层次化设计      | 144 |
| 7.2.1 | 元件与元件例化             | 144 |
| 7.2.2 | 类属参数映射              | 144 |
| 7.2.3 | 配置                  | 147 |



|       |                  |     |
|-------|------------------|-----|
| 7.2.4 | 库和程序包            | 152 |
| 7.3   | 模块的参数化设计         | 156 |
| 7.3.1 | 概述               | 156 |
| 7.3.2 | 参数的分类            | 157 |
| 7.3.3 | 参数传递机制           | 157 |
| 7.4   | 参数化设计中常用的描述语句    | 160 |
| 7.4.1 | FOR LOOP 语句      | 160 |
| 7.4.2 | FOR GENERATE 语句  | 161 |
| 7.4.3 | IF GENERATE 语句   | 162 |
| 7.4.4 | EXIT 语句和 NEXT 语句 | 162 |
| 7.5   | 本章总结             | 162 |
| 第 8 章 | VHDL 程序的综合       | 164 |
| 8.1   | VHDL 基本操作符的综合    | 164 |
| 8.1.1 | 逻辑运算符的综合         | 164 |
| 8.1.2 | 关系运算符的综合         | 164 |
| 8.1.3 | 算术运算符的综合         | 165 |
| 8.1.4 | 其他运算操作符的综合       | 165 |
| 8.1.5 | 带常数的操作符及其综合      | 166 |
| 8.2   | VHDL 基本数据类型的综合   | 167 |
| 8.2.1 | 标准逻辑类型的综合        | 167 |
| 8.2.2 | 整数类型的综合          | 168 |
| 8.2.3 | 枚举类型的综合          | 168 |
| 8.3   | 参数化 VHDL 程序的综合   | 169 |
| 8.3.1 | 类属参数映射的综合        | 169 |
| 8.3.2 | 非限制数据的综合         | 169 |
| 8.4   | VHDL 程序的综合流程     | 169 |
| 8.4.1 | RTL 级综合          | 170 |
| 8.4.2 | 模块生成器            | 170 |
| 8.4.3 | 门级综合             | 171 |
| 8.4.4 | 器件级综合            | 171 |
| 8.5   | 本章总结             | 172 |
| 第 9 章 | 面向仿真的 VHDL 程序设计  | 173 |
| 9.1   | VHDL 的可综合性与可仿真特性 | 173 |
| 9.1.1 | VHDL 的可综合特性      | 173 |
| 9.1.2 | VHDL 的可仿真特性      | 174 |
| 9.2   | 常用的 VHDL 仿真语句    | 174 |
| 9.2.1 | AFTER 语句         | 174 |
| 9.2.2 | WAIT FOR 语句      | 175 |
| 9.2.3 | WAIT 语句          | 175 |
| 9.2.4 | ASSERT 语句        | 176 |
| 9.2.5 | 其他语句             | 176 |
| 9.3   | 仿真激励的产生          | 176 |
| 9.3.1 | 初始化              | 177 |

|        |                                   |     |
|--------|-----------------------------------|-----|
| 9.3.2  | 时钟信号                              | 177 |
| 9.3.3  | 复位信号                              | 178 |
| 9.3.4  | 数据信号                              | 179 |
| 9.4    | VHDL 测试平台                         | 180 |
| 9.4.1  | VHDL 测试平台综述                       | 180 |
| 9.4.2  | VHDL Test Bench 应用举例              | 182 |
| 9.4.3  | 基于 text IO 的 VHDL Test Bench      | 186 |
| 9.5    | 本章总结                              | 192 |
| 第 10 章 | Xilinx Spartan 3E 系列 FPGA 简介      | 193 |
| 10.1   | FPGA 基础知识                         | 193 |
| 10.1.1 | FPGA 简介                           | 193 |
| 10.1.2 | 使用 FPGA 的优势                       | 193 |
| 10.1.3 | FPGA 的分类                          | 194 |
| 10.2   | Xilinx Spartan 3E 系列 FPGA 简介      | 194 |
| 10.2.1 | 内部结构简介                            | 194 |
| 10.2.2 | 基本工作原理                            | 198 |
| 10.3   | 基于 FPGA 的开发与设计                    | 199 |
| 10.3.1 | FPGA 典型开发流程                       | 199 |
| 10.3.2 | FPGA 设计的性能指标                      | 201 |
| 10.4   | 本章总结                              | 202 |
| 第 11 章 | Xilinx ISE Design Suite 10.1 使用指南 | 203 |
| 11.1   | ISE Design Suite 10.1 简介与安装       | 203 |
| 11.1.1 | ISE Design Suite 10.1 版本新特点       | 203 |
| 11.1.2 | ISE Design Suite 10.1 的主要组件       | 203 |
| 11.1.3 | ISE Design Suite 10.1 的安装         | 206 |
| 11.2   | 基于 ISE 的 VHDL 程序开发流程              | 210 |
| 11.2.1 | 软件界面与基本操作                         | 210 |
| 11.2.2 | 新建工程                              | 213 |
| 11.2.3 | VHDL 代码输入                         | 214 |
| 11.2.4 | 功能仿真                              | 218 |
| 11.2.5 | 代码综合                              | 223 |
| 11.2.6 | 用户约束文件                            | 225 |
| 11.2.7 | 设计实现                              | 227 |
| 11.2.8 | 器件配置                              | 229 |
| 11.3   | 在线逻辑分析仪 ChipScope Pro             | 237 |
| 11.3.1 | ChipScope Pro 工具简介                | 238 |
| 11.3.2 | 在 ISE 中调用 ChipScope 组件            | 240 |
| 11.4   | 本章总结                              | 247 |
| 第 12 章 | 基本功能模块的 VHDL 实现                   | 248 |
| 12.1   | 分频电路设计                            | 248 |
| 12.1.1 | 偶数倍分频                             | 248 |
| 12.1.2 | 奇数倍分频                             | 249 |
| 12.2   | 同步整形与按键消抖                         | 252 |

|         |                       |     |
|---------|-----------------------|-----|
| 12.2.1  | 同步整形电路原理与实现           | 252 |
| 12.2.2  | 按键消抖电路原理与实现           | 253 |
| 12.3    | 按键扫描电路设计              | 255 |
| 12.3.1  | 矩阵式键盘简介               | 256 |
| 12.3.2  | 按键扫描电路的 VHDL 实现       | 259 |
| 12.4    | 数码管接口电路设计             | 261 |
| 12.4.1  | 七段数码管简介               | 261 |
| 12.4.2  | 动态扫描数码管接口电路的实现        | 263 |
| 12.4.3  | 特殊显示效果的实现             | 265 |
| 12.5    | 字符型 16×2 LCD 接口电路设计   | 269 |
| 12.5.1  | 字符型 16×2 LCD 简介       | 269 |
| 12.5.2  | 字符型 16×2 LCD 接口电路的实现  | 273 |
| 12.6    | PS/2 键盘接口电路设计         | 279 |
| 12.6.1  | PS/2 键盘标准简介           | 279 |
| 12.6.2  | PS/2 键盘接口电路的实现        | 282 |
| 12.7    | VGA 接口电路设计            | 284 |
| 12.7.1  | VGA 显示原理简介            | 284 |
| 12.7.2  | VGA 接口电路的实现           | 286 |
| 12.8    | 伪随机序列的产生              | 289 |
| 12.8.1  | 伪随机序列及其性质             | 289 |
| 12.8.2  | m 序列发生器的实现            | 291 |
| 12.9    | SPI 同步接口及其应用          | 292 |
| 12.9.1  | SPI 同步接口简介            | 292 |
| 12.9.2  | SPI 同步接口应用实例          | 293 |
| 12.10   | Xilinx FPGA 常用硬核模块的调用 | 296 |
| 12.10.1 | DCM 硬核模块的调用           | 296 |
| 12.10.2 | block RAM 的调用         | 300 |
| 12.11   | 本章总结                  | 305 |
| 第 13 章  | DES 算法的 VHDL 实现       | 306 |
| 13.1    | DES 算法的原理             | 306 |
| 13.1.1  | DES 概述                | 306 |
| 13.1.2  | DES 算法原理              | 306 |
| 13.1.3  | DES 解密方法              | 312 |
| 13.2    | DES 算法的 VHDL 实现       | 312 |
| 13.2.1  | 总体设计                  | 312 |
| 13.2.2  | 子密钥产生模块               | 313 |
| 13.2.3  | 组合逻辑运算模块              | 320 |
| 13.2.4  | 时序控制模块——状态机           | 336 |
| 13.3    | 系统综合与仿真               | 343 |
| 13.3.1  | 系统综合                  | 343 |
| 13.3.2  | 系统仿真                  | 344 |
| 13.4    | 本章总结                  | 348 |

|       |                          |     |
|-------|--------------------------|-----|
| 附录 A  | VHDL 关键字和预定义程序包 .....    | 349 |
| A.1   | VHDL 关键字 .....           | 349 |
| A.2   | VHDL 预定义程序包 .....        | 350 |
| A.2.1 | standard 程序包 .....       | 350 |
| A.2.2 | std_logic_1164 程序包 ..... | 352 |
| 参考文献  | .....                    | 357 |



# 第1章 概述

VHDL 是一种硬件描述语言,用于描述硬件的行为和结构。与软件编程语言被翻译成机器指令不同,VHDL 语言是被翻译成数字电路结构,最终在可编程逻辑器件或专用集成电路 ASIC 中以电路结构的形式实现程序所描述的功能。利用 VHDL 语言设计硬件电路需要一系列先进的 EDA 工具的支持,另一方面大多数 EDA 工具又都以硬件描述语言为主要输入方式。本章简要介绍可编程逻辑器件、EDA 技术、硬件描述语言以及 VHDL 语言的相关知识,为之后读者开始 VHDL 语言的学习与程序设计实践作必要的铺垫。

## 1.1 可编程逻辑器件

可编程逻辑器件(programmable logic device, PLD)是一种由用户编程来实现逻辑功能的新型逻辑器件。与只能实现固定逻辑功能的传统标准逻辑器件(如 74 系列的 TTL 器件)相比,PLD 器件具有可反复修改的特点,并且在满足专用的、个性化的设计需求方面具有更大的灵活性和竞争力。同时,可编程逻辑器件是 VHDL 程序实现的物理器件之一,本节简要介绍可编程逻辑器件的相关知识。



### 1.1.1 可编程逻辑器件概述

可编程逻辑器件是 20 世纪 70 年代发展起来的新型数字逻辑器件,是目前数字系统设计的主要硬件平台。

可编程逻辑器件是一种通用型逻辑器件,其逻辑功能是由使用者对器件进行编程和配置来设定的,且可以反复擦写。可编程逻辑器件采用 CMOS 工艺,其内部集成大量功能独立的分离元件,它们可以是基本逻辑门、由基本逻辑门构成的宏单元或者是“与阵列”、“或阵列”等门阵列。这些分离元件之间有大量可配置的连线,芯片出厂时,内部各个元件、单元间没有相互连接,因此芯片暂不具有任何逻辑功能。芯片内部各元件、单元如何连接,由用户根据需求通过 EDA 工具在计算机上编程实现。可编程逻辑器件根据下载的配置文件而实现不同的逻辑功能。

与传统的数字系统设计方法相比,使用可编程逻辑器件设计数字系统不但能够节省设计面积,而且使得设计具有很强的灵活性。随着可编程逻辑器件技术的发展,每个逻辑器件中门电路的数量愈来愈多,使得一个逻辑器件就能完成本来需要很多分立逻辑器件和存储芯片才能完成的工作。从而,大大减少了系统的功耗和成本,并提高了系统的可靠性。



### 1.1.2 可编程逻辑器件发展历史

可编程逻辑器件是在 20 世纪 70 年代发展起来的,至今已有三十多年。可编程逻辑器件的发展历程可以划分为 4 个阶段。

第 1 阶段为 20 世纪 70 年代初期到 70 年代中期,这一阶段的可编程逻辑器件只有简单的可编程只读存储器(programmable read only memory, PROM)、紫外线可擦除只读存储器(EPROM)、电可擦除只读存储器(EEPROM)3 种。由于结构的限制(只是可编程的 ROM),



它们只能完成简单的数字逻辑功能。

第2阶段为20世纪70年代中期到80年代中期,这一阶段出现了结构上稍微复杂的可编程芯片,正式被称为可编程逻辑器件(PLD),能够完成各种逻辑运算功能。这阶段的产品主要包括可编程阵列逻辑(programmable array logic, PAL)和通用阵列逻辑(generic array logic, GAL)。这时期典型的PLD通常由一个“与门”和一个“或门”阵列组成。由于任意一个组合逻辑都能用“与—或”表达式来描述,因此这些PLD能够以乘积和的形式完成大量的组合逻辑运算。

第3阶段为20世纪80年代中期到90年代末,Xilinx和Altera分别推出了类似于PAL结构的复杂可编程逻辑器件(complex programmable logic device, CPLD)和与标准门阵列类似的现场可编程逻辑门阵列(field programmable gate array, FPGA)。它们都具有体系结构和逻辑单元灵活、集成度高以及使用范围广等特点。与其他基于“门阵列”的ASIC相比,它们具有设计周期短、设计制造成本低、开发工具先进、质量稳定以及可实现在线检验等优点,因此被广泛应用于产品的原型设计和产品生产之中。

第4阶段为20世纪90年代末到目前,这一阶段出现了可编程的片上系统(system on programmable chip, SoPC)和片上系统(system on a chip, SoC)技术,是PLD与ASIC技术融合的结果,涵盖了实时信号处理技术、高速数据收发技术、复杂计算技术以及嵌入式系统设计技术的全部内容。各FPGA厂商都推出了SoC FPGA产品,芯片制造工艺达到65 nm,系统门数也超过了百万门。同时,这一阶段的可编程逻辑器件内嵌了高速硬核乘法器、Gbit差分串行接口、时钟频率高达数百兆的PowerPC微处理器以及软核处理器PicoBlaze和MicroBlaze等,超越了普通ASIC器件的性能和规模和传统的FPGA概念,使PLD的应用范围从单片扩展到系统级。



### 1.1.3 可编程逻辑器件分类

按照可编程逻辑器件内部包含的基本逻辑门数量,可编程逻辑器件可分为低密度可编程逻辑器件和高密度可编程逻辑器件。如图1-1所示,其中低密度器件包括前期的PROM, PLA, PAL和GAL;高密度器件包括EPLD, CPLD和FPGA。

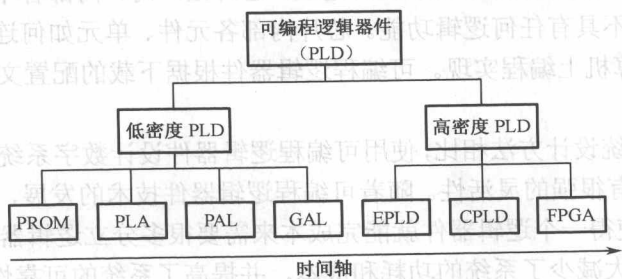


图 1-1 可编程逻辑器件分类

#### 1. 低密度可编程逻辑器件

低密度可编程逻辑器件包括一些可编程逻辑器件发展前期出现的产品,低密度可编程逻辑器件具有以下特征:

- 内部含有的逻辑门数量少,一般为几十门到750门等效基本逻辑门;
- 基本结构均建立在两级“与—或”门电路的基础上;

- 输出电路部分为可编程的输出逻辑宏单元。

可编程逻辑器件于 20 世纪 70 年代发展起来,经历了 PROM, PAL, PLA, 直到 80 年代初期出现的 GAL, 都是处于低密度可编程逻辑器件时期。这一时期, 可编程连线技术由最初的熔丝烧断技术发展到了反熔丝介质导通截止技术; 逻辑器件内的编程数据由不可擦除发展到能反复擦除; 数据擦除方式由速度较慢的紫外线光照擦除发展到电擦除。

PAL 是由一个可编程的“与”平面和一个固定的“或”平面构成, 或门的输出可以通过触发器有选择地被置为寄存状态。PAL 器件是现场可编程的, 实现工艺有反熔丝技术、EPROM 和 EEPROM 技术。还有一类结构更为灵活的逻辑器件是可编程逻辑阵列 (programmable logic array, PLA), 它也由一个“与”平面和一个“或”平面构成, 但是这两个平面的连接关系是可编程的。GAL 采用 EEPROM 技术, 实现了电可擦除、电可改写, 其输出结构是可编程的逻辑宏单元, 因而它的设计具有很强的灵活性。

## 2. 高密度可编程逻辑器件

高密度可编程逻辑器件有以下几种:

### 1) 可擦除的可编程逻辑器件 (erasable programmable logic device, EPLD)

EPLD 产生于 20 世纪 80 年代中期, 是高密度可编程逻辑器件的早期产品。EPLD 是 GAL 的改进, 其基本结构与 GAL 相似, 但是其集成密度、输出宏单元数目、器件内的连接机构都比 GAL 大得多且灵活、方便得多。

### 2) 复杂可编程逻辑器件 (complex programmable logic device, CPLD)

CPLD 产生于 20 世纪 80 年代末, 是 EPLD 的改进产品。CPLD 内部至少含有可编程逻辑宏单元、可编程 I/O 单元、可编程内部连线。这种结构特点是高密度可编程逻辑器件的共同特点。CPLD 是一种基于乘积项的可编程逻辑器件。部分 CPLD 器件内部有 RAM、FIFO 存储器。部分 CPLD 具有在系统编程能力 (in system programmable, ISP)。在系统编程时, 器件的输入/输出管脚暂时被关闭, 编程结束后恢复正常状态。

### 3) 现场可编程逻辑门阵列 (field programmable gate array, FPGA)

FPGA 是 20 世纪 90 年代发展起来的。大部分 FPGA 采用基于 SRAM 的查找表 (look up table, LUT) 结构, 也有一些军品和宇航级 FPGA 采用 Flash 或者反熔丝工艺的查找表结构以提高可靠性。FPGA 的突出特点是现场编程, 即在 FPGA 工作的现场, 不通过计算机, 就能把存于 FPGA 外部 ROM 中的编程数据加载给 FPGA。FPGA 的这个特点, 为工程人员维修、改进和更新电路逻辑功能提供了方便。

目前, 常用的可编程逻辑器件是 CPLD 和 FPGA, 它们都属于高集成度可编程逻辑器件。本书中 VHDL 的程序设计开发都是基于 Xilinx Spartan 3E 系列 FPGA 来实现, 第 10 章简要介绍 FPGA 的基础知识以及 Xilinx Spartan 3E 系列 FPGA 的内部结构。

## 1.2 EDA 技术与硬件描述语言

随着数字电路技术的迅速发展, 数字集成电路已从最初的几个门逻辑发展到现在的千万门级逻辑, 这使得电路系统的设计变得非常复杂, 已经超出设计人员的脑力运算, 必须借助计算机的自动化设计来完成。硬件描述语言是 EDA 工具进行设计输入的主要方式, 以硬件描述语言作为系统设计的工具具有多种优势。本节简要介绍 EDA 技术与硬件描述语言。



## 1.2.1 EDA 技术简介

### 1. 概述

从 20 世纪 60 年代中期（早于 PLD 开始发展的时间 70 年代），人们就开始研究各种计算机辅助设计（computer aided design, CAD）技术。CAD 主要表现为一些印制电路板（printed circuit board, PCB）软件，用于布线设计、电路模拟、版图绘制等。CAD 利用计算机的计算功能，将设计人员从大量繁琐的计算和绘图中解脱出来。

20 世纪 80 年代初，随着集成电路规模的快速发展，出现了计算机辅助工程（computer aided engineering, CAE）技术，主要表现为设计工具和单元库的完备，并具备原理图输入、编译和连接、逻辑模拟、测试代码生成以及版图自动布局等功能。CAE 软件需要针对产品开发，按照设计、分析、生产、测试等划分阶段，不同阶段使用不同软件，通过顺序循环使用这些软件完成整个设计过程。

基于 CAD 或 CAE 技术的设计过程，其自动化和智能化的程度并不高，需要使用多个软件完成一个完整的工程，各种软件千差万别，互不兼容，学习使用困难，直接影响到设计环节的衔接。同时，20 世纪 90 年代中期以后，微电子技术以惊人的速度发展，在单芯片上可集成数百万甚至数千万只晶体管，工作速度达到 GHz 以上，在这种芯片上设计系统就需要更加先进的工具来支持。因此，人们开始追求将整个设计过程自动化，即电子系统设计自动化（electronic design automation, EDA）。

EDA 是指利用计算机来完成电子系统的设计，以计算机和微电子技术为技术先导，汇集了计算机图形学、逻辑学、微电子工艺和结构学以及计算机数学等多种计算机应用学科最新成果的先进技术。目前，EDA 技术主要以硬件描述语言输入、系统级仿真和综合技术为特征，自动将用户以高级语言描述的功能需求转化为基础门电路，将设计封装到 FPGA/CPLD 中或制成 ASIC 芯片，极大提高了系统的设计效率，使设计人员摆脱了大量的辅助性和基础性工作，将精力集中于创造性的方案与算法设计和系统结构优化上。

EDA 工具种类多，系统集成度高，主要可以分为两类：一类是专业的 EDA 软件公司产品，比较著名的有 Mentor Graphics 公司系列产品（ModelSim, PADS, WG 等）、Cadence 公司系列产品（Incisive, Encounter, Allegro 等）、Altium 公司系列产品（Protel DXP 等）等；一类是半导体器件厂家开发的专门用于自身产品开发的 EDA 工具，比较著名的包括 Xilinx 公司的 ISE、Altera 公司的 Quartus II 以及 TI 公司的 CCS 软件等。

### 2. CPLD/FPGA 开发与 EDA 工具

基于 CPLD/FPGA 的开发完全由 EDA 工具来完成，包括 HDL 语言的逻辑编译、化简、分割、综合优化、布局布线、仿真以及对特定目标器件的适配编译和编程下载等工作。

基于 EDA 工具的 CPLD/FPGA 设计主要采用硬件描述语言作为设计输入，包括抽象行为与功能描述，甚至内部的具体线路结构；之后，借助编译综合工具将设计自动转化为底层逻辑模块。这个编译转换过程不需要人工参与，并且可以在系统设计的各个阶段、各个层次进行计算机模拟验证，保证设计过程的正确性。

典型的 CPLD/FPGA EDA 设计工具必须包含特殊的综合器软件包，综合器的功能是将设计者在 EDA 平台上完成的 HDL 语言、原理图描述或状态图描述，针对给定的硬件系统组件进行编译、优化和综合。作为全球最大的 CPLD/FPGA 器件提供商，Xilinx 公司和 Altera 公司





都已针对各自的产品推出自己的 EDA 设计工具。Xilinx 公司的 ISE 集成开发环境和 Altera 公司的 Quartus II 集成开发环境是全面包含 CPLD/FPGA 设计所需组件的 EDA 工具软件,包括设计输入、综合、仿真、布局布线、器件编程。同时,还有诸多 EDA 开发软件也很常用,包括综合软件 Synplify、仿真软件 modelSim 和软件辅助工具 matlab 等。



## 1.2.2 硬件描述语言简介

硬件描述语言 (hardware description language, HDL) 以文本形式来描述数字系统硬件结构和行为,是一种用形式化方法来描述数字电路和系统的语言,可以从上层到下层(从抽象到具体)逐层描述设计者的设计思想。

EDA 要求设计者用硬件描述语言来描述硬件的行为和功能。在利用 EDA 工具进行电子设计时,逻辑图、分立电子元器件作为整个越来越复杂的电子系统的设计已不适应。任何一种 EDA 工具都需要一种硬件描述语言作为其工作语言。这些众多的 EDA 工具软件开发各自推出了自己的硬件描述语言。据统计,目前在美国硅谷约有 90% 以上的 ASIC 和 FPGA 采用硬件描述语言进行设计。

VHDL 和 Verilog HDL 是目前最流行的两种硬件描述语言,均为 IEEE 标准,被广泛应用于基于可编程逻辑器件的项目开发中。VHDL 发展得较早,语法严格,而 Verilog HDL 是在 C 语言的基础上发展起来的一种硬件描述语言,语法较自由。国外电子专业很多在本科阶段教授 VHDL,在研究生阶段教授 Verilog HDL。事实上,两种语言的差别并不大,他们的描述能力也是类似的。掌握其中一种语言以后,可以通过短期的学习,较快地学会另一种语言。选择何种语言主要还是看周围人群的使用习惯,这样可以方便日后的学习交流。集成电路 (ASIC) 设计人员,则必须首先掌握 Verilog HDL,因为在 IC 设计领域,90% 以上的公司都是采用 Verilog HDL 进行 IC 设计。对于 CPLD/FPGA 设计者而言,两种语言可以自由选择。

现在,随着系统级 FPGA 以及系统芯片的出现,软硬件协调设计和系统设计变得越来越重要。传统意义上的硬件设计越来越倾向于与系统设计和软件设计结合。硬件描述语言为适应新的情况,迅速发展,出现了很多新的硬件描述语言,像 Superlog, SystemC, Cynlib C++ 等。究竟选择哪种语言进行设计,整个业界正在进行激烈的讨论。

## 1.3 VHDL 语言简介



### 1.3.1 VHDL 语言概述

VHDL (VHSIC hardware description language, 超高速集成电路硬件描述语言), VHSIC 即 very high speed integrated circuit 的缩写。VHDL 由美国国防部 (DOD) 制定,1987 年被采纳为 IEEE 1076 标准,1993 年被更新为 IEEE 1164 标准。VHDL 是第一个被 IEEE 标准化的硬件描述语言。

美国国防部在 20 世纪 70 年代末和 80 年代初提出了 VHSIC 计划,其目标是为下一代集成电路的生产实现阶段性的工艺极限,以及为完成 10 万门级以上的设计,建立一种新的描述方法。1981 年,美国国防部提出了 VHDL 语言。1986 年,IEEE 致力于 VHDL 的标准化工作,为此成立了 VHDL 标准化小组,经过多次的修改和扩充,直到 1987 年 12 月 VHDL 才被纳入 IEEE 1076 标准。1988 年, Milstd 454 规定,所有为美国国防部设计的 ASIC 产品必须采用 VHDL